Sicherheit in Rechnernetzen:

Mehrseitige Sicherheit in verteilten und durch verteilte Systeme

Skript[1] zu den Vorlesungen Datensicherheit (2+2) und Kryptographie (2+2)[2]

Andreas Pfitzmann

TU Dresden, Fakultät Informatik, D-01062 Dresden,
Tel. 0351 / 463-8277, e-mail: pfitza@inf.tu-dresden.de, http://ikt.inf.tu-dresden.de/

Vorwort

Menschen sind in immer stärkerem Maße auf die Dienstleistungen von Rechnern und ihren Zusammenschluß zu Netzen angewiesen. Deshalb müssen von Rechnernetzen zwei Sicherheitseigenschaften nachgewiesen werden:

1. Sie erbringen die erwarteten Dienste (≈ Totale Korrektheit).
2. Sie erbringen keine verdeckten Dienste (≈ Datenschutz[3]). Andernfalls könnten z.B. Kommunikationsnetze oder Zahlungssysteme vorzüglich als Überwachungsinstrumente genutzt werden.

Beides muß gelten, auch wenn nicht nur dumme Fehler, sondern intelligente Angreifer[4], nämlich manche der immer zahlreicheren Entwerfer, Hersteller, Betreiber, Programmierer, Wartungstechniker, Benutzer von Rechnern etc. versuchen, dies zu verhindern. Wie Rechnernetze so strukturiert werden können, daß beide Sicherheitseigenschaften unter möglichst schwachen Annahmen nachweisbar sind, ist das Thema der Vorlesungen.

Als Einführung werden hierzu verschiedene Angreifermodelle diskutiert und ein kurzer Überblick über Schutz in einzelnen Rechnern gegeben, etwa wie die Gefahr von Computer-Viren auf die durch transitive Trojanische Pferde beschränkt werden kann.

Sodann werden die benötigten kryptographischen und steganographischen Hilfsmittel vorgestellt.

Danach wird die Gestaltung von Datenschutz garantierenden Kommunikationsnetzen vertieft behandelt, denn einerseits sind sie die Trägerstruktur für vielerlei Anwendungsprotokolle, andererseits lassen sich an ihnen die grundlegenden Probleme und Lösungen besonders klar zeigen.

Als Beispiele für Anwendungsprotokolle dienen Werteaustausch- und digitale Zahlungssysteme sowie ihre Verallgemeinerung zu umrechenbaren Autorisierungen (credentials).

Zur Abrundung des mathematischen bzw. politischen Weltbildes werden verteilte Berechnungsprotokolle skizziert und die Regulierbarkeit von Sicherheitstechnologien untersucht.

Eine Zusammenfassung und ein Ausblick, in dem diskutiert wird, inwieweit die dargelegten Datenschutz- und Datensicherungs-Verfahren in verteilten Systemen es nahelegen, selbst bisher zentralisierte Systeme aus Schutzgründen verteilt zu realisieren, beschließen das Skript.

1

2 Datensicherheit umfaßt §1 bis §3.3, §3.6, sowie §5, §6, §9 und §10, Kryptographie §1.2, §3, §4, §6.4.1 (Implementierung (einmal) umrechenbarer Autorisierungen mittels RSA = blindes Leisten von Signaturen), §7 und §8.

3 **Datenschutz** *im engeren Sinne* umfaßt alle Vorkehrungen zur Verhinderung unerwünschter (Folgen der) Datenverarbeitung für *Individuen*. Im weiteren Sinne werden auch die Folgen für Personengruppen, den Staat etc. betrachtet. Oft werden unter Datenschutz vor allem juristische Vorkehrungen verstanden. Bei dieser eingeschränkten Sicht ist ein Gegensatz zu Datenschutz dann **Datensicherung**: Maßnahmen technischer, programmäßiger, organisatorischer, personeller und sonstiger Art zum Schutz der Datenverarbeitung. Zugespitzt dient Datenschutz nicht dem Schutz *von* Daten, sondern eher dem Schutz *vor* Daten. Datensicherung schützt die Daten(verarbeitung).

4 Angreifer wird hier und im folgenden ohne moralisch/rechtliche Bedeutungsassoziation zur Charakterisierung von Einflußmöglichkeiten gebraucht.

Für die Zukunft: Eine Bitte

Dieses Skript enthält sicher etliche Fehler und vermutlich noch mehr schwer verständliche Stellen. An Hinweisen auf solche Unschönheiten habe ich größtes Interesse. Neben einem Dankeschön (und später einer verbesserten Version des Skripts) biete ich ggf. einen mündlichen Erklärungsversuch für Unverständliches an. Merke: Es gibt keine falschen Fragen, nur falsche Skripten!

Über die Vergangenheit

Vieles stände hier nicht oder jedenfalls unverständlicher, manches wäre vielleicht immer noch unbekannt, hätte ich nicht seit einem guten Jahrzehnt die kritische und konstruktive Herausforderung und Begleitung durch Mitglieder unserer[5] Forschungsgruppe. Deshalb

ein herzliches Dankeschön an ...

Prof. Dr. *Birgit Pfitzmann* und Dr. *Michael Waidner*, die mich immer wieder gefragt haben: Und was soll das Verfahren nun genau leisten? Kannst Du das definieren? Oft haben sie entscheidend mitgeholfen, dies herauszufinden und aufzuschreiben. Beide haben an Stoffsammlung und -anordnung meiner ersten Vorlesung über dieses Gebiet deutlichen Anteil. Beide haben mich bei den ersten Vorlesungsdurchläufen hin und wieder vertreten, oftmals während ich in Bonn oder Brüssel Wissenschaftspolitiker zum Handeln, möglichst sogar zum richtigen Handeln, bzgl. Rechner(un)sicherheit zu gewinnen suchte. Bei diesen Gelegenheiten haben beide nicht nur teilweise heute noch benutzte Folien erstellt, sondern Birgit hat auch manche Teile dieses Skripts geschrieben oder korrigiert. Besonderer Dank und Anerkennung gebührt ihr für die gelungene erste Fassung von §3.2 bis §3.5.

Viel profitiert hat dies Skript auch von Fragen und Anregungen der *HörerInnen der Vorlesungen* an der Univ. Karlsruhe, Univ. Hildesheim und TU Dresden.

Unter den HörerInnen des ersten Durchgangs in Karlsruhe war die Zusammenarbeit mit vier damaligen Studenten besonders eng und fruchtbar: *Gerrit Bleumer*, *Manfred Böttger*, *Dirk Fox* und *Andreas Ort* hatten entscheidenden Anteil an der Gestaltung und Realisierung des die Vorlesung ergänzenden (aber auch unabhängig von ihr absolvierbaren) Praktikums. Die gemeinsame Erstellung und Diskussion der Praktikumsunterlagen hatte zahllose Verbesserungen des Vorlesungsskripts zur Folge.

Verbesserungsvorschläge verdanke ich inzwischen auch meiner „neuen“ Umgebung an der TU Dresden: Dr. *Hannes Federrath*, *Anja Jerichow*, Dr. *Herbert Klimant* und *Andreas Westfeld*.

Intensive Diskussionen und gemeinsame Texte über die (Un)Regulierbarkeit von Kryptographie (§9.1-§9.3, §9.5-§9.7) sowie einen kritischen Blick aufs Ganze verdanke ich Dr. *Michaela Huhn*.

Lob an diesem Text gebührt zu einem nicht kleinen Teil den hier Genannten, wie auch eine Entschuldigung meinerseits, manchen Änderungsvorschlag partout nicht als Verbesserungsvorschlag zu begreifen. Folglich gebührt alle Kritik an diesem Text allein mir.

---

5 Auf keinen Fall als Pluralis Majestatis gemeint, sondern als ein partnerschaftliches wir.

Inhaltsverzeichnis

Bilderverzeichnis

1 Einführung

Um das Anwendungsgebiet zu umreißen, wird zuerst ein kurzer Überblick über *Rechnernetze* und ihre Dienste gegeben.

Danach wird ausführlicher darauf eingegangen, was unter *Sicherheit* (security) verstanden wird, insbesondere welche allgemeinen Fragen zu stellen sind: Was ist zu schützen? Vor wem ist zu schützen? Wie und wodurch kann Sicherheit erreicht werden?

Zum Schluß der Einführung wird konkretisiert, was folglich *Sicherheit in Rechnernetzen* bedeuten sollte, und verallgemeinert, warum *mehrseitige Sicherheit* sinnvoll und notwendig ist.

1.1 Was sind Rechnernetze (verteilte offene Systeme)?

Zuerst soll kurz der Anwendungsbereich dieses Skripts skizziert werden:

Was sollen und sind Rechnernetze?
Was versteht man unter verteilten offenen Systemen?
Welche gibt es?
Welche Dienste sind realisiert oder geplant, welche denkbar?

Nachdem der Mensch Rechner geschaffen und zu einer gewissen Vollkommenheit weiterentwickelt hatte, sah er, daß Rechner für ihn nützlicher wären, könnten sie miteinander kommunizieren, um so nicht nur begrenzte Rechen- und Merkfertigkeiten des Menschen, sondern auch seine begrenzten Kommunikationsreichweiten und Fortbewegungsmöglichkeiten zu kompensieren. Menschen begannen, **Rechner** über **Kommunikationsnetze** zu **Rechnernetzen** (*erster Art*) zu verbinden. Nach und nach wurden Teile des Kommunikationsnetzes, etwa elektromechanische Vermittlungseinrichtungen, durch Prozeßrechner ersetzt, so daß heute praktisch alle Kommunikationsnetze vom Funktionieren von Rechnern abhängig sind, und es sich somit um Rechnernetze *zweiter Art* handelt.

In seiner nächsten Kulturstufe entdeckte der Mensch, daß er viele unterschiedliche Rechner geschaffen hatte, die keineswegs elektrischen Signalpegeln, Bits, Zeichen, Datenstrukturen immer die gleiche Bedeutung gaben (denn sie war ja nicht gottgegeben), so daß mit einer physischen Kommunikationsmöglichkeit zwar ein **räumlich verteiltes** informationstechnisches System (IT-System) geschaffen war, dies aber nicht in dem Sinne **offen** war, daß zwei beliebige Rechner miteinander Botschaften (Files, Dokumente, Bilder ...) bedeutungserhaltend austauschen und diese sinnvoll weiterbearbeiten konnten. Seitdem beschäftigen sich Heerscharen von Technikern mit der Normung zunächst eines Referenzmodells für offene Kommunikation, danach einzelner Kommunikationsprotokolle, die in dieses Referenzmodell passen, und schließlich ihrer Implementierung auf den unterschiedlichsten Rechnertypen. Währenddessen entdeckten findige Techniker, daß Rechner, die miteinander kommunizieren können, dazu nicht unbedingt viele Kilometer voneinander entfernt sein müssen. Steigerung von Verfügbarkeit (Fehlertoleranz wird durch die Begrenzung der physischen Auswirkungen von Fehlern unterstützt) oder Durchsatz (Parallelarbeit) durch Entkopplung sprach zunehmend für die Realisierung **bezüglich ihrer Kontroll- und Implementierungsstruktur verteilter** IT-Systeme. Allgemein wird folglich von einem **verteilten** System gesprochen, wenn es keine Instanz gibt, die eine globale Systemsicht hat. Folglich kann ein verteiltes System auch nicht von einer Instanz zentral gesteuert werden.

Zur Geschichte der Rechnernetze

1833 erster elektromagnet. Telegraph [Meye_87 Band 22 Seite 20]
1858 erste Kabelverbindung zwischen Europa und Nordamerika [Meye_87 Band 22 Seite 20]
1876 Fernsprechen über 8,5 km lange Versuchsstrecke [Meye_87 Band 7 Seite 45]

1881 erstes Fernsprechortsnetz [Meye_87 Band 7 Seite 45]
1900 Beginn der drahtlosen Telegraphie [Meye_87 Band 22 Seite 20]
1906 Einführung des Selbstwählferndienstes in Deutschland, realisiert durch Hebdrehwähler, d.h. erste vollautomatische Vermittlung durch Elektromechanik [Meye_87 Band 7 Seite 43]
1928 Fernsprechdienst Deutschland – USA eingeführt (über Funk) [Meye_87 Band 7 Seite 45]
1949 erster funktionierender von-Neumann-Rechner [BaGo_84 Seite 305]
1956 erstes Transatlantikkabel für Fernsprechen [Meye_87 Band 7 Seite 45]
1960 erster Fernmeldesatellit [Meye_87 Band 7 Seite 45]
1967 Beginn des Betriebes des Datex-Netzes durch die Deutsche Bundespost, d.h. des ersten speziell für Rechnerkommunikation realisierten Kommunikationsnetzes (Rechnernetz erster Art). Die Übertragung erfolgt digital, die Vermittlung durch Rechner (Rechnernetz zweiter Art). [Meye_81]
1977 Einführung des Elektronischen Wähl-Systems (EWS) für Fernsprechen durch die Deutsche Bundespost, d.h. erstmals Vermittlung durch Rechner (Rechnernetz zweiter Art) im Fernsprechnetz, aber weiterhin analoge Übertragung [Meye_87 Band 7 Seite 44]
1981 erster persönlicher Rechner (PC) der Rechnerfamilie (IBM PC), die weite Verbreitung auch im privaten Bereich findet [SmRe_90 Seite 382]
1982 Investitionen in die Übertragungssysteme des Fernsprechnetzes erfolgen zunehmend in digitale Technik [ScSc_84 Seite 8, ScSc_86 Seite 7]
1985 Investitionen in die Vermittlungssysteme des Fernsprechnetzes erfolgen zunehmend in rechnergesteuerte Technik, die nunmehr nicht mehr analoge, sondern digitale Signale vermittelt [ScSc_84 Seite 8, ScSc_86 Seite 7]
1988 Betriebsbeginn des ISDN (Integrated Services Digital Network)
1989 erster westentaschengroßer PC: Atari Portfolio; damit sind Rechner im engeren Sinne persönlich und mobil.
1993 Die zellularen Funknetze nach dem GSM-Standard (in Deutschland: D1, D2) entwickeln sich zu einem Massendienst mit mehreren Hunderttausend Teilnehmern. Es werden ausschließlich digitale Signale übertragen und durch Rechner vermittelt.
1994 Das World Wide Web (www), ein verteiltes Hypertextsystem, in das auch Grafiken und Bilder eingebunden werden können, erschließt dem Internet so viele Anwendungsmöglichkeiten auch für ungeübte Benutzer, daß ernsthaft über eine Kommerzialisierung (entsprechend auch Verrechtlichung, leider auch Zensur) diskutiert wird.
1998 Alle Vermittlungsstellen der Deutschen Telekom sind digitalisiert [Tele_00], d.h. der 1985 begonnene Prozeß der Umstellung aller Vermittlungssysteme auf rechnergesteuerte Technik ist abgeschlossen.
2000 Über sogenannte WAP-fähige zellulare Funktelefone ist mobiler Zugriff auf geeignet gestaltete Inhalte des www möglich. Im Frühjahr werden WAP-fähige Handys (140 g) mit einer Guthabenkarte (25 DM Startguthaben) ohne Vertragsbindung für 199 DM verkauft, um die Entwicklung eines Massenmarktes zu fördern.

Folgende Dienste sind 1991 mittels folgender Kommunikationsnetze realisiert:

- Hörfunk, Fernsehen, Videotext z.B. sind Dienste, die (genauer: deren Informationen) vorwiegend über das Rundfunksendernetz, mehr und mehr aber über das entstehende Breitbandkabelverteilnetz **verteilt** werden. Zweck des Breitbandkabelverteilnetzes ist die Verbesserung der Dienstqualität durch Erhöhung der verfügbaren Bandbreite: mehr empfangbare Fernsehprogramme heißt dann Kabelfernsehen, größeres Informationsangebot bei Videotext heißt dann Kabeltext.
- Fernsprechen, Bildschirmtext, elektronisches Postfach (TELEBOX) für elektronische Brief- und Sprachpost, Fernschreiben (TELEX, TELETEX), Fernkopieren (TELEFAX) und Fern-

wirken (TEMEX) sowie Telebanking (electronic cash) sind Dienste, die über das Fernsprechnetz bzw. das digitale Text- und Datennetz **vermittelt** werden, vgl. Bild 1-1. Das 1991 im Teilnehmeranschlußbereich noch fast überall analoge, im Fernbereich bereits weitgehend digitale Fernsprechnetz wird seit 1988 in einigen Ortsnetzen und ab etwa 1993 in der ganzen Bundesrepublik auch im Teilnehmeranschlußbereich nach und nach digitalisiert und danach jeweils alle diese Dienste mit höherer Qualität erbringen.
Das analoge Fernsprechnetz und das dieselben Teilnehmeranschlußleitungen benutzende digitale Fernsprechnetz sind **schmalbandige** Netze (≤ 1 Mbit/s), d.h. sie sind im Gegensatz zu **breitbandigen** Netzen (> 1 Mbit/s) nicht in der Lage, Bewegtbilder hoher Qualität (z.B. Fernsehen) zu übertragen.

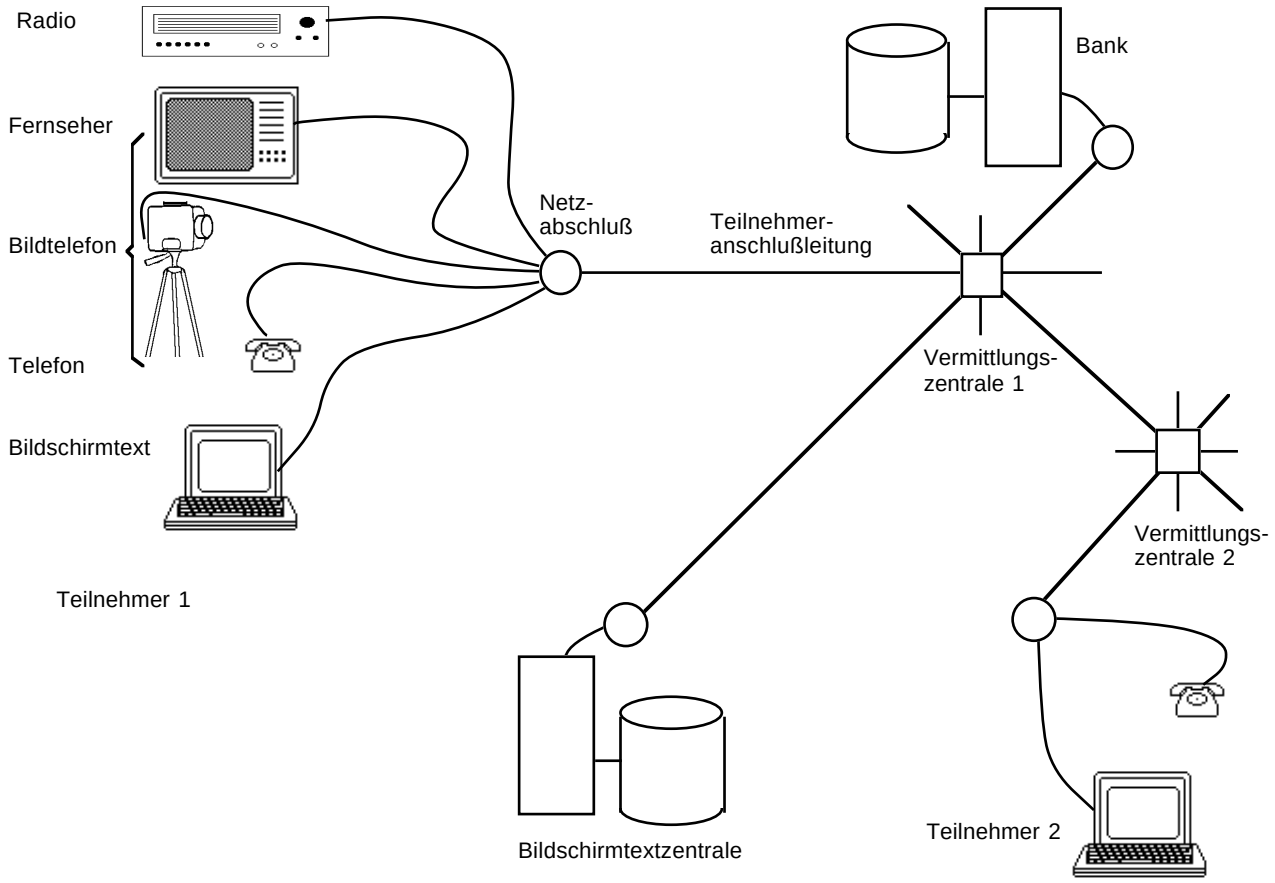


Bild 1-1: Ausschnitt eines Rechnernetzes

Zur Zeit benutzen wir also zwei grundsätzlich verschiedene Typen von Kommunikationsnetzen:

- **Verteilnetze**, in denen alle Teilnehmerstationen vom Netz dasselbe erhalten und jeder Teilnehmer lokal auswählt, ob und, wenn ja, was er tatsächlich empfangen will, und
- **Vermittlungsnetze**, in denen jede Teilnehmerstation vom Netz individuell nur das erhält, was der Teilnehmer angefordert oder ein anderer Teilnehmer an ihn gesendet hat.

In fast allen realisierten und geplanten öffentlichen Verteilnetzen findet Kommunikation nur in einer Richtung, vom Netz zum Teilnehmer, statt [Krat_84, DuD_86]; in Vermittlungsnetzen wird generell in beiden Richtungen kommuniziert.

Da langfristig ein Netz für alle Dienste, ein sogenanntes **diensteintegrierendes** Netz, zumindest im Teilnehmeranschlußbereich preiswerter als mehrere verschiedene Netze ist, und da alle verteilten

Dienste auch vermittelt werden können, wird angestrebt, alle Dienste in einem Netz zu vermitteln [Schö_84, ScSc_84, ScS1_84, Rose_85, Thom_87].

Damit auch breitbandige Dienste vermittelt zum Teilnehmer übertragen werden können, müssen neue Teilnehmeranschlußleitungen (**Glasfaser**) verlegt werden. Nach und nach wird dadurch ein Breitbandkabelverteilnetz überflüssig. Über ein breitbandiges diensteintegrierendes Vermittlungsnetz können nicht nur alle Dienste angeboten werden, die über ein Breitbandkabelverteilnetz und ein schmalbandiges Vermittlungsnetz zusammen angeboten werden können, sondern auch noch zusätzlich Dienste, die breitbandige Kommunikation zwischen Teilnehmern erfordern, z.B. Bildfernsprechen.

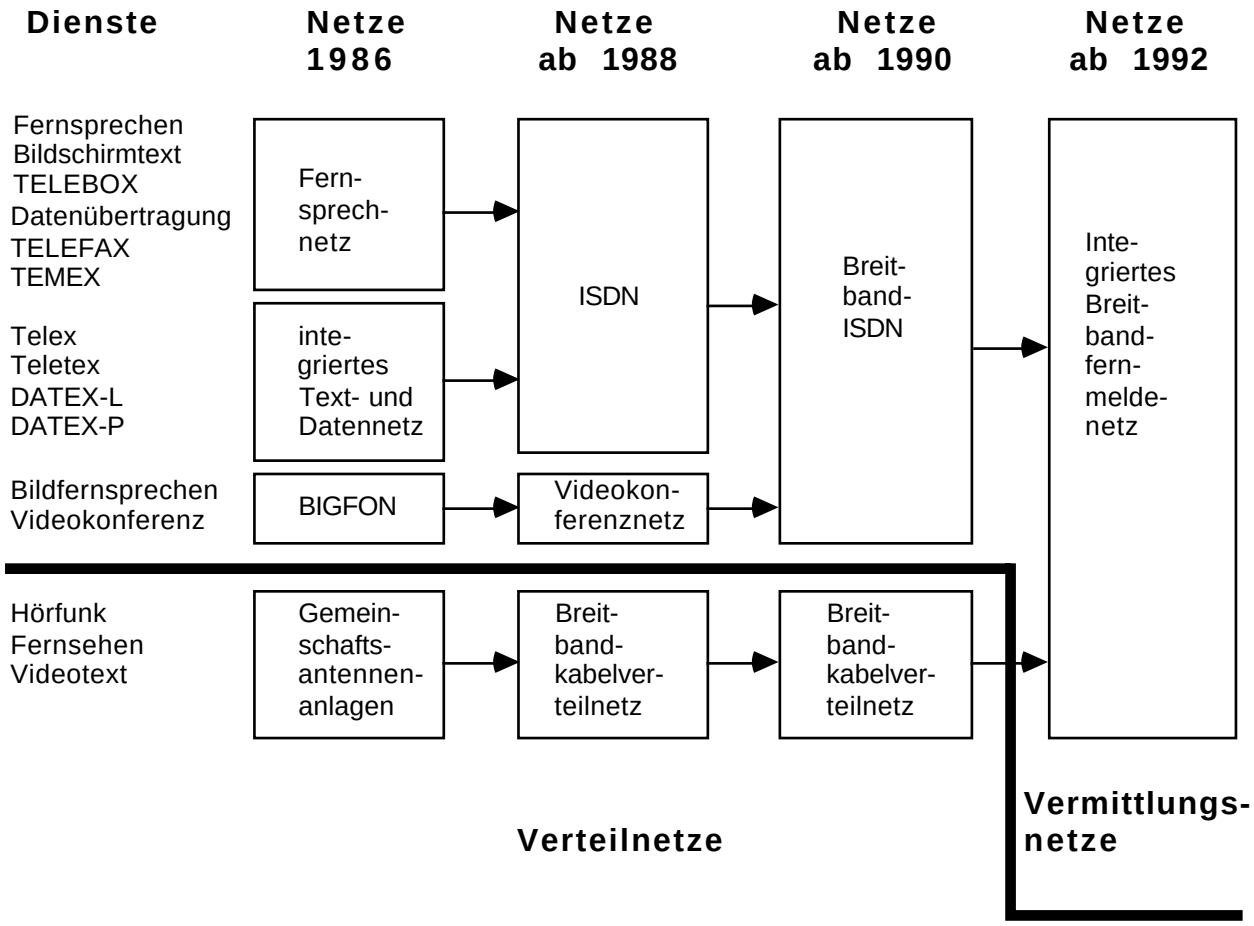


Bild 1-2: Entwicklung der leitungsgebundenen Kommunikationsnetze der Deutschen Bundespost Telekom (jetzt Deutsche Telekom AG) wie 1984 geplant und im Wesentlichen realisiert

Da digitale Werte in modernen technischen Systemen nicht nur leichter übertragen, sondern auch leichter verarbeitet, insbesondere vermittelt, werden können, wird das diensteintegrierende Netz ein von Teilnehmerstation zu Teilnehmerstation **digitales** Netz sein. Solch ein Kommunikationsnetz nenne ich **„diensteintegrierendes Digitalnetz“**, während ich die Abkürzung **ISDN** (Integrated Services Digital Network) nur für das von der Deutschen Telekom AG betriebene, weiteren Einschränkungen genügende diensteintegrierende Digitalnetz verwende. Schon 1986 ersparte die Digitalisierung 40% der Kosten bei der Fernvermittlungstechnik und war bei der Ortsvermittlungstechnik mit fallender Tendenz schon damals nicht teurer [Schö_86]. Beide bringen je 65% Raumersparnis, was die Hochbaukosten enorm senkt, und der schnellere Verbindungsaufbau und die größere Netzflexibilität sparen 15 bis 20% der erforderlichen Übertragungskapazität.

Ersetzt das breitbandige ISDN (abgekürzt Breitband-ISDN oder B-ISDN [Steg_85]) das Breitbandkabelverteilnetz, wodurch einerseits der Unterhalt des Breitbandkabelverteilnetzes eingespart und andererseits hochauflösendes Fernsehen (High Definition TV, HDTV) in größerem Umfang angeboten werden kann, wird es Integriertes Breitbandfernmeldenetz (IBFN) genannt [ScS1_84, Thom_87]. Pilotversuche zur Erprobung des IBFN sind unter der Abkürzung BIGFON (Breitbandiges integriertes Glasfaser-Fernmeldeortsnetz) bekannt [Brau_82, Brau_83, Brau_84, Brau_87]. Inzwischen ist der Name IBFN etwas aus der Mode gekommen und nach Aufhebung des Netzmonopols ist das Ziel einer derart weitreichenden monolithischen Infrastruktur auch weniger öffentlichkeitswirksam. Obwohl es inzwischen mehrere Netzbetreiber gibt, die jeweils nicht mit dem Ziel IBFN werben, geht die technische Entwicklung weiterhin in diese Richtung. Beispielsweise liegen in den Neuen Bundesländern inzwischen Glasfasern bis in die meisten Häuser – wenn auch meistenteils bisher ungenutzt. Letzteres kann vergleichsweise schnell geändert werden.

Seit etwa 1994 wird über die Vision eines alle Dienste integrierenden Netzes weniger aus der Sicht der Übertragungstechnik diskutiert, wie im Jahrzehnt davor, sondern mehr aus der Sicht der Rechnerkommunikation: Das Internet übernimmt die Hoffnungen, Ziele und Illusionen von IBFN, es soll *das* Medium auch für Massenkommunikation werden. Statt gigantischer Bandbreite wie bei IBFN stehen hochwertige Datenkompressionstechniken und adaptive Dienstgestaltung im Mittelpunkt des Interesses.

Die geschilderte Entwicklung ist in Bild 1-2 etwas detaillierter dargestellt. Nicht dargestellt und im folgenden auch erst in §5.7 explizit behandelt sind Funknetze, die als Anschlußnetze für ortsbewegte Teilnehmer, als Ersatznetze in Katastrophenfällen und als Garanten grenzüberschreitender Informationsfreiheit bleibende Bedeutung haben.

Neben diesen digitalen **öffentlichen Weitverkehrsnetzen** gibt es **lokale Rechnernetze** (**LANs**) – ohne sie hätte dieses Skript den Weg vom Rechner zum Drucker nicht so bequem gefunden. Auch sind beispielsweise moderne Krankenhäuser und Verwaltungen bereits heute auf das Funktionieren ihrer LANs auf Gedeih und Verderb angewiesen. Zukünftig werden nicht nur Flugzeuge von Rechnernetzen kontrolliert (Schlagwort: fly-by-wire; erstes ziviles Flugzeug ist der Airbus 320, erster Flug 1988 oder früher), sondern auch Autos. Bei letzteren wird von **CANs** (**Controller Area Networks**[6]) gesprochen, die Rechner an Instrumenten, Motor und Bremsen verbinden.

Bereits heute sind viele LANs, oftmals via öffentlicher Weitverkehrsnetze, verbunden. Und auch bei CANs kann dies sinnvoll sein.

Kurzum: Oftmals bereits heute, in wenigen Jahren aber nahezu umfassend, werden Interaktionen mit technischen Geräten Ein- und Ausgaben an und von Rechnernetzen sein. Daß wir mit einem Rechnernetz interagieren, wird uns dabei immer weniger auffallen, da die Rechner in ihrem Erscheinungsbild in den Hintergrund treten, nicht aber bzgl. ihrer Sicherheit(sprobleme).

1.2 Was bedeutet Sicherheit?

In diesem zweiten Teil der Einführung wird nun darauf eingegangen, welche Gefahren allgemein in und von informationstechnischen Systemen (IT-Systemen) drohen. Man denke an Fehler von Menschen, Ausfälle von integrierten Schaltkreisen, aber auch bewußt böswilliges Verhalten von Menschen, die sich hierzu natürlich auch der Hilfe von IT-Systemen (Rechner, Kommunikationssysteme, etc.) bedienen können.

Nach einer kurzen Aufzählung, *was* zu schützen ist, wird diskutiert, vor *wem* zu schützen ist und auf welchen *Ebenen* die Schutzmechanismen ansetzen müssen.

---

6 Manchmal wird dieser Name nicht für eine Klasse von Rechnernetzen, sondern für ein spezielles benutzt: Bosch entwickelte das „Controller Area Network (CAN)“, das inzwischen international genormt ist (ISO 11898).

1.2.1 Was ist zu schützen?

Eine übliche Einteilung der *Bedrohungen* und korrespondierenden **Schutzziele** für Systeme der Informationstechnik (IT-Systeme) ist folgende Dreiteilung [VoKe_83, ZSI_89]:

1. *unbefugter Informationsgewinn*, d.h. Verlust der **Vertraulichkeit** (confidentiality),
2. *unbefugte Modifikation von Informationen*, d.h. Verlust der **Integrität** (integrity), und
3. *unbefugte Beeinträchtigung der Funktionalität*, d.h. Verlust der **Verfügbarkeit** (availability).

Entsprechende Beispiele mögen dies erläutern:

1. Werden die Krankengeschichten (Untersuchungen, Diagnosen, Therapieversuche, ...) nicht mehr auf Karteikarten, sondern in Rechnern gespeichert, so ist es sicherlich unerwünscht, daß der Rechnerhersteller bei Wartungs- oder Reparaturmaßnahmen lesenden Zugriff auf diese Daten erhält.

2. Lebensgefährlich für die Patienten kann es werden, wenn jemand unbefugt und *unbemerkt* Daten ändert, z.B. die Dosierungsanweisung für ein zu verabreichendes Medikament.

3. Ebenfalls lebensgefährlich kann es werden, wenn die Krankengeschichte nur im Rechner gespeichert ist, dieser aber gerade *erkennbar* ausgefallen ist, wenn eine Abfrage für eine Therapiemaßnahme erfolgen muß.

Leider ist keine Klassifikation der Bedrohungen bekannt. Insbesondere ist die obige Dreiteilung selbst mit den von mir bei den Beispielen kursiv zugefügten Attributen *keine Klassifikation*: Wird ein gerade nicht ausgeführtes Programm im Speicher unbefugt modifiziert, handelt es sich um unbefugte Modifikation von Information. Wird das in gleicher Weise unbefugt modifizierte Programm ausgeführt, handelt es sich um eine unbefugte Beeinträchtigung der Funktionalität.

Wegen dieser durch die Interpretierbarkeit von Information bedingten Abgrenzungsschwierigkeiten wollen manche Wissenschaftler nicht zwischen der 2. und 3. Bedrohung unterscheiden.

Ich persönlich halte diese Unterscheidung aber unter pragmatischen Gesichtspunkten für sinnvoll. Denn bei den später behandelten Schutzmaßnahmen kann man meist klar unterscheiden, welchem der drei Schutzziele sie dienen.

Zusätzlich hat diese Unterscheidung eine **Entsprechung im Korrektheitsbegriff**, wenn man die von mir bei den Beispielen kursiv zugefügten Attribute *unbemerkt* bzw. *erkennbar* in die Definitionen von Integrität bzw. Verfügbarkeit hineinnimmt:

Integrität (= keine unbefugte und unbemerkte Änderung von Information) entspricht dann der *partiellen Korrektheit* eines Algorithmus: Wenn er ein Ergebnis liefert, ist es richtig.

Integrität und Verfügbarkeit zusammen entsprechen dann der *totalen Korrektheit*[7] eines Algorithmus: Es wird ein richtiges Ergebnis geliefert (genügend Betriebsmittel für die Ausführung des Algorithmus vorausgesetzt, s.u.).

Charakteristisch für fast alle Anwendungen mit Verfügbarkeitsanforderungen ist, daß richtige Ergebnisse nicht irgendwann in der Zukunft, sondern zu realen Zeitpunkten (nächste notwendige Therapiemaßnahme im obigen Beispiel) benötigt werden. Zugespitzt sorgen Maßnahmen zur Sicherung der Integrität dafür, daß das Richtige geschieht und Maßnahmen zur Sicherung der Verfügbarkeit, daß es rechtzeitig geschieht [Hopk_89]. Der Nachweis der totalen Korrektheit allein reicht also im allgemeinen als Nachweis für Verfügbarkeit nicht aus. Zusätzlich müssen mindestens noch die benötigten Betriebsmittel analysiert und ihr ausreichendes verfügbar sein nachgewiesen werden. Also gilt:

Integrität und Verfügbarkeit	≈	totale Korrektheit und ausreichend Betriebsmittel verfügbar

Ein paar Anmerkungen zur obigen Einteilung von Bedrohungen und Schutzzielen sind angebracht:

---

7 Bei sequentiellen Algorithmen gilt: totale Korrektheit = partielle Korrektheit und Terminierungsnachweis. Bei reaktiven Systemen ist statt des Terminierungsnachweises der Nachweis von Fairness- und Lebendigkeitseigenschaften notwendig.

Der in [ZSI_89, ITSEC2d_91] verwendete Begriff *unbefugt* muß sehr *weit verstanden* werden, denn sonst sind die obigen drei Bedrohungen unvollständig: Der Ausfall beispielsweise eines Transistors kann durchaus Informationen modifizieren oder die Funktionalität beeinträchtigen, ja sogar einen Verlust der Vertraulichkeit bewirken. Ob er aber nach allgemeinem Sprachgebrauch ein *unbefugtes* Ereignis mit *unbefugten* Folgen darstellt, ist unklar. Denn Befugnisse werden im allgemeinen nur Instanzen mit Verantwortung, also Menschen, übertragen. Instanzen ohne Verantwortung können dann nicht unbefugt handeln, also auch nichts Unbefugtes bewirken.

Die *Beeinträchtigung der Funktionalität* muß auf die *berechtigten Nutzer bezogen* werden. Andernfalls wäre, wenn ein IT-System von Unbefugten intensiv genutzt wird, seine Funktionalität zwar nicht beeinträchtigt, denn es funktioniert ja. Aber je nach Auslastung des IT-Systems durch die Unbefugten wäre die Funktionalität des IT-Systems den berechtigten Nutzern entzogen [Stel_90].

In [Stel_90] wird moniert, daß in [ZSI_89] das Schutzziel *(rechtliche) Verbindlichkeit* nicht aufgeführt ist. Ein ähnliches Schutzziel *Zurechenbarkeit* (accountability) wird später in den Kanadischen Sicherheitsbewertungskriterien [CTCPEC_92] als zusätzliches viertes eingeführt. Man kann damit manche Schutzziele prägnanter beschreiben, vgl. §1.3. Da man aber solche Bedrohungen bzw. Schutzziele auch unter Integrität fassen kann und mit möglichst wenig Bedrohungstypen auskommen sollte, bleibe ich bei der üblichen Dreiteilung.

Positiv und möglichst kurz formuliert lauten die **drei Schutzziele** also:

Vertraulichkeit = Informationen werden nur Berechtigten bekannt.

Integrität = Informationen sind richtig, vollständig und aktuell oder aber dies ist erkennbar nicht der Fall.

Verfügbarkeit = Informationen sind dort und dann zugänglich, wo und wann sie von Berechtigten gebraucht werden.

Hierbei ist „Information“ jeweils in einem weiten und umfassenden Sinn gemeint, so daß Daten und Programme, aber auch Hardwarestrukturen[8] subsumiert werden.

Damit das Reden von „Berechtigten“ Sinn macht, muß zumindest außerhalb des betrachteten IT-Systems geklärt sein, wer in welcher Situation wozu berechtigt ist.

Schließlich sei angemerkt, daß sich „richtig, vollständig und aktuell“ jeweils nur auf das Innere eines betrachteten Systems beziehen kann, da das Zutreffen von dessen Bild von seiner Umwelt nicht innerhalb des betrachteten Systems klärbar ist.

1.2.2 Vor wem ist zu schützen?

Einerseits wirken auf jedes und in jedem technischen System die Naturgesetze und, wenn man es nicht davor schützt, auch die Naturgewalten. Ersteres bedeutet, daß Bauteile altern und schließlich nicht mehr wie vorgesehen funktionieren. Zweiteres bedeutet, daß Vorkehrungen gegen Überspannung (Blitzschlag, EMP), Spannungsausfall, Überschwemmung (Sturmflut, Wasserrohrbruch), Temperaturänderungen etc. zu treffen sind. Um beides kümmert sich das Fachgebiet Fehlertoleranz [Echt_90].

Andererseits können Menschen, sei es aus Unfähigkeit oder Nachlässigkeit, sei es bewußt unbefugt handelnd, auf das System unerwünscht einwirken. Gemäß ihrer Rolle in Bezug auf das IT-System ist es sinnvoll,

Außenstehende,

Benutzer des Systems,

---

8 Da Abläufe – sprich Funktionalität – statt durch Programme auch durch Hardwarestrukturen beschrieben werden können, ist dies nötig.

Betreiber des Systems,
Wartungsdienst,
Produzenten des Systems,
Entwerfer des Systems,
Produzenten der Entwurfs- und Produktionshilfsmittel,
Entwerfer der Entwurfs- und Produktionshilfsmittel,
Produzenten der Entwurfs- und Produktionshilfsmittel der Entwurfs- und Produktionshilfsmittel,
Entwerfer der ...

zu unterscheiden, vgl. Bild 1-3. Es sei darauf hingewiesen, daß es natürlich auch bei allen oben aufgeführten Produzenten und Entwerfern wiederum Benutzer, Betreiber und Wartungsdienst ihres Systems gibt. Auch hiervor ist ggf. zu schützen.

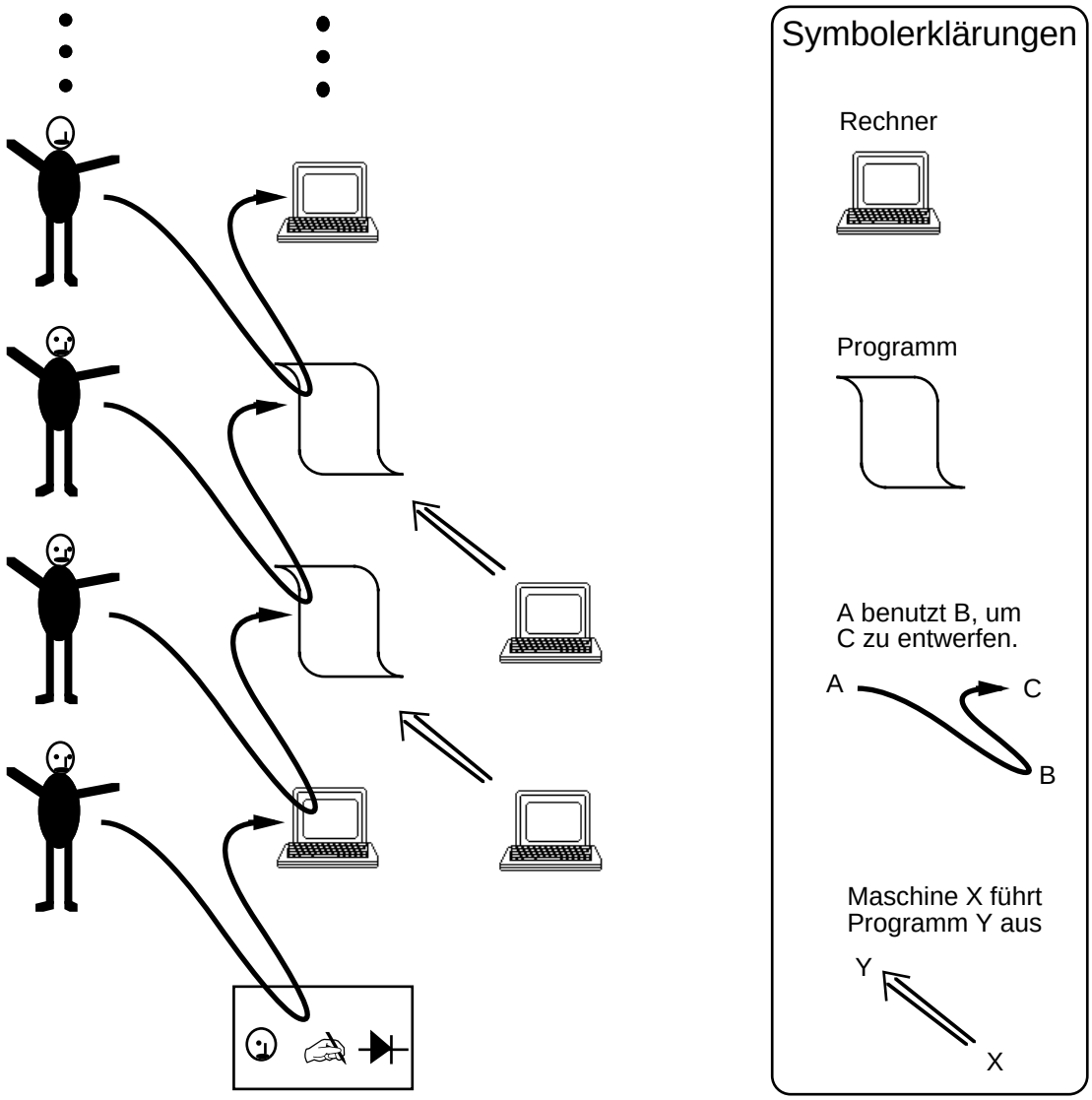


Bild 1-3: Menschen benutzen Rechner und Programme, um weitere Programme und Rechner zu entwerfen

Während es allgemein üblich ist, Außenstehenden und Benutzern des betrachteten IT-Systems zu mißtrauen und deshalb zu versuchen, sich gegen ihre unbewußten Fehler und bewußten Angriffe zu schützen, ist dies bezüglich der anderen Rollenträger (insbesondere auch dem Einfluß weiterer IT-Systeme) weitgehend nicht der Fall. Dies ist, wie gleich begründet wird, eine schwerwiegende

Sicherheitslücke. Denn nicht nur unbewußte Fehler, auch bewußte Angriffe können sich entlang der Entwurfs- und Produktionslinie fortpflanzen. In Bild 1-4 muß beispielsweise mitnichten der Mensch oben schuld sein, wenn der rechnerunterstützt entworfene Rechner fehlerhaft ist. Wie in Bild 1-4 eingezeichnet, könnten es auch der Mensch ganz unten oder der Rechner unten rechts sein. Natürlich kämen auch die anderen beiden Menschen und der Rechner in der zweiten Ebene in Betracht.

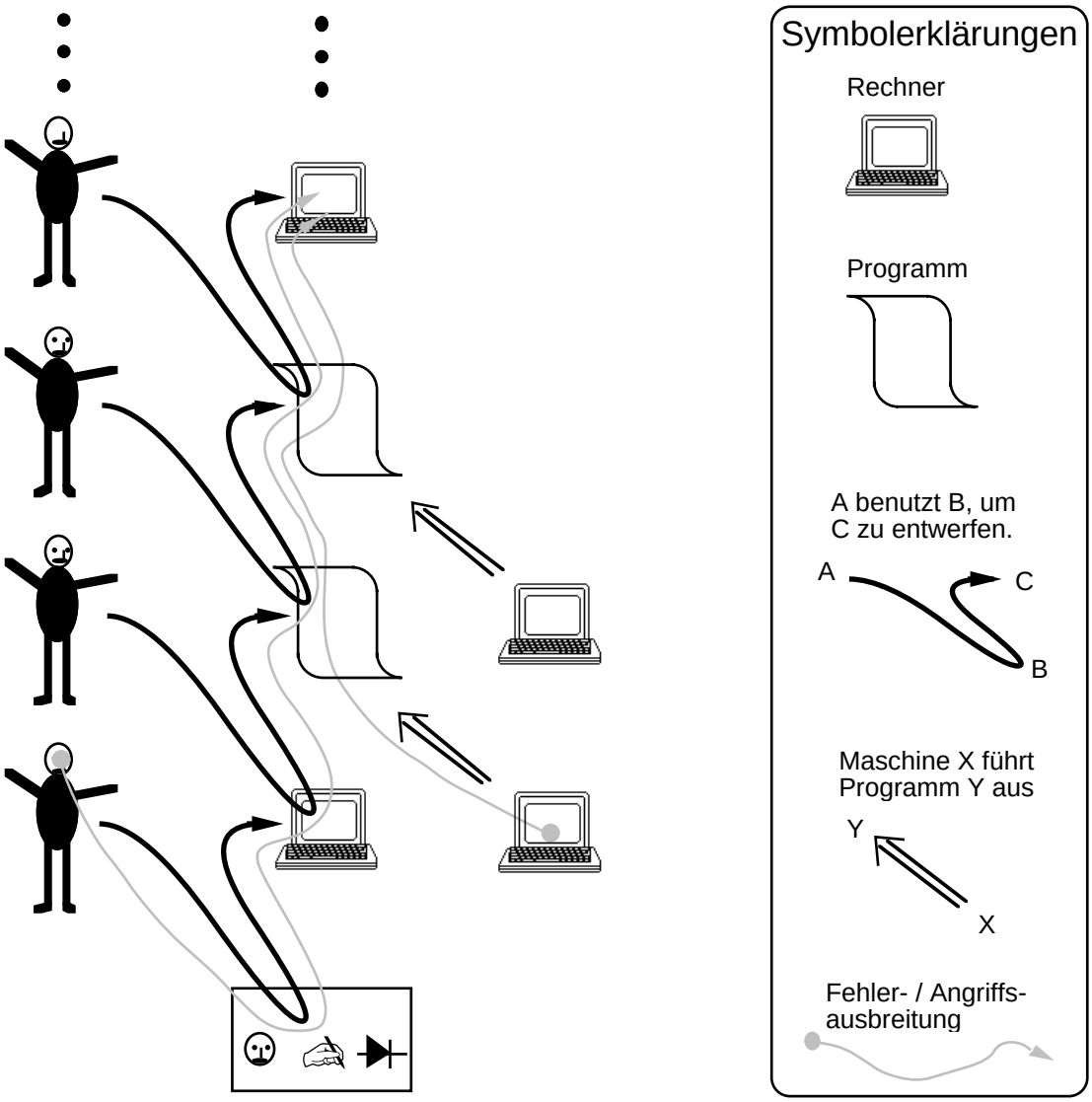


Bild 1-4: Transitive Ausbreitung von Fehlern und Angriffen

Die besonders hohe Komplexität heutiger IT-Systeme verhindert eine genügend rigorose Kontrolle der Systeme selbst. Deshalb müssen auch ihre Produzenten und Entwerfer, seien es Menschen oder wiederum (irgendwann auch von Menschen geschaffene) IT-Systeme, als mögliche **Angreifer**, d.h. unbefugt Handelnde, betrachtet werden. Beispielsweise könnten sie in den von ihnen entworfenen oder produzierten Systemteilen **Trojanische Pferde** [Home_??] verstecken:

Ein Systemteil ist ein Trojanisches Pferd, wenn er unter Ausnutzung der ihm anvertrauten Daten und Rechte *mehr* als das von ihm Erwartete[9] oder von ihm Erwartetes *falsch* oder *nicht* tut[10], vgl. Bild 1-5. Etwa könnte ein Editor die ihm eingegebenen Daten nicht nur in einer Datei seines Benutzers abspeichern, sondern noch an den Programmierer des Editors weitergeben. Hierzu benötigt das Trojanische Pferd einen *verborgenen Kanal*[11] (covert channel) [Cove_90, Denn_82, Lamp_73, Pfit_89 Seite 7, 18]. Kann in unserem Beispiel der Editor als verborgenen Kanal keine zweite Datei benutzen (in vielen Rechnern könnte er das), so kann er etwa seinen Verbrauch an Betriebsmitteln (Speicherbelegung, CPU-Zeit) modulieren, was von anderen Prozessen wahrgenommen werden kann.

Leider sind keine brauchbaren Verfahren bekannt, alle Trojanischen Pferde zu finden oder deren Existenz im untersuchten Systemteil auszuschließen. Im wesentlichen ist man hier auf das „Hinschauen mit bloßem Auge“ angewiesen (was bei der heutigen Komplexität von Rechnern fast genausogut mit geschlossenen Augen geschehen kann).

Die Bandbreite selbst mancher bekannter verborgener Kanäle läßt sich ohne drastischen Mehraufwand nicht beliebig verkleinern. Die höchste Qualitätsanforderung der IT-Sicherheitskriterien läßt ohne Begrenzung der Anzahl der bekannten verborgenen Kanäle eine Bandbreite pro verborgenen Kanal von 1 bit/s zu [ZSI_89 Seite 96]. Nehmen wir an, ein Angreifer kann sich von einem Trojanischen Pferd „nur“ 1 bit/s übertragen lassen: Dann erhält er innerhalb eines Jahres ca. 4 MByte Daten.

Glücklicherweise kann durch den Einsatz von Diversität, d.h. der Entwurf wird durch mehrere unabhängige Entwerfer mit unabhängigen Hilfsmitteln mehrfach durchgeführt, unter plausiblen Annahmen die Bandbreite auf exakt 0 bit/s verkleinert werden [Cles_88]. Allerdings ist der Aufwand hierfür beträchtlich.

---

9 In der Literatur wird ein Trojanisches Pferd oft als ein Programm definiert, das „neben seiner vorgeblichen eigentlichen Funktion eine weitere, *nicht dokumentierte Zusatzfunktion* ausführt“. Neben der unnötigen Beschränkung auf *Programme* sowie *Zusatz*funktion ist diese Definition unglücklich: Zu jeder Programmfunktion gibt es mindestens eine Dokumentation, in der *jede* Einzelheit enthalten ist: den Programmcode selbst [Dier3_91].

10 Diese Definition mag auf den ersten Blick zu allgemein erscheinen, da in ihr *Absicht* oder *böser Wille* eines Urhebers des Trojanischen Pferdes nicht gefordert wird, während die Griechen laut Homer bei der Konstruktion und Nutzung des hölzernen Trojanischen Pferdes vor den Toren Trojas durchaus bewußt planvoll gehandelt haben. Für das hier eingeführte, erweiterte Verständnis Trojanischer Pferde spricht Zweierlei:

1. Das Kriterium, ob hinter einer vorhandenen Funktionalität eines Systemteils Absicht oder gar böser Wille steckt, wäre nicht zu entscheiden – eine auf diese Eigenschaften abstellende Definition also informatisch nicht handhabbar. (Hiervon unberührt bleibt, daß für die moralische und ggf. auch strafrechtliche Verwerflichkeit einer Handlung Absicht und böser Wille durchaus relevant und daher auch Gegenstand der Urteilsbildung sind.)
2. Für die *Auswirkungen* einer unerwarteten Funktionalität eines Systemteils ist es unerheblich, ob sie absichtlich oder gar in böser Absicht geschaffen wurde. Allenfalls, ob sie absichtlich und in böser Absicht nutzbar sowie potentiellen Tätern mit Motiv bekannt ist, ist für die Auswirkungen relevant.

11 In der Literatur sowie hier werden die Namen *verborgener Kanal* und *verdeckter Kanal* bedeutungsgleich verwendet.

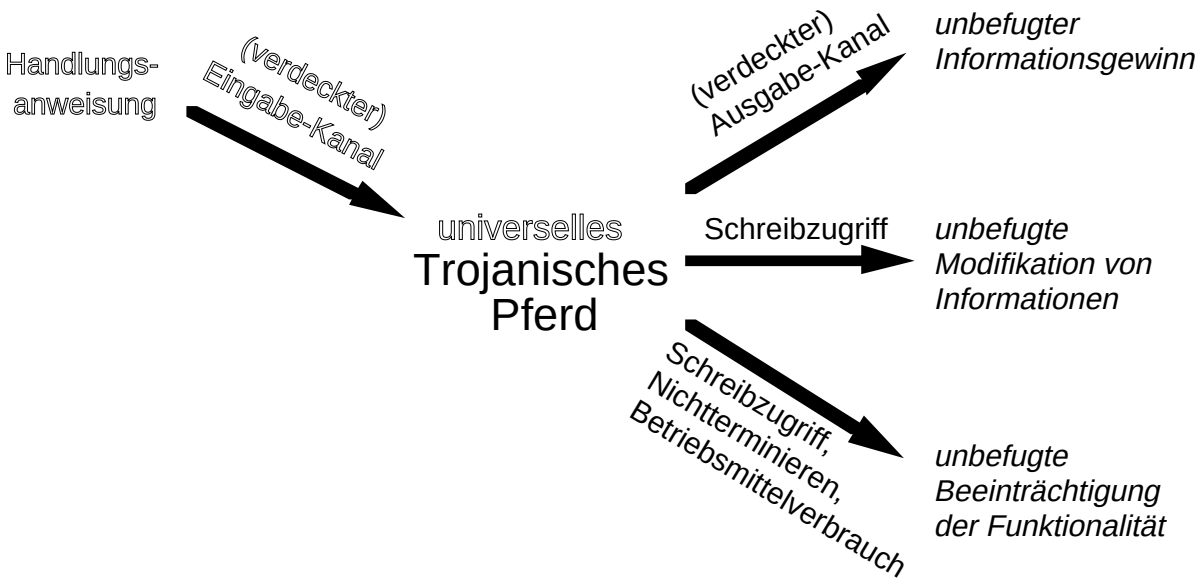


Bild 1-5: Universelles Trojanisches Pferd

Diese sehr weite Definition Trojanischer Pferde läßt viele Spielarten zu, die teilweise mit eigenen martialischen Namen versehen sind, z.B. logische Bomben etc. Für eine grundsätzliche Betrachtung sind jedoch nur die beiden folgenden Eigenschaften universell und/oder transitiv wichtig.

Üblicherweise stellt man sich vor, daß der von einem Trojanischen Pferd anzurichtende Schaden (genauer: seine Schadensfunktion) beim *Entwurf* des Trojanischen Pferdes festgelegt wird. Damit wäre das Trojanische Pferd ein relativ unflexibler und ungerichteter Angreifer, bei Entdeckung wäre die Schädigungsabsicht seines Urhebers (sofern dieser identifizierbar ist) nachweisbar. Es ist aber auch möglich, zum Entwurfszeitpunkt nur einen Handlungsrahmen zu schaffen, der durch Eingaben während des *Betriebs* mit Leben gefüllt wird. Die Eingaben an das Trojanische Pferd werden in zulässigen – und harmlos scheinenden – Eingaben an den das Trojanische Pferd enthaltenden Systemteil verdeckt codiert (Bild 1-5). Die Codierung wird zweckmäßigerweise so redundant gewählt, daß die Wahrscheinlichkeit, daß ein Uneingeweihter durch seine Eingaben an den Systemteil auch Eingaben für das Trojanische Pferd übermittelt, verschwindend gering ist. Im allgemeinsten Fall kann man während des Betriebs das Trojanische Pferd beliebig programmieren. Dorothy Denning hat diese Trojanischen Pferde **universell** (universal) genannt [Denn_85]. Die gezielte Verwendung universeller Trojanischer Pferde ist natürlich nur solchen Personen möglich, die während des Betriebes Zugriff auf das System haben. Dies ist insbesondere bei *offenen* Systemen aber fast jedem möglich. Mit Hilfe universeller Trojanischer Pferde sind sehr flexible und gerichtete Angriffe möglich. Der Benutzer des Trojanischen Pferdes braucht z.B. nicht zu warten, bis ein ihn seit kurzem interessierender Datensatz nach vielen Jahren endlich übertragen wird, sondern er kann das universelle Trojanische Pferd diesen Datensatz sofort übertragen lassen.

Üblicherweise stellt man sich auch vor, daß ein Trojanisches Pferd vom Täter direkt dort untergebracht werden muß, wo es subversiv wirken soll. Auch diese Vorstellung ist falsch: Werden beim Entwurf Hilfsmittel verwendet (z.B. Compiler), so kann sowohl der Entwerfer als auch das Entwurfshilfsmittel ein Trojanisches Pferd im Entwurf unterbringen. Da mit Entwurfshilfsmitteln weitere Entwurfshilfsmittel entworfen werden, gilt dies rekursiv, so daß sich ein Trojanisches Pferd in der transitiven Hülle aller Entwürfe ausbreiten kann, an denen sein Gastgebersystem direkt oder indirekt beteiligt war [Thom_84, Pfit_89]. Ich spreche deshalb von **transitiven** Trojanischen Pferden.

Der Personenkreis, dessen Mitglieder als potentielle Angreifer betrachtet werden müssen, wird durch die *Transitivität* Trojanischer Pferde immer größer, die der obigen *rekursiven* Charakterisierung der Menge der Entwerfer und Produzenten entspricht. Dieses Größerwerden ist aus vielen Gründen mißlich, die sich in zwei Gruppen einteilen lassen:

- Erstens sinkt die Wahrscheinlichkeit, daß alle keine Angreifer sind, einfach durch die Zunahme der Zahl der Menschen. Zusätzlich sinkt mit der Größe und Heterogenität der Gruppe bei jedem Einzelnen das Gefühl der Zusammengehörigkeit und des Füreinander-verantwortlich-Seins. Außerdem nimmt die implizite soziale Kontrolle innerhalb der Gruppe ab.
- Zweitens wird die Durchführung von Kontrollen der Menschen immer aufwendiger, unakzeptabler (Schlagwort: Überwachungsstaat) und durch die Grenzen der Kooperation von Staaten begrenzt (welcher Staat erlaubt einem anderen die Überprüfung eines Teils der eigenen Bevölkerung; andererseits: welcher Staat verbietet den Import von Software?).

1.2.3 Wie und wodurch kann Sicherheit erreicht werden?

Zunächst muß vereinbart werden, was „unbefugt“ bedeutet. Dies kann beispielsweise durch internationale Konventionen, Verfassung, Gesetze, Betriebsvereinbarungen oder berufsständische Ethik geschehen. „Unbefugtes“ Handeln wird mit Strafe bedroht.

Organisationsstrukturen sollten so gewählt werden, daß Unbefugtes möglichst erschwert, am besten verhindert wird.

Reichen Ge- und Verbote sowie organisatorische Maßnahmen?

Ein Verbot ist nur dann wirkungsvoll, wenn seine Einhaltung mit angemessenem Aufwand *überprüft* und durch Strafverfolgung gesichert und der ursprüngliche Zustand durch Schadensersatz *wiederhergestellt* werden kann. Beides ist bei IT-Systemen leider nicht gegeben.

„Datendiebstahl“ allgemein, speziell das direkte Abhören von Leitungen oder Kopieren von Daten aus Rechnern, ist kaum feststellbar, da sich an den Originaldaten nichts ändert. Ebenso ist, wie oben erwähnt, das Installieren Trojanischer Pferde kaum festzustellen und erst recht nicht die unerlaubte Weiterverarbeitung von Daten, die man legal (oder auch illegal) erhalten hat.

Die Wiederherstellung des ursprünglichen Zustands müßte vor allem darin bestehen, alle entstandenen Daten zu löschen. Man ist aber nie sicher, ob nicht noch weitere Kopien existieren. Außerdem können sich Daten im Gedächtnis von Menschen festsetzen, wo das Löschen besser nicht angestrebt werden sollte.

Es sind also zusätzliche wirkungsvollere Maßnahmen nötig. Vorbeugende *technische* Schutzmaßnahmen sind die einzigen bekannten.

1.2.4 Vorausschau auf Schutzmechanismen

Bild 1-6 gibt eine Vorausschau auf Maßnahmen, die in den folgenden Kapiteln erläutert werden. Die Spalte „Unerwünschtes verhindern“ (lies: Unerwünschte Aktionen des Systems sollen verhindert werden) entspricht im wesentlichen dem Schutzziel Vertraulichkeit, die Spalte „Erwünschtes leisten“ (lies: Erwünschte Aktionen des Systems sollen stattfinden) im wesentlichen den Schutzzielen Integrität und Verfügbarkeit.[12]

---

[12] Unbeobachtbarkeit, Anonymität und Unverkettbarkeit können auf folgende subtile Weise zur Verfügbarkeit beitragen: Ist ein Angreifer nicht bereit, alle (oder zumindest sehr viele) Teilnehmer zu beeinträchtigen – sei es, weil er nicht garzusehr auffallen will oder beispielsweise politisch motivierte Terroristen nicht auch die Schwachen einer Gesellschaft angreifen wollen –, so verhindern Unbeobachtbarkeit, Anonymität und Unverkettbarkeit ein *selektives* Beeinträchtigen der Verfügbarkeit für spezielle Teilnehmergruppen oder gar spezielle individuelle Teilnehmer. In solch einer Situation ist von so motivierten Tätern mit keiner Beeinträchtigung der Verfügbarkeit zu rechnen.

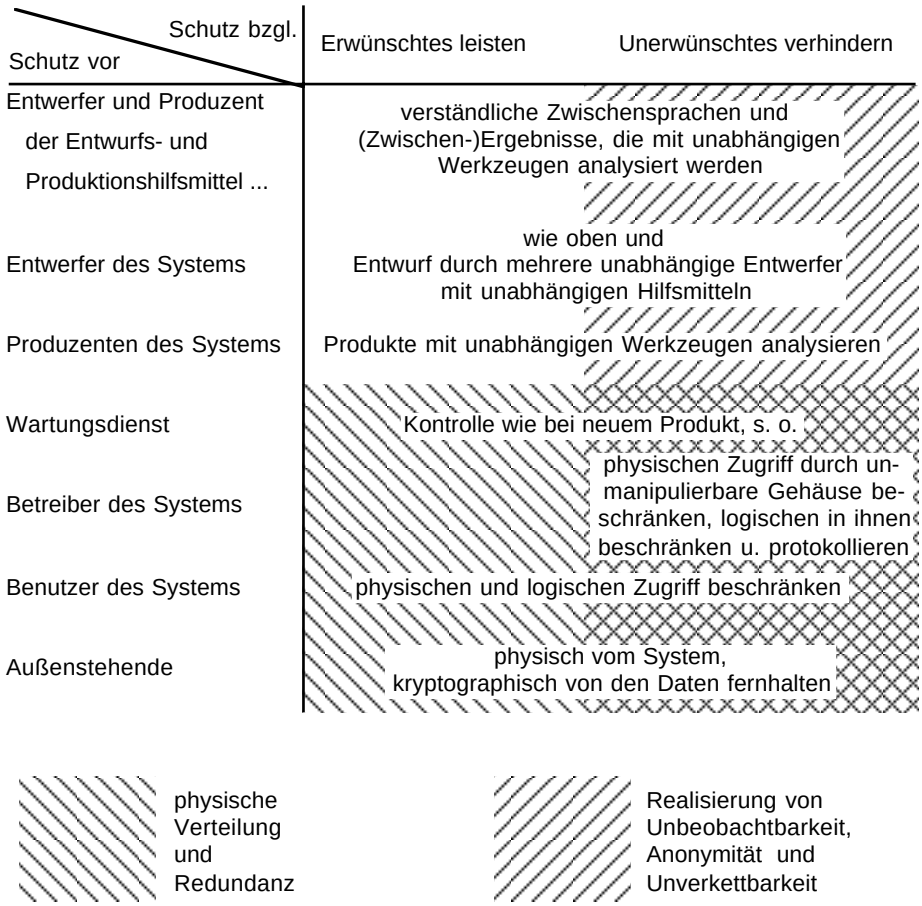


Bild 1-6: Welche Schutzmaßnahmen schützen gegen welche Angreifer

1.2.5 Angreifermodell

Schutz vor einem allmächtigen Angreifer ist natürlich nicht möglich: Ein allmächtiger Angreifer könnte

1. alle Daten, die ihn interessieren, an der Stelle ihrer Entstehung unbefugt erfassen (oder, falls nötig, auch Daten an mehreren Stellen erfassen und sie zu den ihn interessierenden kombinieren, vgl. §5), und so Vertraulichkeit vor ihm verhindern.
2. Daten unbemerkt unbefugt ändern, denn wenn ihr Benutzer ganz genau wüßte, welchen Wert sie haben müssen, könnte er gleich auf sie verzichten.
3. durch physische Zerstörung des gesamten IT-Systems seine Funktionalität unbefugt beeinträchtigen.

Deshalb sind alle folgenden Maßnahmen nur Annäherungen an den perfekten Schutz der Teilnehmer vor jedem möglichen Angreifer. Die Annäherung wird im allgemeinen durch Angabe der **maximal berücksichtigten Stärke eines Angreifers** in der Form eines sogenannten **Angreifermodells** bestimmt.

Auch um einzelne Schutzmechanismen zu charakterisieren, sollte der Angreifer maximaler Stärke, gegen den sie schützen, genau beschrieben werden.

Das Angreifermodell umfaßt neben den möglichen Rollen des Angreifers (Außenstehender, Benutzer, Betreiber, Wartungsdienst, Produzent, Entwerfer ...), die natürlich auch kombiniert auftreten können, auch folgende Attribute:

Um zu beschreiben, **wie weit verbreitet** der Angreifer ist, sollten die möglichen Rollen des Angreifers jeweils quantifiziert werden: Welche physischen oder kryptographischen Abschottungen kann er überwinden? Welche Subsysteme kann er benutzen und dadurch in welcher Hinsicht kontrollieren? Welche Subsysteme kann er als Betreiber, welche als Wartungsdienst in welcher Hinsicht kontrollieren? Wo konnte er als Entwerfer oder Produzent Trojanische Pferde unterbringen? usw.

Sind alle Subsysteme bezüglich einer Rolle des Angreifers gleich, so genügt es, ihr jeweils eine nichtnegative ganze Zahl zuzuordnen. Sie gibt dann an, wie viele dieser Subsysteme vom Angreifer kontrolliert werden können.

Unterscheidet man in einem Rechnernetz als kleinste Subsysteme Leitungen und Stationen (z.B. Vermittlungszentrale, Netzabschluß, Teilnehmerendgerät, etc.), so wäre anzugeben, wie viele und welche Leitungen und/oder Stationen der Angreifer maximal beobachten und/oder gar aktiv verändernd kontrollieren kann.

Verhält sich der Angreifer nach außen (genauer: innerhalb des betrachteten IT-Gesamtsystems, aber außerhalb seiner Systemteile[13], vgl. Bild 1-7) nur so, wie es ihm *erlaubt* ist, d.h. führt er seinen Angriff nur **beobachtend** durch? Dies kann er beispielsweise tun, indem er Leitungen oder Funkstrecken abhört, lokal Daten anders auswertet oder länger speichert als er es darf. Oder riskiert der Angreifer mehr, indem er nach außen ihm *Verbotenes* tut, d.h. seinen Angriff auch **verändernd** durchführt?[14]

Beobachtende Angriffe sind im IT-System prinzipiell nicht erkennbar – ihr Erfolg kann aber, wie wir sehen werden, bezüglich des Erfassens und damit auch Verarbeitens und Speicherns von Daten, die der Angreifer regulär nicht erhält, prinzipiell vollständig verhindert werden. Verändernde Angriffe können zwar prinzipiell erkannt werden, dafür kann ihr Erfolg aber nicht vollständig verhindert werden. Außerhalb des IT-Systems können beobachtende Angriffe möglicherweise erkannt und sollten die Auswirkungen von verändernden Angriffen möglichst begrenzt werden.

Zur genaueren Beschreibung kann es sinnvoll sein, das Attribut beobachtend/verändernd nicht dem Angreifer als Ganzem, sondern einzelnen seiner Rollen im IT-System zuzuordnen.

---

13 Hierbei wird unter den **Systemteilen** des Angreifers der Bereich verstanden, der von ihm exklusiv kontrolliert wird, wo andere also nicht ohne weiteres hineinschauen können. Der **Einflußbereich** und damit der **Verbreitungsbereich** des Angreifers ist üblicherweise größer. Er umfaßt üblicherweise weitere Teile des betrachteten IT-Gesamtsystems.

14 So ganz glücklich bin ich mit den Begriffen **beobachtender** vs. **verändernder** Angreifer nicht, kenne aber bisher keine besseren. Verwendet man stattdessen die Begriffe **protokolltreuer** vs. **protokollverletzender** Angreifer, dann muß man unterstellen, daß alles im Protokoll Vorgesehene auch erlaubt ist. Dies ist bei den heute üblichen Protokollen jedoch nicht der Fall.

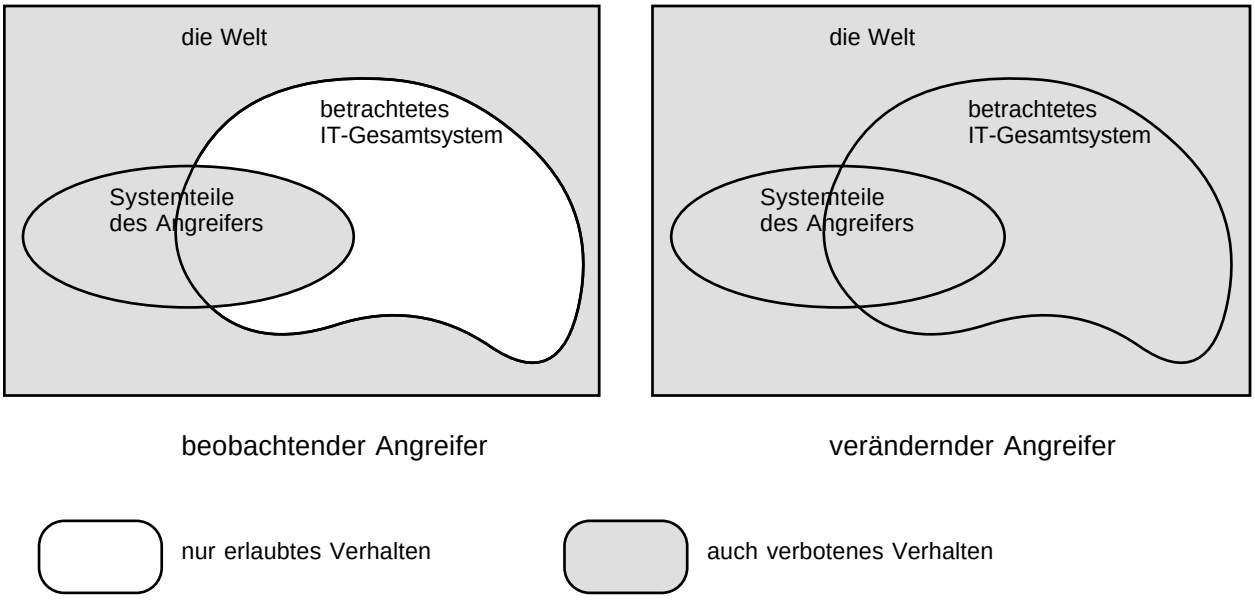


Bild 1-7: Beobachtender vs. verändernder Angreifer

In §1.2.2 wurde schon zwischen dummen oder zufälligen Fehlern und intelligenten zielgerichteten Angriffen unterschieden. Bei letzteren lohnt es sich, noch genauer zu unterscheiden: Wieviel Rechenkapazität wird dem Angreifer zugebilligt?

Auf der sicheren Seite liegt man, wenn man ihm unbeschränkte Rechenkapazität zubilligt. Man spricht dann von einem komplexitätstheoretisch unbeschränkten (oder kürzer: **informationstheoretischen**) Angreifer. Sicherheitsbeweise sind dann in der informationstheoretischen Modellwelt von Claude Shannon zu führen, was in §3 genauer erläutert wird.

Man riskiert etwas, wenn man annimmt, daß der Angreifer nur beschränkte Rechenkapazität (genauer: Berechnungsfähigkeiten) besitzt, z.B. große Zahlen in vernünftiger Zeit nicht in ihre Primfaktoren zerlegen kann, vgl. ebenfalls §3. Man spricht von einem komplexitätstheoretisch beschränkten (oder kürzer: **komplexitätstheoretischen**) Angreifer. Hier kann man sich bezüglich der Hardware-Betriebsmittel und Algorithmenkenntnis des Angreifers irren. Das Hauptrisiko liegt, wie wir insbesondere in §3 sehen werden, in einer falschen Einschätzung der Algorithmenkenntnis.

Als Konkretisierung von komplexitätstheoretischen Angreifern, aber auch allgemein, etwa bzgl. verwendbarer Meßtechnik, kann man Angreifer danach unterscheiden, wieviel **Zeit** und **Geld** sie haben und aufzuwenden bereit sind. Und ganz untechnisch können sie Geld auch außerhalb des IT-Systems einsetzen: Bestechung von Menschen ist in vielen Situationen das wirksamste Mittel, um z.B. Entwerfer zum Einbau Trojanischer Pferde, Betreiber zur Manipulation des IT-Systems oder Zugangsberechtigte zur Zusammenarbeit zu bewegen.

Das Aufstellen eines realistischen Angreifermodells ist eine ähnliche Kunst wie das Abschätzen in mathematischen Beweisen: Ist die Abschätzung zu grob, gelingt der Beweis nicht mehr. Ist die Abschätzung zu fein, wird der Beweis unnötig kompliziert, fehlerträchtig und wenig überzeugend. Bezogen auf Angreifermodelle bedeutet dies, daß oftmals unterstellt wird, daß alle möglichen Angriffspunkte und Angriffsarten zu *einem* Angreifer kombiniert werden, obwohl man für die Realität hoffen kann, daß die diese Angriffspunkte kontrollierenden Instanzen nicht alle miteinander konspirieren und perfekt abgestimmt handeln werden.

> Ein **realistisches Angreifermodell** sollte nicht nur alle momentan erwarteten Angreifer, sondern auch *alle während der Lebensdauer des Systems zu erwartenden Angreifer abdecken*. Es sollte *einfach* sein, damit die nicht abgedeckten Restrisiken durch unerwartet starke Angreifer den vom System Betroffenen verständlich sind.

> Trotz dieser Abschätzung des Angreifers sollte ein gegen ihn sicheres System mit vertretbarem Aufwand gebaut und betrieben werden können sowie für dieses System ein überzeugender Nachweis aller aufgestellten Schutzziele gelingen.

Die Lebensdauer des Systems hat nicht nur Einfluß auf die zu unterstellende Meß- und Gerätetechnik des künftigen Angreifers, sondern auch auf seine Motivation, seinen Zeit- und Geldeinsatz sowie seine Risikobereitschaft. All dies kann durch politische Entwicklungen oder Steigerungen des Wertes (sei es ökonomisch oder politisch) der mit dem System verarbeiteten Daten oder der Abhängigkeit von Individuen, Organisationen oder gar Staaten vom Funktionieren des Systems drastisch zunehmen. Im Zweifelsfall sollte also nicht eine genauere **Risikoanalyse**, die wegen der Unvorhersehbarkeit mancher dieser Entwicklungen nicht möglich ist, versucht werden, sondern der Aufwand sollte lieber in den Bau eines gegen stärkere Angreifer sicheren Systems investiert werden.

1.3 Was bedeutet Sicherheit in Rechnernetzen?

Ohne Anspruch auf Vollständigkeit und Genauigkeit seien hier einige mögliche Schutzziele für Rechnernetze genannt. Sie werden in §5 vertieft behandelt, nachdem in §3 die dazu notwendigen begrifflichen und kryptologischen Grundlagen dargestellt wurden.

Schutzziel Vertraulichkeit

- Nachrichteninhalte sollen vor allen Instanzen außer dem Kommunikationpartner vertraulich bleiben.
- Sender und/oder Empfänger von Nachrichten sollen voreinander anonym bleiben und durch Unbeteiligte (inkl. Netzbetreiber) nicht beobachtet werden können.

Schutzziel Integrität

- Fälschungen von Nachrichteninhalten (inkl. des Absenders) sollen erkannt werden.
- Der Empfänger soll gegenüber Dritten nachweisen können, daß Instanz *x* die Nachricht *y* gesendet hat (und ggf. auch den Zeitpunkt, an dem dies geschah).
- Der Absender soll das Absenden einer Nachricht mit korrektem Inhalt beweisen können, möglichst sogar den Empfang der Nachricht (und ggf. auch den Zeitpunkt, an dem dies geschah).
- Niemand kann dem Netzbetreiber Entgelte für erbrachte Dienstleistungen vorenthalten. Umgekehrt kann der Netzbetreiber nur für korrekt erbrachte Dienstleistungen Entgelte fordern.

Zugegeben, die letzten drei Schutzziele unter Integrität zu fassen, ist etwas gezwungen! Alternativ müßten weitere Schutzziele eingeführt werden, was leicht ins Uferlose ausartet, vgl. das Ende von §1.2.1. Das letzte Schutzziel könnte allerdings auch unter Verfügbarkeit eingeordnet werden.

Schutzziel Verfügbarkeit

- Das Rechnernetz ermöglicht Kommunikation zwischen allen Partnern, die dies wünschen (und denen dies nicht verboten ist).

Diese Schutzziele sind vielschichtig und gelten nicht für alle Beteiligten jeweils gleichstark. Je nach Kommunikationsgegenstand und -umstand werden einzelne Schutzziele sogar für denselben Nutzer eine unterschiedliche Bedeutung haben.

1.4 Warum mehrseitige Sicherheit?

Verallgemeinern wir etwas:

Wir möchten, daß möglichst jede(r) gegen jede(n) geschützt ist. Denn jede(r) kann durch Geräte mit Trojanischen Pferden, Bedienungsfehler, Eigeninteressen oder gar Bösartigkeiten die Schutzinteressen anderer bedrohen und muß deshalb aus deren Sicht als potentieller Angreifer gesehen werden.

Ist jede(r) gegen jede(n) möglichst weitgehend geschützt, nennen wir dies ***mehrseitige* Sicherheit** (multilateral security) [RaPM_96].

> *Mehrseitige Sicherheit* bedeutet die Einbeziehung der Schutzinteressen *aller* Beteiligten sowie das Austragen daraus resultierender Schutzkonflikte, etwa beim Entstehen einer Kommunikationsverbindung.

Die Realisierung von mehrseitiger Sicherheit führt nicht zwangsläufig dazu, daß die Interessen aller Beteiligten erfüllt werden. Da Schutzziele explizit formuliert werden, offenbart sie möglicherweise sogar gegensätzliche, unvereinbare Interessen, die den Beteiligten bisher nicht bewußt waren. Die Realisierung von mehrseitiger Sicherheit gewährleistet jedoch, daß die Partner einer mehrseitig sicheren Kommunikationsbeziehung in einem geklärten Kräfteverhältnis bezüglich Sicherheit miteinander interagieren.

Technisch gesehen kommt es darauf an, dem Nutzer Möglichkeiten an die Hand zu geben, Schutzziele zu formulieren, ggf. auszuhandeln und durchzusetzen. Mehrseitige Sicherheit erfordert also, Verfahren zu entwickeln und nutzbar zu machen, die dem Benutzer informationstechnischer Systeme zumindest aus technischer Sicht die Möglichkeit geben, sich zu schützen.

Selbstverständlich wird der praktische Einsatz solcher Verfahren von den ökonomischen, sozialen wie juristischen Gegebenheiten abhängen. Sicherheit ist ein Querschnittsthema und erfordert daher einen engen Dialog mit anderen wissenschaftlichen Disziplinen.

Leseempfehlungen

Rechnernetze: Tane_96

Sicherheit: MüPf_97, Denn_82, ZSI_89, ZSI_90, ITSEC2_91 (bzw. die fehlerhafte Übersetzung: ITSEC2d_91), GIPr1_92, NRC_99

Sicherheit in Rechnernetzen: VoKe_83

Rechtliche Grundlagen: BFD1_95, BFD5_98, Tele_98

Überwachungstechnologie: http://www.europarl.eu.int/dg4/stoa/de/publi/166499/execsum.htm

2 Sicherheit in einzelnen Rechnern und ihre Grenzen

Nach der Behandlung physischer Sicherheit wird der auf ihr aufbauende „logische“ Schutz isolierter Rechner vor unautorisiertem Zugriff und Computer-Viren dargestellt.

2.1 Physische Sicherheitsannahmen

Alle technischen Schutzmaßnahmen bedürfen einer physischen „Verankerung“ in Form eines Systemteils, auf den der betrachtete Angreifer keinen physischen Zugriff hat – im allgemeinen Fall weder lesenden (Schutz der Vertraulichkeit) noch verändernden (Schutz von Integrität und Verfügbarkeit).

Je nach Schutzkonzept und auch bezüglich einzelner Schutzmaßnahmen kann die Größe dieses für verschiedene Angreifer sinnvollerweise verschiedenen Systemteils vom „Rechenzentrum X“ bis zur „Chipkarte Y“ reichen. Dazwischen liegen Größenordnungen bezüglich Abmessungen und Komplexität. In beiden Fällen agieren Menschen nicht nur als mögliche Angreifer, sondern (in hoffentlich geeigneten Organisationsstrukturen) auch als erwünschte „Verteidiger“.

Als besonders vorteilhaft wird sich erweisen, Schutz nicht nur für das IT-System als Ganzes, sondern insbesondere für seine einzelnen Benutzer zu betrachten. Dann kann Schutz für Benutzer B in einem Systemteil verankert werden, den B vor allen anderen physisch schützt, etwa eine Chipkarte, die B immer mit sich herumträgt, mit unter die Dusche nimmt und selbstverständlich nachts unter sein Kopfkissen legt.

Was nun kann man von physischen Sicherheitsmaßnahmen bestenfalls erwarten? Durch welche Schutzmaßnahmen ist dies zumindest näherungsweise zu erreichen? Sind Chipkarten wirklich physisch sicher? Was also sind sinnvolle Sicherheitsannahmen? Diese Fragen werden nun der Reihe nach behandelt.

2.1.1 Was kann man bestenfalls erwarten?

Die Verfügbarkeit eines räumlich konzentrierten Systemteils wird man gegen durchaus *vorstellbare* mächtige Angreifer, z.B. Terroristen oder allgemeiner Besitzer von genügend viel konventionellem Sprengstoff oder gar Atom- oder Wasserstoffbomben, nicht gewährleisten können. Hier wird es ab einem gewissen physischen Schutzaufwand für dicke Stahlbetonwände, Notstromversorgung, etc. schnell effizienter, auf verteilte Systeme zu setzen und zu hoffen, daß ein Angreifer nicht an vielen verschiedenen Orten gleichzeitig physisch massiv angreifen kann.

Durch Verteilung des IT-Systems eskalieren natürlich die Probleme im Bereich Vertraulichkeit und auch im Bereich Integrität. Bezüglich dieser Schutzziele kann man von physischen Maßnahmen jedoch mehr erwarten, nämlich Schutz gegen alle derzeit *vorstellbaren* Angreifer: Man kann durchaus versuchen, beispielsweise Gerätegehäuse so zu bauen, daß ein nur durch die derzeit bekannten physischen Naturgesetze beschränkter Angreifer weder unbefugt an die in den Gerätegehäusen gespeicherte Information gelangen noch diese unbemerkt unbefugt ändern kann. Gelingt dies hinreichend, dann steht einer physischen Verteilung nichts im Wege.

2.1.2 Gestaltung von Schutzmaßnahmen

Physische Schutzmaßnahmen bestehen aus einer Kombination der folgenden fünf **Grundfunktionen**:

Beobachtende Angriffe, etwa Analyse elektromagnetischer Strahlung, müssen durch *Schirmung* vereitelt werden. (Verfahren und Normen sind unter dem Kürzel TEMPEST bekannt [Horg_85]. Oft vergessen wird, daß auch die Eingaben, beispielsweise der Energieverbrauch, unabhängig von den zu schützenden Geheimnissen sein müssen [Koch_98].)

Verändernde Eingriffe müssen *erkannt* und bezüglich Zulässigkeit *bewertet* werden.

Verändernde Angriffe, d.h. als unzulässig erkannte verändernde Eingriffe, müssen *verzögert* und die Informationen müssen vor Ende der Verzögerung ggf. *gelöscht* werden.

Abhängig von Abmessungen und Komplexität ist es sinnvoll, den physisch zu sichernden Systemteil *schalenförmig* zu gestalten [Chau_84]. Hierbei können in einer Schale, wie z.B. in Bild 2-1 gezeigt, mehrere Grundfunktionen untergebracht werden. Ebenso können Grundfunktionen in einer Schale mehrfach erbracht werden, etwa die Erkennung von Eingriffen durch Sensoren für unterschiedliche physische Größen. Alternativ oder als Ergänzung können dieselben Grundfunktionen wiederholt in weiter innen liegenden Schalen auftreten. Dies ist in Bild 2-1 anhand von 3 Beispielen für den Fall maximal zweifach wiederholten Auftretens von Grundfunktionen gezeigt.

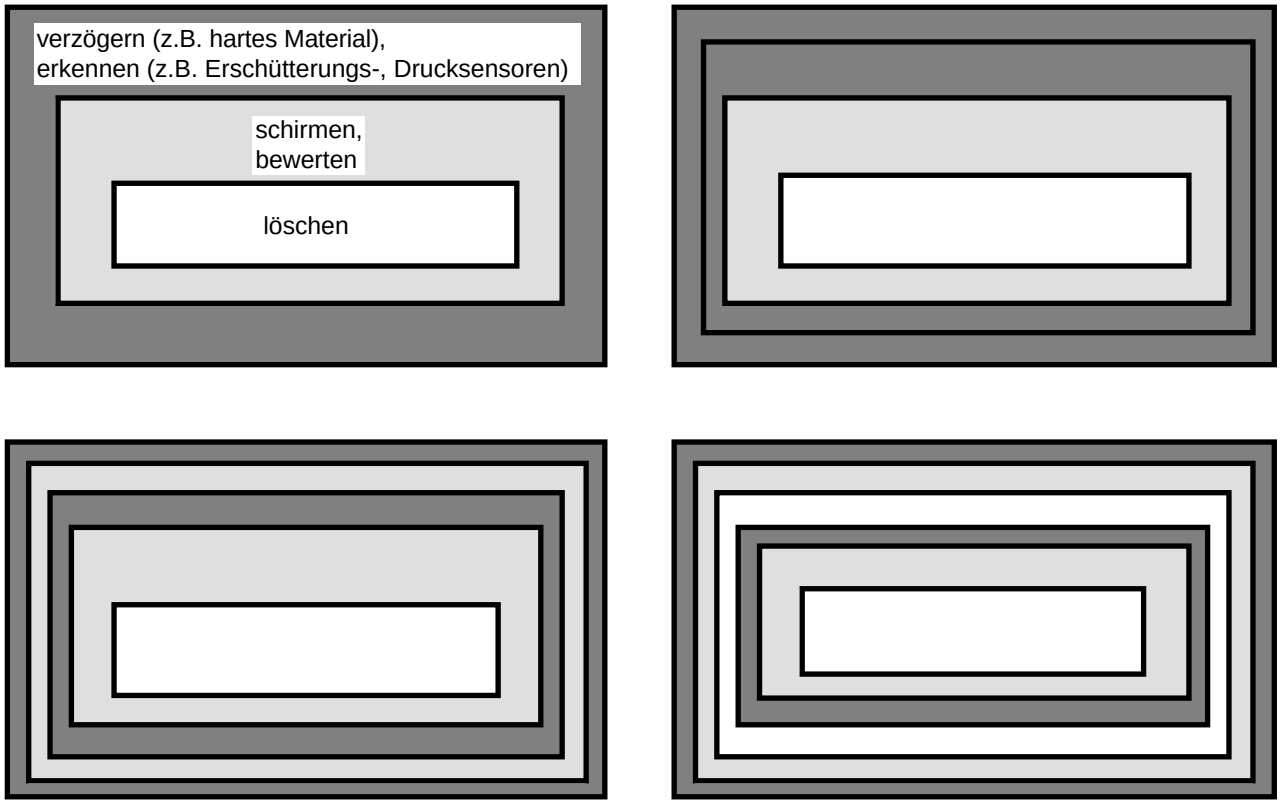


Bild 2-1: Schalenförmige Anordnung der fünf Grundfunktionen (oben links) und drei Möglichkeiten ihrer wiederholten Anwendung

Auch wenn man durch geeignete Kombination und schalenförmige Anordnung von Schutzmaßnahmen die physische Sicherheit stark steigern kann, haben physische Schutzmaßnahmen zwei prinzipielle Nachteile:

– Sie sind sehr schwer objektiv zu validieren. Dies liegt an zwei Gründen:

  Erstens beruht die Sicherheit mancher physischer Schutzmaßnahmen auf der Geheimhaltung ihres genauen Wirkmechanismus – zumindest wird er geheim gehalten. Dies kann

einerseits aus militärischen Gründen, andererseits aus Gründen des Know-how-Schutzes kommerzieller Hersteller sein.[15]

Zweitens ist eine genaue Kenntnis des Standes und der Entwicklung der Meßtechnik wichtig, da physische Maßnahmen (im Gegensatz zu kryptographischen, vgl. §3) nicht preiswert überdimensioniert werden können. Da etwa der militärische Bereich die Qualität seiner Meßtechnik geheimzuhalten versucht, ist solche genaue Kenntnis zumindest problematisch.

Aus diesen Gründen ist es um die subjektive Glaubwürdigkeit physischer Schutzmaßnahmen meist nicht gut bestellt.

– Sie sind vergleichsweise aufwendig und bei weitem nicht so schnell im Preis fallend wie „normale“ Digitaltechnik. Sie erschweren insbesondere Wartungsmaßnahmen.

2.1.3 Ein Negativbeispiel: Chipkarten

Chipkarten sind zwar physisch weitaus sicherer als normale Magnetstreifenkarten, doch sind sie meiner Meinung nach bezüglich wirklicher Sicherheit ein Negativbeispiel:

– Gute Schirmung ist schwierig, da die Chipkarte sehr dünn und biegbar sein soll. Zudem ist der Energieverbrauch unmittelbar meßbar, da die Chipkarte während ihres Betriebes von außen mit Energie versorgt wird – und bei den bisher üblichen Chipkarten ist der Energieverbrauch abhängig von den in der Chipkarte ausgeführten Befehlen, ja sogar von den in ihr verarbeiteten Daten und Schlüsseln [Koch_98]. Außerdem werden Chipkarten heutzutage von außen mit ihrem Takt versorgt, was für spektakuläre Ausforschungen genutzt werden konnte [AnKu_96].
  Dies stellt die Vertraulichkeit der in der Chipkarte verarbeiteten Daten in Frage (zumindest gegenüber dem Chipkartenbesitzer und Geräten, in die die Chipkarte gesteckt wird).

– Normale Chipkarten können Eingriffe nicht erkennen, da unter anderem eine permanente Energieversorgung, z.B. Batterie, fehlt.
  Dies stellt die Vertraulichkeit und ggf. auch Integrität der in der Chipkarte verarbeiteten Daten in Frage.

– Selbst während die Chipkarte vom „Lese“gerät mit Energie versorgt wird, löschen die bisher verbreiteten Chipkarten ihre sensitiven Daten nicht – auch wenn sie unzulässige Eingriffe entdecken.
  Dies stellt die Vertraulichkeit der in der Chipkarte verarbeiteten Daten in Frage.

2.1.4 Sinnvolle physische Sicherheitsannahmen

Aus den in §2.1.2 genannten Gründen sollte mit physischen Sicherheitsannahmen möglichst sparsam umgegangen werden:

> Es sollte eine *möglichst weitgehende Übereinstimmung zwischen organisatorischen und informationstechnischen Strukturen* erzielt werden. Im Idealfall sollte jeder seine eigenen Daten verarbeiten und schützen, auf die anderer aber nur mit deren Zustimmung zugreifen können.

---

15 Eine der ganz wenigen Ausnahmen [Wein_87] sei kurz beschrieben: Der vor physischen Angriffen zu schützende Bereich wird mit dünnen, isolierten Drähten dicht umwickelt und danach mit einer undurchsichtigen, erhärtenden Flüssigkeit vergossen. Im Innern des zu schützenden Bereiches wird der Widerstand der Drähte kontinuierlich gemessen und bei einer verdächtigen Veränderung werden die vertraulich zu haltenden Informationen aktiv gelöscht.

Wird die Schutzschale physisch abgetragen oder angebohrt, werden Drähte unterbrochen. Wird die Schutzschale chemisch gelöst, entstehen Kurzschlüsse zwischen Drähten, da ihre Isolierung chemisch leichter lösbar gewählt ist als die erhärtete Flüssigkeit. In beiden Fällen wird der Widerstand der Drähte deutlich geändert.

All dies bedeutet, möglichst zu vermeiden, daß Geräte gegen ihre Benutzer sicher sein sollen.

Trotz aller physischen Gerätetechnik gilt: Es ist vorsichtig und vermutlich nötig, davon auszugehen, daß es allenfalls im Weltraum Daten gibt, auf die niemand physischen Zugriff hat. Glücklicherweise kann für viele Anwendungen Sicherheit (Vertraulichkeit, Integrität, Verfügbarkeit; je nach Anwendungsgebiet z.B. Rechtssicherheit, Schutz personenbezogener Daten vor unbefugter Kenntnisnahme, usw.) auch unter dieser defensiven Annahme erreicht werden, wie im folgenden dargelegt wird.

2.2 Schutz isolierter Rechner vor unautorisiertem Zugriff und Computer-Viren

Der in diesem Unterkapitel beschriebene **„logische" Schutz setzt** den im vorherigen Unterkapitel beschriebenen **physischen und organisatorischen Schutz voraus**.

Zunächst wird beschrieben, wie Menschen und IT-Systeme identifiziert werden können[16]. Dies bildet die Basis, um Unberechtigten keine Dienste des IT-Systems zu gewähren (Zugangskontrolle) und Berechtigten nur die für sie bestimmten (Zugriffskontrolle). Am Ende des Abschnitts wird gezeigt, wie durch geeignete Zugriffskontrolle und digitales Signieren von Programmen die Bedrohung „Computer-Viren" auf die durch „transitive Trojanische Pferde" eingeschränkt werden kann.

2.2.1 Identifikation

Ein IT-System kann einen *Menschen* daran erkennen, was er *ist*[17], *hat* oder *weiß*, vgl. Bild 2-2. Entsprechendes gilt für die Erkennung von Menschen durch Menschen, wobei hier Handgeometrie und Retina-Muster sowie Magnetstreifenkarte, Chipkarte und Taschenrechner keine wesentliche Rolle spielen dürften.

Natürlich sind Kombinationen möglich und sinnvoll. Z.B. kombiniert ein Reisepaß, was der Inhaber ist (Aussehen mittels Paßbild, schwer nachbildbare, zu einem gewissen Teil unbewußte Handlungsabläufe mittels eigenhändiger Unterschrift) damit, was er hat – nämlich den Reisepaß selbst.

---

16 Manche Autoren unterscheiden zwischen **Identifikation** (bei ihnen: Behauptung einer Identität) und **Authenti(fi)kation** (Nachprüfen einer Identität). Hier wird Identifikation umfassend verstanden, d.h. Ermittlung und Prüfung der Identität. Dies deshalb, da hier immer beides zusammen auftritt und der Begriff Authentikation auf Nachrichten angewandt werden soll, d.h. stammt die vorliegende Nachricht tatsächlich von ihrem vorgeblichen Urheber.

17 Verfahren zur Überprüfung, was man ist, heißen **biometrische Verfahren**. Ein Überblick über Leistung und Kosten der Biometrik (biometrics) ist in [DaPr_89, DuD3_99, Eber_98, FMSS_96, Mill3_94] zu finden.

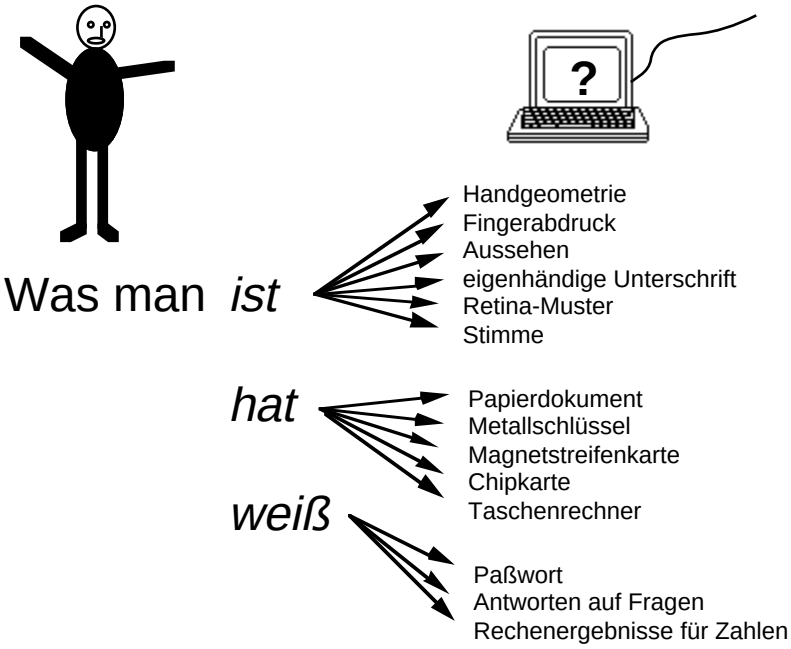


Bild 2-2: Identifikation von Menschen durch IT-Systeme

Bild 2-2 gibt keineswegs eine umfassende Liste:

Einerseits können die genannten Beispiele weiter aufgespalten werden, etwa „eigenhändige Unterschrift“ in „Prüfung des Ergebnisses des Unterschreibens“, also Vergleich zweier statischer Linienzüge, und „Prüfung der Unterschriftsdynamik“, also Vergleich von Druck und Beschleunigungswerten. Letzteres beispielsweise ist deutlich sicherer.

Andererseits gibt es weitere Beispiele, etwa Analyse der Tippverhaltens bei Tastatureingabe oder des Schreibstils bei Peneingabe, beides Maßnahmen der Klasse „Was man ist“.

Bei jeder der Maßnahmen muß dann noch entschieden werden, ob sie

- nur zu Beginn,
- zusätzlich auch nach festen Zeitintervallen oder
- permanent

durchgeführt wird. Unter anderem bei „Metallschlüssel“, Analyse des Tippverhaltens bzw. Schreibstils bietet sich permanente Durchführung an (Schlüssel bleibt im Schloß, Analyseprozeß läuft im Hintergrund weiter mit), während man von niemand verlangen sollte, eine seiner Hände zur permanenten Erfassung ihrer Geometrie auf einem Scanner liegen zu lassen.

Ein Mensch kann ein *IT-System* daran erkennen, was es *ist*, *weiß* oder *wo* es steht, vgl. Bild 2-3. Daß nicht nur das IT-System den Benutzer, sondern auch der Benutzer das IT-System identifizieren muß, und der Ort hierfür allein evtl. nicht ausreicht, zeigt folgende Begebenheit:

> Von Freitag auf Samstag wurde vor einen Bankautomaten eine ähnlich aussehende Attrappe montiert. Diese erfragte zu den eingeschobenen ec-Karten die PIN, speicherte sie und meldete auf dem Display: „Die fehlerhafte ec-Karte wird eingezogen. Bitte wenden Sie sich Montag an Ihre Bank.“ Mit den eingezogenen ec-Karten und den zugehörigen PINs konnte in der Zwischenzeit an richtigen Bankautomaten in aller Ruhe Geld abgehoben werden.

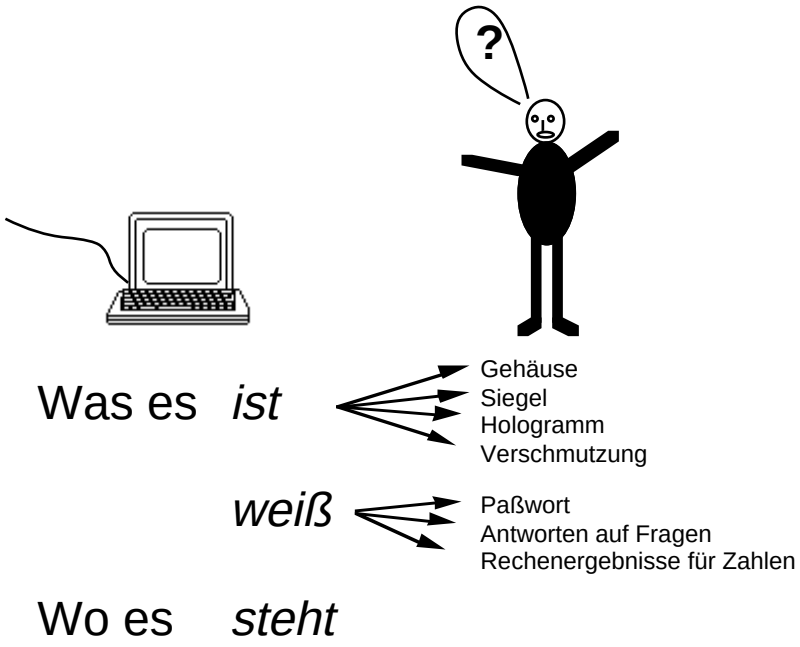


Bild 2-3: Identifikation von IT-Systemen durch Menschen

Ein IT-System kann ein anderes IT-System daran erkennen, was es *weiß* und *woher* die Leitung kommt, vgl. Bild 2-4. Letzteres mag zwischen ICs auf derselben Platine eine genügend wirksame Identifikation sein, ist in offenen Kommunikationsnetzen aber völlig unzureichend.

Wichtig ist, daß zwischen IT-Systemen nicht nur vergleichsweise unsichere Verfahren wie Paßwort, Frage-Antwort-Dialoge und Challenge-Response-Systeme wie „Geschickt wird Zahl, erwartet wird Rechenergebnis“ möglich sind, sondern auch kryptographische Protokolle hoher Komplexität und Sicherheit. Ein Beispiel für letzteres:

> IT-System 1 generiert eine Zufallszahl, sendet sie IT-System 2 und erwartet eine Verschlüsselung mit einem nur diesen beiden IT-Systemen bekannten Schlüssel einer vereinbarten symmetrischen Blockchiffre[18]. Hat IT-System 1 die erwartete Verschlüsselung erhalten, „kennt“ es IT-System 2. Entsprechend kann IT-System 2 das IT-System 1 identifizieren. (In Aufgabe 3-18a finden Sie einen Hinweis auf Angriffe, die hierdurch nicht abgewehrt werden.)

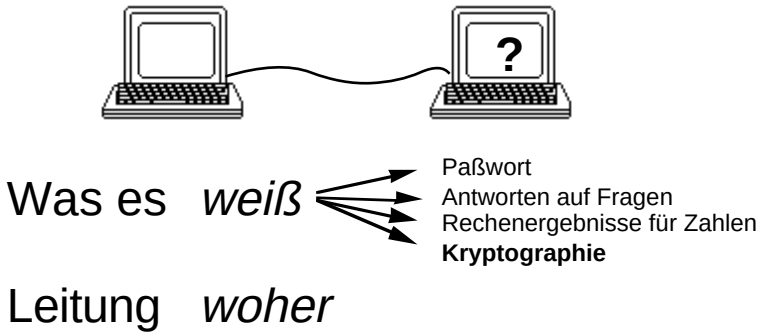


Bild 2-4: Identifikation von IT-Systemen durch IT-Systeme

---

18 Bei gegebenem Alphabet ver- bzw. entschlüsselt eine **Blockchiffre** Blöcke *fester* Länge, während eine **Stromchiffre** dies mit Zeichenketten *beliebiger* Länge kann, vgl. §3.8. Blockchiffren haben in der Regel Schlüssel fester Länge (z.B. 128 Bit lang), mit denen Millionen von Blöcken ver- bzw. entschlüsselt werden

2.2.2 Zugangskontrolle

Unter **Zugangskontrolle** versteht man, daß ein IT-System die Identitäten seiner Kommunikationspartner erfragt, prüft und nur mit berechtigten Partnern weiterkommuniziert. Es verhindert so unbefugte Inanspruchnahme seiner Betriebsmittel und ggf. auch noch wesentlich Schlimmeres.

Es ist bemerkenswert, daß diese Identitäten für viele Dienste nicht an die Identität einer natürlichen Person gekoppelt sein müssen, sondern daß mit sogenannten Pseudonymen gearbeitet werden kann [Chau_87, PWP_90]. Wie dies möglich ist, wird in §6 erläutert.

Es sei darauf hingewiesen, daß es sich wie bei Identifikation auch bei Zugangskontrolle um ein symmetrisches Problem handelt. Auch der Mensch wird in der Regel Interesse daran haben, seine Zeit nicht mit ihn nasführenden IT-Systemen zu verbringen oder ihnen gar seine Geheimnisse zu verraten.

2.2.3 Zugriffskontrolle

Unter **Zugriffskontrolle** versteht man, daß ein IT-System auch berechtigten Partnern nicht alles erlaubt: Jedes *Subjekt* (Mensch, IT-System, Prozeß) hat nur bestimmte *Rechte*, Operationen auf *Objekten* (Prozesse, Daten, Peripherie-Geräte, etc.) auszuführen. Ein möglichst kleiner und gut abgegrenzter Teil des IT-Systems kontrolliert vor Ausführung aller Operationen, ob ihr Urheber die dafür nötigen Rechte hat. Dieser Teil des IT-Systems wird **Zugriffsmonitor** genannt, vgl. Bild 2-5. Der Zugriffsmonitor merkt sich ihm vorgelegte oder implizit entstehende Rechte und muß auch deren Ungültigwerden erkennen.

Die Rechtevergabe selbst wird üblicherweise **Autorisierung** (authorization) genannt. Sie muß natürlich mindestens so gut geschützt werden wie der eigentliche Zugriff auf die Objekte.

Da man leider meist nicht im voraus genau genug weiß oder formal spezifizieren will, wer unter welchen Umständen was darf, und deshalb die Rechte im IT-System eigentlich zu großzügig einräumt, hat der Zugriffsmonitor oftmals auch die Aufgabe, alle oder zumindest eine geeignete Auswahl der Operationen zu *protokollieren*. Hat ein Datenschutzbeauftragter[19] geeignete Programme zur Auswertung der Protokolle zur Verfügung, so kann man hoffen, daß er Übertretungen der nur in der Umwelt des IT-Systems definierten Rechte erkennen und ggf. Konsequenzen veranlassen kann.

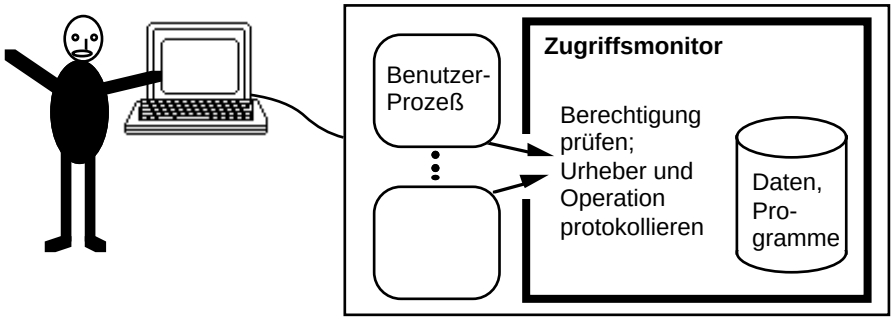


Bild 2-5: Zugriffskontrolle durch Zugriffsmonitor

Zweierlei sei hervorgehoben:

---

können, ohne daß dadurch selbst einem adaptiven aktiven Angreifer (vgl. §3.1.3.2) ein Brechen des Systems gelingen sollte.

19 Wenn es nicht um personenbezogene Information geht und Datenschutz eng verstanden wird, sollte man den weniger üblichen Begriff *Sicherheitsbeauftragter* verwenden.

Der Zugriffsmonitor kontrolliert den Zugriff von Rechenprozessen, die Menschen oder IT-Systeme vertreten können, auf Daten und Programme (inkl. Ausgabe), und nicht den unmittelbaren Zugriff von Menschen auf sie oder Geräte.

Benutzer-Prozesse tun auch auf einer korrekten Betriebssystemmaschine nicht unbedingt das, was der Benutzer will oder von ihnen vorgespiegelt bekommt – denn sie können Trojanische Pferde enthalten. Aus diesem Grund sitzt der Zugriffsmonitor an der richtigen Stelle, nämlich zwischen Benutzer und Benutzer-Prozeß einerseits und den Daten und Programmen, die anderen gehören können, andererseits. Dies wird im folgenden Abschnitt am Beispiel der „Computer-Viren“ noch etwas deutlicher.

Soll Zugriffskontrolle nicht nur bei der Realisierung von Vertraulichkeit und Integrität helfen, sondern auch bei der Realisierung von *Verfügbarkeit*, dann muß der Zugriffsmonitor nicht nur das Vorhandensein von Rechten prüfen, sondern auch, wie häufig sie ausgeübt wurden und werden dürfen. Etwas allgemeiner gesagt: Für Verfügbarkeit ist nicht nur Zugriffskontrolle im engeren Sinne nötig, sondern auch eine *Kontrolle des Ressourcenverbrauchs* – wozu selbstverständlich auch die Benutzung der der Zugriffskontrolle unterliegenden Objekte gehört.

2.2.4 Beschränkung der Bedrohung „Computer-Viren“ auf die durch „transitive Trojanische Pferde“

Die in aller Munde und Zeitschriften befindlichen „*Computer-Viren*“ sind spezielle, durch ihren Ausbreitungsmechanismus definierte Trojanische Pferde:

> Hat ein ausgeführtes Programm, in dem sich ein Computer-Virus befindet, Schreibzugriff auf ein beliebiges anderes Programm, so kann er dieses Programm so ändern, daß bei dessen Ausführung eine (optional veränderte) Kopie des Virus mit ausgeführt wird.

Computer-Viren können sich in der transitiven Hülle der (meist üppig gewährten) Schreibzugriffsrechte ausbreiten.

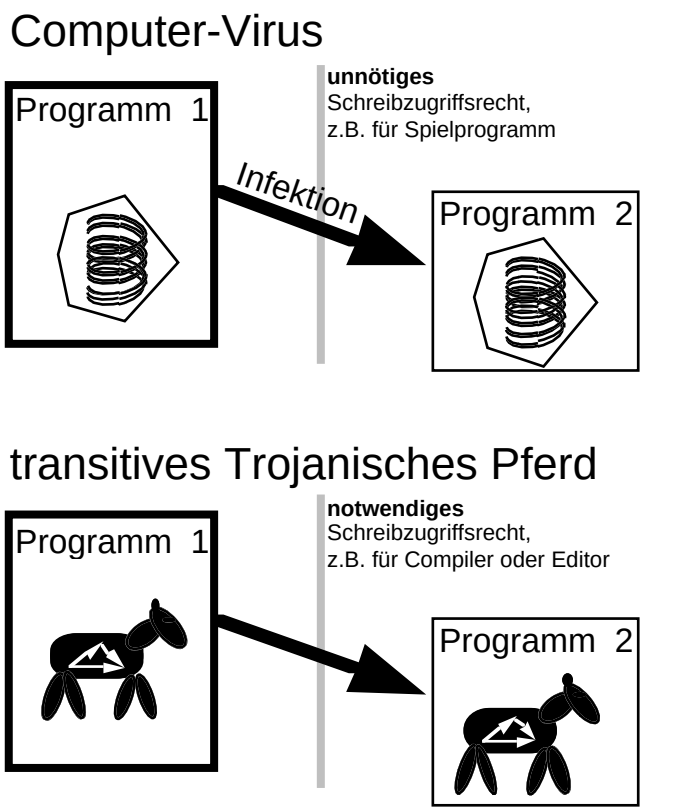


Bild 2-6: Computer-Virus, transitives Trojanisches Pferd und Beschränkung der Bedrohung durch ersteres auf die durch letzteres

Grundsätzliche Schutzmechanismen zur Verhinderung der Ausbreitung von Computer-Viren waren bereits vor deren Erfindung bekannt [Denn_82 Seite 317f]:

> Jedem Programm wird bei seiner Ausführung nur das ermöglicht, was es können muß (Prinzip der geringstmöglichen Privilegierung, principle of least privilege).
>
> Um Programme vor unbemerkter Veränderung auch außerhalb physischer Schutzbereiche zu schützen, werden alle Programme vom Generierer digital signiert, vgl. §3 und [ClPf_91, Cles_88]. Vor der Ausführung jedes Programms wird geprüft, daß es nicht unautorisiert verändert wurde.

Da diese Schutzmechanismen die Bedrohung durch Computer-Viren genau auf die durch transitive Trojanische Pferde vermindern [Pfit_89 Seite 7, 17], sind Computer-Viren kein zusätzliches Forschungsproblem der Informatik. Leider wurden diese grundsätzlichen Schutzmechanismen jedoch aus Denkfaulheit und zu einem geringeren Teil auch Aufwands- und damit Kostengründen bisher nicht eingesetzt. Daraus resultiert ein schwerwiegendes Problem, da gegen Computer-Viren weitgehend ungeschützte Rechner weit verbreitet sind.

Anmerkung mit Begründungen: Es ist absehbar, daß alle Maßnahmen, gegen Computer-Viren vorzugehen, ohne das Prinzip der geringstmöglichen Privilegierung durchzusetzen, versagen werden:

1. Beh.: Es ist *nicht entscheidbar*, ob ein Programm ein Computer-Virus ist oder nicht.
   Bew. indirekt (nach [Cohe_84, Cohe_87] vgl. auch [Cohe1_89, Cohe2_92])
   Nehmen wir an, wir hätten eine Entscheidungsprozedur decide(•), die zu jedem Programm TRUE liefert, wenn es ein Computer-Virus ist[20], und FALSE andernfalls. Dann liefert decide zu folgendem Programm ein falsches Ergebnis:

   ```
   program Gegenbeispiel;
   if decide(Gegenbeispiel) then keine_Virusfunktionen_ausführen
   else Virusfunktionen_ausführen;
   ```

   Also kann es keine solche Entscheidungsprozedur decide(•) geben.

2. Da Computer-Viren spezielle Trojanische Pferde sind, gilt:
   Es ist nicht entscheidbar, ob ein Programm ein Trojanisches Pferd ist oder nicht.

Also ist es nicht allgemein möglich, Computer-Viren oder Trojanische Pferde zu erkennen. Man kann sich nur gegen sie schützen, indem man bei Ausführung ihre Zugriffsmöglichkeiten beschränkt und/oder verdächtige Programme erst gar nicht ausführt. Kurzum: Statt genauem Entscheiden: Vorsicht und im Zweifelsfall lieber ein gutartiges Programm für potentiell bösartig halten!

Leider sind auch die zwei bescheideneren Ziele (*bekannte* Computer-Viren erkennen und nach erkanntem Virenbefall den Schaden bzgl. der Daten ermitteln), die heute allgemein angestrebt werden, nicht erreichbar:

3. Beh. Selbst für *bekannte* Computer-Viren gibt es keine wirksamen Erkennungsmechanismen!
   Begründung: Computer-Viren können sich selbst modifizieren und dem Erkennungsmechanismus unbekannte, weil erst zu ihrer Laufzeit eintretende, Ereignisse benutzen, um die Selbstmodifikation zu steuern. Ein Erkennungsmechanismus müßte also *alle* möglichen Selbstmodifikationen des bekannten Computer-Virus analysieren, was aus Aufwandsgründen bei geschickter Programmierung des Computer-Virus nicht durchführbar ist. Schätzt man deshalb ab und läßt einen sogenannten **Viren-Scanner** nur nach kurzen Mustern suchen, gibt es kaum noch Programme, die nicht virenverdächtig sind. Erste (allerdings noch nicht wirklich gute) Werkzeuge, die solche Selbstmodifikation zu generieren gestatten, sind in einschlägigen Kreisen verbreitet, z.B. Mutation Engine [Adam2_92, DuDR2_92].

4. Beh. Selbst für *bekannte* transitive Trojanische Pferde gibt es keine wirksamen Erkennungsmechanismen!
   Begründung wie bei 3.

5. Beh. Es kann bei keinem Virenbefall hinterher mit Sicherheit festgestellt werden, ob vom Computer-Virus Daten unbefugt geändert und/oder unbefugt weitergegeben wurden.
   Begründung: Nach Durchführung seiner sogenannten **Schadensfunktion** kann das Computer-Virus seine Schadensfunktion selbst modifizieren und danach diesen Modifikationsmechanismus.

2.2.5 Restprobleme

1. Man muß genau spezifizieren, was ein IT-System tun und *unterlassen* soll. Letzteres wird meist vergessen.

---

[20] Hier kommt es, wie bei vielen Begründungen, auf den genauen Wortlaut von Definition und Behauptung an. Die obige Definition von Computer-Virus lautete (ein Wort hier hervorgehoben):

> Hat ein ausgeführtes Programm, in dem sich ein Computer-Virus befindet, Schreibzugriff auf ein beliebiges anderes Programm, so *kann* er dieses Programm so ändern, daß bei dessen Ausführung eine (optional veränderte) Kopie des Virus mit ausgeführt wird.

Es wird also nicht gefragt, ob ein Programm Virusfunktionen *enthält*, sondern ob es sie *ausführen kann*. Nur hierfür liefert obiges Programm `Gegenbeispiel` ein Gegenbeispiel.

2. Danach ist eigentlich die *totale Korrektheit der Implementierung* nachzuweisen, was aus Komplexitätsgründen heutzutage oft nicht möglich ist.
3. Wäre dies richtig durchgeführt, bliebe „nur“ noch das Problem übrig, ob man alle *verborgenen Kanäle erkannt* hat

Das erste und dritte Problem können nie mit Sicherheit richtig gelöst werden, denn man kann immer etwas übersehen haben. Und auch das zweite Problem ist noch nicht befriedigend gelöst. Deshalb empfiehlt sich:

> IT-Systeme sollten prinzipiell derart als verteiltes System entworfen und realisiert werden, daß ein Angreifer, der eine begrenzte Zahl Rechner kontrolliert (sei es als Betreiber, sei es als Benutzer eines universellen Trojanischen Pferdes), keinen wesentlichen Schaden anrichten kann.

Wie dies für einige ausgewählte IT-Systeme geschehen kann, wird in den folgenden Kapiteln erläutert.

Leseempfehlungen

Physische Sicherheitsannahmen: Chau_84, Clar_88, Wein_87

Identifikation: DaPr_89 §7, Mill3_94, DuD3_99

Zugangskontrolle: Chau_87, PWP_90

Zugriffskontrolle: Denn_82, ZSI_89, ZSI_90

Aktuelle Forschungsliteratur:

Computers & Security, Elsevier

Tagungsserien IEEE Symposium on Security and Privacy, ESORICS, IFIP/Sec, VIS; Tagungsbände erscheinen bei IEEE, in der Serie LNCS des Springer-Verlags Heidelberg, bei Chapman & Hall London, in der Serie Datenschutz-Fachberichte des Vieweg-Verlags Wiesbaden

Hinweise auf aktuelle Forschungsliteratur und Kommentare zu aktuellen Entwicklungen:

http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/ oder
abonnieren unter cipher-request@itd.nrl.navy.mil

Datenschutz und Datensicherheit DuD, Vieweg-Verlag

3 Kryptologische Grundlagen

Kann oder soll[21] ein IT-System nicht vollständig physisch geschützt werden, so ist Kryptographie *das* effiziente Hilfsmittel, um trotzdem *Vertraulichkeit* und *Integrität* zu erreichen. Zur Verfügbarkeit trägt Kryptographie nicht oder allenfalls indirekt bei.

Kommt es auf Effizienz nicht allzusehr an, so steht mit der Steganographie ein sogar weiterreichendes, allerdings bei weitem noch nicht vergleichbar gut untersuchtes Hilfsmittel zur Verfügung, vgl. §4.

Begrifflich kann **Kryptologie**, d.h. die Wissenschaft (dem griechischen Wortsinn nach: die Weisheit) der Geheimschriften, in **Kryptographie** und **Kryptoanalyse**[22] unterschieden werden: Kryptographie beschreibt, wie die Geheimschrift funktioniert. Kryptoanalyse versucht, Geheimschriften zu analysieren, sprich: zu entschlüsseln.

3.1 Systematik

Wir unterscheiden hier die wichtigsten kryptographischen Systeme nach drei Kriterien:

1. Dem **Zweck**.
   Hier betrachten wir vor allem
   - **Konzelationssysteme**, d.h. Systeme zur Geheimhaltung von Nachrichten (in der Literatur auch oft Verschlüsselungssysteme oder einfach Kryptosysteme genannt). Diese dienen also dem Schutzziel *Vertraulichkeit*.
   - **Authentikationssysteme**, d.h. Systeme, die Nachrichten vor unbemerkter Veränderung schützen, ähnlich wie fehlererkennende Codes, aber gegen bösartigere Fehler. Sie dienen also dem Schutzziel *Integrität*. Ein besonders starker Spezialfall sind
     - **digitale Signatursysteme**, bei denen ein Empfänger, der eine korrekt „codierte“ Nachricht bekommt, nicht nur selbst sicher ist, daß sie vom behaupteten Sender stammt, sondern dies auch Außenstehenden beweisen kann, z.B. einem Gericht.

   Daneben gibt es z.B. Pseudozufallsbitfolgengeneratoren (§3.4) und kollisionsresistente Hashfunktionen (§3.6.4 und §3.8.3) [Damg_88] und vieles andere.

2. Der nötigen **Schlüsselverteilung**.
   Diese Unterscheidung bezieht sich vor allem auf solche kryptographischen Systeme, bei denen es zwei Sorten von Parteien gibt, Sender und Empfänger (wie eben Konzelations- und Authentikationssysteme). Hier unterscheidet man **symmetrische** und **asymmetrische** Systeme, je nachdem, ob beide Parteien den gleichen oder verschiedene Schlüssel haben, siehe unten.

3. Dem **Sicherheitsgrad**.

---

21 Besonders bei Rechnernetzen ist dies der Fall: Nehmen wir an, wir brauchen für physischen Schutz eine Person Wachpersonal alle 100 m, damit sie die Leitung im Auge behalten kann. Nehmen wir an, wir hätten gerne Sicherheit rund um die Uhr, dann brauchen wir 4-Schicht-Betrieb. Nehmen wir an, Wachpersonal arbeitet für den sehr niedrigen Lohn von 50 000 DM/Jahr brutto. Dann kostet der physische Schutz einer Übertragungsstrecke 2 Millionen DM pro Jahr und Kilometer. Da macht sich das Durcharbeiten dieses Kapitels und das Anschaffen von ein paar kryptographischen Geräten (oder auch nur Software) schnell bezahlt. Sie können jetzt natürlich ausrechnen, was der Ersatz von Menschen durch Hunde und der Ersatz von kontinuierlichem im Auge Behalten durch physische Hindernisse wie unterirdisches Verlegen (und dann nur noch regelmäßige Inaugenscheinnahme alle Stunde) einspart, aber so viel Sie auch rechnen, Sie werden nie so preiswert werden wie Kryptographie – großes Kryptologenehrenwort.

22 Manche Autoren sparen eine Silbe ein, indem sie das o weglassen. Sie sprechen von **Kryptanalyse**.

Die Hauptunterscheidung hier ist in **informationstheoretische** und **kryptographische** Sicherheit. Erstere ist absolut (soweit man nur das kryptographische System betrachtet, also z.B. nicht Schlüsseldiebstahl). Letztere ist schwächer, kann aber noch sinnvoll weiter unterteilt werden, siehe unten.

Zwei Einschränkungen von *Konzelationssystemen* seien gleich zu Beginn erwähnt:

- Sie schützen nur die Vertraulichkeit der Nachrichten*inhalte*, nicht aber Kommunikationsumstände, beispielsweise wer von wo wann mit wem kommuniziert. Sollen auch die Kommunikationsumstände geschützt werden, sind zusätzliche Schutzmaßnahmen nötig, vgl. §5.
- Oftmals werden auch die Nachrichteninhalte nicht vollständig geschützt, sondern manche Attribute des Nachrichteninhalts, beispielsweise seine Länge, schlicht ignoriert[23]. Besonders deutlich wird dies beim Beweis der perfekten Sicherheit der Vernam-Chiffre, vgl. Aufgabe 3-7 b).

Was mag diese weit verbreiteten, von den meisten Fachleuten überhaupt nicht bemerkten Fehler (oder beschönigend: Ungenauigkeiten) verursacht haben? Vielleicht folgendes Vorgehen:

Auf der höchsten Entwurfsebene, der Anforderungsdefinition, wurde festgelegt: Nachrichten (auf dieser Ebene atomare Objekte) sollen vertraulich bleiben. Später wird auf niedrigeren Ebenen der atomare Typ Nachricht verfeinert, indem über die Repräsentation von Nachrichten entschieden und dabei beispielsweise implizit die Länge von Nachrichten eingeführt wird. Schließlich wird die Anforderung „Vertraulichkeit von Nachrichten“ auf der tieferen Ebene nur auf das entsprechende Hauptattribut bezogen, implizit eingeführte Nebenattribute wie z.B. „Länge“ geraten entweder überhaupt nicht ins Blickfeld oder werden für nicht wichtig erachtet.

Was lernen wir daraus? *Alle* verfeinerten, auch alle implizit eingeführten Attribute eines vertraulich zu haltenden Objektes müssen vertraulich gehalten werden.[24] Aus einer Vertraulichkeitsanforderung werden im Zuge der Verfeinerung also *mehrere*. Sollte dies dem Entwerfer (d.h. dem Verfeinerer) zu aufwendig erscheinen, so muß er die Ersteller der Anforderungsdefinition fragen, ob sie mit der Nichtrealisierung der Vertraulichkeitsforderung bzgl. mancher Attribute einverstanden sind. Dies klingt jedoch wesentlich einfacher, als es sein dürfte, denn der Entwerfer muß den Erstellern der Anforderungsdefinition erst einmal seine (verfeinerte) Welt umfassend erklären. Und mit der wollten sich in aller Regel die Ersteller der Anforderungsdefinition gerade nicht befassen.

---

23 Wenn manche Attribute des Nachrichteninhalts ignoriert werden, muß darauf geachtet werden, daß sie über den gesamten Nachrichteninhalt nicht „zu viel“ verraten. Ein drastisches Beispiel wäre, wenn Nachrichteninhalte unär codiert werden und das Attribut Nachrichtenlänge nicht geschützt wird. Dann verrät die Nachrichtenlänge alles über den Nachrichteninhalt, so daß Verschlüsselung etwa mit der Vernam-Chiffre überhaupt nichts nützt. (**Unäre Codierung** bedeutet: Es gibt nur ein Codezeichen – beispielsweise würde 17 also durch 17-faches Hintereinanderschreiben dieses Codezeichens dargestellt.)

24 Dies gilt auch für abgeleitete Attribute anderer Objekte, wie die folgenden zwei Beispiele zeigen:

1. Bei der in §3.2 beschriebenen Vernam-Chiffre darf die Darstellung des Schlüsseltextes, die der Angreifer erfährt, nicht davon abhängen, wie sich die modulo-Summen zusammensetzen. Bei der binären Vernam-Chiffre muß also $0 \oplus 1$ bzgl. aller Eigenschaften wie analoge Signalpegel, Timing etc. genau gleich wie $1 \oplus 0$ aussehen, gleiches gilt für $0 \oplus 0$ und $1 \oplus 1$. In der Praxis ist dies leicht zu erreichen, beispielsweise Zwischenspeichern des Ergebnisses vor der Ausgabe – bei einer Softwareimplementierung passiert das in jedem Rechner automatisch. Bei einer Hardwareimplementierung muß mensch allerdings dran denken.
2. Auch bei Softwareimplementierung kryptographischer Systeme ist Timing kritisch – etwa darf die Ausführungszeit kryptographischer Operationen nicht vom Wert des geheimen Schlüssels abhängen. Dies kann erreicht werden, indem bei der Programmierung kryptographischer Software darauf geachtet wird, daß in keiner Sprungbedingung der Wert (oder auch nur Teile des Wertes) des geheimen Schlüssels vorkommen. Alternativ kann über Zeitabfragen sichergestellt werden, daß alle kryptographischen Operationen gleich lange dauern.

Beide Beispiele sind schon seit langem bekannt – aber anscheinend in der Praxis nicht durchgehend beachtet. Sonst hätte die Publikation von P. Kocher über *Timing Attacks* kein so breites Echo gefunden. Mehr über diesen Angriff steht in [EnHa_96].

Die wichtigsten, im folgenden betrachteten kryptographischen Systeme bestehen aus 3 Algorithmen (z.B. Schlüsselgenerierung, Verschlüsselung, Entschlüsselung in Bild 3-1), die öffentlich bekannt sein dürfen. Zunächst werden die Funktionen der verschiedenen Systeme mit der benötigten Schlüsselgenerierung und -verteilung dargestellt.

3.1.1 Kryptographische Systeme und ihre Schlüsselverteilung

Zunächst wird ein Überblick über symmetrische und asymmetrische Konzelationssysteme gegeben, danach einer über symmetrische und asymmetrische Authentikationssysteme.

Allen vier Bildern ist folgender Grundaufbau unterlegt (Bild 3-1): Links und rechts befindet sich je ein **Vertrauensbereich** der jeweiligen Teilnehmer[25]. In der Mitte unten befindet sich der **Angriffsbereich**, d.h. der Bereich, wo Angriffe mittels des kryptographischen Systems pariert werden sollen, da sie dort nicht durch physische und/oder organisatorische Maßnahmen verhindert oder zumindest vereitelt werden.

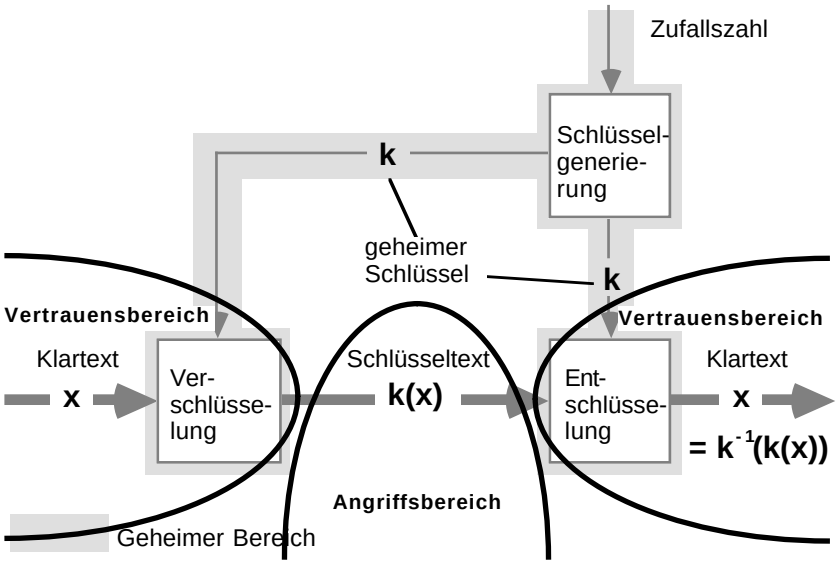


Bild 3-1: Grundschema kryptographischer Systeme am Beispiel symmetrisches Konzelationssystem

3.1.1.1 Konzelationssysteme, Überblick

Die bekannteste und älteste Sorte kryptographischer Systeme ist das **symmetrische Konzelationssystem**, s. Bild 3-2.

---

25 Dieser Vertrauensbereich muß jeweils durch andere als kryptographische Maßnahmen gesichert, insbesondere durch physische Mittel und organisatorische Maßnahmen abgegrenzt und ggf. verteidigt werden. In ihm sollte das und nur das vorgehen, was der jeweilige Teilnehmer autorisiert. Und den Vertrauensbereich sollte nur die Information verlassen, deren Verlassen der jeweilige Teilnehmer autorisiert. Dies bedeutet, daß der Vertrauensbereich physisch abgeschirmt sein muß (vgl. §2.1): Ihn darf keine auswertbare elektromagnetische Strahlung verlassen, und auch seine Eingaben – etwa sein Energieverbrauch – darf nicht davon abhängen, was in ihm vorgeht. Wird dies nicht beachtet, so kann der unterschiedliche Stromverbrauch einzelner Transistoren genutzt werden, um beispielsweise an geheime Schlüssel zu kommen [Koch_98]. Vertrauensbereiche sollten von ihrer Umgebung also so weit wie möglich entkoppelt sein. Ein Vertrauensbereich wird üblicherweise durch ein dem jeweiligen Teilnehmer (hoffentlich) vertrauenswürdiges Benutzerendgerät realisiert, vgl. [PPSW_95, PPSW_97].

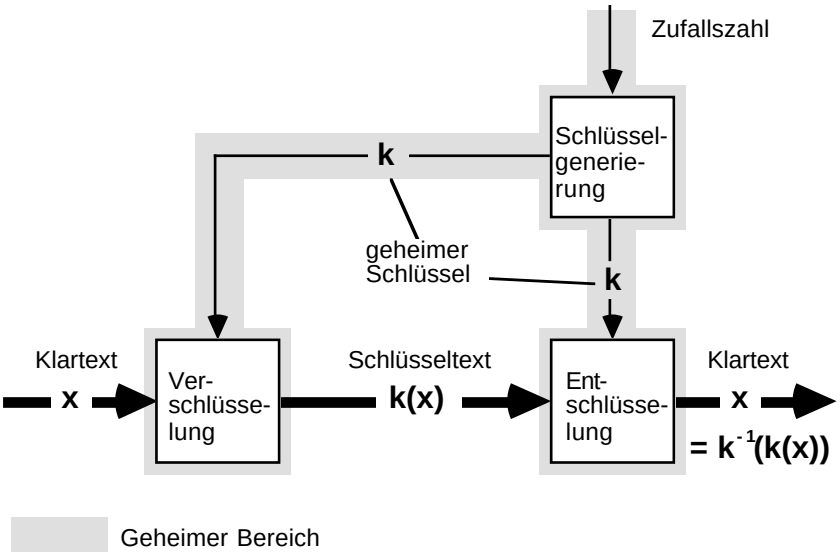


Bild 3-2: Symmetrisches Konzelationssystem
(≈ undurchsichtiger Kasten mit Schloß, es gibt zwei gleiche Schlüssel)

Anmerkung: Die Schreibweise $k(x)$ ist nur eine Abkürzung, weil das k, das die Schlüsselgenerierung generiert, i.allg. kein ganzer Algorithmus ist. Es ist nur ein Schlüssel, d.h. Parameter, der in vorgegebene Ver- und Entschlüsselungsalgorithmen eingesetzt wird, z.B. *ver* und *ent*. (In der Realität würden also *ver* und *ent* ein für allemal programmiert oder in Hardware implementiert.) Der Schlüsseltext ist dann

$$S := ver(k, x)$$

und die Entschlüsselung ist

$$ent(k, S) = ent(k, ver(k, x)) = x.$$

Das Problem beim symmetrischen Konzelationssystem ist, daß, wenn die Nachricht verschlüsselt werden soll, um über ein unsicheres Netz gesendet zu werden, zuvor der Schlüssel k bei Sender und Empfänger vorliegen muß.

Wenn sich Sender und Empfänger vorher getroffen haben, konnten sie k bei der Gelegenheit austauschen.

Wenn aber einfach einer die Netzadresse des anderen (z.B. eines Dienstanbieters in einem offenen System) aus einer Art Telefonbuch entnimmt und beide nun vertraulich kommunizieren wollen, wird es schwierig. In der Praxis geht man dafür meist von der Existenz einer vertrauenswürdigen „**Schlüsselverteilzentrale**“ Z aus. Jeder Teilnehmer A tauscht bei der Anmeldung zum offenen System einen Schlüssel mit Z aus, etwa k_{AZ}. Wenn nun A mit B kommunizieren will und noch keinen Schlüssel mit B gemeinsam hat, so fragt er bei Z an. Z generiert einen Schlüssel k und schickt ihn sowohl an A als auch an B, und zwar mit k_{AZ} bzw. k_{BZ} verschlüsselt. Von da an können A und B den Schlüssel k benutzen, um in beide Richtungen Nachrichten zu schicken. Die Vertraulichkeit ist natürlich nicht sehr groß: Außer A und B kann auch Z alle Nachrichten entschlüsseln. (Und es ist sogar sehr wahrscheinlich, daß Z alle Nachrichten zu sehen bekommt, denn meistens ist der Betreiber der Schlüsselzentrale gleich dem Netzbetreiber.)

Man kann dies verbessern, indem man *mehrere* Schlüsselverteilzentralen so einsetzt, daß sie nur dann die Nachrichten lesen können, wenn sie alle zusammenarbeiten, siehe Bild 3-3. Wieder tauscht jeder Teilnehmer A zu Beginn mit jeder Zentrale ζ einen Schlüssel $k_{A\zeta}$ aus. Wenn A mit B kommunizieren will, so bittet er alle Zentralen, einen Teilschlüssel zu generieren. Diese schicken die Teilschlüssel k_i wieder vertraulich an A und B. A und B verwenden die *Summe* k aller k_i (in einer geeigneten Gruppe) als eigentlichen Schlüssel. Will man im Endeffekt z.B. einen 128 Bit langen

Schlüssel, so würden auch alle k_i die Länge 128 haben, und k könnte das bitweise XOR aller k_i sein (oder auch die Summe modulo 2^{128}). Solange zumindest eine der Zentralen ihren Teilschlüssel k_i geheimhält, haben die restlichen Zentralen keine Information über k, da durch das Hinzuaddieren von einem geeigneten k_i immer noch alle k möglich sind. (Dies heißt, daß aus der Sicht der restlichen Zentralen k noch mit einer Vernam-Chiffre perfekt verschlüsselt ist, vgl. §3.2.)

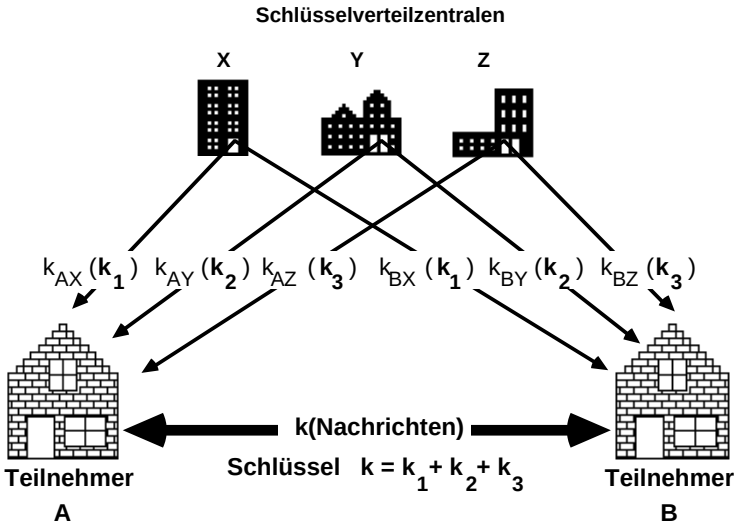


Bild 3-3: Schlüsselverteilung bei symmetrischen Konzelationssystemen

Asymmetrische Konzelationssysteme wurden genau erfunden, um die Schlüsselverteilung zu vereinfachen, s. Bild 3-4.

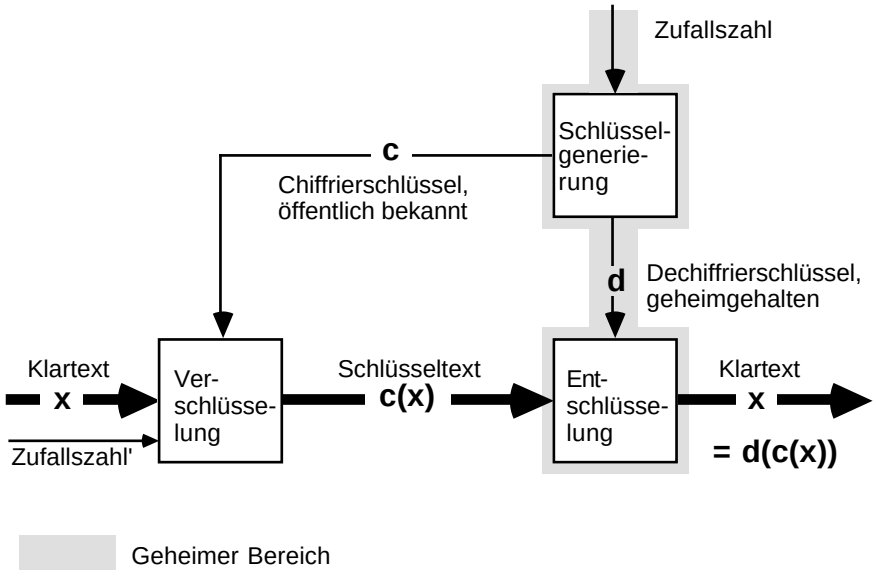


Bild 3-4: Asymmetrisches Konzelationssystem
(≈ undurchsichtiger Kasten mit Schnappschloß, es gibt nur einen Schlüssel: Jeder kann etwas hineintun, aber nur einer kann es herausholen.[26])

[26] Die Analogie zwischen einem asymmetrischen Konzelationssystem und einem undurchsichtigen Kasten mit Schnappschloß erreicht ihre Grenze, wenn der Kasten zugeschnappt ist. Dann kann ihm ohne Mithilfe des Schlüs-

Hier sind also zum Ver- und Entschlüsseln verschiedene Schlüssel *c* und *d* erforderlich, und nur *d* muß geheimgehalten werden.

Anmerkung 1: Wieder sind die Schreibweisen $c(x)$ und $d(c(x))$ nur Abkürzungen. Heißen die Ver- und Entschlüsselungsalgorithmen wieder *ver* und *ent*, so sind *c* und *d* eigentlich Eingabeparameter dafür, der Schlüsseltext ist $S := ver(c, x)$ und die Entschlüsselung $ent(d, S) = ent(d, ver(c, x)) = x$.

Anmerkung 2: Die Zufallszahl links unten hat folgende Bedeutung: Nehmen wir an, die Verschlüsselung sei deterministisch. Wenn dann ein Angreifer einen Schlüsseltext *S* sieht und einige Klartexte x_i kennt, von denen wahrscheinlich einer dahintersteckt, so kann er dies prüfen, indem er selbst alle $c(x_i)$ bildet und mit *S* vergleicht. (Dies ist bei langen Privatbriefen sicher kaum möglich, wohl aber bei standardisierten Schritten aus Protokollen, z.B. im Zahlungsverkehr.) Deswegen muß die **Verschlüsselung indeterministisch** gemacht werden, d.h. *S* hängt auch noch von einer Zufallszahl ab, die der Angreifer nicht erraten kann. (Im Englischen wird dies als **probabilistic encryption** bezeichnet.)

Man kann diese Zufallszahl als Teil des Klartextes auffassen. Da der Algorithmus Entschlüsselung den ganzen Klartext herausbekommt, erhält er auch die enthaltene Zufallszahl, gibt sie aber nicht nach außen.

Damit man *c* tatsächlich nicht geheimhalten muß, darf natürlich *d* nicht mit vernünftigem Aufwand aus *c* zu bestimmen sein. (Genauer später bei Sicherheitsbegriffen und in §3.6)

Nun kann auf jeden Fall jeder Benutzer *A* sich selbst ein Schlüsselpaar (c_A, d_A) generieren und muß d_A nie jemand anderem mitteilen. Es geht nur noch darum, c_A so zu verteilen, daß jeder andere Teilnehmer *B*, der *A* eine vertrauliche Mitteilung schicken will, es hat bzw. finden kann. Wenn *A* und *B* sich schon vorher kannten, können sie es sich wieder privat mitteilen; *B* kann nun c_A offen in sein Adreßbuch schreiben. Auch können Bekannte sich c_A weitererzählen.

Für Kontakte mit Unbekannten könnte c_A gleich mit im Telefonbuch stehen, wo *B* die Netzadresse von *A* nachschaut. Gibt es kein solches Verzeichnis außerhalb des Netzes, so kann ein im Netz agierendes Schlüsselregister an seine Stelle treten, s. Bild 3-5.

---

selbesitzers (erneutes Öffnen) nichts Geheimzuhaltendes mehr anvertraut werden, während dies beim asymmetrischen Konzelationssystem auch nach Verschlüsselung vieler Klartexte weiterhin möglich ist. Unter diesem Aspekt wäre vielleicht die Analogie mit einem undurchsichtigen Kasten mit einem Schloß, zu dem es nur einen Schlüssel gibt, der aber zusätzlich einen nur von außen nach innen passierbaren, keinen Einblick gewährenden Einwurfschlitz besitzt, besser. Der Preis hierfür ist, daß in der Analogie nun eine mechanische Sicherungseinrichtung mehr vorkommt.

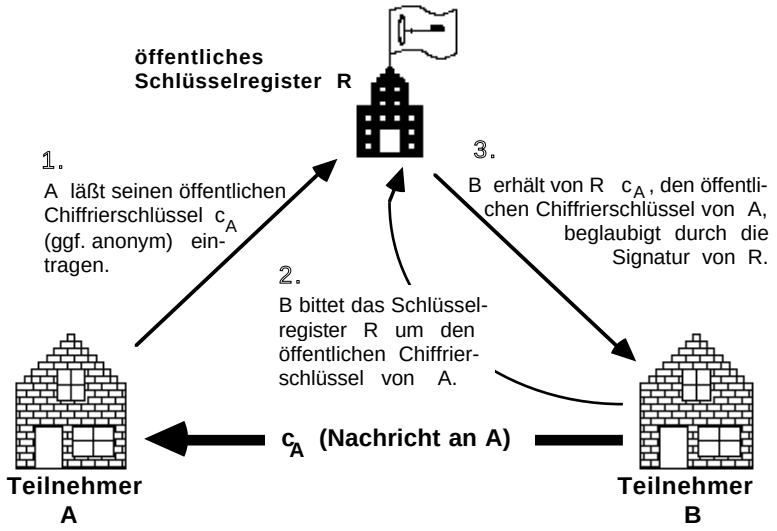


Bild 3-5: Schlüsselverteilung mit öffentlichem Register bei asymmetrischem Konzelationssystem

Man beachte, daß im Fall von Bild 3-5 das Schlüsselregister *R* (bzw. dessen Betreiber) die Nachricht nicht entschlüsseln kann. Allerdings gibt es eine verändernde Angriffsmöglichkeit für *R*, siehe Aufgabe 3-4 c).

Damit Teilnehmer *B* sicher sein kann, daß c_A auch wirklich zu Teilnehmer *A* gehört, beglaubigt das öffentliche Schlüsselregister durch seine Signatur nicht einfach c_A, sondern den *Zusammenhang* zwischen *A* und c_A. Andernfalls könnte ein Angreifer die Antwortnachricht (in Bild 3-5 Schritt 3.) des Schlüsselregisters abfangen und *B* einen anderen „beglaubigten" Schlüssel schicken, etwa einen, zu dem der Angreifer den Dechiffrierschlüssel kennt.

3.1.1.2 Authentikationssysteme, Überblick

Ein **symmetrisches Authentikationssystem** ist in Bild 3-6 dargestellt.

Hier wird also die Nachricht durch den kryptographischen Algorithmus links in Bild 3-6 nicht unleserlich gemacht, sondern es wird ein Prüfteil an sie angehängt. Dieser wird nach seiner englischen Bezeichnung oft MAC genannt. Der Empfänger kann anhand von *x* auch den richtigen MAC bilden und prüfen, ob der mit der Nachricht mitgekommene damit übereinstimmt.

Anmerkung: Wieder ist die Schreibweise $k(x)$ nur eine Abkürzung. Heißt der kryptographische Algorithmus *code*, so ist $MAC := code(k, x)$, und der Empfänger einer Nachricht (x, MAC) testet, ob $MAC = code(k, x)$.

Die Schlüsselverteilung kann wie bei symmetrischen Konzelationssystemen erfolgen. Es gibt auch die analogen Probleme, z.B. könnte eine einzelne Schlüsselverteilzentrale diesmal gefälschte Nachrichten unterschieben.

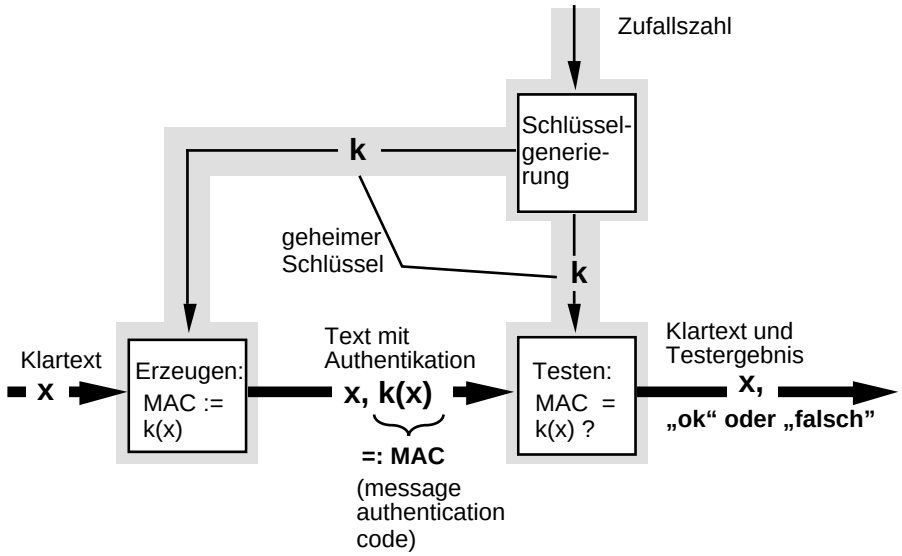


Bild 3-6: Symmetrisches Authentikationssystem
(≈ Glasvitrine mit Schloß, es gibt zwei gleiche Schlüssel, um etwas hineinzutun)

Asymmetrische Authentikationssysteme, also **digitale Signatursysteme**, vereinfachen zunächst die Schlüsselverteilung analog zu asymmetrischen Konzelationssystemen, s. Bild 3-7 und 3-8.

Ihr **Hauptvorteil** ist aber ein anderer: Jener ist, daß nun der Empfänger *B* einer signierten Nachricht von *A* jedem anderen, der auch *A*'s Schlüssel t_A kennt, beweisen kann, daß er diese Nachricht von *A* bekam. Dies geht bei einem symmetrischen Authentikationssystem nicht, selbst wenn z.B. vor Gericht die Schlüsselverteilzentrale bestätigen würde, welchen Schlüssel *A* und *B* hatten: Bei symmetrischen Authentikationssystemen kann ja *B* den MAC genausogut selbst erzeugt haben. Bei digitalen Signatursystemen ist jedoch *A* der einzige, der die Signatur erzeugen kann.

Deswegen sind digitale Signatursysteme unumgänglich, wenn man rechtlich relevante Dinge digital in sicherer Weise abwickeln will, z.B. bei digitalen Zahlungssystemen. Sie entsprechen dort der Funktion der handgeschriebenen Unterschrift in heutigen Rechtsgeschäften (daher natürlich der Name). Im Folgenden verwende ich die Begriffe „Unterschrift“ und „Signatur“ weitgehend synonym.

Dem gerade beschriebenen Hauptvorteil digitaler Signatursysteme bzgl. Integrität steht ein prinzipieller Nachteil bzgl. Vertraulichkeit gegenüber: Der Empfänger einer digitalen Signatur kann Nachricht und Signatur herumzeigen – und wenn der Schlüssel zum Testen der Signatur öffentlich ist, kann *jeder* die Gültigkeit der Signatur testen. Je nach Nachrichteninhalt kann dies demjenigen, der die Nachricht signiert hat, wesentlich unangenehmer sein, als wenn hinter seinem Rücken nur einfach seine Nachricht herumgezeigt werden könnte – beliebige Nachrichten erfinden und als nicht nachprüfbares Gerücht verbreiten kann jeder. Diese Prüfung der Integrität einer Nachricht hinter dem Rücken des Senders ist bei symmetrischen Authentikationssystemen nicht möglich: Bei ihnen könnte ja auch der Empfänger der Nachricht den Prüfteil (MAC) generiert haben, so daß Dritte die Authentizität der Nachricht bei symmetrischen Authentikationssystemen nicht prüfen können. Es sei hier schon darauf hingewiesen, daß es Authentikationssysteme gibt, die beide Vorteile kombinieren: Bei **nicht herumzeigbaren Signaturen** (undeniable[27] signatures) ist zur Prüfung der Signatur die aktive Mithilfe des Signierers nötig, so daß ein Prüfen hinter seinem Rücken nicht möglich ist. Andererseits muß dann festgelegt werden, unter welchen Umständen der (vorgebliche) Signierer zur aktiven Mithilfe bei der Prüfung „seiner“ Signatur verpflichtet ist. Denn sonst könnte der Empfänger

[27] Da *jede* digitale Signatur „undeniable“, d.h. unleugbar, nicht abstreitbar, sein soll, da sie andernfalls für den Rechtsverkehr unbrauchbar ist, halte ich den vom Erfinder dieses Signatursystemtyps – David Chaum – gewählten Namen für ungeschickt. Wenigstens im Deutschen möchte ich einen aussagekräftigen etablieren.

der signierten Nachricht im Konfliktfall mit dem Sender daraus, daß die Nachricht signiert ist, keine Vorteile ziehen, da er niemand hiervon überzeugen könnte. Mehr über diesen Typ digitaler Signatursysteme steht in §3.9.4.

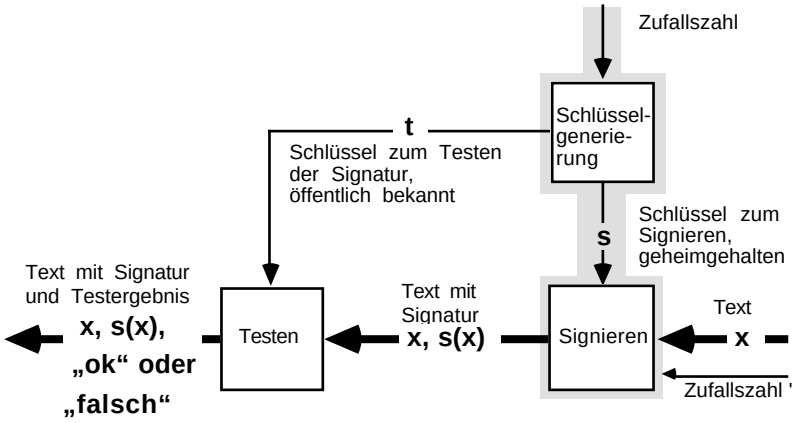


Bild 3-7: Signatursystem
(≈ Glasvitrine mit Schloß, es gibt nur einen Schlüssel, um etwas hineinzutun)

Anmerkung 1: In ausführlicher Schreibweise könnten die Signier- und Testalgorithmen *sign* und *test* heißen. Dann ist die Signatur $Sig := sign(s, x)$, und der Empfänger einer Nachricht (x, Sig) berechnet $test(t, x, Sig) \in \{ok, falsch\}$.

Anmerkung 2: Die Zufallszahl unten rechts wird unabhängig von der Zufallszahl oben gewählt. Wozu man sie braucht, wird erst in §3.5 einsichtig.

Man beachte, daß hier im Gegensatz zur symmetrischen Authentikation ein eigener Testalgorithmus benötigt wird, weil der Empfänger dort den Schlüssel *k* hat, mit dem er aus *x* selbst den korrekten Wert *MAC* bestimmen kann; hier hat er aber *s* nicht.

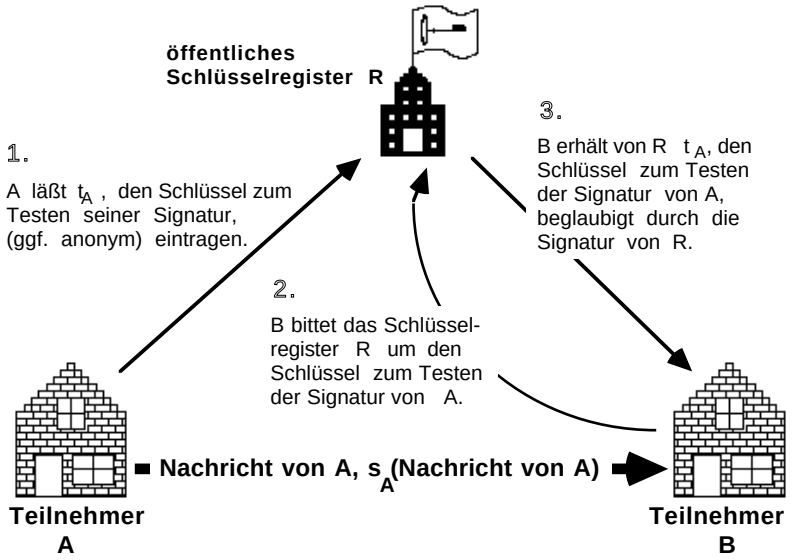


Bild 3-8: Schlüsselverteilung mit öffentlichem Register bei Signatursystem

Man beachte, daß die Möglichkeit, den Testschlüssel privat mit seinem Kommunikationspartner auszutauschen, nur in dem Fall genügt, daß man diesem Partner vertraut, d.h. das Signatursystem nur

als bequeme Form gegenseitiger Authentikation benutzt. Will man aber sicher sein, daß eine Signatur später ggf. vor Gericht anerkannt wird, muß man sich versichern, daß man den richtigen Testschlüssel hat, d.h. zumindest zur Kontrolle bei einem Schlüsselregister nachfragen.

Wieder hätte ein einzelnes Schlüsselregister Möglichkeiten zu verändernden Angriffen, vgl. Aufgabe 3-4 c).

Die Beglaubigung des öffentlichen Testschlüssels bezieht sich – wie bei den öffentlichen Chiffrierschlüsseln – nicht auf den Schlüssel allein, sondern auf den *Zusammenhang* zwischen Schlüssel und Teilnehmer. Ignorieren wir sonstige Information, etwa Zeitstempel, vgl. Aufgabe 3-4 f), so lautet die Beglaubigung des Schlüsselregisters R für t_A (in Bild 3-8 Schritt ③): $A,t_A , s_R(A,t_A)$. Diese Beglaubigung des öffentlichen Schlüssels wird **Zertifizierung des öffentlichen Schlüssels** genannt, die Instanz, die sie durchführt, Zertifizierungsinstanz (certification authority, CA).

Zusammenfassend besteht an die Schlüsselverteilung eine Forderung mehr als bei asymmetrischen Konzelationssystemen: Sie muß **konsistent** sein. Dies ist ein gemeinsames Ziel mehrerer Empfänger, bzgl. dessen Erfüllung sie der Zertifizierungsinstanz nicht vertrauen: Selbst wenn diese betrügt, müssen alle Empfänger denselben Wert t erhalten. Man stelle sich vor, ein Empfänger prüft mit einem Schlüssel t, später das Gericht aber mit einem ganz anderen t'! Daneben muß die Verteilung weiterhin integer sein, d.h., wenn die Zertifizierungsinstanz nicht betrügt, erhalten alle das von ihr zertifizierte t.

Die Konsistenz der Schlüsselverteilung sollte dadurch unterstützt werden, daß derjenige, der sich einen öffentlichen Schlüssel zertifizieren läßt, die Kenntnis des zugehörigen geheimen Schlüssels nachweist. Andernfalls könnte ein Angreifer fremde öffentliche Schlüssel einer Zertifizierungsinstanz vorlegen und sie sich als seine öffentlichen Schlüssel zertifizieren lassen. Zwar kann er die geheime Operation nicht ausführen, aber die Zertifizierung des gleichen öffentlichen Schlüssels für mehrere unterschiedliche Teilnehmer kann Verwirrung und Irritationen auslösen. Eine elegante Möglichkeit bei digitalen Signatursystemen, all dies zu vermeiden, ist, daß im Beispiel der Teilnehmer A nicht t_A zur Zertifizierung vorlegt, sondern den Zusammenhang zwischen seinem Namen A und seinem öffentlichen Schlüssel t_A zunächst mit dem zugehörigen geheimen Schlüssel s_A **selbst zertifiziert**. Er legt der Zertifizierungsinstanz also $A,t_A , s_A(A,t_A)$ vor und erhält als Beglaubigung $A,t_A , s_A(A,t_A)$, $s_R(A,t_A , s_A(A,t_A))$. Oftmals wird dies als Schlüsselzertifikat bezeichnet.

Zusammenfassend hat die **Zertifizierungsinstanz** also folgende Aufgaben:

1. Überprüfung der Identität des Teilnehmers. (Wie gründlich dies genau geschieht, legt die Zertifizierungsinstanz in einer sogenannten *certification policy* fest. Oftmals geschieht es durch Vorlage eines amtlichen Ausweises.)
2. Überprüfung des Antrags des Teilnehmers. (Wie gründlich dies genau geschieht, legt die Zertifizierungsinstanz ebenfalls in ihrer *certification policy* fest. Oftmals geschieht es durch einen schriftlichen Vertrag mit eigenhändiger, ggf. sogar notariell beglaubigter Unterschrift des Teilnehmers.)
3. Prüfung der Eindeutigkeit des Namens des Teilnehmers, so wie er im Schlüsselzertifikat erscheinen soll.
4. Prüfung, daß der Teilnehmer den zugehörigen geheimen Schlüssel anwenden kann.
5. Digitales Signieren von Namen des Teilnehmers und öffentlichem Schlüssel, ggf. unter Beifügung des öffentlichen, selbst wieder zertifizierten Schlüssels der Zertifizierungsinstanz.

3.1.2 Weitere Anmerkungen zum Schlüsselaustausch

In §3.1.1 wurden die wichtigen Klassen kryptographischer Systeme zusammen mit ihrer Schlüsselverteilung vorgestellt. Bevor in §3.1.3 genauer diskutiert wird, was unter der Sicherheit eines kryptographischen Systems zu verstehen ist, werden hier drei Aspekte des Schlüsselaustauschs vertieft.

3.1.2.1 Wem werden Schlüssel zugeordnet?

Interessant ist die Unterscheidung von drei möglichen Antworten:

1. einzelnen Teilnehmern,
2. Paarbeziehungen oder
3. Gruppen von Teilnehmern.

Bei *asymmetrischen* kryptographischen Systemen werden Schlüsselpaare normalerweise *einzelnen* Teilnehmern zugeordnet. Der geheimzuhaltende Schlüssel (*d* bzw. *s*) ist dann nur einem einzelnen Teilnehmer, der öffentliche Schlüssel (*c* bzw. *t*) potentiell allen bekannt, vgl. Bild 3-9.

Bei *symmetrischen* kryptographischen Systemen werden Schlüssel üblicherweise *zwei* Teilnehmern, d.h. einer *Paarbeziehung*, zugeordnet: Vernachlässigt man den für die Schlüsselverteilung pathologischen (d.h. extrem gebildeten, krankhaften) Fall, daß jemand Nachrichten an sich selbst schickt, vgl. Aufgabe 3-6 c), so müssen bei symmetrischen Systemen immer mindestens zwei Teilnehmer den Schlüssel kennen. Da symmetrische Schlüssel vertraulich bleiben müssen, sollten sie nur möglichst wenigen bekannt sein. Hieraus folgt, daß jeder symmetrische Schlüssel genau zwei Teilnehmern bekannt ist.

Außer unter speziellen Umständen werden Schlüssel also keinen Gruppen, sondern entweder einzelnen oder Paarbeziehungen zugeordnet.

kryptographische Systeme und ihre Ziele \ Kenntnis des Schlüssels	einer (Generierer)	zwei (beide Partner)	alle (öffentlich bekannt)
asymmetrisches Konzelationssystem: Vertraulichkeit	**d** Dechiffrierschlüssel		**c** Chiffrierschlüssel
symmetrisches kryptographisches System: Vertraulichkeit und / oder Integrität		**k** geheimer Schlüssel	
digitales Signatursystem: Integrität	**s** Signierschlüssel		**t** Testschlüssel

Bild 3-9: Übersichtsmatrix der Ziele und zugehörigen Schlüsselverteilungen von asymmetrischem Konzelationssystem, symmetrischem kryptographischem System und digitalem Signatursystem

3.1.2.2 Wieviele Schlüssel müssen ausgetauscht werden?

Möchten ***n* Teilnehmer** miteinander vertraulich kommunizieren, benötigen sie bei Verwendung eines **asymmetrischen** kryptographischen Systems hierfür *n* Schlüsselpaare. Möchten sie ihre Nachrichten digital signieren, sind *n* zusätzliche Schlüsselpaare nötig. Insgesamt werden also **2*n* Schlüsselpaare** benötigt. (Sollen Teilnehmer anonym bleiben können und ihre öffentlichen Schlüssel keine Personenkennzeichen sein, vgl. §6, so benötigt jeder Teilnehmer nicht nur zwei, sondern viele digitale Pseudonyme, sprich: öffentliche Schlüssel.)

Verwenden *n* Teilnehmer **symmetrische** kryptographische Systeme, so benötigen sie ***n* • (*n*-1) Schlüssel**, wenn jeder mit jedem gesichert kommunizieren können soll. Hierbei ist unterstellt, daß Schlüssel *Paaren von Teilnehmern* zugeordnet sind, vgl. §3.1.2.1, und für jede Kommunikations-

richtung ein separater Schlüssel verwendet wird, vgl. die Lösung von Aufgabe 3-7 e). Teilnehmer werden also oftmals jeweils gleich 2 symmetrische Schlüssel austauschen, so daß **Schlüsselaustausch** nur $n \cdot (n-1)/2$ mal stattfindet.

Insbesondere für IT-Systeme mit vielen Teilnehmern stellt sich die Frage, **wann** diese Schlüssel(paare) generiert und ausgetauscht werden sollen?

Bei asymmetrischen kryptographischen Systemen kann die Erzeugung problemlos vor dem eigentlichen Betrieb des IT-Systems oder – bei wechselnden Teilnehmern – der Registrierung des einzelnen Teilnehmers erfolgen. Die Verteilung der öffentlichen Schlüssel kann dann nach Bedarf erfolgen.

Bei symmetrischen kryptographischen Systemen kann aufgrund der wesentlich größeren Zahl der nötigen Schlüssel deren Erzeugung oder gar Verteilung nicht für alle möglichen Kommunikationsbeziehungen vorgenommen werden – zumindest für IT-Systeme mit vielen Teilnehmern. Bei ihnen werden daher nur die Schlüssel zwischen Teilnehmer und Schlüsselverteilzentrale(n) bei der Registrierung des Teilnehmers erzeugt und ausgetauscht. Alle anderen Schlüssel können und müssen dann bei Bedarf erzeugt und verteilt werden. Dies spart erheblichen Aufwand, da in IT-Systemen mit vielen Teilnehmern üblicherweise nur ein winziger Bruchteil aller möglichen Kommunikationsbeziehungen überhaupt jemals wahrgenommen wird.

3.1.2.3 Sicherheit des Schlüsselaustauschs begrenzt kryptographisch erreichbare Sicherheit

Da im folgenden meist von einem sicheren Ur-Schlüsselaustausch außerhalb des zu schützenden Kommunikationssystems ausgegangen wird, sei dessen Wichtigkeit hier deutlich hervorgehoben:

> Die durch Verwendung eines kryptographischen Systems erreichbare Sicherheit ist *höchstens* so gut wie die des Ur-Schlüsselaustauschs!

Deshalb ist es sinnvoll und – wie in §3.1.1.1 sowie den Aufgaben 3-4 c) und d) gezeigt – möglich, nicht auf einem einzigen Ur-Schlüsselaustausch aufzubauen, sondern auf mehreren. Hierdurch kann die erreichbare Sicherheit erheblich gesteigert werden.

***Anmerkung*: Auch die Sicherheit der Schlüsselgenerierung begrenzt die kryptographisch erreichbare Sicherheit:**
Ein geheimer Schlüssel (*k*, *d*, oder *s*) kann höchstens so geheim sein wie die *Zufallszahl*, aus der er erzeugt wird. Diese muß – wenn irgend möglich – also beim zukünftigen Inhaber des Schlüssels erzeugt werden und darf für einen Angreifer nicht zu raten sein. Auf keinen Fall darf man Datum und Uhrzeit als *Zufallszahl* wählen oder die einfache Funktion *random* seiner Programmierumgebung benutzen. Gute Zufallszahlenerzeugung ist nicht trivial.

Zum einen kann man **physikalische Phänomene** verwenden. Hierzu braucht man Spezialhardware, die nicht jeder persönliche Rechner hat. Beispiele sind Rauschdioden, radioaktive Prozesse oder Turbulenzen im Lüfter eines normalen Rechners [GlVi_88, DaIF_94].

Braucht man nur wenige echte Zufallszahlen, können sie auch vom **Benutzer** kommen. Erstens kann dieser würfeln. Das ist etwas mühsam, aber für die Erzeugung eines Signierschlüssels, mit dem man wirklich wichtige Dinge tun kann, vielleicht immer noch die beste Idee. Zweitens kann man den Benutzer einfach „zufällig“ tippen lassen und die Eingabe im Rechner komprimieren. Drittens werden manchmal statt der Eingabezeichen die niederwertigen Bits der Abstände zwischen den Tastendrücken genommen, weil man hofft, daß diese zufälliger sind. Zumindest in Kombination mit der zu niedrigen Abtastrate mancher Betriebssysteme ist dies aber problematisch.

Vertraut man keinem vorhandenen Verfahren ganz, sollte man mehrere Zufallszahlen nach verschiedenen Verfahren erzeugen und ihr **XOR** verwenden (Bild 3-10). Ist auch nur eine zufällig

und geheim, so ist bewiesenermaßen auch das XOR zufällig und geheim. Insbesondere sollte man so vorgehen, wenn staatliche Stellen an der Zufallszahlenerzeugung beteiligt werden möchten, weil sie befürchten, daß manche Benutzer zu schlechte Zufallszahlen eingeben, denn man kann andererseits nicht verlangen, daß Benutzer fremden Zufallszahlen vertrauen.

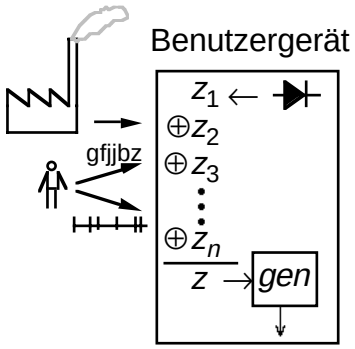


Bild 3-10 Erzeugung einer Zufallszahl *z* für die Schlüsselgenerierung: XOR aus einer im Gerät erzeugten, einer vom Hersteller gelieferten, einer vom Benutzer gewürfelten und einer aus Zeitabständen errechneten

3.1.3 Grundsätzliches über Sicherheit

3.1.3.1 Angriffsziele und -Erfolge

Bei der Etablierung eines kryptographischen Systems können 4 Phasen unterschieden werden:

1. Definition des kryptographischen Systems
2. Wahl des Sicherheitsparameters
3. Wahl des Schlüssel(paare)s
4. Bearbeiten von Nachrichten

Während mit allgemeinen Aufwandsparametern ein Brechen schon nach der 1. Phase untersucht werden kann, lohnt sich eine Untersuchung für konkreten vom Angreifer erbringbaren Aufwand erst nach der 2. Phase.

Bei den folgenden Angriffen ist unterstellt, daß sie nach der 3. oder auch erst nach der 4. Phase stattfinden. In *absteigender Reihenfolge der Angriffserfolge* wird unterschieden:

a) Finden des Schlüssels (**vollständiges Brechen**, total break)

b) Finden eines zum Schlüssel äquivalenten Verfahrens (**universelles Brechen**, universal break)

c) Brechen (also Fälschen bzw. Entschlüsseln) nur für manche einzelne Nachrichten[28] (**nachrichtenbezogenes Brechen**)

   c1) für eine selbst gewählte[29] Nachricht (**selektives Brechen**, selective break)[30] bzw.

---

28 Beim „Fälschen“ ist die „einzelne Nachricht“ üblicherweise ein Klartext, beim „Entschlüsseln“ ein Schlüsseltext.

29 Bei Konzelationssystemen ist dies typischerweise eine abgehörte verschlüsselte Nachricht, deren Klartext den Angreifer interessiert, der sie also in diesem Sinne auswählt. Denn generierte der Angreifer den Schlüsseltext selbst, dann verriete ihm der zugehörige Klartext nichts über andere. Bei Authentikationssystemen hingegen kann er einen Klartext zum Authentisieren völlig frei wählen. Dessen Authentikation kann ihm gegenüber anderen nützen.

30 Ein klarer Unterschied zum universellen Brechen besteht nur, wenn das selektive Brechen einen aktiven Angriff beinhaltet, siehe §3.1.3.2.

c2) für irgendeine Nachricht (**existentielles Brechen**, existential break).[31]

Es ist klar, daß Brechen im Sinne von a), b) und c1) nicht akzeptabel ist. Schließt man auch c2) aus, ist man auf der sicheren Seite. Es gibt aber schwächere kryptographische Systeme, wo man nur hofft, daß die sinnvollen und den Angreifer interessierenden Nachrichten in der Gesamtnachrichtenmenge so selten sind, daß er mit seinem Erfolg im Sinne von c2) nie eine davon erwischt.

Bei *Konzelationssystemen* kann bzgl. b) und c) noch jeweils unterschieden werden:

1. **Komplette Entschlüsselung**, d.h. der Angreifer erfährt den ganzen Klartext.
2. **Partielle Information:** Der Angreifer erfährt nur manche Eigenschaften, z.B. einzelne Bits oder die Quersumme des Klartextes.

3.1.3.2 Angriffstypen

Nach *zunehmender Stärke*:

a) **Passiv**. Bei Konzelationssystemen heißt das, daß der Angreifer die ganze Zeit nur zuschaut. Bei Authentikationssystemen heißt es, daß der Angreifer solange nur zuschaut, bis er eine Nachricht zu fälschen versucht.
   Bei Konzelationssystemen unterscheidet man noch:
   a1) **Reiner Schlüsseltext-Angriff** (ciphertext-only attack): Der Angreifer sieht die ganze Zeit nur Schlüsseltexte.
   a2) **Klartext-Schlüsseltext-Angriff** (known-plaintext attack): Der Angreifer sieht zu einigen Schlüsseltexten auch die zugehörigen Klartexte, z.B. weil manche Nachrichten nach einer Weile veröffentlicht werden. In abgeschwächter Weise kommt es vor, wenn der Angreifer Teile der Klartexte mit großer Wahrscheinlichkeit raten kann (z.B. anhand der Reaktion des Empfängers oder gerade geschehener Ereignisse beim Sender, oder weil es typische Briefanfänge sind); so wurden viele klassische Konzelationssysteme gebrochen.

b) **Aktiv**. Der Angreifer greift schon vor dem eigentlichen Brechen in das System ein, z.B. bringt er einen Unterzeichner dazu, ihm zunächst einmal einige harmlos aussehende Nachrichten zu unterzeichnen.

   Man kann noch unterscheiden, bei welchem Kommunikationspartner der Angreifer angreift: Bei asymmetrischen Systemen lohnt sich ein Angriff natürlich nur auf den Partner mit Geheimnis; Verschlüsseln bzw. Signaturen testen kann der Angreifer ja selbst.

   Beim *symmetrischen Konzelationssystem* unterscheidet man:
   b1) Klartext → Schlüsseltext (**gewählter Klartext-Schlüsseltext-Angriff**, chosen-plaintext attack): Der Angreifer kann Klartexte wählen, die der Sender dann verschlüsselt verschickt. Bsp.: Er besucht als Kunde eine Bankfiliale und führt gewisse Transaktionen durch, die dann verschlüsselt an die Zentrale gesendet werden.
   b2) Schlüsseltext → Klartext (**gewählter Schlüsseltext-Klartext-Angriff**, chosen-ciphertext attack): Der Angreifer läßt dem Empfänger Schlüsseltexte zukommen[32] und sieht die Entschlüsselungen. Bsp.: Der Angreifer ist Mitarbeiter in der Bankzentrale, soll aber

---

31 Bei Konzelationssystemen ist nur *selektives* nachrichtenbezogenes Brechen sinnvoll – denn es ist sinnlos, irgendwelche Chiffretexte zu entschlüsseln, die niemand geschickt hat. Letzteres genau tut existentielles Brechen.

32 Damit der Empfänger beim Entschlüsseln den passenden Schlüssel verwendet, muß der Angreifer hierzu im allgemeinen einen passenden, von ihm also gefälschten Absender angeben (können). Denn würde der Empfänger bei einem symmetrischen Konzelationssystem beim Entschlüsseln den mit dem Angreifer ausgetauschten Schlüssel verwenden, erführe der Angreifer durch seinen aktiven Angriff nichts - mit dem ihm bekannten Schlüssel könnte er die Aktion auch selbst durchführen.

nicht den Schlüssel wissen. Nun schickt er mit fingierter Absenderangabe seltsame Schlüsseltexte, und beobachtet, was der Entschlüsselungsrechner damit anfängt.

Beim *asymmetrischen Konzelationssystem* ist entsprechend obiger Bemerkung nur b2) relevant.

Beim *symmetrischen Authentikationssystem* kann

b1) Klartext → MAC (**gewählter Klartext-MAC-Angriff**, chosen-plaintext attack[33]) und

b2) MAC → (ein möglicher[34]) Klartext (**gewählter MAC-Klartext-Angriff**, chosen-MAC attack)[35] unterschieden werden.

Beim *digitalen Signatursystem* ist nur b1) Klartext → Signatur (**gewählter Klartext-Signatur-Angriff**, chosen-plaintext attack) relevant.

Adaptivität:

- Nicht adaptiv (d.h. der Angreifer muß alle seine Nachrichten zu Beginn wählen)
- Adaptiv (d.h. der Angreifer kann zunächst einige Nachrichten wählen, dann je nach dem Ergebnis einige weitere, usw.)

Ein Beispiel möge den Sinn dieser Unterscheidung verdeutlichen: Ein Angreifer gehe mit einem Kasten Bier zu einem trinkfreudigen Fernmelder und gieße ihm kräftig solange ein, bis er auf's Klo muß. In aller Regel wird der Fernmelder, schon angetrunken, das Kryptogerät entgegen der Dienstanweisung nicht mit auf's Örtchen nehmen. Nun hat der Angreifer die Gelegenheit, schnell ein paar Texte mit dem Kryptogerät zu bearbeiten – aber auch bei anfangs gut gefüllter Fernmelderblase sicherlich nicht die Ruhe und Zeit, sich in Abhängigkeit von den Ausgaben des Kryptogerätes clevere neue Eingaben auszudenken. Unser Bierkasten-Toiletten-Angriff ist also ein nichtadaptiver aktiver Angriff. Ersetzen wir Bier durch Schlafmittel, so erhalten wir einen adaptiven aktiven Angriff – zumindest für Angreifer mit guten Nerven und Konzentrationsfähigkeit auch in ungemütlichen Umgebungen.

Wünschenswert ist, daß ein kryptographisches System selbst bei sehr starken Angriffen keine oder nur minimale Angriffserfolge erlaubt. Am besten bedient ist man also mit einem kryptographischen System, das selbst bei adaptiven aktiven Angriffen nicht einmal existentiell brechbar ist.

3.1.3.3 Zusammenhang zwischen beobachtendem/veränderndem und passivem/aktivem Angreifer

Während in §1.2.5 bei der informellen Definition der Angreiferattribute beobachtend/verändernd die *Erlaubnis* für Handlungen entscheidend ist, unterscheidet passiv/aktiv Handlungen unabhängig davon, ob sie dem Angreifer erlaubt oder verboten sind. Beobachtender und passiver Angreifer meinen also, ebenso wie verändernder und aktiver Angreifer, *nicht* genau dasselbe.

Beispielsweise kann ein beobachtender Angreifer ein passiver Angreifer sein, wenn er sich für Kommunikationsinhalte einer durch ein symmetrisches Konzelationssystem geschützten Kommunikationsbeziehung interessiert, an der er nicht teilnehmen darf. Oder er kann ein aktiver Angreifer sein, indem er – wie im Beispiel oben – als Kunde eine Bankfiliale besucht und erlaubte, aber von ihm gewählte Transaktionen durchführt, die dann verschlüsselt an die Zentrale gesendet werden.

Umgekehrt kann ein verändernder Angreifer ein passiver Angreifer sein, wenn er sich durch Umgehen der Zugangskontrolle und Brechen der Zugriffskontrolle in einem fremden IT-System unerlaubterweise Kopien der verschlüsselten Paßwortdatei verschafft (verändernder Angriff) und den

---

33 Bisher wird dieser englische Begriff in der Literatur nur bei Konzelationssystemen verwendet.

34 Bei kurzen MACs aber potentiell langen Klartexten müssen einzelnen MACs im Mittel viele Klartexte entsprechen.

35 Ein gewählter MAC-Klartext-Angriff ist zugegeben etwas theoretisch: Denn ein entwendetes Authentikationsgerät erwartet als Eingabe den Klartext und generiert dazu den MAC – und nicht umgekehrt. Aber es geht doch nichts über eine durchgehaltene Systematik ...

Klartext herauszubekommen versucht (passiver Angriff).[36] Ist der verändernde Angreifer aber ein legaler Benutzer des IT-Systems und sei das Kopieren der Paßwortdatei (verändernder Angriff) auch ihm verboten, so ist er als aktiver Angreifer zu betrachten, da er sein Paßwort beliebig ändern kann.

Die Beispiele sind in Bild 3-11 dargestellt.

Angreifer	passiv	aktiv
beobachtend	Interesse an Kommunikationsinhalten einer fremden Kommunikationsbeziehung ohne Eingriff in diese Beziehung	Erlaubte Wahl von Klartexten, die dann mit einem ihm unbekannten Schlüssel verschlüsselt übertragen werden und ihm zum Brechen des Kryptosystems dienen
verändernd	Zugriffskontrolle brechen und verschlüsselte fremde Datei zu entschlüsseln versuchen	Zugriffskontrolle brechen und verschlüsselte Datei mit partiell selbst gewählten Klartextwerten zu entschlüsseln versuchen

Bild 3-11 Kombinationen der Angreiferattribute beobachtend/verändernd und passiv/aktiv

3.1.3.4 Grundsätzliches über „kryptographisch stark“

Wenn möglich, wird man unter dem Gesichtspunkt Sicherheit des kryptographischen Systems *informationstheoretisch sichere* verwenden, also das in §3.2 beschriebene symmetrische Konzelationssystem „Vernam-Chiffre“ (one-time pad) oder die in §3.3 beschriebenen Authentikationscodes. Dann kann ein Angreifer so viel rechnen wie er will – es wird ihm nichts nützen.[37]

Drei Gründe können erzwingen, auf lediglich *komplexitätstheoretisch sichere* kryptographische Systeme auszuweichen:

- Der benötigte Sicherheitsdienst ist durch informationstheoretisch sichere kryptographische Systeme nicht zu erbringen, etwa eine digitale Signatur im strengen Sinn, vgl. §3.1.1.2.
- Das Risiko der unbefugten Kenntnisnahme der ausgetauschten Schlüssel ist höher als die zusätzliche Sicherheit eines informationstheoretisch sicheren kryptographischen Systems gegenüber einem alternativen asymmetrischen (und damit zwangsläufig nur komplexitätstheoretisch sicheren, s.u.).
- Der Aufwand des Austauschs der für informationstheoretisch sichere kryptographische Systeme nötigen, mit der Gesamtlänge der zu schützenden Nachrichten unbegrenzt wachsenden Schlüssel ist zu groß.

„Kryptographisch stark“ nennt man die nach den informationstheoretisch sicheren nächstsicheren Systeme. Man kann hier, unabhängig vom Systemtyp, schon einiges sagen, was Sicherheit im kryptographischen Sinne bedeutet:

---

36 In einem gewissen Sinne wird hier gemogelt, da sich „verändernd“ und „passiv“ auf unterschiedliche Dinge beziehen: „Verändernd“ auf das Beschaffen der Paßwortdatei und „passiv“ auf das Entschlüsseln der verschlüsselten Paßworte. Will man diese Verschiebung des Bezugs vermeiden, dann sind passive verändernde Angriffe genau solche, wo etwas geschehen sollte (verändernd), aber nicht geschieht (passiv). Beispielsweise verweigert ein Trojanisches Pferd innerhalb einer Komponente auf einmal den Dienst der Komponente an ihre Außenwelt (denial of service attack).

37 Die Sicherheit ist also (soweit es die algorithmischen Teile des Systems betrifft) absolut.

1) **Viele kryptographische Systeme sind im Prinzip brechbar**: Wenn Schlüssel der festen Länge l verwendet werden, und genügend Information zur Verfügung steht, daß der Schlüssel im Prinzip eindeutig ist, kann ein Angreiferalgorithmus im Prinzip immer alle 2^l Schlüssel durchprobieren und damit das System vollständig brechen. (Dies betrifft die meisten asymmetrischen Systeme, sowohl bzgl. Konzelation als auch Authentikation, weil i.allg. bei bekanntem c bzw. t nur ein d bzw. s möglich ist, und außer der Vernam-Chiffre auch symmetrische Konzelationssysteme bei Klartext-Schlüsseltext-Angriff mit genügend viel „Material“. Außerdem gilt analoges z.B. für das Fälschen der Signatur zu einer bestimmten Nachricht: Der Angreifer weiß, daß seine gefälschte Signatur *Sig* zur Nachricht x gut genug ist, wenn sie den Test *test*(t, x, *Sig*) besteht. Da es i.allg. eine Obergrenze für die Länge von Signaturen gibt, muß er wieder nur endlich viele *Sig*s probieren.)

   Dieses erschöpfende Durchprobieren erfordert aber **exponentiell** viele Operationen, ist also z.B. für $l > 100$ nicht realistisch.

   Es bedeutet aber, daß das Beste im Sinne der Komplexitätstheorie, was der Entwerfer kryptographischer Systeme erhoffen kann, ein für den Angreiferalgorithmus in l exponentieller Aufwand ist.

2) **Nicht nur asymptotische „worst-case“-Komplexität:**

Komplexitätstheorie	liefert hauptsächlich asymptotische Resultate.
"	behandelt hauptsächlich „worst-case“-Komplexität.

   Es ist schön zu wissen, wie sich die Sicherheit verhält, wenn der Sicherheitsparameter l (Verallgemeinerung der Schlüssellänge; praktisch nützlich) genügend groß ist. Beim praktischen Einsatz kryptographischer Systeme muß man die Werte aller Parameter aber konkret festlegen. Was einen dann interessiert ist: Wie sicher ist das System bei konkreten Parameterwerten? Oder anders formuliert: Wo beginnt „genügend groß“?

   Die „worst-case“-Komplexität ist für Sicherheit ungenügend, ebenso „average-case“-Komplexität. (Denn es soll ja nicht nur ein paar Schlüssel geben, die nicht gebrochen werden können, sondern die meisten sollen nicht gebrochen werden.)

   Man wünscht sich: Das Problem soll fast überall, d.h. bis auf einen verschwindenden Bruchteil der Fälle, schwer sein. (D.h. wenn ein braver Benutzer seinen Schlüssel zufällig wählt, soll die Wahrscheinlichkeit überwältigend sein, daß er nicht gebrochen wird.)

   Ist der Sicherheitsparameter l , so verlangt man:

$$\text{Wenn } l \to \infty, \text{ dann Brechwahrscheinlichkeit} \to 0.$$

   Man hofft: Die Brechwahrscheinlichkeit sinkt schnell, so daß man l nicht allzu groß machen muß.

3) **Wie definiert man leicht und schwer?** Im wesentlichen braucht man bei einem kryptographischen System 2 Komplexitätsklassen: Die Dinge, die die Benutzer tun müssen, müssen leicht gehen; hingegen muß das Brechen schwer sein. Formal faßt man dies meist durch den Unterschied zwischen polynomial und nicht polynomial, also:

Schlüsselgenerierung, Ver-/Entschlüsseln:	leicht	= polynomial in l
Brechen:	schwer	= nicht polynomial in l $\approx$ exponentiell in l

   Warum gerade diese Trennung? (Die Punkte b und c gelten ganz allgemein für die Komplexitätstheorie.)

   a) Schwerer als exponentiell geht nicht, siehe 1).

   b) Abgeschlossen: Einsetzen von Polynomen in Polynome ergibt Polynome, d.h. vernünftige Kombinationen polynomialer Algorithmen sind wieder polynomial (ohne daß man etwas Genaues rechnen muß).

c) Vernünftige Berechnungsmodelle (Turing-, RAM-Maschine) sind polynomial äquivalent, man muß sich also nicht auf ein genaues Maschinenmodell festlegen.

Für die Praxis würde ein Polynom von hohem Grad als untere Schranke für Angreiferalgorithmus auf RAM-Maschine reichen. (Und die leichten Algorithmen gehen in der Praxis meist höchstens bis l^3.)

Anmerkung zu: nicht polynomial in l $\approx$ exponentiell in l
Hier steht kein Gleichheitszeichen, da es zwischen polynomial (l^k mit festem k) und exponentiell (2^l) weitere Funktionen gibt, z.B. $2^{\sqrt{l}}$.

4) **Warum braucht man komplexitätstheoretische Annahmen?** (z.B. „Faktorisierung schwer“, s. §3.4.1)

Die Komplexitätstheorie kann bisher keine brauchbaren unteren Schranken beweisen.

Für Kenner: Genauer liegt das Brechen bei vielen solchen Systemen offensichtlich innerhalb von NP. (Im Prinzip heißt NP gerade: Wenn man eine Lösung geraten hat, kann man schnell testen, ob sie richtig ist. So war das z.B. in 1) mit der zu fälschenden Signatur.) Damit ist für solche Fälle klar, daß ein Beweis, daß das Brechen eines solchen Systems exponentiell ist, P≠NP implizieren würde. Mit solch einem Beweis ist also zunächst nicht zu rechnen, da seit vielen Jahren viele Wissenschaftler P≠NP zu beweisen oder zu widerlegen versuchen.

Innerhalb von NP sind keine passenden unteren Schranken bekannt, also würde es auch hier nichts nützen, wenn man sich z.B. mit einer unteren Schranke l^{100} zufriedengäbe.

5) **Was für Annahmen macht man gern?**
Je kompakter und je länger (und von je mehr Leuten) untersucht, desto vertrauenswürdiger. In gewisser Hinsicht werden also kryptographische Systeme, wenn sie nicht informationstheoretisch sicher sind, mit zunehmendem Alter entweder gebrochen oder vertrauenswürdiger. Allerdings muß man, schon wegen des Fortschritts der Rechnerleistungen, die Parameter l allmählich erhöhen. (Vorsicht, wenn Sachen lange geheim bleiben sollen, oder Signaturen für Jahre gültig, also auch unfälschbar, sein sollen!)

Ein NP-vollständiges Problem wäre natürlich besonders nett, aber man hat bisher keins entdeckt, das auch im Sinne von 2) hart wäre.

6) **Was, wenn sich Annahme als falsch herausstellt?**
   a) Andere Annahme treffen.
   b) Genauere Analyse, z.B. Berechnungsmodell genau fixieren (RAM- oder Turingmaschine? Wie viele Bänder bei der Turingmaschine? usw.) und dann untersuchen, ob Polynom von genügend hohem Grad.

7) **Beweisziel:** Wenn der Angreiferalgorithmus das kryptographische System brechen kann, dann kann er auch das als schwer angenommene Problem lösen.

3.1.3.5 Überblick über im folgenden vorgestellte kryptographische Systeme

Zu den durchgestrichenen Feldern in Bild 3-12: Mit den Spezifikationen aus den Bildern 3-4 bzw. 3-7 gibt es solche Systeme nicht, wie in Punkt 1) oben begründet wurde. (Allerdings gibt es mittlerweile ein System, das fast die Eigenschaften von digitalen Signaturen hat und informationstheoretisch sicher ist. Von der Effizienz her ist es aber derzeit wohl nicht praktikabel. Vgl. §3.9.3 und [ChRo_91].)

Das Feld 3 ist seit 1998 nicht mehr vollständig leer. Bis dahin war gegen aktive Angriffe kein praktikables und gegen adaptive aktive Angriffe überhaupt kein System bekannt ist [BlFM_88,

NaYu_90, Damg_92]. Das 1998 von Ronald Cramer und Victor Shoup veröffentlichte System CS [CrSh_98] ist allerdings nicht als äquivalent zu einem reinrassigen Standardproblem wie dem Faktorisieren großer Zahlen (vgl. §3.4.1) oder dem Ziehen diskreter Logarithmen (vgl. §3.9.1) bewiesen. Seine Sicherheit ist nur so stark wie das Diffie-Hellman-Entscheidungsproblem schwierig ist. Zwar ist auch diese Annahme nicht unüblich, sie ist aber eine stärkere Annahme als die Diffie-Hellman-Annahme und diese ist eine stärkere Annahme als die Diskreter-Logarithmus-Annahme. Aus diesen Gründen und da Systeme basierend auf dem diskreten Logarithmus nicht im Zentrum dieses Skriptes stehen, wird auf CS im folgenden nicht weiter eingegangen, sondern mit dem s^2-mod-n-Generator ein System beschrieben, das auf der Faktorisierungsannahme beruht.

Die Felder 10 und 11 sind leer, weil man keine solchen Systeme kennt. Im Gegensatz zu Feld 1 und 2 ist eine Erfindung aber nicht ausgeschlossen.

Sicherheit \ Systemtyp		Konzelation: sym.	Konzelation: asym.	Authentikation: sym.	Authentikation: asym.
		sym. Konzelationssystem	asym. Konzelationssystem	sym. Authentikationssystem	digitales Signatursystem
informationstheoretisch		Vernam-Chiffre (one-time pad)	1	Authentikationscodes	2
kryptographisch stark gegen …	aktiver Angriff	Pseudo-one-time-pad mit s^2-mod-n-Generator	3 CS	4	GMR
	passiver Angriff	5	System mit s^2-mod-n-Generator	6	7
wohluntersucht	Mathematik	8	RSA	9	RSA
	Chaos	DES	10	DES	11

kann es nicht geben

wird von bekanntem System majorisiert

Bild 3-12: Überblick über die folgenden kryptographischen Systeme

Zu den Teilmengenzeichen:

Horizontal: Jedes asymmetrische System läßt sich auch als symmetrisches nutzen, indem der Erzeuger des geheimen Schlüssels diesen seinem Partner mitteilt.

Vertikal: Jedes informationstheoretisch sichere System ist auch kryptographisch stark, und letztere sind automatisch wohluntersucht. Auch ist, was gegen aktive Angriffe sicher ist, erst recht gegen

passive sicher. (Aber Vorsicht: Es gilt nicht, daß jedes *einzelne* kryptographisch starke System besser als jedes einzelne wohluntersuchte sein muß; z.B. ist denkbar, daß der s^2-mod-n-Generator gebrochen wird und das auf einer vollkommen anderen Annahme beruhende DES nicht.)

Dies erklärt die restlichen leeren Felder: Dort kann man das rechts davon bzw. darüberstehende System nehmen, z.B. GMR für 4, 6, 7, 9 und 11, oder für 4, 6 und 9 auch Authentikationscodes. Allerdings gibt es bei schwächeren Anforderungen manchmal effizientere Systeme. Nur deshalb kann sich z.B. ein Pseudo-one-time-pad anstelle eines echten One-time-pads, RSA als Signatursystem statt GMR, oder DES statt eines Authentikationscodes lohnen.

Kryptographische Systeme, die nicht einmal wohluntersucht sind, wurden nicht aufgenommen. Der Begriff ist natürlich etwas umstritten. Insbesondere wurden hier auch solche Systeme nicht wohluntersucht genannt, bei denen mehrere ähnliche Systeme bereits gebrochen wurden. (Manche Leute verwenden aber trotzdem nichtlineare Schieberegister für symmetrische Konzelation; diese sind noch schneller als DES.)

Noch weniger Vertrauen kann man kryptographischen Systemen entgegenbringen, die nicht einmal veröffentlicht wurden, sondern die nur der Erfinder selbst und seine „Vertrauten“ untersucht haben.

3.1.4 Hybride kryptographische Systeme

Da symmetrische kryptographische Systeme sowohl in Hardware als auch in Software um 2 bis 3 Größenordnungen effizienter als asymmetrische implementiert werden können, werden asymmetrische Konzelationssysteme (und ggf. digitale Signatursysteme) meist nur dazu verwendet, einen Schlüssel eines symmetrischen kryptographischen Systems dem Partner vertraulich (und ggf. digital signiert) zu übermitteln. Mit dem übermittelten Schlüssel können dann die eigentlichen Daten wesentlich effizienter, d.h. schneller, verschlüsselt und/oder authentiziert werden. Das aus einem asymmetrischen Konzelationssystem, ggf. einem digitalen Signatursystem und einem symmetrischen kryptographischen System zusammen gebildete kryptographische System wird **hybrides kryptographisches System** genannt. Es verbindet das bequeme Schlüsselmanagement der ersteren kryptographischen Systeme mit der Effizienz des letzteren. Das hybride kryptographische System ist natürlich nur so sicher wie das schwächste zu seiner Bildung verwendete.

Ein hybrides kryptographisches System kann nicht nur wegen der höheren Effizienz verwendet werden, sondern auch aus Gründen einer effizienteren Nutzung des Übertragungskanals: lange Blöcke könnten zuviel Verschnitt verursachen; die Fehlerausbreitung des symmetrischen kryptographischen Systems könnte auf die Anwendung und den Kanal besser abgestimmt sein, vgl. §3.8.2. (Letzteres ist nur dann sinnvoll, wenn Authentizität für die Anwendung nicht wichtig ist.)

3.2 Vernam-Chiffre (one-time pad)

Dieser Abschnitt behandelt informationstheoretisch sichere symmetrische Konzelationssysteme. Sie bestehen also aus den in Bild 3-2 gezeigten Komponenten. Es zeigt sich, daß das einfachste Beispiel, die Vernam-Chiffre, so perfekt ist, daß man kein weiteres benötigt.

Grob heißt Sicherheit hier, daß ein Angreifer, der einen (beliebig langen) Schlüsseltext S sieht, dadurch nichts über den Klartext erfährt. Dies wird erreicht, indem man verlangt, daß es zu jedem möglichen Klartext x mindestens einen Schlüssel k gibt, so daß $k(x) = S$. (Das heißt: x ist tatsächlich ein möglicher Inhalt von S, und zwar gerade in dem Fall, wo der geheime Schlüssel k ist.) Damit die verschiedenen Klartexte nicht verschieden wahrscheinlich werden, wird in dem normalen Fall, daß alle Schlüssel gleichwahrscheinlich sind, noch verlangt, daß es zu jedem x gleichviele solche k gibt, meistens genau 1.

Am Beispiel der Vernam-Chiffre sieht man, daß diese Definition sehr natürlich ist.

Beispiel: Wir nehmen an, wir wollen 2 Bits verschlüsseln und haben 2 Schlüsselbits. Die Verschlüsselungsfunktion ist Addition mod 2 (= bitweises XOR). Nun sehe der Angreifer z.B. den Schlüsseltext *S*=01. Dies kann nun entweder der Klartext *x*=00 mit dem Schlüssel *k*=01 gewesen sein, oder *x*=01 und *k*=00, oder *x*=10 und *k*=11, oder *x*=11 und *k*=10, vgl. Bild 3-13. Alle Klartexte sind also gleich gut möglich.

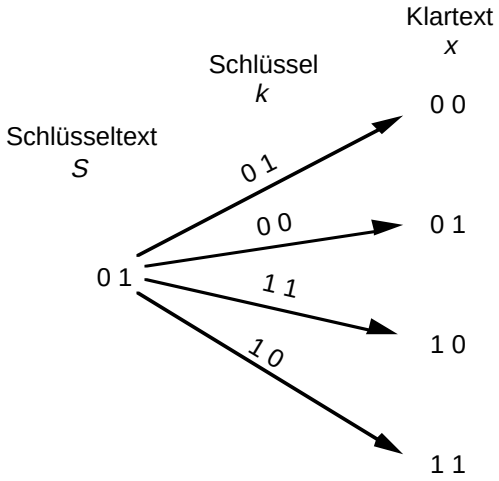


Bild 3-13: Schlüsseltext kann allen möglichen Klartexten entsprechen

Dies war die binäre Vernam-Chiffre für 2 Bit. Man kann dies auf beliebig viele Bits erweitern[38], und auf andere Additionen statt modulo 2. (Beispielsweise mod 26 für buchstabenweise Chiffrierung von Hand.)

Bild 3-14 zeigt links den in Bild 3-13 dargestellten Sachverhalt für alle möglichen Schlüsseltexte, wobei aus Platzmangel die Werte des Schlüssels weggelassen wurden.

Rechts in Bild 3-14 ist eine unsichere Chiffre dargestellt. Dies ist daran erkennbar, daß nicht jeder Schlüsseltext auf jeden Klartext abgebildet werden kann. Ein Angreifer, der z.B. den Schlüsseltext 10 beobachtet, erfährt dadurch, daß z.B. der Klartext 11 gerade nicht übertragen wird. (Natürlich ist auch die unsichere Chiffre in Bild 3-14 rechts besser als nichts. Wird der 1-bit-Schlüsselabschnitt jeweils nur einmal zum Ver- bzw. Entschlüsseln eines nichtredundant codierten[39] 2-bit-Textabschnitts verwendet, wird ein Angreifer nie erfahren, welcher Klartext genau übertragen wird. Er kann rechnen, solange er will. In diesem *eingeschränkten* Sinn ist dann auch diese Chiffre informationstheoretisch sicher: Der Angreifer kann zwar Information über den Klartext gewinnen, aber nicht genug, um ihn genau zu bestimmen. In seiner bahnbrechenden Arbeit über informationstheoretisch sichere Konzelation [Sha1_49] nennt Claude Shannon diese eingeschränkte informationstheoretische

---

38 Erinnert sei an eine Warnung aus §3.1: Eigentlich sollte perfekte Konzelation (und in gewissem Sinne ist die Vernam-Chiffre die bestmögliche Konzelation) unabhängig von der Wahrscheinlichkeitsverteilung des Klartextraumes *und der für den Klartextraum gewählten (Quell-)Codierung* funktionieren. Die *binäre* Vernam-Chiffre zeigt nun besonders deutlich, daß dies erschreckenderweise nicht der Fall ist: Werden die Klartextnachrichten *unär* codiert, dann ist die binäre Vernam-Chiffre vollkommen wirkungslos. Also müßte eigentlich zusätzlich zur Beschreibung der Vernam-Chiffre auch noch die Nachrichtenlänge (genauer: die Schlüsseltextlänge) a-priori festgelegt werden, damit sie nichts über den Nachrichteninhalt verraten kann. Dann bleibt nur noch das Problem, daß, wenn diese Länge zu kurz gewählt wird, lange Nachrichten in mehrere kurze aufgeteilt werden und dann die Nachrichtenhäufigkeit Information über den Nachrichteninhalt verraten könnte. Bei genügend groß gewählter Schlüsseltextlänge sollte dieses Problem aber nun wirklich ein vernachlässigbares Problem sein ...

39 Enthält der 2-bit-Textabschnitt genügend Redundanz, so kann die unsichere Chiffre vollständig gebrochen werden. Ist bekannt, daß die Klartexte 00 und 10 nicht vorkommen, dann verbirgt sich hinter dem Schlüsseltext 00 immer der Klartext 11, hinter 01 immer 01, hinter 10 immer 01 und hinter 11 immer 11.

Sicherheit *ideale Konzelation* (ideal secrecy) – eine sehr unglückliche Begriffsbildung. Die informationstheoretische Sicherheit, die in dieser Arbeit zugrunde gelegt und definiert wird, nennt Claude Shannon *perfekte Konzelation* (perfect secrecy). In neuerer Literatur wird diese perfekte Konzelation auch als *unbedingte Konzelation* (unconditional secrecy) bezeichnet.)

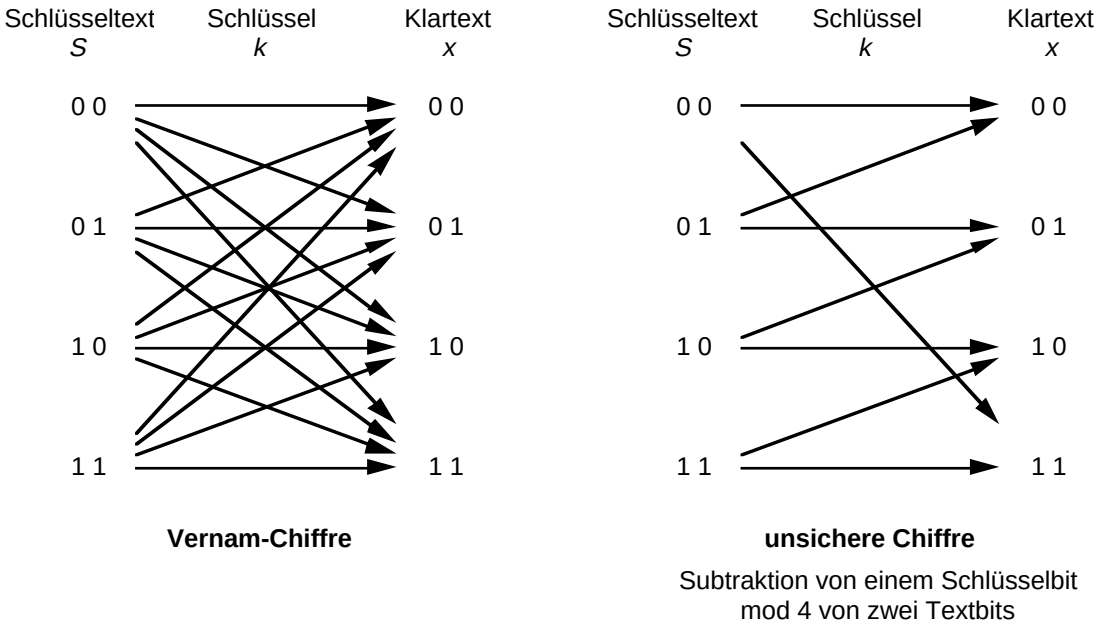


Bild 3-14: Jeder Schlüsseltext sollte allen möglichen Klartexten entsprechen können

Vernam-Chiffre allgemein: Will man die Vernam-Chiffre in einer Gruppe G benutzen und eine Zeichenkette der Länge n (von Zeichen aus G) verschlüsseln, so muß man vorweg als Schlüssel ebenfalls eine Zeichenkette der Länge n zufällig wählen und vertraulich austauschen[40], etwa $k = (k_1,\ldots,k_n)$. Das i-te Klartextzeichen x_i wird verschlüsselt als

$$S_i := x_i + k_i$$

Entschlüsselt werden kann es durch

$$x_i := S_i - k_i.$$

Für den Sicherheitsbeweis siehe Aufgabe 3-7 b). (Es ist klar, daß man jedes Klartextzeichen einzeln betrachten kann, da sie alle unabhängig verschlüsselt werden.) Auch ist klar, daß diese Chiffre selbst gegen aktive Angriffe sicher ist (vgl. Aufgabe 3-7 e)): Selbst wenn der Angreifer zunächst einige Klartext-Schlüsseltext-Paare seiner Wahl erhält, so erfährt er damit nur etwas über die dort verwendeten Schlüsselbits, die aber später (also wenn er wirklich etwas entschlüsseln will) nie wieder vorkommen[41].

---

40 Um die Frage „Was bringt denn die Vernam-Chiffre überhaupt, wenn ich, um eine Nachricht vertraulich austauschen zu können, vorher einen Schlüssel *gleicher* Länge vertraulich austauschen muß? Da könnte ich doch gleich die Nachricht vertraulich austauschen!“ gleich zu beantworten: Bzgl. der Länge bringt die Vernam-Chiffre gar nichts, aber der Schlüssel kann zu einem *beliebigen*, d.h. für beide Kommunikationspartner günstigen, Zeitpunkt ausgetauscht werden, *bevor* die Nachricht vertraulich ausgetauscht werden soll. Wenn die Situation für vertrauliche Kommunikation günstig ist, wird die Nachricht üblicherweise noch nicht vorliegen. Also wird die günstige Situation zur vertraulichen Kommunikation „auf Vorrat“, sprich Schlüsselaustausch, genutzt.

41 „Nie wieder vorkommen“ bezieht sich auf die konkreten, gegenständlich gedachten Schlüsselbits, nicht etwa ihre Werte. Denn natürlich kann, ja muß jedes Bit, wenn nur sein Wert als das Charakteristische betrachtet wird, in einer zufällig gewählten, unendlichen Bitfolge immer wieder vorkommen. Entsprechendes gilt nicht nur für einzelne Bits, sondern auch für jede endliche Bitkette.

Bezüglich der Berechnung ist die Vernam-Chiffre einfach und schnell, und sie verursacht auch keine Nachrichtenexpansion. Ihr einziger Nachteil ist die Länge des Schlüssels: Er ist genauso lang wie die Nachricht. Daher gilt die Vernam-Chiffre für viele Zwecke als unpraktikabel. (Mit Verbesserung der Speichermedien wird sie aber immer besser einsetzbar, jedenfalls in Fällen, wo die Teilnehmer vorweg z.B. Disketten austauschen können.)

Man kann leicht einsehen, daß für informationstheoretische Sicherheit die Schlüssel so lang sein müssen: Sei K die Schlüsselmenge, X die Klartextmenge und S die Menge der mindestens einmal auftretenden Schlüsseltexte. Zunächst muß $|S| \geq |X|$ gelten, denn bei festem Schlüssel müssen alle Klartexte verschieden verschlüsselt werden, damit man eindeutig entschlüsseln kann. Nun haben wir für die Sicherheit verlangt, daß für alle *S* und *x* ein *k* existiert mit $k(x) = S$. Bei festem *x* muß dies für jeden Schlüsseltext *S* ein anderer Schlüssel sein. Also gilt $|K| \geq |S|$, und insgesamt $|K| \geq |X|$.

Es muß also mindestens so viele Schlüssel wie Klartexte geben. Sind die Klartexte geschickt codiert[42], so müssen also die Schlüssel genauso lang sein.

Anmerkung 1: Mit den gerade eingeführten Bezeichnungen lautet die Definition für informationstheoretische Sicherheit für den Fall, daß alle Schlüssel mit gleicher Wahrscheinlichkeit gewählt werden, also kurz:

$$\forall S \in S\ \exists const \in \mathbb{N}\ \forall x \in X:\ |\{k \in K \mid k(x) = S\}| = const. \qquad (1)$$

Da es sich hier um eine zentrale Definition handelt, sei sie noch etwas problematisiert und dadurch erläutert:

Da $\mathbb{N} = \{1, 2, 3, 4, ...\}$, gilt: $const \geq 1$.

Die Konstante *const* darf in obiger Definition vom konkreten Schlüsseltext *S* abhängen. Bei vielen informationstheoretisch sicheren Konzelationssystemen, etwa dem One-time pad, tut sie das nicht. Dort ist, wie bereits erwähnt, $const = 1$. Man mag deshalb versucht sein, die Reihenfolge der Quantoren in der Definition zu tauschen: $\exists const \in \mathbb{N}\ \forall S \in S\ \forall x \in X:\ |\{k \in K \mid k(x) = S\}| = const$. Dadurch wird die Definition aber restriktiver, ohne daß ihr genügende Konzelationssysteme sicherer werden. Also bleibt man geschickterweise bei der schwächeren Definition. Aus den gleichen Gründen nimmt man auch nicht folgende noch stärker vereinfachte Definition: $\forall S \in S\ \forall x \in X:\ |\{k \in K \mid k(x) = S\}| = 1$.

Nun seien auch noch – zugegeben eher exotische – Konzelationssysteme angegeben, die der allgemeinen Definition (1) genügen, den engeren aber nicht:

Beispiel für $const = 2$: Wir erweitern den Schlüsselraum des One-time pads um ein Bit, das bei der Ver- und Entschlüsselung überhaupt nicht verwendet wird. Dann kann jedem Schlüsseltext jeder Klartext jeweils durch 2 Schlüssel zugeordnet werden, die sich genau im nicht verwendeten Bit unterscheiden.

Beispiel für *const* abhängig von *S*: Wir verdoppeln die Schlüssellänge des One-time pads, verwenden aber jedes zweite Schlüsselbit bei der Ver- und Entschlüsselung überhaupt nicht. Dann kann jedem Schlüsseltext der Länge *l* Bit jeder Klartext jeweils durch 2^l Schlüssel zugeordnet werden, die sich genau in den *l* nicht verwendeten Bits unterscheiden. Also ist *const* vom Schlüsseltext, genauer: seiner Länge, abhängig. Es gilt dann: $const = 2^l$.

Anmerkung 2: Das informationstheoretische, also absolute, an der Sicherheit ist, daß wir uns in keiner Weise darum gekümmert haben, ob etwas vielleicht schwierig auszurechnen ist: Wir haben den schlimmsten Fall angenommen, daß der Angreifer eine genaue Tabelle hätte, welche *k*, *x* und

---

42 Um einem Mißverständnis vorzubeugen: Hier hat „Codieren“ nichts mit Vertraulichkeit oder Authentikation zu tun, sondern es geht nur darum, daß mittels Bitketten der Länge *l* insgesamt 2^l Werte unterschieden, das bedeutet 2^l Klartexte geschickt codiert – oder anders ausgedrückt: repräsentiert –, werden können. Entsprechendes gilt auch für die Werte der Schlüssel, so daß die entsprechende Länge der Schlüssel(codierung) resultiert.

S zusammengehören. Die späteren Konzelationssysteme mit kurzen Schlüsseln werden alle in diesem Sinn nicht sicher sein; z.B. gibt es bei vielen asymmetrischen Konzelationssystemen zu jedem *c* nur genau ein mögliches *d* (vgl. Bild 3-4), und damit, weil der Angreifer *c* kennt, auch zu jedem Schlüsseltext *S* nur genau einen möglichen Klartext *x*. Dort muß man sich also voll und ganz darauf verlassen, daß *x* schwierig auszurechnen ist.

Anmerkung 3: Die übliche Definition informationstheoretisch sicherer Konzelation [Sha1_49] holt etwas weiter aus; unsere kann man dann als äquivalent beweisen. (Und in konkreten Beweisen verwenden auch andere Autoren meist die obige einfachere Formulierung (1).) Sie besagt: Egal, was der Angreifer a priori an Information über den Klartext hat, er gewinnt durch das Sehen des Schlüsseltextes keine Information hinzu.

Die a-priori-Information wird durch eine Wahrscheinlichkeitsverteilung dargestellt, die die Klartexte aus seiner Sicht haben (z.B. bei Briefanfängen W(Liebe) = 0,4, W(Sehr) = 0,4, W(xrq,q) = 0,0001). Auch die Schlüssel haben Wahrscheinlichkeiten und werden unabhängig vom Klartext gewählt. Damit haben auch die Schlüsseltexte Wahrscheinlichkeiten. Die a-posteriori-Wahrscheinlichkeit eines Klartextes *x*, wenn der Angreifer den Schlüsseltext *S* gesehen hat, ist die bedingte Wahrscheinlichkeit von *x*, wenn man *S* kennt, W(*x*|*S*). Die Sicherheitsforderung ist also

$$\forall S \in \mathcal{S}\ \forall x \in \mathcal{X}:\ \mathrm{W}(x|S) = \mathrm{W}(x). \qquad (2)$$

Dies bedeutet, daß für jemand, der den Schlüssel nicht kennt, Klartext *x* und Schlüsseltext *S* stochastisch unabhängig sind.

Zur Äquivalenz der Definitionen (1) und (2): Nach der Bayesschen Formel ist

$$\mathrm{W}(x|S) = \frac{\mathrm{W}(x)\bullet\mathrm{W}(S|x)}{\mathrm{W}(S)}.$$

(2) ist also äquivalent zu[43]

$$\forall S \in \mathcal{S}\ \forall x \in \mathcal{X}:\ \mathrm{W}(S|x) = \mathrm{W}(S). \qquad (3)$$

Wir zeigen, daß dies äquivalent ist zu

$$\forall S \in \mathcal{S}\ \exists\, const' \in \mathbb{R}\ \forall x \in \mathcal{X}:\ \mathrm{W}(S|x) = const'. \qquad (4)$$

(3) $\Rightarrow$ (4) ist klar mit *const'* := W(*S*). Umgekehrt zeigen wir *const'* = W(*S*):

$$\mathrm{W}(S) = \sum_x \mathrm{W}(x)\bullet\mathrm{W}(S|x) = \sum_x \mathrm{W}(x)\bullet const' = const' \bullet \sum_x \mathrm{W}(x) = const'.$$

(4) sieht (1) schon sehr ähnlich: Allgemein ist W(*S*|*x*) = W({*k*|*k*(*x*) = *S*}), und wenn alle Schlüssel gleichwahrscheinlich sind, W(*S*|*x*) = |{*k*|*k*(*x*) = *S*}| / |$\mathcal{K}$|. Dann ist (4) äquivalent (1) mit *const* = *const'*•|$\mathcal{K}$|.

3.3 Authentikationscodes

Authentikationscodes sind informationstheoretisch sichere symmetrische Authentikationssysteme. Sie bestehen also aus den in Bild 3-6 gezeigten Komponenten. Bevor wir ein Beispiel geben (Bild 3-15), betrachten wir informell, was informationstheoretische, also absolute, Sicherheit in diesem Fall heißt.

Sicherheit heißt, daß, wenn ein Angreifer eine Nachricht unterzuschieben versucht, der Empfänger anhand des (nicht passenden) MAC bemerkt, daß sie falsch ist.

Dies kann nicht mit Wahrscheinlichkeit 1 gelten: Irgendeinen korrekten MAC muß es geben, und der Angreifer kann immer Glück haben und gerade diesen erraten. Genauer gilt, wenn der Wertebereich für die MACs *W* Elemente hat, daß man mit rein zufälligem Raten mit der Wahrscheinlichkeit

---

43 Wir betrachten nur solche Schlüsseltexte *S*, die vorkommen können, so daß gilt: W(*S*) > 0.

$1/W$ den richtigen MAC erwischt. (Insbesondere also bei σ-Bit-MACs mit Wahrscheinlichkeit mindestens $2^{-\sigma}$.)

Die beste erreichbare Sicherheit ist also die, daß es für einen Angreifer keine bessere Möglichkeit gibt, als rein zufällig zu raten.

Anmerkung: Beachte aber, daß der Angreifer hier nicht lokal prüfen kann, ob er richtig geraten hat, d.h. es trifft nicht der Fall aus §3.1.3.4 1) zu. Hier werden wir beweisbar erreichen, daß ein Fälschungsversuch fast sicher erkannt wird. Im Gegensatz dazu könnte ein Angreifer mit beliebig viel Berechnungsfähigkeit bei einem digitalen Signatursystem im Sinne von Bild 3-7 lauter solche Signaturen erzeugen, die wirklich richtig aussehen, weil er ja den Test kennt, den sie bestehen müssen. (Vgl. aber §3.9.2 und §3.9.3.)

Authentikationscodes haben also einen prinzipiellen Sicherheitsvorteil gegenüber digitalen Signaturen; dafür sind sie unpraktischer, s. §3.1.1.2.

Beispiel 1: Bevor wir die Sicherheitsaussage formalisieren, betrachten wir Bild 3-15, das einen kleinen Authentikationscode darstellt (aus der Sicht des Angreifers). Hier soll 1 Nachrichtenbit $\in$ {H, T} (von Head, Tail) authentisiert werden; der Schlüssel ist 2 Bit lang, und der MAC 1 Bit. Wie bei allen folgenden Systemen werden alle Schlüssel mit gleicher Wahrscheinlichkeit gewählt.

Die möglichen Ergebnisse stehen oben; in der Tabelle sieht man, bei welchem Klartext sich dieses Ergebnis ergibt. (Man kann diesen Code übrigens sehr viel kürzer darstellen, s. Aufgabe 3-8 b).) Wie bei der Vernam-Chiffre wird jeder Schlüsselabschnitt (hier jeweils die beiden Bits) nur einmal verwendet.

		Text mit MAC			
		H,0	H,1	T,0	T,1
Schlüssel	00	H	–	T	–
	01	H	–	–	T
	10	–	H	T	–
	11	–	H	–	T

Bild 3-15: Ein kleiner Authentikationscode

Anhand der MAC-Länge wissen wir, daß das Erkennen von Fälschungen in diesem Beispiel höchstens mit Wahrscheinlichkeit 1/2 möglich ist. Wir wollen nun zeigen, daß tatsächlich jede Fälschung mit Wahrscheinlichkeit 1/2 erkannt wird.

Wir betrachten nur den Fall, daß der Angreifer gerne die Nachricht T unterschieben würde. Wenn der richtige Absender noch gar nichts gesendet hat, sind alle 4 Schlüssel gleichwahrscheinlich, und bei je zweien hat T als MAC 0 bzw. 1. Mit Wahrscheinlichkeit 1/2 wählt der Angreifer also den falschen Wert.

Das interessante an einem Authentikationscode ist aber, daß die Sicherheit auch dann noch gilt, wenn der richtige Absender die Nachricht H mit ihrem richtigen MAC sendet, und der Angreifer diese abfängt und durch T zu ersetzen versucht.

1. Fall: Der Angreifer fängt H,0 ab. Dann sind noch die Schlüssel k=00 und k=01 möglich. Bei k=00 muß der Angreifer 0 als MAC für T wählen, bei k=01 hingegen 1. Wieder rät also der Angreifer mit Wahrscheinlichkeit 1/2 falsch.
2. Fall: Der Angreifer fängt H,1 ab. Analog … (Weiter s. Aufgabe 3-8 b)).

Genauere Sicherheitsdefinition: Eine präzisere Definition, wann ein Authentikationscode *code* sicher mit Fehlerwahrscheinlichkeit ε ist, beschreibt genau das, was wir im obigen Beispiel getan haben:

„Selbst wenn ein Angreifer eine richtige Nachricht (x, *MAC*) sieht, und diese beseitigt und statt dessen versucht, den Klartext y zu schicken, gilt, egal wie geschickt er dazu *MAC'* wählt: Die Fälschung wird mit Wahrscheinlichkeit $\geq 1-\varepsilon$ erkannt, d.h. die Wahrscheinlichkeit für *MAC'* = *code*(k, y) ist höchstens ε (wobei k der korrekte geheime Schlüssel ist)."

Wir müssen nur noch präzis angeben, worüber eigentlich die Wahrscheinlichkeit gebildet wird. Dies ist genau die Menge der aus Angreifersicht noch möglichen Schlüssel. (Wie im Beispiel im 1. Fall die Menge aus 00 und 01.) Die Angreifersicht besagt, daß nur noch solche Schlüssel k in Frage kommen, für die *MAC* = *code*(k, x) ist.

Ganz formal also: Ein Authentikationscode *code* mit Schlüsselmenge K (und Wahrscheinlichkeitsverteilung W auf K) heißt sicher mit Fehlerwahrscheinlichkeit ε, wenn gilt: $\forall\, x\ \forall\, MAC \in code(K, x)\ \forall\, y \neq x\ \forall\ MAC'$:

$$W(MAC' = code(k, y) \mid MAC = code(k, x)) \leq \varepsilon.$$

Beispiel 2 (Verbesserung des Codes aus Bild 3-15): Will man die Fehlerwahrscheinlichkeit des Codes aus Bild 3-15 von 1/2 auf $1/2^{\sigma}$ senken, so kann man an ein Nachrichtenbit σ MACs im bisherigen Sinne hängen, etwa

$$x, MAC_1, \ldots, MAC_{\sigma}.$$

Man muß also vorher 2σ Schlüsselbits ausgetauscht haben. Jeden Einzel-MAC rät der Angreifer nur mit Wahrscheinlichkeit 1/2 richtig, und sie sind alle unabhängig, folglich rät er nur mit Wahrscheinlichkeit $(1/2)^{\sigma}$ jedesmal richtig.

Will man den Code auf Nachrichten der Länge l Bit anwenden, so braucht man $l \cdot 2\sigma$ Schlüsselbits, und jedes Bit wird einzeln in obigem Sinne authentisiert. (Die Gesamtfehlerwahrscheinlichkeit ändert sich etwas!)

Ein wesentlich effizienterer Code wird in Aufgabe 3-8 f) betrachtet.

Ausblick: Grenzen von Authentikationscodes. Wie informationstheoretisch sichere Konzelationssysteme, so brauchen auch informationstheoretisch sichere Authentikationscodes Schlüssel einer gewissen Mindestlänge.

Wir haben schon fast gezeigt, daß man für eine Fehlerwahrscheinlichkeit ε mindestens eine Schlüsselmenge mit $\lceil 1/\varepsilon \rceil$ Elementen braucht: Klar war, daß $\lceil 1/\varepsilon \rceil$ MACs für die zu fälschende Nachricht y möglich sein müssen. Zu jedem möglichen MAC muß aber auch ein möglicher Schlüssel gehören.

Es ist auch einleuchtend, daß es noch mehr Schlüssel sein müssen, denn anhand der richtigen Nachricht (x, *MAC*) konnte der Angreifer ja i.allg. schon viele Schlüssel ausschließen. Man kann (nicht allzu schwierig) zeigen, daß es $\lceil 1/\varepsilon^2 \rceil$ Schlüssel sein müssen [GiMS_74].

Wählt man $\varepsilon = 2^{-\sigma}$, so braucht man also mindestens Schlüssel der Länge 2σ. Dies ist lange nicht so schlimm wie die untere Schranke für Konzelationssysteme: Dort mußte der Schlüssel so lang wie die Nachricht sein. Für die Fehlerwahrscheinlichkeit wird hingegen in der Praxis ein Wert von etwa 2^{-100} genügen. (Sonst wird allmählich die Wahrscheinlichkeit größer, daß der zur Prüfung des MAC benutzte Rechner unbemerkt ausfällt und gefälschte Nachrichten als gültig passieren läßt u.ä., oder überhaupt sämtliche Risiken des praktischen Lebens.)

Für 1-Bit-Nachrichten ist der Code aus Bild 3-15 also optimal, und auch die erste Erweiterung aus Beispiel 2. Allerdings bräuchte die Schlüssellänge nicht, wie in der 2. Erweiterung aus Beispiel 2, mit der Nachrichtenlänge zu wachsen. Mit dem wesentlich effizienteren Code aus Aufgabe 3-8 f) kann man z.B. noch Nachrichten der Länge 100 mit 200 Schlüsselbits und Fehlerwahrscheinlichkeit 2^{-100} authentisieren. Dieser Code wurde in [WeCa_81 §3] so erweitert, daß man eine Nachricht der Länge l mit Fehlerwahrscheinlichkeit $2^{-\sigma+1}$ mit einem Schlüssel der Länge etwa $4\sigma\,\mathrm{ld}(l)$ authentisieren kann. Das ist zwar nicht optimal, aber praktikabel.

Authentisiert man n Nachrichten hintereinander mit einem Schlüssel und will eine Fehlerwahrscheinlichkeit $\leq 2^{-\sigma}$, so gilt, daß man pro Nachricht mindestens σ Schlüsselbits braucht; man braucht also nicht mehr 2σ pro Nachricht: Es gibt ein Verfahren, das für n kurze Nachrichten mit $(n+2)\sigma$ Bit auskommt [WeCa_81 §4].

Die Idee ist folgende: Immer mit *denselben* 2σ Bits des Schlüssels wird zu jeder der kurzen Nachrichten ein *geheimzuhaltender* MAC berechnet. Diese werden aber nie direkt ausgegeben (sonst könnte ein Angreifer nach zwei beobachteten MACs beliebig fälschen), sondern jeweils nur *One-time-pad verschlüsselt*. Für die One-time-pad Verschlüsselung der n MACs werden $n\sigma$ Bits Schlüssel benötigt, insgesamt also $(n+2)\sigma$ Bits.

3.4 Der s^2-mod-n-Pseudozufallsbitfolgengenerator: Kryptographisch starke Konzelation

Wir wissen nun: Wenn man informationstheoretisch sichere Konzelation will, kann man erstens nur ein symmetrisches Verfahren nehmen, hat also Probleme mit der Schlüsselverteilung, und zweitens müssen die Schlüssel mindestens so lang wie die Nachricht sein. Deswegen begnügt man sich in der Praxis fast immer mit kryptographischer Sicherheit. (Rote Telefone zwischen Moskau und Washington haben aber angeblich wirklich Vernam-Chiffren.)

In diesem Abschnitt betrachten wir die zweitbeste Sicherheitsstufe, nämlich kryptographisch starke symmetrische und asymmetrische Konzelation. Das asymmetrische System hat allerdings den Nachteil, daß es gegen aktive Angriffe definitiv unsicher ist. In vielen Fällen in der Praxis würde man daher RSA bevorzugen, obwohl das nicht kryptographisch stark ist (§3.6).

Das symmetrische und das asymmetrische Konzelationssystem beruhen auf demselben kryptographisch starken Pseudozufallsbitfolgengenerator. Ein solcher kryptographisch starker Pseudozufallsbitfolgengenerator ist auch sonst von Interesse, weil die Generierung von echten Zufallszahlen in der Praxis ein beträchtliches Problem ist.

Außerdem sind die mathematischen und kryptographischen Grundlagen für dieses System dieselben, die auch für GMR und RSA benötigt werden, da alle diese Systeme auf der Faktorisierungsannahme beruhen.

3.4.1 Grundlagen für Systeme mit Faktorisierungsannahme

Bei allen asymmetrischen kryptographischen Systemen muß sich jemand ein Geheimnis wählen (einen Dechiffrier- oder Signierschlüssel), zu dem er irgendwelche Information veröffentlichen kann (den Chiffrier- oder Testschlüssel). Letzteres muß einerseits ermöglichen, das zu prüfen, was er mit seinem Geheimnis tut, darf andererseits aber nicht erlauben, mit vernünftigem Aufwand das Geheimnis herzuleiten. Die Idee, daß so etwas überhaupt möglich ist, galt 1976 bei Erscheinen von [DiHe_76] als ziemlich sensationell.

Fast alle derzeit ernsthaft diskutierten asymmetrischen Systeme beruhen auf nur zwei verschiedenen Ideen für solche Geheimnisse. Wir betrachten davon die auf der Faktorisierungsannahme beru-

henden. (Die andere heißt diskreter Logarithmus, vgl. §3.9.1, und beruht auf ähnlicher elementarer Zahlentheorie.)

Idee für die Geheimnisse: Das Geheimnis besteht im wesentlichen aus zwei großen, unabhängig zufällig gewählten

Primzahlen p und q.

(Groß heißt derzeit je etwa 300 bis 1000 Bit lang.) Die öffentliche Information ist im wesentlichen das Produkt

$$n = p \bullet q.$$

Damit daraus etwas Sinnvolles werden kann, muß man also

1. große Primzahlen in vernünftiger Zeit wählen können (multiplizieren kann man sie sowieso),
2. aber es darf nicht mit vernünftigem Aufwand möglich sein, solche Produkte zu faktorisieren.
3. Außerdem reicht es natürlich nicht, ein Geheimnis zu haben, sondern es muß irgendwelche nachrichtenabhängigen Funktionen, z.B. zum Entschlüsseln oder Signieren geben, die man nur mit Hilfe von p und q effizient berechnen kann, während für die umgekehrten Operationen n genügt.

Für einige der Systeme werden noch kleine Spezialeigenschaften von p und q verlangt, z.B. in diesem §3.4, daß

$$p \equiv q \equiv 3 \bmod 4.$$

Zur Faktorisierungsannahme: Entsprechend §3.1.3.4 kann man zur Zeit nicht beweisen, daß Faktorisierung schwer ist, sondern man muß eine Annahme treffen. Diese besagt, daß für jeden schnellen (Faktorisierungs-)Algorithmus F die Wahrscheinlichkeit, daß F eine Zahl der speziellen Gestalt $p \bullet q$ tatsächlich faktorisieren kann, schnell immer kleiner wird, je größer die Länge l der Faktoren wird.

Dieses l ist also der Sicherheitsparameter. Man sieht ein, daß worst- oder average-case-Komplexität nicht genügen würde: Selbst wenn F z.B. bei allen Längen immer nur jede 1000-te Zahl schnell faktorisieren könnte, hieße das, daß er für 1/1000 der Teilnehmer ihre Geheimnisse bestimmen könnte. Genauer verlangt man, daß die Restwahrscheinlichkeit schneller sinkt als 1 dividiert durch jedes beliebige Polynom.

Annahme: Für jeden (probabilistischen) polynomialen Algorithmus F und jedes Polynom Q gilt: Es existiert ein L, so daß für alle $l \geq L$ gilt: Wenn p, q als unabhängig zufällige Primzahlen der Länge l gewählt werden und $n := p \bullet q$

$$W(F(n) = (p; q)) \leq \frac{1}{Q(l)}.$$

Dies ist die Annahme, mit der man in Beweisen arbeitet; für konkrete Systeme in der Praxis lautet das, was man annimmt, natürlich eher: Es gibt keinen Algorithmus F, der für die ganz bestimmte Länge l, die gerade verwendet wird (z.B. 1024), mehr als einen ganz bestimmten Anteil (z.B. 2^{-100}) in weniger als einem Jahrhundert auf dem größten verfügbaren Rechner faktorisiert.

Während keine „schnellen“ Algorithmen bekannt sind, die *alle* Zahlen n faktorisieren, sind schnelle bekannt, *wenn* $p+1$, $p-1$, $q+1$ oder $q-1$ ihrerseits nur kleine Primfaktoren haben [DaPr_89 Seite 232]. Um dies zu vermeiden, wird gefordert:

$p+1$ muß (mindestens) einen großen Primfaktor haben.
$p-1$ muß (mindestens) einen großen Primfaktor haben.
$q+1$ muß (mindestens) einen großen Primfaktor haben.
$q-1$ muß (mindestens) einen großen Primfaktor haben.

Hierbei sollten die großen Primfaktoren jeweils mindestens 100 Bit lang sein [BrDL_91 Seite 268].

Für sehr große p und q (und damit insbesondere asymptotisch) sind diese Forderungen „von allein“ mit sehr großer Wahrscheinlichkeit erfüllt. Für den praktischen Einsatz hinreichend wahrscheinlich ist dies für Primzahlen, die wesentlich länger als 500 Bit sind [BrDL_91 Seite 268].

Um konkrete Hinweise für die Dimensionierung kryptographischer Systeme zu geben, die auf der Faktorisierungsannahme beruhen, sei der Stand der Faktorisierung kurz berichtet. Mittels

- einzelner „Superrechner“ (Vektorrechner),
- hundert durch ein lokales Netz verbundener Workstations und
- per elektronischer Post verbundener weltweit verteilter UNIX-Systeme, deren Leerzeit genutzt wurde,

konnten 1989 innerhalb eines Monats bereits je Zahlen mit 100 bis 130 Dezimalstellen faktorisiert werden [LeMa1_89, LeMa1_90, Silv_91, Schn_96 Seite 257]. 1999 ist man bei 155 Dezimalstellen angelangt [Riel_99].

Laut [Schn_96 Seite 256] ist der Faktorisierungsalgorithmus *quadratic sieve*, mit dem 1989 die oben genannten Faktorisierungsrekorde erzielt wurden, der schnellste bekannte für Zahlen n bis 110 Dezimalstellen. Seine Laufzeit beträgt

$$L_{\textit{quadratic sieve}}(n) = \exp(\sqrt{\ln(n) \ln\ln(n)}).$$

Die Laufzeit von Faktorisierungsalgorithmen ist wie üblich in der Zeiteinheit angegeben, die zur Multiplikation von zwei Zahlen halber Länge benötigt wird, beispielsweise also für die Berechnung von $n = p \cdot q$.

Für Zahlen n mit mehr als 110 Dezimalstellen[44] ist der Faktorisierungsalgorithmus *general number field sieve*, der erst seit 1993 eingesetzt wird, schneller und der schnellste bisher bekannte. Seine Laufzeit beträgt

$$L_{\textit{general number field sieve}}(n) = \exp(1{,}923 \cdot \sqrt[3]{\ln(n) (\ln\ln(n))^2}).$$

Bei Zahlen mit etwa 130 Dezimalstellen benötigt der Faktorisierungsalgorithmus *quadratic sieve* für 4 weitere Dezimalstellen doppelten Rechenaufwand, der Faktorisierungsalgorithmus *general number field sieve* benötigt dies erst für 5 weitere Dezimalstellen.

In der Praxis sollte l heutzutage im Bereich von 350 bis 2048 Bit Länge gewählt werden, vgl. Aufgabe 3-11 a), so daß die zu faktorisierende Zahl n eine Länge zwischen 700 und 4096 Bit hat.

Zur Primzahlerzeugung:

- Zunächst ist festzustellen, daß es auch unter sehr großen Zahlen noch genügend viele Primzahlen gibt: Nach dem **Primzahlsatz** [Kran_86 §1.15] gilt für die Anzahl $\pi(x)$ der Primzahlen bis zu einem gewissen Maximum x:

  $$\frac{\pi(x)}{x} \approx \frac{1}{\ln(x)}.$$

  Betrachtet man also die Zahlen bis zur Länge l Bit, so ist im Mittel jede ($l \cdot \ln(2)$)-te eine Primzahl.
- Nach dem Dirichletschen Primzahlsatz [Kran_86 §1.15] ist im Mittel die Hälfte aller Primzahlen $\equiv 3 \bmod 4$ (und je ein Viertel $\equiv 3$ bzw. $7 \bmod 8$).

Diese beiden Sätze bedeuten zum einen, daß zumindest keine Gefahr besteht, daß jemand alle Zahlen n der speziellen Form $p \cdot q$ und einer festen Länge l faktorisiert, indem er sich eine Liste aller in Frage kommenden Primzahlen macht. Zum anderen werden sie für die Primzahlerzeugung benötigt.

---

44 Bei Gleichsetzen der angegebenen Laufzeiten von *quadratic sieve* und *general number field sieve* und Lösen nach $\ln(n)$ ergibt sich: $\ln(n) \approx 286$. Dies bedeutet 124 Dezimalstellen. Der Unterschied zu den von Bruce Schneier angegebenen 110 Dezimalstellen mag sich dadurch erklären, daß auch beim *quadratic sieve* ein Faktor vor der Wurzel steht, der, da er ziemlich nahe bei 1 liegt, in der angegebenen Laufzeitformel zu 1 gesetzt wurde.

- Das Prinzip der schnellen Erzeugung zufälliger Primzahlen p ist nämlich einfach, ebenso das Erzeugen von Primzahlen mit zusätzlichen Eigenschaften, etwa $p \equiv 3 \bmod 4$:

  WHILE noch keine Primzahl gefunden
  DO Wähle Zufallszahl p passender Größe[45] (ggf. z.B. mit $p \equiv 3 \bmod 4$)
  Teste, ob p prim
  OD.

  Man hat also nach etwa $l \cdot \ln(2)$ Durchläufen Erfolg.

- Der Test, ob p prim ist, kann sicherlich nicht einfach erfolgen, indem man p faktorisiert und schaut, daß es keine echten Teiler hat, denn Faktorisierung haben wir gerade als schwierig angenommen.

  Zum Glück gibt es schnellere Tests, wenn diese auch wieder eine exponentiell kleine Fehlerwahrscheinlichkeit haben. Der beste ist der RABIN-MILLER-Test. Speziell für Zahlen $p \equiv 3 \bmod 4$ ist er besonders einfach[46]. Es gilt

  $$p \text{ prim} \Rightarrow \forall a \in \mathbb{Z}_p^*: a^{\frac{p-1}{2}} \equiv \pm 1 \bmod p$$

  (Herleitung in §3.4.1.5), während es für zusammengesetztes p höchstens für 1/4 aller entsprechenden a gilt. (Die im folgenden ausführlich eingeführte Notation $\mathbb{Z}_p^*$ bedeutet $\{1, 2, ..., p-1\}$, wenn p eine Primzahl ist.) Man prüft also diese Formel für σ zufällige Werte a nach. Gilt sie mindestens einmal nicht, ist klar, daß p nicht prim ist. Andernfalls kann man in gewisser Weise sagen, daß p mindestens mit Wahrscheinlichkeit $1 - 4^{-\sigma}$ eine Primzahl ist.

  Die Fehlerwahrscheinlichkeit stört zumindest bezüglich der Faktorisierungsannahme nicht, da es dort sowieso auch eine Fehlerwahrscheinlichkeit ähnlicher Größe gibt, nämlich die Wahrscheinlichkeit, daß F faktorisieren kann.

  In der Praxis prüft man allerdings, bevor man diesen Test nimmt, ob p ganz kleine Primfaktoren enthält. Dazu legt man (ein für allemal) eine Tabelle der ersten Primzahlen an, z.B. alle von höchstens Wortlänge des verwendeten Rechners. Durch alle diese wird probeweise dividiert. Erst dann lohnt sich das Exponentieren, vgl. z.B. [Kran_86 §1, §2].

Zum Rechnen mit und ohne Kenntnis von p und q:
Die Rechnungen mit p und q bzw. mit n sind fast alles Rechnungen mit Restklassen. Deswegen sind im folgenden die wichtigsten Tatsachen darüber dargestellt. Insbesondere wird danach unterschieden, wie leicht die Operationen auszuführen sind. (Bei Bedarf nach mehr Information vgl. z.B. [Lips_81, Lüne_88, ScSc_73, Knut_97] oder eine beliebige Einführung in sog. Elementare Zahlentheorie.)

3.4.1.1 Rechnen modulo n

Hier ist n völlig beliebig. Diese Operationen können also sowohl die Teilnehmer mit Geheimnis als auch die ohne durchführen.

- $\mathbb{Z}_n$ bezeichnet den Restklassenring modulo n. I. allg. wird er durch die Zahlen $\{0,\ldots,n-1\}$ repräsentiert, und vermutlich ist das Rechnen darin den meisten Lesern intuitiv klar. Trotzdem seien hier kurz die grundlegenden Definitionen wiederholt:

  Formaler definiert man den Restklassenring modulo n über eine Kongruenzrelation $\equiv$:

  Für $a, b \in \mathbb{Z}$ sei

---

45 In diesem Moment haben wir noch nicht unbedingt eine Primzahl, obwohl der Bezeichner p dies suggeriert. Nach Ende des Algorithmus ist p dann hoffentlich prim.

46 Auch hier suggeriert die Bezeichnerwahl p das gewünschte Ergebnis und nicht unbedingt den Ist-Zustand.

$$a \equiv b \bmod n \quad :\Leftrightarrow \quad n \mid (a-b).$$

(Das Zeichen „|“ steht für „ist ein Teiler von“, d.h. $x|y :\Leftrightarrow \exists\, z \in \mathbb{Z}: y = xz$) Es gilt $a \equiv b \bmod n$ genau dann, wenn sie bei Division durch n denselben Rest lassen.

Nun definiert man zu jedem $a \in \mathbb{Z}$ eine Restklasse $\bar{a} := \{b \in \mathbb{Z} \mid a \equiv b \bmod n\}$. Mit diesen Klassen kann man rechnen, weil gilt

$$a_1 \equiv a_2 \bmod n \ \wedge\ b_1 \equiv b_2 \bmod n \ \Rightarrow\ a_1 \pm b_1 \equiv a_2 \pm b_2 \bmod n \ \wedge\ a_1 \cdot b_1 \equiv a_2 \cdot b_2 \bmod n,$$

wie man recht leicht nachrechnet. (D.h. um zwei Klassen $\bar{a}, \bar{b}$ zu addieren o.ä., kann man zwei beliebige Elemente daraus nehmen und erhält immer dasselbe Ergebnis.) Konkret nimmt man i. allg. die Vertreter aus $\{0,\ldots,n-1\}$ und bildet vom Ergebnis den Rest bei Division durch n, insbesondere ist das die üblichste Darstellung im Rechner. Zum „Rechnen von Hand“ nimmt man häufig die *symmetrische Repräsentation* um die Null herum, für ungerade Zahlen n also $\{-(n-1)/2, ..., 0, ..., (n-1)/2\}$ und für gerade Zahlen n etwas weniger symmetrisch $\{-n/2+1, ..., 0, ..., n/2\}$.

Die meisten üblichen Rechenregeln gelten, genauer ist $\mathbb{Z}_n$ ein kommutativer Ring mit 1.

- Addition, Subtraktion und Multiplikation in $\mathbb{Z}_n$ sind „leicht“. Man kann mit den Langzahlen aus mehreren Rechnerwörtern im Prinzip rechnen wie mit Dezimalzahlen aus mehreren Ziffern in der Schule. (Ist l die Länge von n, so ist der Aufwand von plus und minus $O(l)$, der von mal $O(l^2)$ – egal ob innerhalb der ganzen Zahlen oder mod n gerechnet wird [Lips_81 Seite 202]. Es gibt auch schnellere Algorithmen, aber für Zahlen der in der Kryptographie üblichen Größenordnungen sind die Unterschiede noch nicht groß.)

  Auch die Exponentiation $a^b \bmod n$ mit einem Exponenten b derselben Größenordnung ist „leicht“.[47] Sie wird oft gebraucht (z.B. schon oben im Primzahltest). Allerdings würde hier einfaches b-maliges Multiplizieren zu lange dauern. Man verwendet sog. „square-and-multiply“-Algorithmen, bei denen der Exponent binärstellenweise abgearbeitet wird; sei $b = (b_{l-1} \ldots b_0)_2$ die Binärdarstellung. Hier wird die Berechnung von links skizziert (man kann auch von rechts anfangen):

  Es wird der Reihe nach $a^{(b_{l-1})_2}$, $a^{(b_{l-1}b_{l-2})_2}$ usw. berechnet. Dabei erhält man aus $a^{(b_{l-1}b_{l-2} \cdots b_{l-i})_2}$ den nächsten Wert, je nach dem nächsten Exponentenbit, als

$$a^{(b_{l-1}b_{l-2} \ldots b_{l-i}0)_2} = a^{2\cdot(b_{l-1}b_{l-2} \ldots b_{l-i})_2} = \left(a^{(b_{l-1}b_{l-2} \ldots b_{l-i})_2}\right)^2$$

  bzw.
$$a^{(b_{l-1}b_{l-2} \ldots b_{l-i}1)_2} = a^{2\cdot(b_{l-1}b_{l-2} \ldots b_{l-i})_2+1} = \left(a^{(b_{l-1}b_{l-2} \ldots b_{l-i})_2}\right)^2 \cdot a.$$

  Es wird also pro Binärstelle quadriert (square) und bei einer 1 noch zusätzlich multipliziert (multiply) – jeweils mod n.

  Also ist der Aufwand für eine Exponentiation $a^b \bmod n$ mit einem Exponenten b derselben Größenordnung höchstens $O(l^3)$, denn Quadrieren (und ggf. auch Multiplizieren) wird l mal durchlaufen. Ist der Exponent b kürzer – führende Nullen streichen wir weg –, so braucht für jedes Bit, das er kürzer ist, einmal weniger quadriert (und ggf. multipliziert) zu werden. Ist $|b|$ die Länge des Exponenten b, so ist der Aufwand zur Berechnung von $a^b \bmod n$ folglich höchstens $O(l^2 \cdot |b|)$.

  Beispiel: $7^{22} \bmod 11$. 22 ist binär $(10110)_2$. Also berechnen wir

$$7^{(1)_2} := 7;$$
$$7^{(10)_2} := 7^2 \equiv 5;$$
$$7^{(101)_2} := 5^2 \cdot 7 \equiv 3 \cdot 7 \equiv -1;$$

---

47 Als Anhaltspunkt: Praktisch dauert eine solche Exponentiation, wenn alle drei Parameter 512 bit lang sind, in Software auf 1997 verfügbaren PCs größenordnungsmäßig 0,1 Sekunde. Es hängt natürlich von der genauen Rechnerleistung und der Qualität der Implementierung ab.

$$7^{(1011)_2} := (-1)^2 \bullet 7 \equiv 7;$$
$$7^{(10110)_2} := 7^2 \equiv 5.$$

- $\mathbb{Z}_n^*$ bezeichnet die multiplikative Gruppe von $\mathbb{Z}_n$, d.h. die Menge aller Elemente, die ein Inverses haben. (Man kann leicht zeigen, daß das wirklich eine Gruppe ist.) Es gilt

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \text{ggT}(a, n) = 1\}.$$

Auch das Berechnen der Inversen ist „leicht". Man verwendet den sog. erweiterten Euklidschen Algorithmus. Dieser berechnet allgemein zu $a, b \in \mathbb{Z}$ den ggT und zwei ganze Zahlen u, v mit

$$u\,a + v\,b = \text{ggT}(a, b)$$

Das sei nur an einem Beispiel dargestellt: $a = 11$, $b = 47$. Dazu wenden wir zunächst den normalen Euklidschen Algorithmus an, bei dem iterativ der vorige Divisor durch den vorigen Rest geteilt wird:

$$47 = 4 \bullet 11 + 3;$$
$$11 = 3 \bullet 3 + 2;$$
$$3 = 1 \bullet 2 + 1.$$

Nun beginnt man mit der letzten Zeile, den ggT als Linearkombination des Divisors und Dividenden der jeweiligen Zeile darzustellen:

$1 = 3 - 1 \bullet 2$ (Jetzt 2 durch 3 und 11 ersetzen)
$= 3 - 1 \bullet (11 - 3 \bullet 3) = -11 + 4 \bullet 3$ (Jetzt 3 durch 11 und 47 ersetzen)
$= -11 + 4 \bullet (47 - 4 \bullet 11) = 4 \bullet 47 - 17 \bullet 11.$ (Probe: 4•47= 188, 17•11=187)

(Dies kann man, v.a. bezüglich Speicherplatz, noch optimieren, s. Literatur.)

Speziell zum Invertieren von a berechnen wir mit dem erweiterten Euklidschen Algorithmus ganze Zahlen u, v mit

$$u\,a + v\,n = 1.$$

Dann gilt offenbar

$$u\,a \equiv 1 \bmod n,$$

und das heißt gerade, daß u das Inverse von a ist. In unserem Beispiel haben wir also

$$11^{-1} \equiv -17 \equiv 30 \bmod 47$$

erhalten (und auch $47^{-1} \equiv 4 \bmod 11$ sowie $11^{-1} \equiv -17 \equiv 3 \bmod 4$ und $47^{-1} \equiv 4 \bmod 17$.)[48]

3.4.1.2 Elementanzahl von $\mathbb{Z}_n^*$ und Zusammenhang mit Exponentiation

Hier sind erste Eigenschaften von $\mathbb{Z}_n$, bei denen Kenntnis des Geheimnisses (p, q) hilft:

- Die Eulersche ϕ-Funktion ist definiert als

$$\phi(n) := \left|\{a \in \{0,\ldots,n-1\} \mid \text{ggT}(a, n) = 1\}\right|,$$

wobei für beliebige ganze Zahlen $n \neq 0$ gilt: $\text{ggT}(0,n) = |n|$, vgl. z.B. [Lüne_87 Seite 12].
Es folgt sofort aus den beiden Definitionen, daß

$$\left|\mathbb{Z}_n^*\right| = \phi(n).$$

Speziell für $n = p \bullet q$ mit p, q prim und $p \neq q$ kann man $\phi(n)$ leicht ausrechnen:

---

48 Damit sieht man auch, daß die obige Gleichung $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \text{ggT}(a, n) = 1\}$ stimmt: Gerade haben wir für jedes a mit $\text{ggT}(a, n) = 1$ tatsächlich ein Inverses bestimmt. Umgekehrt: Wenn es ein Inverses gibt, also $a \bullet a^{-1} \equiv 1 \bmod n$, gilt $n \mid (a \bullet a^{-1} - 1)$, also gibt es v mit $a \bullet a^{-1} - 1 = v \bullet n$ bzw. $a \bullet a^{-1} - v \bullet n = 1$. Hätten a und n einen gemeinsamen Teiler d, so würde dieser also auch 1 teilen.

$$\phi(p \bullet q) = (p-1)(q-1).$$[49]

Ist n ($=p \bullet q$) gegeben, so ist $\phi(n)$ auszurechnen gleich schwer wie p und q zu finden [Hors_85 Seite 187]:

$\phi(p \bullet q) = (p-1)(q-1) = p \bullet q - (p+q) + 1 = n - (p+q) + 1$ zusammengefaßt gilt

$p+q = n - \phi(p \bullet q) + 1$ (**)

$(p-q)^2 = p^2 - 2pq + q^2 = p^2 + 2pq + q^2 - 4pq = (p+q)^2 - 4n$ durch Wurzelziehen ergibt sich

$p-q = \sqrt{(p+q)^2 - 4n}$

Einsetzen von (**) ergibt $p-q = \sqrt{(n - \phi(p \bullet q) + 1)^2 - 4n}$

Addition mit (**) ergibt $2p = n - \phi(p \bullet q) + 1 + \sqrt{(n - \phi(p \bullet q) + 1)^2 - 4n}$

- Für manche kryptographischen Systeme (z.B. RSA) braucht man folgenden Zusammenhang zwischen der Exponentiation und der Elementanzahl: Der kleine Fermatsche Satz sagt: Für alle $a \in \mathbb{Z}_n^*$:

$$a^{\phi(n)} \equiv 1 \bmod n.$$[50]

3.4.1.3 Zusammenhang zwischen $\mathbb{Z}_n$ und $\mathbb{Z}_p$, $\mathbb{Z}_q$

(Für $n = p \bullet q$ mit p, q prim und $p \neq q$.) Dies wird später dazu dienen, daß der Teilnehmer mit dem Geheimnis (p, q) geheime Dinge modulo p und q einzeln rechnen kann, aber dann modulo n, wo alle anderen Teilnehmer rechnen, verwenden kann.

Die Grundlage für diesen Zusammenhang ist folgende Formel (ein Spezialfall des „Chinesischen Restsatzes“):

$$a \equiv b \bmod n \;\Leftrightarrow\; a \equiv b \bmod p \;\wedge\; a \equiv b \bmod q. \qquad (\divideontimes)$$

Dies sieht man leicht ein: Nach Definition von „≡“ ist dies äquivalent zu

$$n \mid (a-b) \;\Leftrightarrow\; p \mid (a-b) \;\wedge\; q \mid (a-b),$$

und dies ist klar, weil $p \bullet q$ die Primfaktorzerlegung von n ist.[51]

Wenn man nun z.B. $y := f(x) \bmod n$ berechnen möchte, aber die Berechnung von $f(x)$ nur modulo Primzahlen leicht ist, so kann der Geheimnisbesitzer $y_p := f(x) \bmod p$ und $y_q := f(x) \bmod q$ bestimmen, und es gilt:

$$y \equiv f(x) \bmod n \;\Leftrightarrow\; y \equiv y_p \bmod p \;\wedge\; y \equiv y_q \bmod q.$$

Wir müssen also nur noch schauen, ob wir aus y_p und y_q effizient y zusammensetzen können. Das tut der „Chinesische Restealgorithmus (CRA)“ (der eigentliche Chinesische Restsatz besagt nur, daß ein solches y für beliebige y_p, y_q existiert):

Zunächst bestimmt man mit dem erweiterten Euklidschen Algorithmus ganze Zahlen u, v mit

---

49 Die nicht zu n teilerfremden Zahlen (d.h. mit ggT≠1) sind nämlich einmal 0, dann p, $2p$, …, $(q-1)p$, und zuletzt q, $2q$, …, $(p-1)q$, und diese $1+(q-1)+(p-1)=p+q-1$ Zahlen sind für $p \neq q$ alle verschieden.

50 Dies ist ein allgemeiner Satz aus der Gruppentheorie [Lips_81 Seite 85 Cor. 4, Waer_71 Seite 26]: In jeder endlichen Gruppe G gilt für jedes Element a: $a^{|G|} = 1$. Der Beweis ist nicht sehr schwer. Die beiden Grundlagen sind:

1. Für jede Untergruppe H gilt: $|H| \mid |G|$.
2. Die Potenzen eines Elementes, also a, a^2, … bilden eine Untergruppe. Es gibt eine erste Potenz $a^v = 1$, und genau ab da wiederholt sich die Folge zyklisch. v wird „Ordnung von a“ genannt. Diese Untergruppe hat also genau v Elemente, und nach 1. folgt $v \mid |G|$.

51 Es ist leicht zu sehen, daß der Chinesische Restsatz nicht nur für n = Produkt zweier verschiedener Primzahlen gilt, sondern auch für n = Produkt beliebig vieler, paarweise teilerfremder natürlicher Zahlen.

$$u\,p + v\,q = 1.$$[52]

Die Behauptung ist, daß

$$y = \mathrm{CRA}(y_p, y_q) :\equiv up\,y_q + vq\,y_p \bmod n.$$

Wir müssen also $y \equiv y_p \bmod p \wedge y \equiv y_q \bmod q$ zeigen. Dies sieht man am besten mit folgender Tabelle:

	mod p	mod q
up	0	1
vq	1	0
$up\,y_q + vq\,y_p$	$0 \bullet y_q + 1 \bullet y_p$ $\equiv y_p$	$1 \bullet y_q + 0 \bullet y_p$ $\equiv y_q$

(Die oberen beiden Zeilen folgen aus $u\,p + v\,q = 1$ und besagen, daß up und vq in $\mathbb{Z}_n$ etwas wie zwei „Basisvektoren“ darstellen, wenn man $\mathbb{Z}_n$ aus $\mathbb{Z}_p$ und $\mathbb{Z}_q$ zusammensetzt. Die Berechnung von up und vq kann ein für allemal bei Erzeugung des Geheimnisses geschehen.)

3.4.1.4 Quadrate und Wurzeln allgemein

Aus §3.4.1.3 wissen wir, wie ein Geheimnis p, q helfen kann, eine Funktion f modulo n auszurechnen, wenn sie modulo Primzahlen viel leichter zu berechnen ist. Eine solche Funktion wird modulares Wurzelziehen sein. Man wird also das Geheimnis benötigen, um Wurzeln mod n zu ziehen, während jeder andere allein anhand von n Quadrieren kann, also die inverse Operation ausführen. Dazu betrachten wir zunächst Wurzeln und Quadrate in Restklassenringen genauer.

- Definiert sind sie genau wie man sich das vorstellen würde, außer daß man sich meistens nur um invertierbare Elemente kümmert:

  $$\mathrm{QR}_n := \{x \in \mathbb{Z}_n^* \mid \exists\, y \in \mathbb{Z}_n^* : y^2 \equiv x \bmod n\}.$$

  Die x'e aus dieser Menge heißen quadratische Reste (also das, was in $\mathbb{N}$ die Quadratzahlen sind), und ein entsprechendes y heißt Wurzel aus x.

- Es gilt wie üblich

  $$y \text{ ist Wurzel aus } x \implies -y \text{ ist auch Wurzel aus } x,$$

  denn es ist $(-1)^2 = 1$.

- Andere Gesetze, die man aus $\mathbb{N}$ oder $\mathbb{R}$ gewöhnt ist, gelten aber i.allg. nicht. Beispielsweise kann eine Zahl durchaus mehr als zwei Wurzeln mod $n \neq p \bullet q$ haben, z.B. gilt mod 8: $1^2 \equiv 1 \wedge 3^2 \equiv 1 \wedge 5^2 \equiv 1 \wedge 7^2 \equiv 1$.

- Die quadratischen Reste bilden aber wenigstens eine (multiplikative) Gruppe: Sind $x_1, x_2 \in \mathrm{QR}_n$, und y_1, y_2 ihre Wurzeln, so sind die Wurzeln von $x_1 \bullet x_2$ und x_1^{-1} gerade $y_1 \bullet y_2$ und y_1^{-1}: $(y_1 \bullet y_2)^2 = {y_1}^2 \bullet {y_2}^2 = x_1 \bullet x_2$ und $(y_1^{-1})^2 = ({y_1}^2)^{-1} = x_1^{-1}$.

3.4.1.5 Quadrate und Wurzeln mod *p*

Während modulo beliebigem n nur wenig über Wurzeln und Quadrate zu sagen war (s. §3.4.1.4), ist modulo Primzahlen alles hübsch geordnet, vor allem, weil $\mathbb{Z}_p$ ein Körper ist.[53]

---

52 Zwei verschiedene Primzahlen haben ja den ggT 1. Es ist leicht zu sehen, daß der CRA auch funktioniert, wenn p und q keine Primzahlen, aber weiterhin teilerfremd sind. Durch wiederholte Anwendung des CRA kann der Chinesische Restsatz auch für Produkte bestehend aus mehr als zwei Faktoren abgedeckt werden.

53 Klar: Jedes Element außer 0 hat ein multiplikatives Inverses, wie wir oben sahen.

- Vor allem hat jetzt jede Zahl wieder höchstens 2 Wurzeln (denn in Körpern hat ein Polynom vom Grad 2 immer höchstens 2 Nullstellen)[54]. Wir wissen auch, daß $\pm y$ immer gemeinsam Wurzeln sind. I.allg. gilt auch $y \not\equiv -y$, außer für $y \equiv 0$ oder $p=2$. (Bei geraden Zahlen p, auch zusammengesetzten, gilt diese Ausnahme noch für $y \equiv p/2$.)

  Also haben für $p > 2$ alle quadratischen Reste genau 2 Wurzeln.

- Daraus folgt, daß für $p > 2$ genau die Hälfte aller Zahlen mod p quadratische Reste sind, d.h.

$$|QR_p| = \frac{p-1}{2},$$

  denn die Quadrierfunktion ist genau $2 \rightarrow 1$. Man kann das so darstellen:

x	0	1	2	…	$\frac{p-1}{2}$	$-\frac{p-1}{2}$	…	−2	−1
x^2	0	1	4	…	$(\frac{p-1}{2})^2$	$(\frac{p-1}{2})^2$	…	4	1

  (Zur Erinnerung: Da mod p alle Zahlen höchstens 2 Wurzeln haben, s.o., sind die Werte in jedem der beiden unteren Felder paarweise verschieden. Denn andernfalls hätte diese in einem Feld mehrfach auftretende Zahl, d.h. dieses Quadrat, mehr als 2 Wurzeln mod p.)

- Zunächst als reine Bezeichnungsweise zur Unterscheidung von Quadraten und Nichtquadraten definieren wir das Legendre-Symbol (später in einer Verallgemeinerung Jacobi-Symbol genannt) für alle $x \in \mathbb{Z}_p^*$ als

$$\left(\frac{x}{p}\right) := +1 \text{ falls } x \in QR_p \text{ und } \left(\frac{x}{p}\right) := -1 \text{ sonst.}$$

- Einen schnellen Algorithmus zur Quadratprüfung liefert für $p > 2$ das Euler-Kriterium:

$$\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \bmod p.$$[55]

  (Bisher hätten wir ja $(p-1)/2$ mögliche Wurzeln durchprobieren müssen, um festzustellen, ob x ein Quadrat ist, was exponentiell in der Länge l von p wäre; das Exponentieren geht aber in $O(l^3)$, siehe §3.4.1.1.)

- Manchmal braucht man, daß das Legendre-Symbol multiplikativ ist, d.h.

$$\left(\frac{xy}{p}\right) = \left(\frac{x}{p}\right) \cdot \left(\frac{y}{p}\right).$$

  Dies folgt sofort aus dem Euler-Kriterium.

---

54 Wenn Sie es lieber explizit bewiesen hätten: Der Beweis wird indirekt geführt. Nehmen wir an, es gäbe eine Zahl, die mehr als 2 Wurzeln hat. Dann gibt es insbesondere zwei wesentlich verschiedene Wurzeln, d.h. zwei Wurzeln, die sich in mehr als dem Vorzeichen unterscheiden. (Gäbe es keine zwei wesentlich verschiedenen Wurzeln, dann gäbe es höchstens zwei Wurzeln.) Nennen wir die zwei wesentlich verschiedenen Wurzeln x und y. Es gilt dann: $x^2 \equiv y^2 \bmod p$ und $x \not\equiv \pm y \bmod p$. Nach der Definition der Kongruenz gilt also: $p \mid x^2 - y^2$. Dies ist äquivalent mit $p \mid (x+y)\cdot(x-y)$. Da p eine Primzahl ist, muß p entweder $x+y$ oder $x-y$ teilen. Damit ist dann aber $x \equiv \pm y \bmod p$. Widerspruch!

55 Wenn man den kleinen Fermatschen Satz (s. §3.4.1.2) glaubt, kann man das leicht beweisen: Für Primzahlen ist $\phi(p) = p-1$, es gilt also mod p: $x^{p-1} \equiv 1$. $x^{(p-1)/2}$ ist also eine Wurzel aus 1, also ±1 (liegt also zumindest im Wertebereich des Jacobi-Symbols).

a) Wenn x ein Quadrat ist, etwa $x \equiv y^2$, dann gilt $x^{(p-1)/2} \equiv y^{p-1} \equiv 1$.

b) Die Gleichung $x^{(p-1)/2} \equiv 1$ kann als Polynomgleichung im Körper $\mathbb{Z}_p$ höchstens $(p-1)/2$ Lösungen haben. Wegen a) wissen wir schon, daß alle quadratischen Reste Lösungen sind, und wir wissen, daß es $(p-1)/2$ quadratische Reste gibt. Also kann es keine weiteren Lösungen geben. Für die quadratischen Nichtreste ist also $x^{(p-1)/2} \not\equiv 1$; es bleibt also nur $x^{(p-1)/2} \equiv -1$.

3.4.1.6 Quadrate und Wurzeln mod p für $p \equiv 3 \bmod 4$

Einige Sachen werden noch leichter, wenn man $p \equiv 3 \bmod 4$ voraussetzt, wie in den meisten folgenden kryptographischen Systemen.

- Zunächst gibt es einen einfachen Algorithmus zum Wurzelziehen:[56] (Bisher hätten wir $(p-1)/2$ mögliche Wurzeln durchprobieren müssen.) Für jedes $x \in QR_p$ gilt:

$$w := x^{\frac{p+1}{4}} \text{ ist Wurzel aus } x \bmod p.$$

  Erstens ist diese Formel wegen $p \equiv 3 \bmod 4$ sinnvoll: $(p+1)/4$ liegt in $\mathbb{N}$. Zweitens müssen wir prüfen, daß $w^2 \equiv x \bmod p$:

$$w^2 \equiv \left(x^{\frac{p+1}{4}}\right)^2 \equiv x^{\frac{p+1}{2}} \equiv x^{\frac{p-1}{2}} \cdot x \equiv 1 \cdot x,$$

  wobei die letzte Gleichung aus dem Euler-Kriterium folgt.

  Außerdem wissen wir, daß die so erhaltene Wurzel w selbst wieder ein Quadrat ist: Sie ist ja eine Potenz des Quadrats x. Das wird wichtig, wenn wir mehrfach hintereinander Wurzeln ziehen wollen, z.B. um 4-te, 8-te Wurzeln usw. zu ziehen.

- Es gilt

$$-1 \notin QR_p,$$

  denn wenn etwa $p = 4r + 3$ (mit $r \in \mathbb{N}_0$), so gilt nach dem Euler-Kriterium:

$$\left(\frac{-1}{p}\right) \equiv (-1)^{\frac{p-1}{2}} \equiv (-1)^{2r+1} \equiv -1 \mod p.$$[57]

- Das hat folgende Bedeutung für die zwei Wurzeln $\pm w$ aus einem x: Ist w die mit obigem Verfahren gefundene, also $w \in QR_p$, so gilt:

$$-w \notin QR_p,$$

  denn wenn beide Wurzeln quadratische Reste wären, so würde folgen (weil QR_p eine Gruppe ist): $-1 \equiv (-w) \cdot w^{-1} \in QR_p$; Widerspruch.

  Es gibt also auch genau zwei 4-te Wurzeln aus x, nämlich die zwei Wurzeln $\pm w'$ aus w, und genau zwei 8-te Wurzeln, nämlich die zwei Wurzeln $\pm w''$ aus w', usw. für jede Zweierpotenz.

3.4.1.7 Quadrate und Wurzeln mod n mit Kenntnis von p, q

(Für $n = p \cdot q$ mit p, q prim und $p \neq q$.) Jetzt setzen wir unsere Erkenntnisse aus §3.4.1.3, §3.4.1.5 und §3.4.1.6 zusammen, um mögliche Operationen zu erhalten, die nur Geheimnisbesitzer ausführen können: Diese können nämlich nun auch mod n testen, ob ein x ein Quadrat ist, und ggf. Wurzelziehen:

- Quadrattest: Es gilt

$$x \in QR_n \Leftrightarrow x \in QR_p \wedge x \in QR_q.$$

  (Und die rechten beiden Bedingungen kann der Besitzer von p und q mit dem Euler-Kriterium auswerten.) Beweis:

  (B1) Für $x \in QR_n$ sei etwa w eine Wurzel, d.h. $w^2 \equiv x \bmod n$. Dann gilt erst recht $w^2 \equiv x \bmod p$ und $w^2 \equiv x \bmod q$ (wegen (✻)).

---

56 Auch modulo anderen Primzahlen ist Wurzelziehen leicht im Sinne von schnell; der Algorithmus ist aber viel komplizierter [Kran_86 Seite 22].

57 Hierfür ist die Wahl von $p \equiv 3 \bmod 4$ wesentlich. Denn sonst wäre für große p nämlich $p \equiv 1 \bmod 4$ und damit nach dem Euler-Kriterium $-1 \in QR_p$.

(B2) Sei umgekehrt $x \in \mathrm{QR}_p \wedge x \in \mathrm{QR}_q$, etwa $w_p^2 \equiv x \bmod p$ und $w_q^2 \equiv x \bmod q$. Nach dem chinesischen Restsatz gibt es ein w mit

$$w \equiv w_p \bmod p \ \wedge \ w \equiv w_q \bmod q.$$

Damit gilt, wieder mit (❋),

$$w^2 \equiv w_p^2 \equiv x \bmod p \ \wedge \ w^2 \equiv w_q^2 \equiv x \bmod q \ \Rightarrow w^2 \equiv x \bmod n.$$

- Jedes $x \in \mathrm{QR}_n$ hat genau 4 Wurzeln: Es liegt ja in QR_p und in QR_q und hat dort jeweils zwei Wurzeln $\pm w_p$ und $\pm w_q$. Jede der 4 Kombinationen läßt sich nach (B2) zu einer Wurzel von x zusammensetzen: $w := \mathrm{CRA}(\pm w_p, \pm w_q)$.
- Wir erweitern das Legendre-Symbol nun zum Jacobi-Symbol modulo n, zunächst wieder als reine Bezeichnungsweise:

  $$\left(\frac{x}{n}\right) := \left(\frac{x}{p}\right) \cdot \left(\frac{x}{q}\right).$$

  Es gilt *nicht*, daß das Jacobi-Symbol von x mod n genau dann +1 ist, wenn $x \in \mathrm{QR}_n$, wie bei Primzahlen, sondern wenn entweder ($x \in \mathrm{QR}_p \wedge x \in \mathrm{QR}_q$) oder ($x \notin \mathrm{QR}_p \wedge x \notin \mathrm{QR}_q$). (Erstere Bedingung allein entspricht $x \in \mathrm{QR}_n$.) Alle Quadrate haben also das Jacobi-Symbol 1, aber nicht umgekehrt.[58]

  Das Jacobi-Symbol ist multiplikativ, d.h.

  $$\left(\frac{xy}{n}\right) = \left(\frac{x}{n}\right) \cdot \left(\frac{y}{n}\right).$$

  Dies folgt sofort aus seiner Definition und der Multiplikativität des Legendre-Symbols.

3.4.1.8 Quadrate und Wurzeln mod n mit Kenntnis von $p, q \equiv 3 \bmod 4$

Sei $n = p \cdot q$ mit p, q prim und $p \neq q$.

- Wurzelziehen ist leicht: Die Wurzeln $\pm w_p$ und $\pm w_q$ kann man mit dem Verfahren aus §3.4.1.6 bestimmen, und daraus jeweils w ausrechnen als

  $$w := \mathrm{CRA}(\pm w_p, \pm w_q).$$

- –1 ist ein Nichtquadrat mit Jacobi-Symbol 1:

  $$\left(\frac{-1}{n}\right) := \left(\frac{-1}{p}\right) \cdot \left(\frac{-1}{q}\right) = (-1) \cdot (-1) = +1.$$

3.4.1.9 Quadrate und Wurzeln mod n ohne Kenntnis von p, q

Damit Wurzelziehen und Quadrattesten als geheime Operationen taugen, müssen wir jetzt noch beweisen oder annehmen, daß sie ohne Kenntnis von p und q schwer sind. Beim Wurzelziehen kann man es beweisen. Dazu hilft folgendes, auch sonst nützliches Sätzchen:

- Wenn jemand zwei wesentlich verschiedene Wurzeln w, w' aus derselben Zahl x mod n kennt, d.h. $w \not\equiv \pm w'$, dann kann er definitiv n faktorisieren:
  $w^2 \equiv w'^2 \equiv x \bmod n \Rightarrow n \mid (w^2 - w'^2) = (w + w')(w - w')$. Andererseits heißt $w \not\equiv \pm w'$, daß n kein Teiler von $(w + w')$ oder $(w - w')$ ist. Also muß in jedem dieser beiden Faktoren genau ein Primfaktor von n stecken. Man erhält also einen davon z.B. als

  $$\mathrm{ggT}(n, w + w').$$

---

[58] Da mod p und q jeweils die Hälfte aller Zahlen $\not\equiv 0$ quadratische Reste sind, folgt, daß es mod n genau 1/4 sind, daß aber gerade die Hälfte das Jacobi-Symbol 1 haben.

- Nun folgt eine Beweisskizze, daß unter der Faktorisierungsannahme auch das Wurzelziehen mod *n* schwer ist.[59] Diese kann als Muster für sämtliche Beweise dieser Art dienen.

  Zu zeigen ist also „Faktorisieren schwer ⇒ Wurzelziehen schwer". Man zeigt die äquivalente Umkehrung „Wurzelziehen leicht ⇒ Faktorisieren leicht". Man nimmt also an, es gäbe einen guten Wurzelzieh-Algorithmus W und leitet daraus einen guten Faktorisierungsalgorithmus F her. Dazu konstruiert man F explizit und läßt F als Unterprogramm W verwenden, s. Bild 3-16.[60]

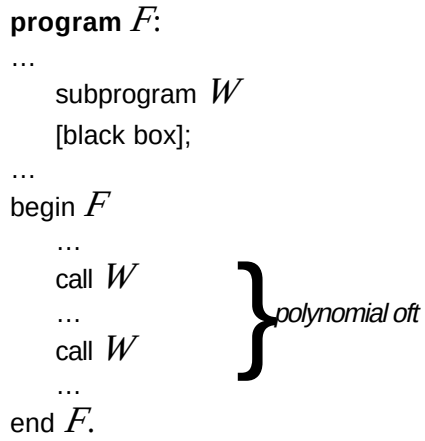


Bild 3-16: Struktur einer Turing-Reduktion des Faktorisierens F aufs Wurzelziehen W.

Unser spezielles F sieht so aus:

```
begin F
    wähle w ∈ ℤ_n* zufällig; setze x := w²;
    ziehe Wurzel: w' := W(x, n);
    if w ≢ ±w' then begin
        p := ggT(n, w + w');
        q := n div p
    end
end.
```

Zunächst ist klar, daß mit W auch F polynomial ist.

Die andere Behauptung ist: Wenn W bei einem bestimmten n mit Wahrscheinlichkeit ε eine Wurzel findet, dann faktorisiert F dasselbe n mit Wahrscheinlichkeit $\varepsilon/2$.[61] Aus dem vorigen Punkt folgt, daß F definitiv faktorisiert, wenn

1. W tatsächlich eine Wurzel findet und
2. $w' \not\equiv \pm w$.

Ersteres gilt mit Wahrscheinlichkeit ε. Wir müssen also nur noch zeigen, daß die gefundene Wurzel w' mit Wahrscheinlichkeit 1/2 wesentlich verschieden ist von dem w, was man schon

---

59 Formal heißt die Behauptung (ähnlich der Faktorisierungsannahme): Für jeden (probabilistischen) polynomialen Algorithmus W und jedes Polynom Q gilt: Es existiert ein L, so daß für alle $l \geq L$ gilt: Wenn p, q als zufällige Primzahlen der Länge l gewählt werden; $n := p \cdot q$, und x zufällig aus QR_n gewählt wird, so gilt

$$W((W(n, x)) \text{ ist Wurzel aus } x) \leq \frac{1}{Q(l)}.$$

60 Der Unterschied zur in der Komplexitätstheorie verwendeten sog. Karp-Reduktion [GaJo_80 Seite 118, Paul_78 Seite 60] ist, daß hier W in F beliebig oft aufgerufen werden darf (natürlich nur polynomial), während dort nur ein einziger Aufruf zugelassen ist.

61 Wir könnten die Wahrscheinlichkeit verbessern, indem wir F σ mal hintereinander ausführen; dann ist die Wahrscheinlichkeit $\varepsilon(1-2^{-\sigma})$, und der Algorithmus bei festem σ immer noch polynomial.

kannte. Dies liegt anschaulich daran, daß W nicht weiß, welche Wurzel außerhalb bekannt ist. Genauer: Das Ergebnis w' von W hängt nur von den Eingaben zu W ab, d.h. (n, x), und evtl. internen Zufallsentscheidungen von W. Heißen also die vier Wurzeln von x etwa $\pm w'$ und $\pm w''$, so hätte W als Ausgabe w' gehabt, ganz egal ob außen $w = \pm w'$ oder $w = \pm w''$ gewählt wurde, denn das sieht man x nicht an. Da außen zufällig gewählt wird, treten alle 4 Werte gleichwahrscheinlich als w auf. Mit Wahrscheinlichkeit 1/2 ist also $w \neq \pm w'$, was noch zu zeigen war.[62]

- Man nimmt an, daß es auch schwer ist, zu testen, ob $x \in \mathbb{Z}_n^*$ ein Quadrat ist, aber dies konnte man bisher nicht beweisen. Man macht daher eine Quadratische-Reste-Annahme („quadratic residuosity assumption“, ähnlich der Faktorisierungsannahme). Man muß dabei zwei Einschränkungen machen:
  1. Es gibt einen schnellen Algorithmus zur Berechnung von Jacobi-Symbolen, auch wenn p und q nicht bekannt sind (mit dem sog. quadratischen Reziprozitätsgesetz). Dieser wird im folgenden nicht gebraucht. Es bedeutet aber, daß jemand, dem man x mit $(\frac{x}{n}) = -1$ vorlegt, sehr wohl bestimmen kann, daß $x \notin \mathrm{QR}_n$. Man beschränkt die Annahme daher auf die x mit $(\frac{x}{n}) = +1$.
  2. Man kann für zufälliges x mit $(\frac{x}{n}) = +1$ natürlich schon durch rein zufälliges Raten mit Wahrscheinlichkeit 1/2 richtig raten, ob $x \in \mathrm{QR}_n$. Man verlangt daher nur, daß kein polynomialer (Rate-) Algorithmus R es wesentlich besser kann.

  Sei noch qr das Prädikat, ob etwas ein quadratischer Rest ist, d.h.

  $$qr(n, x) := \text{true, falls } x \in \mathrm{QR}_n, \text{ und } qr(n, x) := \text{false sonst.}$$

  Die Annahme lautet damit

  Annahme: Für jeden (probabilistischen) polynomialen Algorithmus R und jedes Polynom Q gilt: Es existiert ein L, so daß für alle $l \geq L$ gilt: Wenn p, q als zufällige Primzahlen der Länge l gewählt werden, $n := p \cdot q$, und x zufällig unter den Restklassen mit Jacobi-Symbol +1, so gilt:

  $$W(R(n, x) = qr(n, x)) \leq \frac{1}{2} + \frac{1}{Q(l)}.$$

- Wählen eines zufälligen Quadrats ist leicht: Man wählt einfach ein beliebiges $w \in \mathbb{Z}_n^*$ und quadriert es. Da jedes x gleich viele (für Spezialisten: genau 4) Wurzeln hat, wird jedes Quadrat gleichwahrscheinlich gewählt.

---

62 Ganz formal: Die Annahme, W sei ein „guter“ Wurzelziehalgorithmus, ist gerade der Gegensatz zur Fußnote zu "Wurzelziehen mod n schwer". Es existiert also doch ein Polynom Q, so daß doch für unendlich viele l gilt: Wenn p, q als zufällige Primzahlen der Länge l gewählt werden, $n := p \cdot q$, und x zufällig aus QR_n, so gilt

$$W(W(n, x) \text{ ist Wurzel aus } x) > \frac{1}{Q(l)}. \quad (1)$$

Zeigen müssen wir, daß F der Faktorisierungsannahme widerspricht, d.h. daß es ein Polynom R gibt, so daß für unendlich viele l gilt: Wenn p, q als zufällige Primzahlen der Länge l gewählt werden und $n := p \cdot q$, so gilt

$$W(F(n) = (p; q)) > \frac{1}{R(l)}. \quad (2)$$

Dies wird für $R := 2Q$ und dieselben l's wie in (1) gezeigt: Sei ein solches l gegeben. Man entnimmt der Beschreibung von F, daß dort x zufällig aus QR_n gewählt wird (denn jedes dieser x'e hat genau 4 Wurzeln, kommt also bei 4 der gleichwahrscheinlichen w's vor). Beim Aufruf von W liegen also alle Voraussetzungen von (1) vor, so daß (1) gilt. Nun folgt wie oben im Haupttext, daß in der Hälfte der Fälle $w \neq \pm w'$ und somit F erfolgreich ist. Insgesamt gilt dies also mit Wahrscheinlichkeit $1/(2 \cdot Q(l))$, wie behauptet.
Damit haben wir ganz formal gezeigt: „Wurzelziehen leicht $\Rightarrow$ Faktorisieren leicht“.

3.4.2 Anforderungen an Pseudozufallsbitfolgengeneratoren

Das Folgende steht ausführlicher in [BlBS_86].

Was ist ein Pseudozufallsbitfolgengenerator (PBG)?

Ein Pseudozufallsbitfolgengenerator (PBG) tut folgendes:

Er erzeugt aus einem echt zufällig gewählten *kurzen* Startwert (seed) eine *lange* Pseudozufallsbitfolge.

Also (siehe Bild 3-17):

- Es gehört dazu ein Schlüssel- und Startwertgenerieralgorithmus. Er bilde aus dem Sicherheitsparameter l einen Schlüssel n und einen Startwert s der Länge l.
  (An dieser Stelle würde ein Schlüssel *oder* ein Startwert genügen. Aber bei späterer Verwendung in asymmetrischen Systemen ist es interessant, daß der Schlüssel n öffentlich sein darf, während der Startwert s immer geheim bleiben muß.)
- Der eigentliche Bitfolgengenerieralgorithmus *PBG* ist ein Algorithmus, der mit dem Schlüssel n aus dem kurzen Startwert s eine (beliebig lange) Bitfolge $b_0\ b_1\ b_2\ \ldots$ erzeugt. Allerdings darf nur ein polynomial (in l) langes Anfangsstück dieser Folge verwendet werden.
- *PBG* ist **deterministisch**, d.h. aus dem gleichen Startwert und Schlüssel wird jedesmal dieselbe Bitfolge.
- *PBG* ist **effizient**, also in polynomialer Zeit berechenbar.
- *PBG* ist **sicher**, d.h. die Ergebnisse sind durch keinen polynomialen Test von echten Zufallsfolgen unterscheidbar. (Genauer siehe unten.)

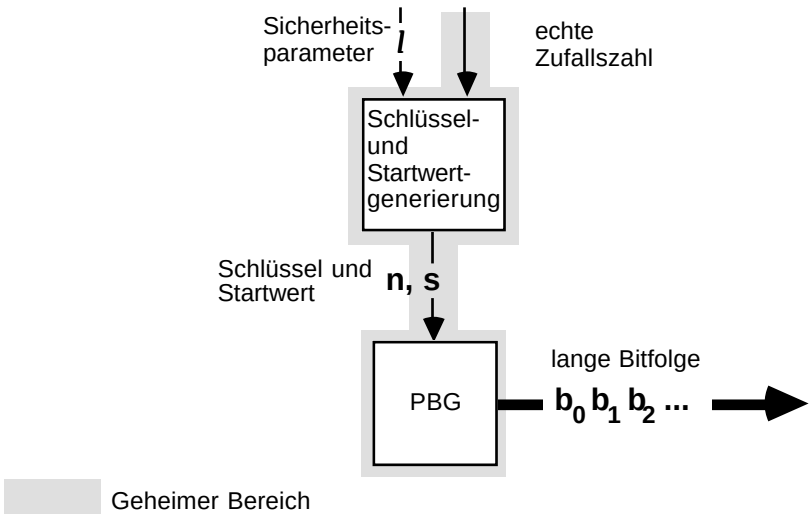


Bild 3-17: Pseudozufallsbitfolgengenerator

Verwendung als Pseudo-one-time-pad

Eine wichtige Anwendung ist, die Pseudozufallsbits statt einem one-time-pad (siehe §3.2) als symmetrisches Konzelationssystem zu verwenden (siehe Bild 3-18): Statt einer echten Zufallsfolge wird eine Pseudozufallsfolge bitweise modulo 2 zum Klartext addiert. Diese Idee stammt von de Vigenère (1523-1596), dem zu Ehren das Pseudo-one-time-pad manchmal *Vigenère-Chiffre* genannt wird.

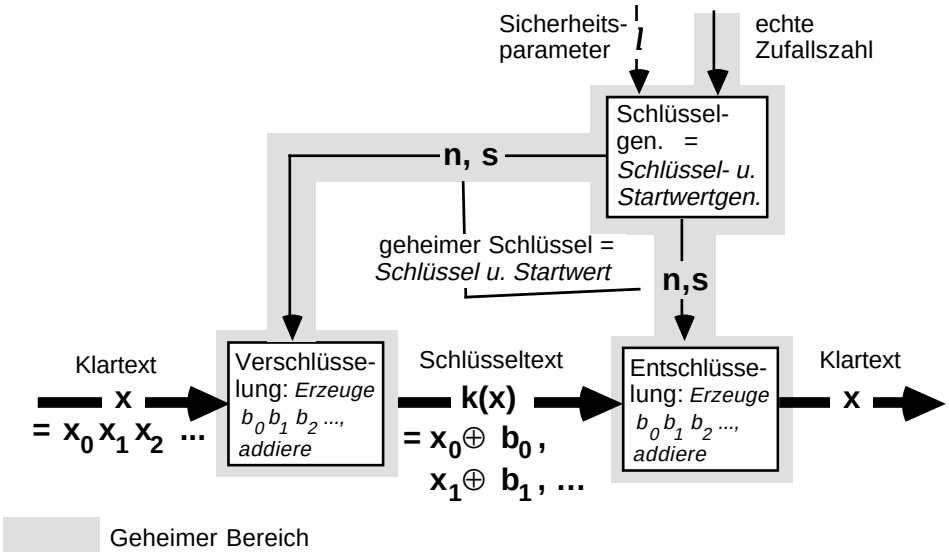


Bild 3-18: Pseudo-one-time-pad

Dabei muß statt des langen one-time-pads nur noch der kurze Schlüssel und Startwert des PBG ausgetauscht werden.

Forderung an Pseudo-one-time-pad: Für jeden polynomial beschränkten Angreifer soll das Pseudo-one-time-pad bis auf einen verschwindenden Anteil der Fälle so sicher wie one-time-pad sein.

Die Idee dazu ist, daß wenn kein polynomialer Test (also auch kein polynomialer Angreifer) die Pseudo-one-time-pads und die echten one-time-pads unterscheiden kann, es dann bzgl. polynomialer Angreifer auch keinen Unterschied machen kann, ob man mit einem Pseudo-one-time-pad oder einem echten one-time-pad verschlüsselt.

Forderungen an PBG

„Stärkste“ Forderung: PBG besteht *jeden* probabilistischen Test *T* polynomialer Laufzeit.

Dabei heißt „**besteht**“: Der Test kann Folgen, die der PBG erzeugt, nicht mit signifikanter Wahrscheinlichkeit von echten Zufallsbitfolgen unterscheiden.

Ein probabilistischer **Test** *T* polynomialer Laufzeit ist ein probabilistischer, polynomial zeitbeschränkter Algorithmus, der jeder Eingabe aus $\{0, 1\}^*$ (also einer endlichen Folge, die er testen soll) eine reelle Zahl aus [0, 1] zuordnet. (Der Wert hängt im allgemeinen von den Zufallsentscheidungen in *T* ab.)

Die Grundidee ist nun, daß nicht jede einzelne Folge betrachtet wird (denn jede einzelne endliche Folge kann ja mit gleicher Wahrscheinlichkeit von einem echten Zufallsgenerator erzeugt werden, d.h. von der Zufälligkeit einer einzelnen Folge zu sprechen, ist für die Kryptographie sinnlos). Statt dessen vergleicht man den Durchschnittswert des Tests für die PBG-Folgen mit dem Durchschnittswert für echte Folgen. Wenn im Mittel bei beiden Sorten von Folgen dasselbe herauskommt, kann der Test nicht helfen, die PBG-Folgen von den echten Zufallsfolgen zu unterscheiden.

Sei dazu

- α_m der durchschnittliche Wert, den *T* einer echt zufälligen *m*-Bit-Folge zuordnet (d.h. der Erwartungswert bei Gleichverteilung über alle *m*-Bit-Folgen überhaupt), und

- $\beta_{l,m}$ der durchschnittliche Wert, den T einer von PBG generierten m-Bit-Folge zuordnet, wenn der Sicherheitsparameter l ist (wobei der Durchschnitt über die Schlüssel und Startwerte gebildet wird, die beim Sicherheitsparameter l erzeugt werden).

Damit gilt formaler, d.h. bei Umstellung des formalen Parameters m auf beliebiges Polynom $Q(l)$:

Ein PBG besteht T gdw.
Für alle Polynome Q und alle $t > 0$ gilt: Für alle genügend großen l liegt $\beta_{l,Q(l)}$ im Intervall $\alpha_{Q(l)} \pm 1/l^t$.

Zu dieser „stärksten" Forderung sind die folgenden 3 äquivalent (aber leichter beweisbar):
Für jede erzeugte endliche Anfangs-Bitfolge, bei der ein beliebiges (das rechte, linke) Bit fehlt, kann jeder polynomial zeitbeschränkte Algorithmus P (**Prädiktor**) das fehlende Bit „nur raten" (d.h. nicht besser voraussagen als mit Wahrscheinlichkeit $1/2 \pm 1/l^t$.).

Beweisideen (indirekte Beweise):

1. (Einfacher Teil) Jeder Prädiktor ist auch als Test zu verwenden: Man nimmt von der zu testenden Folge alle Bits bis auf eines, gibt sie als Eingabe in den Prädiktor und vergleicht das Ergebnis mit dem richtigen Bit aus der Folge. Das Testergebnis ist 1, wenn der Prädiktor richtig geschätzt hat, und 0 sonst. Der Wert α_m ist 1/2, denn bei echten Zufallsfolgen kann jeder Prädiktor nur mit Wahrscheinlichkeit 1/2 richtig schätzen. Wenn der Prädiktor also auf PBG-Folgen wesentlich anders (besser) schätzt, hat der PBG diesen Test nicht bestanden.

2. (Schwierigerer Teil) Zu zeigen: Aus jeder der 3 schwächeren Forderungen folgt die „stärkste".
   Wir nehmen also an, es gebe irgendeinen Test T, den PBG nicht besteht, und zeigen, daß es dann auch einen Prädiktor gibt, der ein Bit von PBG zu gut voraussagt. Hier wird das für das linke Bit gezeigt.
   Für ein $t > 0$, ein Polynom Q und unendlich viele l liegt $\beta_{l,Q(l)}$ außerhalb, z.B. oberhalb, des Intervalls $\alpha_{Q(l)} \pm 1/l^t$.
   Man wirft nun T eine Bitfolge aus 2 Teilen vor, mit den Längen $j + k = Q(l)$, und zwar ist der linke Teil echt zufällig und der rechte von PBG generiert. Man vergleicht immer zwei benachbarte Typen von Folgen.

	echt zufällig				
A_{j+1} =	$\{r_1 \ldots r_j$	r_{j+1}	$b_1 \ldots b_k\}$		ergibt Testergebnisse näher bei $\alpha_{Q(l)}$
A_j =	$\{r_1 \ldots r_j$	b_0	$b_1 \ldots b_k\}$		ergibt Testergebnisse weiter weg, z.B. höher.
		von PBG generiert			

   Obiges muß mindestens für ein j gelten, da A_0 die PBG-Folgen sind und $A_{Q(l)}$ die echten Zufallsfolgen

   Daraus konstruiert man einen Prädiktor für die Bitfolge $b_1 \ldots b_k$ folgendermaßen: Man wählt sich eine echte Zufallsfolge $r_1 \ldots r_j$ dazu und wendet zweimal den Test T an, nämlich
   T auf $\{r_1 \ldots r_j \quad 0 \quad b_1 \ldots b_k\}$: Nenne das Ergebnis α^0
   T auf $\{r_1 \ldots r_j \quad 1 \quad b_1 \ldots b_k\}$: Nenne das Ergebnis α^1
   Rate $b_0 = 0$ mit Wahrscheinlichkeit $1/2 + 1/2\,(\alpha^0 - \alpha^1)$.

(Genauer: [BlBS_86 Seite 375f])

3.4.3 Der s^2-mod-n-Generator

Der s^2-mod-n-Generator ist ein bestimmter Pseudozufallsbitfolgengenerator, der mit Quadraten (und der Schwierigkeit des Wurzelziehens bzw. des Entscheidens, ob etwas ein Quadrat ist) arbeitet.

(In der Original-Publikation [BlBS_86] wird er x^2-mod-N-Generator genannt. Da hier x als Variablenbezeichner für den Klartext benutzt wird und da in vielen anderen Papieren statt des großen N ein kleines n als Bezeichner für den aus zwei großen Primzahlen zusammengesetzten Modulus verwendet wird, führe ich der Konsistenz wegen diese neue Schreibweise ein.)

Def. des s^2-mod-n-Generators:

Als Schlüssel wähle $n = p \cdot q$ mit $|p| \approx |q|$ und p, q Primzahlen $\equiv 3 \bmod 4$.
(Betragsstriche bedeuten hier „Länge“ in Bits. p und q werden zufällig und unabhängig gewählt, wobei die in §3.4.1 genannten Bedingungen an p und q eingehalten werden müssen – sonst kann n faktorisiert werden.)

Als Startwert wähle s zufällig in $\mathbb{Z}_n^* = \{a \in \mathbb{Z} \mid 0 < a < n, \text{ggT}(a, n) = 1\}$.

Generierung der Folge:
Berechne sukzessive die inneren Zustände $s_0 := s^2 \bmod n$, $s_{i+1} := s_i^2 \bmod n$,
und bilde in jedem Schritt $b_i := s_i \bmod 2$ (= letztes Bit von s_i)

Ausgabe: $b_0\, b_1\, b_2 \ldots$

(Als Merkhilfe für den Bezeichner s_i: innerer Zustand = Status = Stand des Generators)

Beispiel:

$n = 3 \cdot 11 = 33, \quad s = 2$

s_i:	4	16	25	31	4	…
b_i:	0	0	1	1	0	…

Sicherheitsbetrachtung

Die Sicherheit des s^2-mod-n-Generators kann unter der Faktorisierungsannahme bewiesen werden [ACGS_84, ACGS_88]. Hier wird aber nur der etwas einfachere Beweis unter der Quadratische-Reste-Annahme (QRA) gezeigt (siehe §3.4.1.9).

Beh. Unter QRA ist der s^2-mod-n-Generator ein unvorhersagbarer (kryptographisch starker) PBG.

Gemäß dem obigen Absatz über Forderungen an PBG genügt es, zu beweisen, daß es keinen Prädiktor gibt, der das linkeste Bit von Folgen, die der s^2-mod-n-Generator erzeugt, gut aus den restlichen Bits voraussagen kann. Genauer heißt das:

a) Ein Prädiktor $P[\bullet, \bullet]$ ist ein probabilistischer, polynomial zeitbeschränkter Algorithmus, der bei Eingabe von $n, b_1 \ldots b_k$ ($b_i \in \{0, 1\}$) den Wert 0 oder 1 ausgibt.

b) Man sagt, P habe für ein gewisses n einen ε-Vorteil beim Vorhersagen nach links von k-Bit-Folgen des s^2-mod-n-Generators gdw.

$$\frac{\sum_{s \in QR_n} W(P[n, b_1(s) \ldots b_k(s)] = b_0(s))}{\phi(n) \,/\, 4} \quad > \quad \frac{1}{2} + \varepsilon$$

mit $b_i(s)$ = letztes Bit von $s^{2^i} \bmod n$. (Beachte, daß $\phi(n)/4$ die Stichprobengröße ist, da s beim Vorhersagen nach links ein quadratischer Rest sein muß – sonst kann man nicht Wurzelziehen. Im Gegensatz dazu darf bei der Generierung der Folge nach rechts s beliebig aus $\mathbb{Z}_n^*$ sein.)

c) Der s^2-mod-n-Generator ist sicher gdw. für jeden Prädiktor P, jede Konstante $0 < \delta < 1$, jedes Polynom Q und jede natürliche Zahl t gilt: Sofern l genügend groß ist, hat P für alle (außer einem

δ-Anteil) Zahlen n der Länge l höchstens einen $1/l^t$ Vorteil für n beim Vorhersagen von $Q(l)$-bit-Folgen nach links.

Bew. (Skizze)

Annahme: Es gibt einen Prädiktor P für den s^2-mod-n-Generator mit ε-Vorteil auf Zahlen der Länge l. (Um den Beweis zu vervollständigen, müssen alle Quantoren über das ε wie bei c) gesetzt werden.)

1. Schritt: P kann effizient und uniform in ein Verfahren P^* transformiert werden, das mit einem ε-Vorteil das letzte Bit von s_0 zu gegebenem s_1 aus QR_n rät:

   Gegeben sei s_1. Bilde $b_1\ b_2\ b_3$ ... mit dem s^2-mod-n-Generator. Nun wende P auf diese Folge an. Dann rät P das Bit b_0 mit ε-Vorteil. Genau dies ist das passende Ergebnis von P^*.

2. Schritt: Sei nun das Verfahren P^* gegeben. Daraus wird effizient und uniform ein Verfahren R konstruiert, das mit ε-Vorteil rät, ob ein gegebenes s^* mit Jacobi-Symbol +1 ein Quadrat ist:

   Gegeben sei s^*. Setze $s_1 := s^{*2}$. Wende P^* auf s_1 an. P^* berechnet also das letzte Bit von s_0 mit ε-Vorteil. Dabei sind s^* und s_0 Wurzeln von s_1, und zwar ist s_0 genau die eine, die selbst ein Quadrat ist[63]. Also gilt:

   $$s^* \in QR_n \Leftrightarrow s^* = s_0.$$

   Anhand der letzten Bits, also des bekannten b^* von s^* und des geratenen b_0 von s_0 soll nun entschieden werden, ob $s^* = s_0$ ist. Klar ist: Wenn $s^* = s_0$, dann müssen auch ihre letzten Bits gleich sein. Gezeigt wird nun, daß umgekehrt aus $s^* \neq s_0$ auch folgt, daß die letzten Bits verschieden sind: Wenn $s^* \neq s_0$, dann ist $s^* \equiv -s_0 \bmod n$ (wegen den gleichen Jacobi-Symbolen), also $s^* = n - s_0$ in $\mathbb{Z}$. n ist aber ungerade, also haben s^* und s_0 verschiedene letzte Bits.

   Also endet das Verfahren R so: Genau dann, wenn $b^* = b_0$, wird geraten, daß s^* ein Quadrat ist, und rät damit genausogut wie P^*.

3. Schritt: Das so konstruierte Verfahren R steht im Widerspruch zur QRA. □

Anmerkung: Man kann zeigen, daß man auch mehr als ein Bit pro Quadrierungsschritt nehmen kann, nämlich $O(\log(l))$.

3.4.4 s^2-mod-n-Generator als asymmetrisches Konzelationssystem

Man kann den s^2-mod-n-Generator auch als asymmetrisches Konzelationssystem verwenden (Bild 3-19):

- Der geheime Schlüssel besteht aus den Faktoren p und q von n.
- Der öffentliche Schlüssel ist n.
- Ein Sender und ein Empfänger haben keinen geheimen Startwert s mehr miteinander ausgetauscht, sondern zu jeder Nachricht wählt sich der Sender einen neuen, zufälligen Startwert s. (Dadurch wird die *Verschlüsselung indeterministisch*, d.h. gleiche Klartexte werden nicht notwendigerweise als gleiche Schlüsseltexte verschlüsselt, vgl. §3.1.1.1 Anmerkung 2.) Damit der Empfänger trotzdem entschlüsseln kann, soll ausgenutzt werden, daß er mittels p und q Wurzeln modulo n ziehen kann.

---

63 Dies wurde schon in Aufgabe 3-9 k) gezeigt: Daß nur eine der 4 Wurzeln selbst ein Quadrat ist, folgt, weil $p \equiv q \equiv 3 \bmod 4$. Wie in §3.4.1.6 gezeigt, ist dann von den zwei Wurzeln $\pm w_p$ aus $s_1 \bmod p$ nur eine ein Quadrat, und genauso für $\pm w_q$. Gemäß §3.4.1.7 sind die 4 Wurzeln von s_1 Kombinationen aus $\pm w_p$ und $\pm w_q$. Zuletzt ist (auch nach §3.4.1.7) eine Kombination genau dann ein Quadrat, wenn beide Teile es sind.

- Damit der Empfänger auch etwas hat, woraus er Wurzeln ziehen kann, schicken ihm die Sender außer dem bisherigen Schlüsseltext

$$x_0 \oplus b_0, x_1 \oplus b_1, \ldots, x_k \oplus b_k$$

auch den Wert

$$s_{k+1}$$

zu, also den Wert, dessen letztes Bit b_{k+1} geworden wäre, wenn man das noch gebraucht hätte.

- Der Empfänger entschlüsselt nun so: Er berechnet aus s_{k+1} rückwärts die ganze Folge der s_i, jeweils

$$s_{i-1} := \text{die Wurzel aus } s_i, \text{ die selbst ein Quadrat ist}$$
$$= \text{CRA}(s_i^{(p+1)/4} \bmod p, s_i^{(q+1)/4} \bmod q).^{64}$$

Dann kann er die letzten Bits b_i von s_i bestimmen und damit die Klartextbits x_i zurückerhalten.

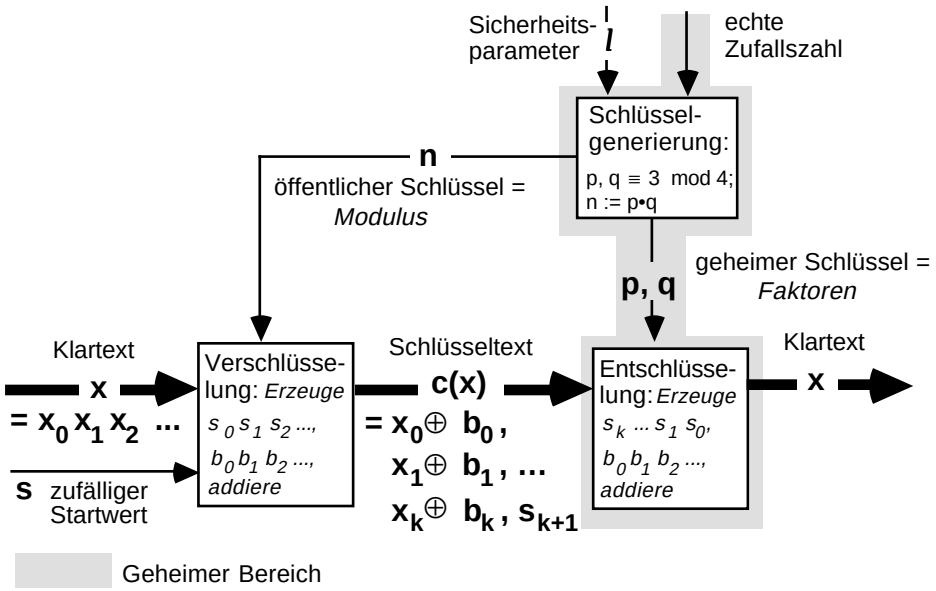


Bild 3-19: s^2-mod-n-Generator als asymmetrisches Konzelationssystem

Sicher gegen passive Angriffe: Ähnlich wie oben kann man beweisen, daß das Verfahren gegen passive Angriffe sicher ist. (Man muß nur noch berücksichtigen, daß der Angreifer jeweils auch s_{k+1} sieht.)

Unsicher gegen aktive Angriffe: Ziel des aktiven Angriffs ist es, einen vom Angreifer beobachteten Schlüsseltext $x_0 \oplus b_0, x_1 \oplus b_1, \ldots, x_k \oplus b_k, s_{k+1}$ zu entschlüsseln. Hierzu erhält der Angreifer beliebige, hiervon verschiedene Schlüsseltexte entschlüsselt.

Ein einfacher *gewählter Schlüsseltext-Klartext-Angriff* kann folgendermaßen geschehen (Bild 3-20): Der Angreifer quadriert s_{k+1} und erhält so s_{k+2}. Er schickt dem Opfer

$$x_0 \oplus b_0, x_1 \oplus b_1, \ldots, x_k \oplus b_k, 1, s_{k+2},$$

also den beobachteten Schlüsseltext bis auf den inneren Zustand s_{k+1}, danach ein eingefügtes Bit, hier 1, und danach den „nächsten“ inneren Zustand s_{k+2}. Die ersten k+1 Bits, die der Angreifer als Antwort erhält, sind der ihn interessierende Klartext.

---

64 Hierfür benötigt man eigentlich noch die Bedingung $p \neq q$. Sie kann bei der Schlüsselgenerierung aber weggelassen werden, da sie bei zufälliger und unabhängiger Wahl von p und q nur in einem exponentiell kleinen Anteil aller Fälle verletzt wäre.

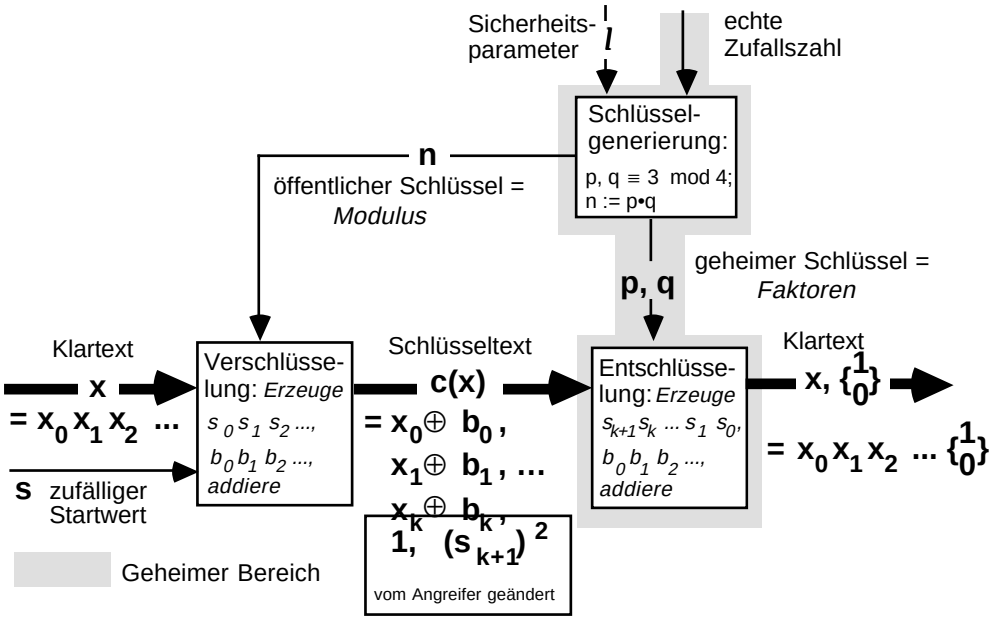


Bild 3-20: Gewählter *Schlüsseltext-Klartext-Angriff* auf s^2-mod-n-Generator als asymmetrisches Konzelationssystem

Dieser einfache Angriff könnte bei einer praktischen Anwendung durch Vergleich erkannt und dann durch Nichtantwort des Opfers vereitelt werden. Er kann aber in zweierlei Weise variiert und dann nur viel schwerer erkannt werden:

1. Der Angreifer ersetzt $x_0 \oplus b_0, x_1 \oplus b_1, \ldots, x_k \oplus b_k$ durch k+1 zufällige Bits. Aus der Antwort des Opfers kann er dann die Pseudozufallsbitfolge errechnen und damit den ursprünglichen Schlüsseltext entschlüsseln.
2. Der Angreifer quadriert nicht einmal, sondern j mal, berechnet also s_{k+1+j}. Er fügt entsprechend auch j Bits ein, deren Wert bei der Antwort ihn nicht interessiert. Selbst wenn Antworten jeweils nur k+1-Bit-Nachrichten zulassen, muß der Angreifer nur auf die ersten 1+j Bits des Klartextes verzichten.

Noch gefährlicher als diese beschriebenen *nachrichtenbezogenen* Angriffe ist, daß mittels eines aktiven Angriffs sogar ein vollständiges Brechen, d.h. das Faktorisieren des Modulus, möglich ist. Dies ist hier noch nicht zu sehen, in [GoMT_82 Algorithm 5] aber nachzulesen.

3.5 GMR: Ein kryptographisch starkes Signatursystem

GMR, benannt nach den ersten Buchstaben der Nachnamen seiner Erfinder Shafi Goldwasser, Silvio Micali und Ronald L. Rivest, ist historisch das erste praktikable, kryptographisch starke Signatursystem. Man kann zeigen, daß es selbst bei einem adaptiven aktiven Angriff unmöglich ist, auch nur eine neue Signatur zu fälschen (egal unter was für eine sinnlose Nachricht), wenn die Faktorisierungsannahme gilt. (Vgl. in §3.1.3: Es ist sicher gegen den schwächsten Erfolg c2 und beim stärksten Angriff b. adaptiv.) Veröffentlicht wurde es in [GoMR_88].

Der Sicherheitsbeweis ist relativ komplex; deswegen werden hier nur die Funktionsweise des Systems und einige wesentliche Schritte des Beweises dargestellt.[65] Man kann auch einen Teil der Eigenschaften von GMR aus der Sicherheit gegen adaptive aktive Angriffe herleiten; hier wird aber ein einfacher bottom-up-Zugang gewählt:

1. werden Permutationenpaare eingeführt, bei denen unter einer kryptographischen Annahme nur, wer ein Geheimnis kennt, Kollisionen finden kann. Solche Permutationenpaare werden *kollisionsresistent (collision resistant)* genannt[66].
2. wird gezeigt, wie mit Hilfe eines kollisionsresistenten Permutationenpaares eine Ein-Bit-Nachricht signiert werden kann. Hierzu wird neben dem Permutationenpaar eine authentische *Referenz* benötigt, die etwa als Teil des öffentlichen Testschlüssels veröffentlicht wird.
3. werden aus *einem* kollisionsresistenten Permutationenpaar *viele* kollisionsresistente Permutationen konstruiert.
4. wird erklärt, wie ausgehend von *einer* authentischen Referenz *viele* baumförmig authentisiert werden können, so daß dann mittels vieler kollisionsresistenter Permutationen viele Nachrichten signiert werden können.
5. werden Effizienzverbesserungen aufgezeigt.

3.5.1 Grundfunktion: Kollisionsresistente Permutationenpaare

Die Grundfunktion von GMR sind sog. ***kollisionsresistente Permutationenpaare mit Geheimnis*** („claw-resistant permutation pairs with trap-door“).[67]

Eine Kollision von zwei Permutationen f_0, f_1 auf demselben Definitionsbereich ist ein Paar (x, y) von Werten mit $f_0(x)=f_1(y)$ (Bild 3-21).

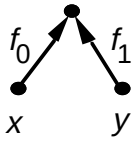


Bild 3-21: Kollision eines Paars von Permutationen

„Kollisionsresistent“ heißt, daß es unter einer kryptographischen Annahme unmöglich sein soll, solche Kollisionen zu finden, wenn man das Geheimnis nicht kennt. (Obwohl sie existieren: Da es Permutationen auf demselben Definitionsbereich sind, gibt es zu jedem z genau ein x mit $f_0(x) = z$ und genau ein y mit $f_1(y) = z$.)

---

65 Das folgende wurde mit freundlicher Genehmigung teilweise kopiert aus [FoPf_91].

66 In der Literatur findet sich oft noch eine ältere, ungeschickte Begriffsbildung. Statt von kollisionsresistent (collision resistant), was bedeuten soll kollisions„schwierig“, wird von kollisionsfrei (collision free) gesprochen – was im Wortsinn natürlich nicht zutreffen kann.

67 Hier wird in der Originalpublikation von kollisions„freien“ Permutationenpaaren (claw-free permutation pairs with trap-door) gesprochen, die es im Wortsinn nicht gibt. Deshalb verwende ich auch hier den Begriff kollisionsresistent.

Der Besitzer des Geheimnisses soll aber die Inversen der Funktionen berechnen können.

Konstruktion unter Faktorisierungsannahme: Solche kollisionsresistenten Permutationenpaare kann man unter der Faktorisierungsannahme mit den üblichen Geheimnissen konstruieren (vgl. §3.4.1): Das Geheimnis sind wieder zufällig und stochastisch unabhängig gewählte große Primzahlen p und q, diesmal sogar mit der Anforderung

$$p \equiv 3 \bmod 8,\ q \equiv 7 \bmod 8.$$

Wie in §3.4.1 unter „Zur Primzahlerzeugung“ erwähnt, gibt es auch mit diesen Eigenschaften genug Primzahlen.

Die Permutationen sind im wesentlichen die Quadrierfunktion, von der wir ja schon wissen, daß das Geheimnis zum Invertieren, d.h. hier zum Wurzelziehen hilft.

Erstens ist aber die Quadrierfunktion keine Permutation, sondern bildet immer 4 Wurzeln auf dasselbe Quadrat ab (vgl. §3.4.1.7). Deswegen müssen wir den Definitionsbereich so einschränken, daß von diesen vieren immer nur noch genau eine darin liegt. Dazu wählt man

$$D_n = \{x \in \mathbb{Z}_n^* \mid (\tfrac{x}{n})=1,\ x < n/2\}.^{68}$$

Das Zeichen „<“ auf Zahlen modulo n bezieht sich auf die Standarddarstellung durch $0,\ldots,n-1$.

Zweitens müssen wir nun auch dafür sorgen, daß das Ergebnis wieder im Definitionsbereich liegt. Das geht aber einfach (vgl. §3.4.1.8): Das Ergebnis y beim Quadrieren ist ein Quadrat, hat also Jacobi-Symbol +1. Wenn $y \geq n/2$ ist, nimmt man $-y$. Das ist dann $< n/2$, und sein Jacobi-Symbol ist auch +1, weil $(\tfrac{-1}{n}) = +1$.

Drittens brauchen wir eine zweite Permutation mit ähnlichen Eigenschaften wie x^2. Dazu nimmt man $4x^2 = (2x)^2$. Damit ist folgende Wahl motiviert:

$$f_0(x) := \begin{cases} x^2 \bmod n, \text{ falls } (x^2 \bmod n) < n/2 \\ -x^2 \bmod n \text{ sonst.} \end{cases}$$

$$f_1(x) := \begin{cases} 4 \cdot x^2 \bmod n, \text{ falls } (4 \cdot x^2 \bmod n) < n/2 \\ -4 \cdot x^2 \bmod n \text{ sonst.} \end{cases}$$

Skizze, daß dies kollisionsresistente Permutationenpaare sind: Daß f_0 eine Permutation ist, wurde schon gezeigt. Für f_1 zeigt man es analog.

Für die Kollisionsresistenz wird gezeigt, daß man mit jeder Kollision sofort faktorisieren kann. Dazu braucht man allerdings noch einen unbewiesenen Hilfssatz, nämlich daß für $p \equiv 3 \bmod 8$, $q \equiv 7 \bmod 8$:

$$(\tfrac{2}{p}) = -1 \quad \text{und} \quad (\tfrac{2}{q}) = +1.$$

(Zweiter Ergänzungssatz zum quadratischen Reziprozitätsgesetz [ScSc_73 Seite 83])

---

68 Daß tatsächlich von den 4 Wurzeln immer genau eine diese Eigenschaft hat, folgt aus $p \equiv q \equiv 3 \bmod 4$ so:

a) Nach §3.4.1.8 sind die 4 Wurzeln CRA($\pm w_p$, $\pm w_q$), wobei $\pm w_p$, $\pm w_q$ die Wurzeln mod p bzw. q sind.

b) Nach §3.4.1.6 gilt: Von $\pm w_p$ hat eine das Jacobi-Symbol +1, die andere –1. Gleiches gilt für $\pm w_q$. Seien o.B.d.A. w_p,w_q diejenigen mit Jacobi-Symbol +1.

c) Allgemein gilt, wenn z = CRA(x, y): Nach Definition des Jacobi-Symbols ist $(\tfrac{z}{n}) = (\tfrac{z}{p}) \cdot (\tfrac{z}{q})$. Das Jacobi-Symbol von z modulo p hängt nur von der Restklasse mod p ab, also $(\tfrac{z}{p}) = (\tfrac{x}{p})$ und analog $(\tfrac{z}{q}) = (\tfrac{y}{q})$. Also ist $(\tfrac{z}{n}) = (\tfrac{x}{p}) \cdot (\tfrac{y}{q})$.

d) Also sind die Jacobi-Symbole von w := CRA(w_p, w_q) und von $-w$ = CRA($-w_p$, $-w_q$) jeweils +1, die der anderen zwei Wurzeln –1. Unter w und $-w$ ist genau eine $< n/2$.

Daraus folgt $\left(\frac{2}{n}\right) = -1.$

Nehmen wir nun an, jemand habe eine Kollision gefunden, also x, y mit $f_0(x)=f_1(y)$. Dann gilt

$$x^2 \equiv \pm (2y)^2 \bmod n.$$

Das Vorzeichen „–“ ist nicht möglich, weil –1 kein Quadrat mod n ist. Also sind x und $2y$ zwei Wurzeln derselben Zahl. Nach §3.4.1.9 können wir damit definitiv faktorisieren, wenn sie wesentlich verschieden sind, d.h.

$$x \neq \pm 2y.$$

Dies folgt daraus, daß sie schon verschiedene Jacobi-Symbole haben: Wegen $x, y \in D_n$ haben x und y das Jacobi-Symbol +1, und wegen §3.4.1.8 gilt

$$\left(\frac{\pm 2y}{n}\right) = \left(\frac{\pm 1}{n}\right) \cdot \left(\frac{2}{n}\right) \cdot \left(\frac{y}{n}\right) = 1 \cdot (-1) \cdot 1 \neq 1 = \left(\frac{x}{n}\right).$$

Das Umkehren mittels Geheimnis geht so: Ist $x = f_0^{-1}(y)$ gesucht, so testet man zunächst, ob y selbst ein Quadrat ist (s. §3.4.1.7). Andernfalls ist nach Definition von f_0: $y = -x^2$, d.h. $-y = x^2$. Man zieht also die Wurzel aus y bzw. $-y$ nach dem Verfahren aus §3.4.1.8. Die so erhaltene Wurzel w ist selbst ein Quadrat (vgl. §3.4.1.6), hat also das Jacobi-Symbol +1. Je nachdem, ob sie $< $ oder $\geq n/2$ ist, setzt man $x = +w$ bzw. $-w$.

Für $f_1^{-1}(y)$ ist zusätzlich nach dem Wurzelziehen modulo n durch 2 zu dividieren. Da 2 das Jacobi-Symbol –1 hat, $f_1^{-1}(y)$ aber im Definitionsbereich der Permutationen liegen, also Jacobi-Symbol 1 haben muß, muß wegen der Multiplikativität des Jacobi-Symbols der Wert nach dem Wurzelziehen Jacobi-Symbol –1 haben. Hierzu werden nach dem Wurzelziehen mod p und mod q die Ergebnisse mit unterschiedlichem Vorzeichen in den CRA eingesetzt. □

Allgemeineres: Wenn man kollisionsresistente Permutationenpaare mit Geheimnis präzis definiert (s. [GoMR_88]), benötigt man nicht nur ein Paar (f_0, f_1), sondern eine ganze Familie davon in Abhängigkeit von einem Schlüssel, damit jeder, der so ein Paar braucht, eins wählen kann, von dem nur er das Geheimnis kennt. Dazu gehört ein Schlüsselgenerierungsalgorithmus *gen*, mit dem man das Geheimnis und den öffentlichen Schlüssel wählen kann.

Bei der konkreten Konstruktion war das automatisch der Fall: *gen* ist der Algorithmus zur Wahl von p und q, die das Geheimnis sind, und zur Berechnung des öffentlichen Schlüssels n. Für jedes n ergibt sich ein anderes Paar, man müßte also genauer $f_0^{(n)}, f_1^{(n)}$ schreiben.

Die Kollisionsresistenz wird wieder so definiert, daß kein polynomialer Algorithmus für lange Schlüssel mit signifikanter Wahrscheinlichkeit Kollisionen findet, analog zur Faktorisierungsannahme.

Außerdem braucht man, daß man allein mit dem öffentlichen Schlüssel effizient ein zufälliges Element des jeweiligen Definitionsbereichs wählen kann. Konkret geht das so:

Wahl eines zufälligen Elementes aus D_n: Man wählt $z \in \mathbb{Z}_n^*$ zufällig und quadriert es; ist $z^2 < n/2$, so wählt man $x := z^2$, andernfalls $x := -z^2$.

3.5.2 Mini-GMR für eine Nachricht aus nur einem Bit

Wenn nur eine Nachricht signiert werden soll, und diese nur ein Bit lang sein wird, kann man einfach ein kollisionsresistentes Permutationenpaar nehmen: Der geheime Signierschlüssel s ist (p, q); der öffentliche Testschlüssel t besteht aus n und einer sogenannten Referenz R, die der Unterzeichner zufällig aus D_n wählt.

Die Signatur unter die Nachricht $m = 0$ ist $f_0^{-1}(R)$, die unter $m = 1$ ist $f_1^{-1}(R)$ (Bild 3-22).

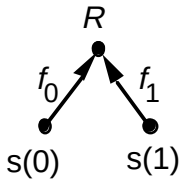


Bild 3-22: Mini-GMR für eine 1-Bit-Nachricht. *s*(0) und *s*(1) sind die möglichen Signaturen.

Sicherheit, Skizze: Einen echten aktiven Angriff gibt es hier nicht, da nur eine Nachricht signiert wird. Es kann aber sein, daß ein Fälscher die Signatur unter $m = 0$ erhalten hat und daraus die unter $m=1$ fälschen möchte oder umgekehrt. Man ahnt anhand von Bild 3-22, daß das heißt, daß der Angreifer eine Kollision findet.

Ganz so einfach ist es aber nicht: Er hat ja bei der einen Hälfte der Kollision die Hilfe des Unterzeichners gehabt, der das Geheimnis kennt und sowieso Kollisionen berechnen kann. Wir müssen zeigen, daß der Fälscher auch ohne diese Hilfe an *irgendeine* Kollision kommt:

O.B.d.A. wird nur der erste Fall betrachtet, also $m = 0$ ist gegeben. Wir nehmen an, daß es hierfür einen guten Fälschungsalgorithmus F mit Eingabe (n, R, $s(0)$) gäbe, und zeigen, daß es dann auch einen guten Kollisionsfinde-Algorithmus K gibt:

K: Eingabe n.
Wähle eine „Signatur“ $S_0 \in D_n$ zufällig. (Hier wird gebraucht, daß das allein mit dem öffentlichen Schlüssel geht. Da S_0 nicht mit dem Signieralgorithmus gewählt wird, wird es nicht mit $s(0)$ bezeichnet. Entsprechendes gilt unten für die Bezeichnung S_1 statt $s(1)$.)
Berechne $R := f_0(S_0)$.
Verwende nun F mit Eingabe (n, R, S_0). Wenn F erfolgreich ist, nenne das Ergebnis S_1.
Ausgabe: (S_0, S_1).

Klar ist, daß dieses K immer eine Kollision findet, wenn sein Aufruf von F erfolgreich ist. Zu zeigen ist also nur, daß F hier genauso oft erfolgreich ist, wie wenn es mit einer „echten“ Referenz und Signatur aufgerufen wird. Echte Referenzen sind zufällig, und das hiesige R ist auch zufällig (weil S_0 zufällig ist und f_0 eine Permutation ist, also jedes R bei genau einen S_0 vorkommt). Bei diesem R ist S_0 die (einzige) richtige Signatur. □

3.5.3 Grundsignaturen: Große Tupel kollisionsresistenter Permutationen

Wir bauen jetzt als zweiten Schritt GMR so aus, daß eine beliebig lange Nachricht signiert werden kann (aber nach wie vor nur eine einzige).

Die Idee ist dieselbe wie oben für ein Bit: Man hätte gern für jede dieser Nachrichten eine Funktion f_m (so wie oben f_0 und f_1 für die zwei Nachrichten $m = 0$ bzw. 1). Als öffentlichen Schlüssel wählt man wieder zusätzlich zu n eine Referenz R[69], und die Signatur unter die Nachricht m soll

$$s(m) := f_m^{-1}(R)$$

sein (Bild 3-23), wobei man f_m^{-1} wieder nur mit Hilfe des Geheimnisses berechnen kann.

69 Für diesen vereinfachten Fall, daß die Referenz dem Angreifer *vor* seiner Wahl der zu signierenden Nachricht bekannt ist, ist die Sicherheit von GMR nicht bewiesen; wir sehen auch nicht, wie sie bewiesen werden könnte.

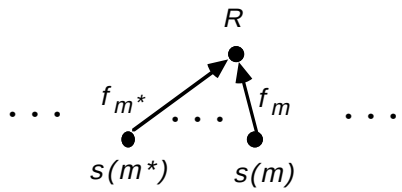


Bild 3-23: Prinzip der Grundsignatur von GMR

Nun kann man natürlich nicht jede der vielen Funktionen f_m einzeln angeben. Deswegen werden sie aus einem Permutationenpaar konstruiert:

Grundkonstruktion der Funktionen f_m aus einem Paar f_0, f_1: Hat man ein kollisionsresistentes Permutationenpaar (f_0, f_1) gewählt, so läßt sich zu jeder Nachricht m eine Funktion f_m konstruieren, indem f_0 und f_1 bitweise in Abhängigkeit von m komponiert werden.

Beispiel: m=100101, $f_{100101}(x) := f_1(f_0(f_0(f_1(f_0(f_1(x))))))$.

Mit Kenntnis des Geheimnisses lassen sich die f_0^{-1} und f_1^{-1} berechnen und damit auch f_m^{-1}; die Auswertungsrichtung von m kehrt sich dabei um.

Beispiel: m=100101, $f_{100101}^{-1}(x) := f_1^{-1}(f_0^{-1}(f_1^{-1}(f_0^{-1}(f_0^{-1}(f_1^{-1}(x))))))$.

Ein Vorteil dieses Vorgehens ist, daß f_m für beliebig lange Nachrichten m existiert. Es ist also keine Aufteilung von m in Blöcke oder der Einsatz einer Hashfunktion notwendig.

Die Komposition von f_0 und f_1 birgt jedoch eine Gefahr: Der „zurückrechnende" Empfänger, der die Signatur mit f_m testet, erhält mit jeder Anwendung von f_0 und f_1 wieder eine gültige Signatur zu der um jeweils das letzte Bit verkürzten Nachricht.

Beispiel: $f_1(Sig_{100101}) = f_1(f_{100101}^{-1}(R)) = f_{10010}^{-1}(R) = Sig_{10010}$.

Dies läßt sich verhindern, indem die Form einer „gültigen Nachricht" durch eine präfixfreie Abbildung *präf*(•) festgelegt wird: Kein Präfix, d.h. Anfangsstück, einer so abgebildeten Nachricht darf gleichzeitig präfixfreie Abbildung einer anderen Nachricht sein. Die ursprüngliche Nachricht erhält so einen „Gültigkeitsvermerk", der bei Verlust des (oder der) letzten Bits automatisch erlischt. Signiert wird nun *präf*(m) mit $f_{präf(m)}^{-1}(R)$.

Aus der Kollisionsresistenz der Paare (f_0, f_1) kann man herleiten, daß die Familie der $f_{präf(m)}$ in ähnlichem Sinne kollisionsresistent ist, d.h. daß es schwer ist, zwei Nachrichten m, m^* und „Signaturen" S, S^* zu finden mit $f_{präf(m)}(S) = f_{präf(m^*)}(S^*)$. (s. Aufgabe 3-15 c)).

Präfixfreie Abbildung: Eine effiziente präfixfreie Abbildung läßt sich durch Voranhängen eines Längenprädikates vor die Nachricht m erreichen, also eines Feldes fester Länge (etwa ein Rechnerwort), das die Länge der eigentlichen Nachricht enthält.[70]

Im Fall, daß die Länge der Nachricht a priori feststeht, erübrigt sich eine zusätzliche präfixfreie Abbildung.

3.5.4 Gesamt-GMR: Authentisierung vieler Referenzen

Die einfachste Idee, wie man erreichen könnte, daß mehrere Nachrichten signiert werden können, wäre, im öffentlichen Schlüssel eine bestimmte Anzahl von Referenzen bekanntzugeben und später

[70] Der Vorzug des GMR-Signatursystems, Nachrichten beliebiger Länge signieren zu können, geht dabei nicht verloren, wenn z.B. über ein „Erweiterungsbit" das Einfügen weiterer Längenfelder ermöglicht wird. Konkret kann man dazu das Vorzeichenbit des Längenfeldes wählen.

Signatur für Signatur zu „verbrauchen“. Dieses Vorgehen hätte den Nachteil, daß die öffentliche Referenzenliste sehr lang würde.[71]

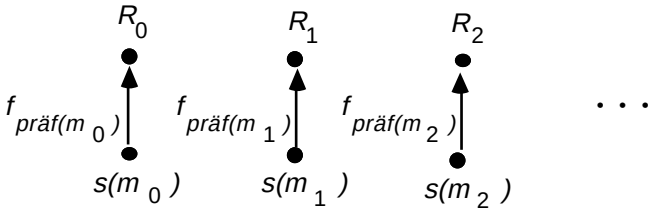


Bild 3-24: Grundidee, wie mit GMR viele Nachrichten signiert werden

Von dieser Idee bleibt, daß tatsächlich für jede auszugebende Signatur eine andere, zufällig gewählte Referenz verwendet wird, also etwa R_0, R_1, ... für die 1-te, 2-te usw. Signatur. (Dadurch wird erreicht, daß ein aktiver Angreifer seine Signatur jedesmal zu einer anderen Referenz erhält. Dies erschwert die Zielgerichtetheit eines aktiven Angriffs; dies ist eine der Grundideen von GMR.)

Die einzelnen R_i können aber auch nicht erst mit der Nachricht bekanntgegeben werden – andernfalls könnte ein Angreifer zu einer Nachricht *m* seiner Wahl eine Signatur *Sig* zufällig wählen und von dort mittels f_0 und f_1 die passende Referenz $R := f_{präf(m)}(Sig)$ berechnen.

Es liegt also nahe, die Referenzen selbst mittels Signaturen zu authentisieren. Hierzu wäre wieder die einfachste Idee, eine einzige Referenz r_ε in den öffentlichen Schlüssel aufzunehmen. Dann signiert man die Liste der Referenzen R_0, R_1, ... bezüglich r_ε, und nun kann man das passende R_i zum Signieren der eigentlichen Nachricht verwenden. Der Nachteil hierbei wäre, daß man jeder Signatur die Liste aller Referenzen beilegen müßte.

Deswegen geht man zu baumförmiger Authentikation der Referenzen über, d.h. mit jeder authentisierten Referenz werden zwei neue signiert. Man legt vorweg die Anzahl 2^b an Referenzen fest, die man so erhalten möchte.

Nur eine Referenz r_ε wird veröffentlicht. Die 2^b für Nachrichtensignaturen (im weiteren kurz N-Signaturen genannt) vorgesehenen Referenzen R_0, R_1, ... werden nun an die Blätter eines Binärbaumes der Tiefe *b* angehängt; jeder Knoten des Baumes enthält wiederum eine zufällig gewählte Referenz r_j . Der Baum wird im folgenden Referenzenbaum genannt (Bild 3-25).

---

71 Für diesen vereinfachten Fall, daß die Referenzen dem Angreifer vor seiner Wahl der zu signierenden Nachrichten bekannt sind, ist die Sicherheit von GMR außerdem nicht bewiesen; wir sehen auch nicht, wie sie bewiesen werden könnte.

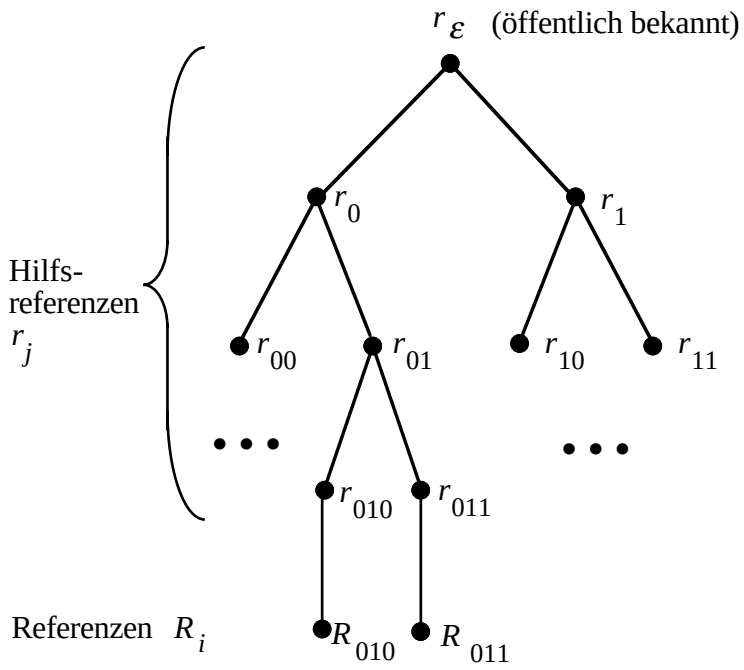


Bild 3-25: Referenzenbaum

Für jede Signatur wird nun ein Signaturkopf berechnet, der die aktuelle Referenz R_i bezüglich des öffentlichen r_ε authentisiert. Dies geschieht folgendermaßen:

Beginnend mit der öffentlichen Referenz r_ε, der Wurzel des Referenzenbaumes also, wird jede Hilfsreferenz r_j auf dem Pfad von r_ε nach R_i benutzt, um ihre beiden Nachfolgeknoten r_{j0} und r_{j1} zu signieren (Beispiel in Bild 3-26): r_{j0} und r_{j1} werden dazu (mit Trennzeichen) konkateniert, als „Nachricht“ interpretiert und präfixfrei abgebildet. Dann wird die Signatur $KSig := f_{präf(r_{j0}|r_{j1})}{}^{-1}(r_j)$ gebildet. Diese Knotensignaturen werden im folgenden kurz K-Signaturen genannt. Schließlich wird die angehängte Referenz R_i mit $RSig := f_{präf(R_i)}{}^{-1}(r_i)$ authentisiert; diese Referenzsignatur wird im folgenden als R-Signatur bezeichnet.[72]

Da alle Referenzen r_j an den Knoten des Referenzenbaumes, die im Signaturkopf zu authentisieren sind, etwa die gleiche Länge haben, kann hier eine feste Länge angenommen werden (kürzere Referenzen werden durch führende Nullen auf die korrekte Länge gebracht). Als Länge wird sinnvollerweise gerade der Sicherheitsparameter l gewählt, der die Länge des Modulus n (in Bit) festlegt. Eine zusätzliche präfixfreie Abbildung erübrigt sich also.

Beim Testen ist also nicht nur die N-Signatur zu überprüfen, sondern auch die b K-Signaturen zu sämtlichen Knoten r_j des Pfades durch den Referenzenbaum und die R-Signatur der Referenz R_i. Der dadurch entstehende zusätzliche Aufwand fällt jedoch, wie Aufwandsabschätzungen zeigen, bei längeren Nachrichten kaum ins Gewicht.

---

72 Für K- und R-Signaturen ist ein anderes Geheimnis zu verwenden als für die N-Signaturen; dies wird im Sicherheitsbeweis benötigt und ist in Bild 3-28 gezeigt. Im folgenden wird diese Unterscheidung aus Übersichtlichkeitsgründen meist nicht erwähnt, z.B. schreiben wir einfach p und q, obwohl es für N-Signaturen andere sind als für K- und R-Signaturen. Welche gemeint sind, ist aus dem Kontext klar.

Auch die Notwendigkeit der R-Signaturen wird nur im Beweis klar.

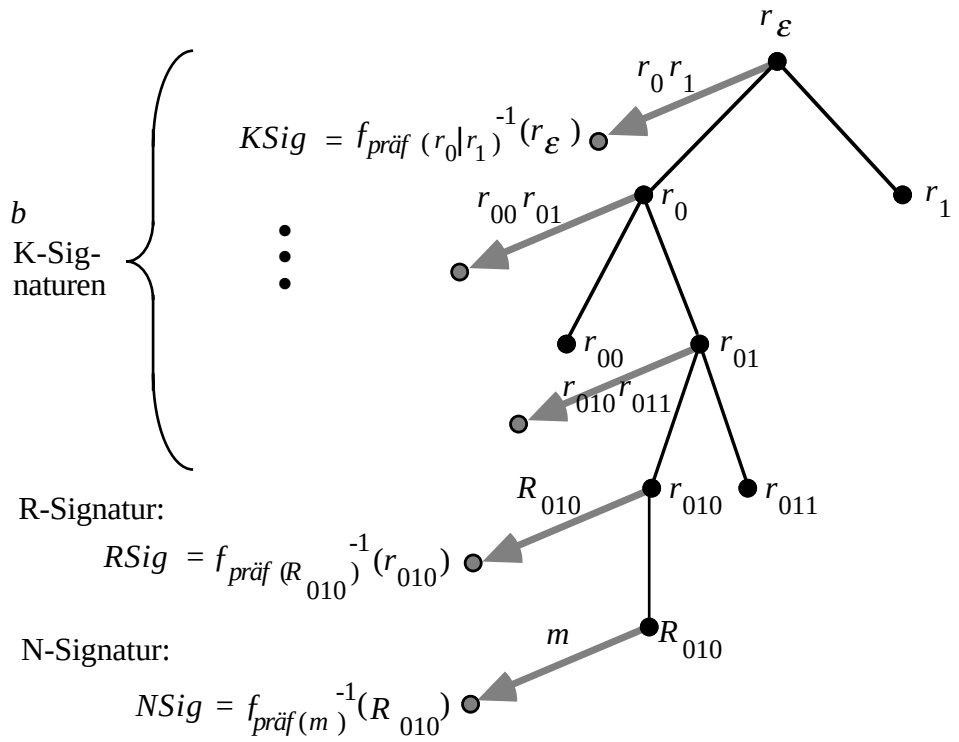


Bild 3-26: Signatur unter m bezüglich Referenz R_2

Effizienzverbesserungen: Zunächst muß man beachten, daß der Signierer nicht den ganzen Referenzenbaum vorweg generieren muß, sondern immer nur die Referenz R_i, die er gerade braucht, und die im Baum darüberliegenden Hilfsreferenzen und deren Geschwister (da diese jeweils zusammen authentisiert werden). Was zu einer Zeit zu speichern ist, sieht man am Beispiel R_2 also gerade in Bild 3-26.

Der Unterzeichner kann für jede Signatur aus der jeweils vorigen den gemeinsamen Anfangsabschnitt des Pfades im Referenzenbaum mit allen zugehörigen, bereits berechneten K-Signaturen in den neuen Signaturkopf übernehmen. Es empfiehlt sich daher, nicht nur die weiterhin benötigten Hilfsreferenzen, sondern den gesamten Signaturkopf der vorherigen Signatur im „Gedächtnis“ zu behalten.

Da die Berechnung der Umkehrfunktionen (f_0^{-1}, f_1^{-1}) aufwendiger ist als das Rechnen mit f_0 und f_1, sollte auf f_0^{-1}, f_1^{-1} weitestmöglich verzichtet werden. Es zeigt sich, daß tatsächlich nur einmal je Signatur eine Authentisierung mit f^{-1}, also f_0^{-1} und f_1^{-1} erfolgen muß:

Bei jeder neuen Signatur kann der Signaturkopf zunächst „von unten“ mittels f_0 und f_1 berechnet werden: Die N-Signatur *NSig* für die Nachricht *m* wird zufällig gewählt und die Referenz als $R_i := f_{präf(m)}(NSig)$ berechnet. (Da f_0 und f_1 Permutationen sind, garantiert die Zufälligkeit von *NSig* auch die von R_i.) Anschließend wird die R-Signatur *RSig* zufällig gewählt und r_i berechnet usw., bis ein „Verbindungsknoten“ erreicht ist, an dem der übernommene Pfadteil mit dem soeben neu berechneten verknüpft werden muß. Hier ist ein einziges Mal mit f_0^{-1} und f_1^{-1} zu rechnen (Bild 3-27).

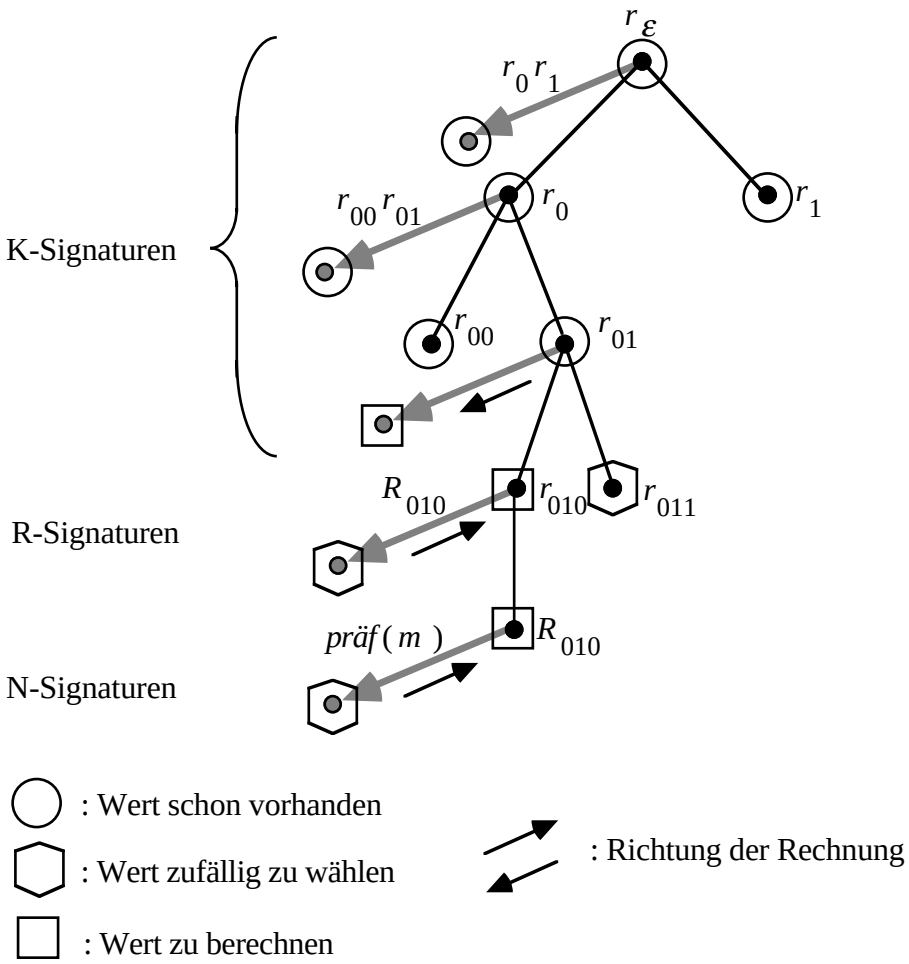


Bild 3-27: Rechnen „von oben“ und „von unten“

Um am Schluß der schrittweisen Einführung einen Gesamtüberblick über GMR zu geben, ist es in Bild 3-28 in der üblichen Darstellung kryptographischer Systeme aus §3.1.1 dargestellt. Je nach Implementierung kann es sinnvoll sein, die Baumtiefe *b* als Teil des öffentlichen Testschlüssels zu führen.

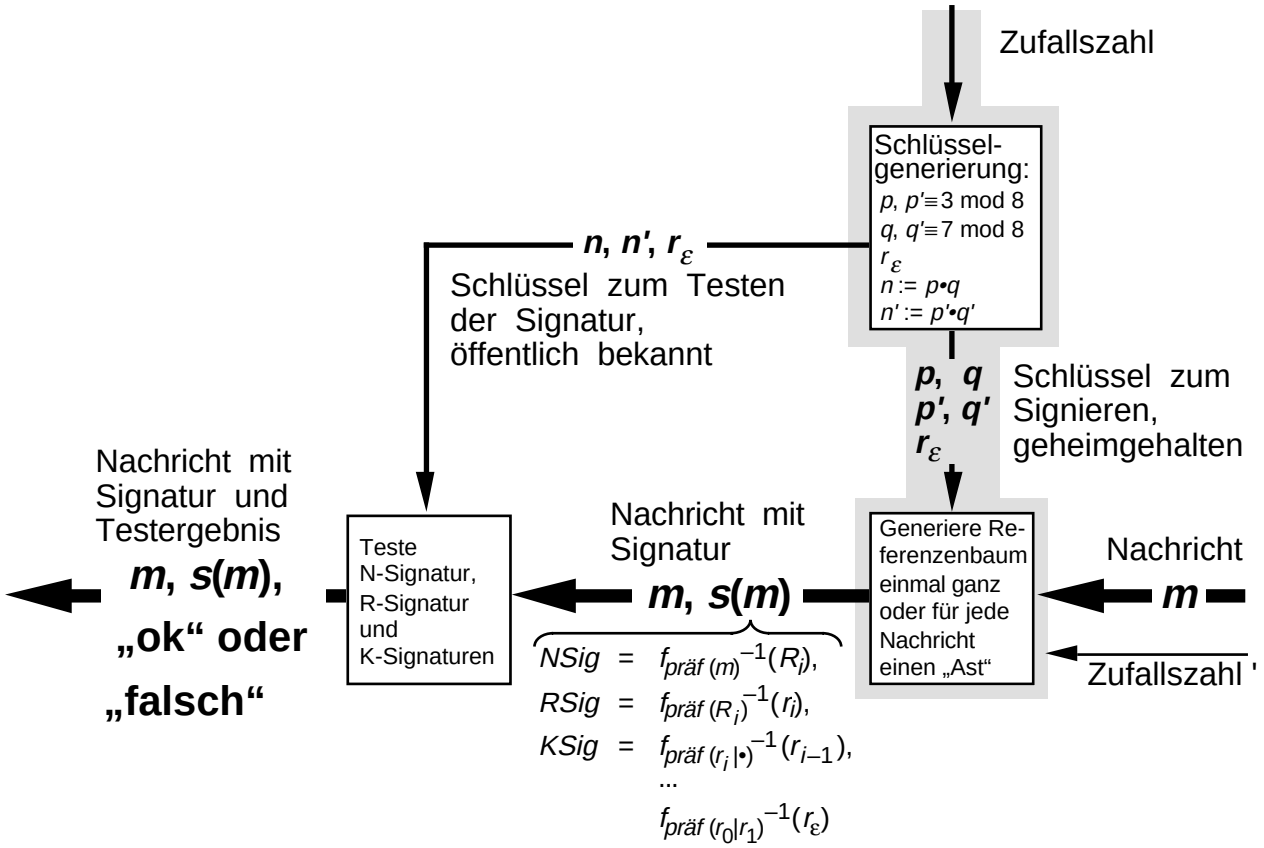


Bild 3-28: Das digitale Signatursystem GMR

3.5.5 Weitere Effizienzverbesserungen

Man kann insbesondere das aufwendige Rechnen „von oben“ noch wesentlich beschleunigen, vor allem mit einem Trick von Goldreich, der erlaubt, anstatt bitweise f_0^{-1} und f_1^{-1} anzuwenden, $f_w^{-1}(R)$ für beliebige Bitketten w effizienter in einem Schritt zu bestimmen. Auch gibt es einige weitere Stellen, wo der Unterzeichner die Kenntnis des Geheimnisses (p, q) ausnutzen kann. Für Einzelheiten vgl. [FoPf_91].

Für eine Andeutung der mit diesem Verfahren erreichbaren Effizienz: In einer Software-Implementierung

- auf dem Mac IIfx mit dem Prozessor MC68030, 40MHz
- für eine Nachricht von 8192 Bit
- bei Baumtiefe 10 und Moduluslänge 512

benötigt das Signieren im Mittel etwa 3,2 und das Testen 16,6 Sekunden [FoPf_91]. Das Testen dauert wesentlich länger, da nicht mod der Primzahlen, sondern nur mod der Produkte gerechnet werden kann (vgl. auch §3.6.5.2) und jeweils auch den ganzen Baum hinauf getestet werden muß. Beim Signieren muß neben der N-Signatur und der R-Signatur im Mittel nur eine K-Signatur berechnet werden.

Hardwarerealisierungen wären natürlich wesentlich schneller.

3.6 RSA: Das bekannteste System für asymmetrische Konzelation und digitale Signaturen

RSA, benannt nach den ersten Buchstaben der Nachnamen seiner Erfinder Ronald L. Rivest, Adi Shamir und Leonard M. Adleman, ist das bekannteste asymmetrische kryptographische System. Es wurde 1978 publiziert [RSA_78] und ist sowohl als asymmetrisches Konzelationssystem wie auch als digitales Signatursystem verwendbar.

Wie der s^2-mod-n-Generator (§3.4) und die bei GMR beschriebenen kollisionsresistenten Permutationenpaare mit Geheimnis (§3.5.1) beruht die Sicherheit von RSA auf der Faktorisierungsannahme (§3.4.1). Jedoch gibt es im Gegensatz zu diesen Systemen bisher keinen Beweis, daß ein Brechen von RSA einen Faktorisierungsalgorithmus liefert. Deshalb kann RSA nur in die Klasse „wohluntersucht“ in Bild 3-12 eingeordnet werden.

3.6.1 Das asymmetrische kryptographische System RSA

Die Kernidee von RSA besteht darin, (in einem Restklassenring) Nachrichten zu exponenzieren, zueinander inverse Exponenten zu bestimmen und das Wissen über die Exponenten geeignet zu verteilen.

Schlüsselgenerierung für Sicherheitsparameter l:

1. Wähle 2 Primzahlen p und q zufällig und stochastisch unabhängig, so daß gilt: $|p| \approx |q| = l$ und $p \neq q$. [73]
2. Berechne $n := p \bullet q$.
3. Wähle c mit $3 \leq c < \phi(n)$ und $ggT(c, \phi(n)) = 1$. [74]
4. Berechne d mittels p, q, c als multiplikatives Inverses von c modulo $\phi(n)$.[75] Dann gilt:

$$c \bullet d \equiv 1 \mod \phi(n).$$

Veröffentliche c und n als öffentlichen Schlüssel, behalte d, p, q als geheimen Schlüssel.

---

[73] $|p|$ bedeutet die Länge von p in Bits.
Die Bedingung $p \neq q$ könnte weggelassen werden, da sie nur in einem exponentiell kleinen Anteil aller Fälle verletzt wäre. Da für $p \neq q$ aber $\phi(n)$ besonders einfach als $(p-1)(q-1)$ dargestellt werden kann und die Entschlüsselung dann nicht nur fast immer, sondern immer klappt, wird $p \neq q$ hier gefordert. (Wer nicht glaubt, daß andernfalls die Entschlüsselung nicht immer klappt, rechne folgendes Beispiel: $p = q = 5$; $\phi(p^2) = p(p-1)$, also $\phi(25) = 20$. Zu $c = 7$ ergibt sich $d = 3$, so daß $c \bullet d = 21 \equiv 1 \mod 20$. $(15^3)^7$ ist dann $\equiv 0 \mod 25$ statt $\equiv 15$.)
Während p und q in etwa gleich *lang* gewählt werden sollten und auch exakt gleiche Länge keinerlei Problem darstellt, sollten p und q – wie gerade begründet – auf keinen Fall gleich *groß* (sonst klappt die Entschlüsselung nicht) und auch nicht näherungsweise gleich *groß* sein (sonst kann n faktorisiert werden). Dies Faktorisieren könnte z.B. folgendermaßen geschehen: Ziehe in $\mathbb{R}$ die Wurzel aus n und suche von dort aus abwärts nach einem Faktor von n. Wenn p und q näherungsweise gleich sind, ist dies effizient durchführbar. Selbst wenn p und q exakt gleich lang gewählt werden, ansonsten ihre Wahl aber stochastisch unabhängig ist, sind sie nur in einem exponentiell kleinen Anteil aller Fälle „näherungsweise gleich“, so daß auch dies nicht explizit geprüft zu werden braucht.
Um das Brechen von RSA zu erschweren, sollten die Primzahlen p und q nicht nur die in §3.4.1 geforderten Eigenschaften haben, die das Faktorisieren des Modulus erschweren. Zusätzlich sollte auch jeweils für mindestens einen der großen Primfaktoren von $p-1$ und $q-1$ gelten: Um eins vermindert hat er selbst wieder jeweils mindestens einen großen Primfaktor. Dies wehrt einen speziellen Angriff auf RSA ab [DaPr_89 Seite 232]. Auch mit diesen Zusatzeigenschaften gibt es immer noch genügend viele Primzahlen p und q und sie können auch effizient gewählt werden [BrDL_91].

[74] Oft wird c in diesem Intervall unter den zu $\phi(n)$ teilerfremden Zahlen *zufällig* gewählt. Eine andere Art, die obigen Bedingungen an c hinzuschreiben, lautet: Wähle c aus $\mathbb{Z}_{\phi(n)}^*$, aber $\neq 1$.

[75] Dies geschieht mit dem erweiterten Euklidschen Algorithmus, der in §3.4.1.1 beschrieben ist.

Ver- bzw. **Entschlüsselung** erfolgt durch Exponentiation mit c bzw. d in $\mathbb{Z}_n$.

Zu beweisen ist nun, daß Ver- und Entschlüsselung für alle Nachrichten m zueinander invers sind:

Beh. Für alle $m \in \mathbb{Z}_n$ gilt: $(m^c)^d \equiv m^{c \cdot d} \equiv (m^d)^c \equiv m \mod n$.

Bew. Die ersten zwei Kongruenzzeichen gelten nach den Rechenregeln für Kongruenzen. Das dritte ist zu zeigen.

$c \cdot d \equiv 1 \mod \phi(n) \Rightarrow$
$c \cdot d \equiv 1 \mod p-1 \Leftrightarrow$
$\exists k \in \mathbb{Z}: c \cdot d = k \cdot (p-1) + 1.$
Also gilt für dieses k und für alle $m \in \mathbb{Z}_n$: $m^{c \cdot d} \equiv m^{k \cdot (p-1) + 1} \equiv m \cdot (m^{p-1})^k \mod p$.
Mittels des kleinen Fermatschen Satzes (§3.4.1.2) folgt für alle zu p teilerfremden m
$m^{p-1} \equiv 1 \mod p$.
Also ist für alle zu p teilerfremden m: $m \cdot (m^{p-1})^k \equiv m \cdot 1^k \equiv m \mod p$.
Dies gilt trivialerweise auch für $m \equiv 0 \mod p$.
Zusammen gilt $\forall m \in \mathbb{Z}_n$:
$m^{c \cdot d} \equiv m \mod p$.

Die entsprechende Argumentation für q ergibt
$m^{c \cdot d} \equiv m \mod q$.

Da die Kongruenz sowohl bezüglich p als auch q gilt, gilt sie nach §3.4.1.3 auch bezüglich $p \cdot q = n$. Insgesamt ergibt sich:
Für alle $m \in \mathbb{Z}_n$ gilt: $m^{c \cdot d} \equiv m \mod n$. □

Da jedes Element von $\mathbb{Z}_n$ als Nachricht vorkommen und nach der Verschlüsselung durch modulare Exponentiation mit c wieder eindeutig entschlüsselt werden kann, ist die modulare Exponentiation mit c *injektiv*. Da alle Schlüsseltexte in $\mathbb{Z}_n$ liegen, Bild- und Urbildraum also gleich, insbesondere also gleich mächtig sind, ist die modulare Exponentiation mit c also auch *surjektiv*, insgesamt also eine *Permutation*. Also kann aus *jedem* Element von $\mathbb{Z}_n$ eine c-te Wurzel gezogen werden.

Leider ist (bisher?) *kein* Sicherheitsbeweis von RSA bekannt. Er müßte darin bestehen zu zeigen: Wenn RSA leicht zu brechen ist, dann kann leicht faktorisiert werden. Für die schärfste Form des Brechens (und damit die schwächste Form von Sicherheitsbeweis) also: Wenn jemand c-te Wurzeln modulo n ziehen kann, kann er n faktorisieren. Leider konnte nicht einmal dies bisher gezeigt werden.

3.6.2 Naiver und unsicherer Einsatz von RSA

Hier wird der naive (und wie wir in §3.6.3 sehen werden) unsichere Einsatz von RSA als asymmetrisches Konzelationssystem und danach der als digitales Signatursystem beschrieben. In beiden Fällen werden die (Klar-)Texte als Zahlen m mit $0 \le m < n$ dargestellt. Hierzu müssen lange (Klar-)Texte in mehrere Blöcke (von jeweils maximal der Länge $|n|$-1 Bits) unterteilt werden (**Blockung**).

3.6.2.1 RSA als asymmetrisches Konzelationssystem

Die Bezeichner aus §3.6.1 sind schon passend gewählt:

Verschlüsselung erfolgt durch modulare Exponentiation mit c.
Dies ergibt für Klartextblock m: $m^c \mod n$.

Entschlüsselung erfolgt durch modulare Exponentiation mit d.
Dies ergibt für Schlüsseltextblock m^c: $(m^c)^d \mod n = m$.

Bild 3-29 veranschaulicht das Gesamtsystem.

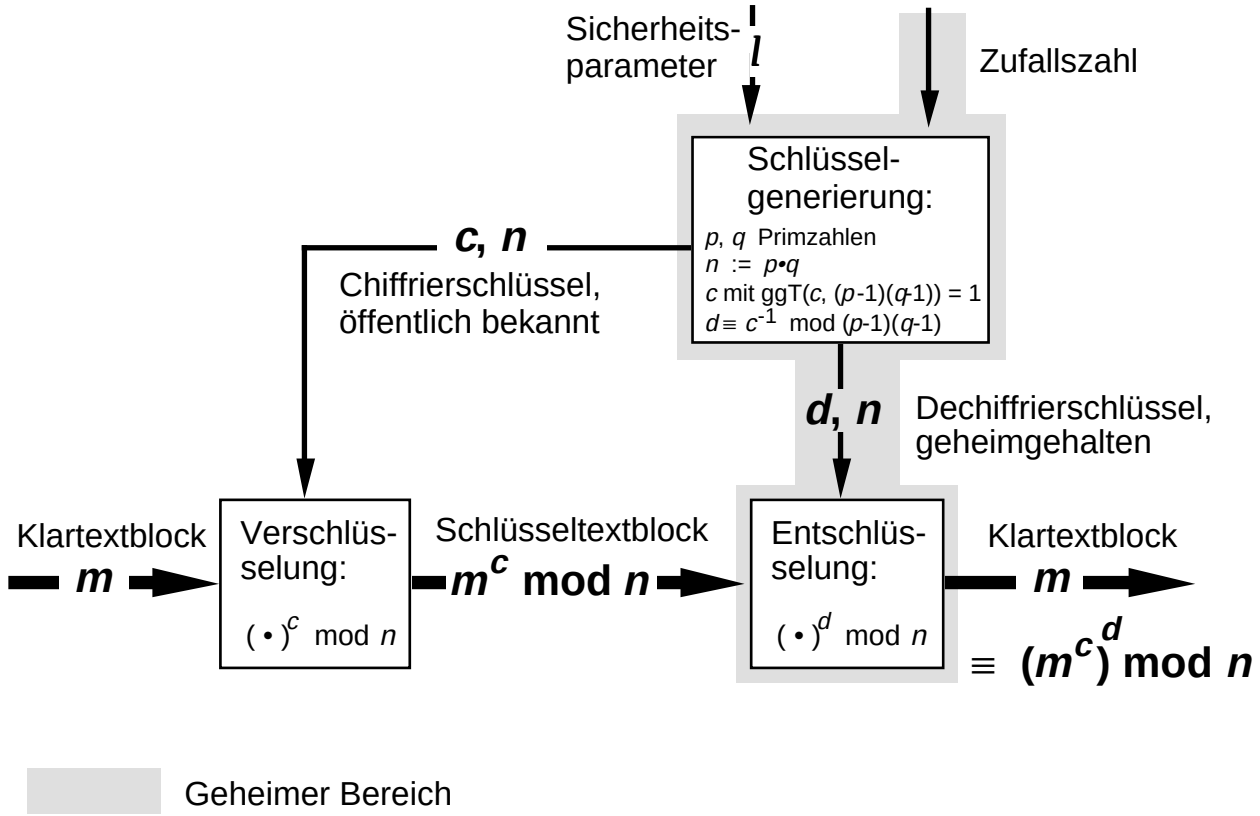


Bild 3-29: Naiver und unsicherer Einsatz von RSA als asymmetrisches Konzelationssystem

Bsp. *Schlüsselgenerierung:* Sei $p = 3$, $q = 17$, also $n = 51$.
Also ist $\phi(n) = (p-1) \cdot (q-1) = 32$.
Sei $c = 5$; prüfe ob ggT(5, 32) = 1. (Euklidscher Algorithmus) Dies ist der Fall.
Gesucht $d = c^{-1} \bmod \phi(n)$, also $5^{-1} \bmod 32$. Mit dem erweiterten Euklidschen Algorithmus ergibt sich $d = 13$.

Verschlüsselung: Sei die Klartextnachricht $m = 19$. Dann ist der zugehörige Schlüsseltext $S = 19^5 \equiv 19^2 \cdot 19^2 \cdot 19 \equiv 361 \cdot 361 \cdot 19 \equiv 4 \cdot 4 \cdot 19 \equiv 4 \cdot 76 \equiv 4 \cdot 25 \equiv 49 \bmod 51$.

Entschlüsselung: $S^d = 49^{13} \equiv (-2)^{13} \equiv 1024 \cdot (-8) \equiv 4 \cdot (-8) \equiv -32 \equiv 19 \bmod 51$, was tatsächlich der Klartext ist.

3.6.2.2 RSA als digitales Signatursystem

Die Bezeichner aus §3.6.1 werden folgendermaßen umbenannt:
$c \rightarrow t, \quad d \rightarrow s$.

Dann gilt:

Signieren erfolgt durch modulare Exponentiation mit s.
Dies ergibt für Textblock m: $m^s \bmod n$.

Testen erfolgt durch modulare Exponentiation der Signatur mit t und anschließendem Vergleich des Ergebnisses mit dem zugehörigen Textblock.
Dies ergibt für Textblock m mit Signatur m^s: $(m^s)^t \bmod n = m$?

Bild 3-30 veranschaulicht das Gesamtsystem.

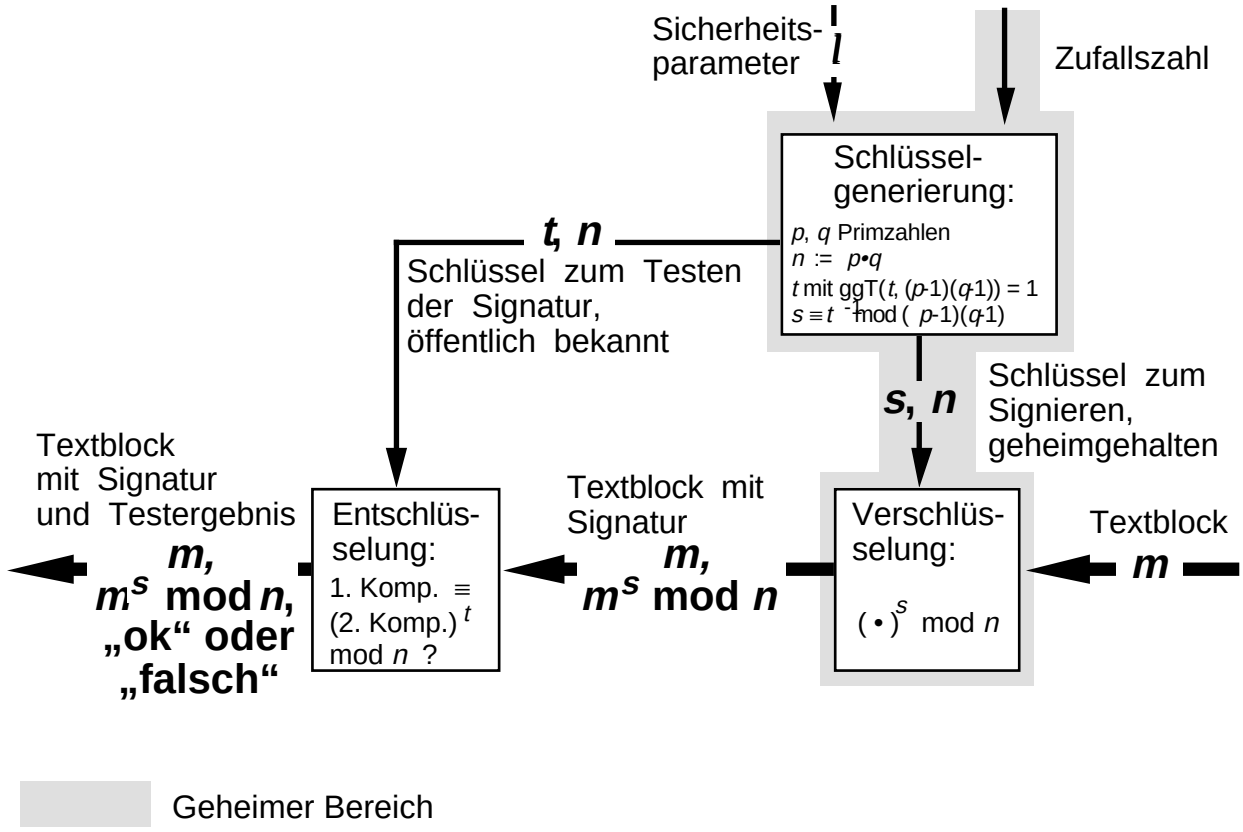


Bild 3-30: Naiver und unsicherer Einsatz von RSA als digitales Signatursystem

3.6.3 Angriffe, insbesondere multiplikative Angriffe von Davida und Moore

Leider ist der Einsatz von RSA, wie er in §3.6.2 beschrieben wurde, zwar leicht zu verstehen, aber weitgehend unsicher. Da RSA eine multiplikative Struktur besitzt (Genaueres wird gleich erklärt), kann sowohl das Konzelationssystem als auch das Signatursystem gebrochen werden, ohne daß der Angreifer irgendetwas über den geheimen Schlüssel herausbekommen müßte.

Da der multiplikative Angriff für RSA als Signatursystem systematischer erklärt werden kann, wird damit begonnen.

3.6.3.1 RSA als digitales Signatursystem

Jeder kann in dem in §3.6.2.2 beschriebenen digitalen Signatursystem Signaturen fälschen, indem er rückwärts rechnet: Er wählt sich eine Signatur, exponenziert sie modular mit t und erhält so den passenden Textblock. Dieser *passive Angriff* bricht RSA als digitales Signatursystem also *existentiell*, nicht aber selektiv. Denn der Angreifer kann nicht die Nachricht vorgeben, zu der er gerne die Signatur hätte, vgl. §3.1.3.1 und §3.1.3.2. Das Mindeste, was also zusätzlich zu dem in §3.6.2.2 Beschriebenen gefordert werden muß, ist, daß „sinnvolle“ Textblöcke sich bei diesem Angriff nur sehr, sehr selten ergeben.

Um aktive Angriffe, die ein *selektives* Brechen des in §3.6.2.2 beschriebenen digitalen Signatursystems erlauben, systematisch herzuleiten, wird zunächst ein einfacher passiver Angriff beschrieben. Er fälscht zwar Signaturen nicht besser als der gerade beschriebene, erläutert aber die multiplikative Struktur von RSA. Diese wird dann für die aktiven Angriffe benötigt.

Kennt ein Angreifer zwei Signaturen m_1^s und m_2^s für die Textblöcke m_1 und m_2, so kann er eine dritte Signatur m_3^s und den passenden Textblock m_3 durch modulare Multiplikation gewinnen:

$m_3^s := m_1^s \cdot m_2^s \mod n$,

$m_3 := m_1 \cdot m_2 \mod n$.

m_3^s ist tatsächlich die Signatur zu m_3. Denn es gilt: $m_1^s \cdot m_2^s = (m_1 \cdot m_2)^s$. Mit anderen Worten: RSA besitzt eine **multiplikative Struktur**, oder genauer: RSA ist ein **Homomorphismus** bezüglich der Multiplikation. Wie oben bereits erwähnt, gelingt auch mit diesem passiven Angriff nur *existentielles* Brechen.

Ist ein *aktiver* Angriff möglich, kann mittels dieser multiplikativen Struktur ein *selektives Brechen* erfolgen (Angriff von Davida [Merr_83, Denn_84, Bras_88]):

Der Angreifer wählt den Textblock m_3 frei nach seinen Wünschen.

Er wählt ein beliebiges m_1, zu dem mod n ein multiplikatives Inverses m_1^{-1} existiert.

Er berechnet $m_2 := m_3 \cdot m_1^{-1} \mod n$.

Er läßt m_1 und m_2 signieren.

Er erhält m_1^s und m_2^s.

Er berechnet $m_3^s := m_1^s \cdot m_2^s \mod n$.

Dieser (nichtadaptive) aktive Angriff benötigt *zwei* Signaturen des Opfers, um *eine* selektiv zu fälschen. Der folgende, von Judy Moore verbesserte Angriff [Denn_84], benötigt nur noch *eine* Signatur des Angegriffenen, um eine selektiv zu fälschen.

Der Angreifer wählt den Textblock m_3 frei nach seinen Wünschen.

Er wählt eine Zufallszahl r gleichverteilt in $\mathbb{Z}_n^*$. (Zur Erinnerung: Dies bedeutet $1 \le r < n$, und r besitzt mod n ein multiplikatives Inverses r^{-1}.)

Er berechnet $m_2 := m_3 \cdot r^t \mod n$.

Er läßt m_2 signieren.[76]

Der Angreifer erhält m_2^s.

Er weiß: $m_2^s \equiv (m_3 \cdot r^t)^s \equiv m_3^s \cdot r^{t \cdot s} \equiv m_3^s \cdot r \mod n$.

Er berechnet $m_3^s := m_2^s \cdot r^{-1} \mod n$.

Die Idee dieses (nichtadaptiven) aktiven Angriffs besteht also darin, die Nachricht m_3, die der Angreifer gerne signiert hätte, so mit einer „aufbereiteten“ Zufallszahl r^t zu multiplizieren, daß der Angegriffene sie **blind** (d.h. ohne zu wissen, was sich dahinter verbirgt) signiert und der Angreifer die erhaltene Signatur nur noch durch r dividieren muß. Eine Nutzanwendung dieses Angriffs von Judy Moore wurde 1985 von David Chaum publiziert [Cha8_85]. Sie wird in §3.9.5 und §6.4.1 ausführlich behandelt.

3.6.3.2 RSA als asymmetrisches Konzelationssystem

Wird RSA wie in §3.6.2.1 beschrieben eingesetzt, wird nur ein *deterministisches* asymmetrisches Konzelationssystem realisiert. Ein Angreifer kann also wahrscheinliche Klartextblöcke raten, sie mit dem öffentlich bekannten Schlüssel deterministisch verschlüsseln und an den Schlüsselinhaber geschickte Schlüsseltexte mit den von ihm selbst generierten vergleichen. Sind welche gleich, so kann er sie „entschlüsseln“, vgl. Anmerkung 2 zu Bild 3-4.

Da die in §3.6.3.1 beschriebenen multiplikativen Angriffe nur Eigenschaften von RSA ausnutzen, die in gleicher Weise beim naiven Einsatz als Signatursystem wie auch als Konzelationssystem bestehen,

---

[76] Ist $m_3 \in \mathbb{Z}_n^*$, so erfährt der Angegriffene dadurch nichts über m_3, da r^t gleichverteilt in $\mathbb{Z}_n^*$ ist. Andernfalls benötigt der Angreifer die Hilfe des Angegriffenen überhaupt nicht: Für $m_3 = 0$ ist für alle n nämlich m_3^s ebenfalls $= 0$. In allen anderen Fällen ergibt $\mathrm{ggT}(m_3,n)$ einen nichttrivialen Faktor von n, konkret also p oder q. Damit hat der Angreifer RSA vollständig gebrochen, vgl. §3.1.3.1.

können sie alle auf den Einsatz als Konzelationssystem übertragen werden. Dies wird hier nur für den von Judy Moore verbesserten aktiven Angriff gezeigt.

Nichtadaptiver aktiver Angriff zum *selektiven* Brechen von RSA als naiv eingesetztes asymmetrisches Konzelationssystem:

Der Angreifer wählt den Schlüsseltextblock S_3 frei nach seinen Wünschen (z.B. einen Schlüsseltextblock, den er beobachtet hat).
Er wählt eine Zufallszahl r gleichverteilt in $\mathbb{Z}_n^*$. (Zur Erinnerung: Dies bedeutet $1 \le r < n$, und r besitzt mod n ein multiplikatives Inverses r^{-1}.)
Er berechnet $S_2 := S_3 \cdot r^c \mod n$.
Er läßt S_2 entschlüsseln.[77]
Der Angreifer erhält S_2^d.
Er weiß: $S_2^d \equiv (S_3 \cdot r^c)^d \equiv S_3^d \cdot r^{c \cdot d} \equiv S_3^d \cdot r \mod n$.
Er berechnet $S_3^d := S_2^d \cdot r^{-1} \mod n$.

Die Idee dieses (nichtadaptiven) aktiven Angriffs besteht also darin, den Schlüsseltext S_3, den der Angreifer gerne entschlüsselt hätte, so mit einer „aufbereiteten" Zufallszahl r^c zu multiplizieren, daß der Angegriffene sie **blind** (d.h. ohne zu wissen, was sich dahinter verbirgt) entschlüsselt und der Angreifer den erhaltenen Klartext nur noch durch r dividieren muß.

3.6.4 Vereitelung der Angriffe

In diesem Abschnitt wird kurz beschrieben, wie alle bekannten Angriffe auf RSA als Konzelationssystem bzw. Signatursystem vereitelt werden können. Allerdings sei explizit auf die Einschränkung auf „alle bekannten Angriffe" hingewiesen, darauf, daß nicht einmal hierfür Beweise existieren, und daß natürlich kein noch so geschickter Einsatz von RSA sicherer sein kann als RSA selbst, vgl. die einschränkende Bemerkung über die Sicherheit von RSA am Ende von §3.6.1.

Sowohl beim Einsatz von RSA zur Konzelation, als auch von RSA zur Authentikation ist die Verwendung einer **kollisionsresistenten Hashfunktion** h nützlich. Solche Funktionen haben wir in §3.5.3 kennengelernt, nur daß sie dort etwas anders aufgefaßt wurden, nämlich als Permutationenfamilie. Zur Erinnerung: Beliebig langen Bitketten m wurden kollisionsresistente Permutationen f_m zugeordnet. Faßt man diese Bitketten m als Eingabe einer Funktion auf, die die kollisionsresistente Permutation auf einen festen Wert x anwendet und den permutierten Wert y als Ausgabe, hat man eine kollisionsresistente Hashfunktion konstruiert. Etwas präziser hingeschrieben, lautet dies:

$$f_m(x) = y$$

ist eine	**Permutation**	für m fest:	Eingabe x,	Ausgabe y
und eine	**Hashfunktion**	für x fest:	Eingabe m,	Ausgabe y.

Eine Kollision der Hashfunktion, d.h.

$f_m(x) = f_{m*}(x)$ mit $m \ne m*$,

ist auch eine Kollision der Permutationenfamilie, d.h.

$f_m(x) = f_{m*}(z)$ mit $m \ne m*$,

da $x = z$ erlaubt ist. Das Umgekehrte gilt natürlich nicht, da im allgemeinen nicht $x = z$ gelten wird. Die konstruierte Hashfunktion ist also mindestens so kollisionsresistent wie die zugrundegelegte Permutationenfamilie.

---

[77] Auch hier erfährt der Angegriffene dadurch nichts über S_3, da r^c gleichverteilt in $\mathbb{Z}_n^*$ ist. Andernfalls benötigt der Angreifer die Hilfe des Angegriffenen überhaupt nicht: Für $S_3 = 0$ ist für alle n nämlich S_3^d ebenfalls = 0. In allen anderen Fällen ergibt ggT(S_3,n) einen nichttrivialen Faktor von n, konkret also p oder q. Damit hat der Angreifer RSA vollständig gebrochen, vgl. §3.1.3.1.

Für $f_m(x)$ mit x fest und Eingabe m, schreiben wir kurz h(m) und nennen h eine kollisionsresistente Hashfunktion (konstruiert aus einer kollisionsresistenten Permutationenfamilie).

Effizientere, aber nicht beweisbar sichere kollisionsresistente Hashfunktionen können beispielsweise aus Blockchiffren konstruiert werden, vgl. §3.8.3.

3.6.4.1 RSA als asymmetrisches Konzelationssystem

RSA wird zu einem **indeterministisch** verschlüsselnden Konzelationssystem, indem mit jedem Klartextblock m eine neue Zufallszahl z (sie hat nichts zu tun mit der Zufallszahl für die Schlüsselgenerierung, deshalb als Zufallszahl' in Bild 3-31 bezeichnet) verschlüsselt wird.

Aktive Angriffe werden verhindert, indem vor der Verschlüsselung jedem Klartextblock **Redundanz** zugefügt wird, die vom Entschlüsseler nach der Entschlüsselung geprüft wird. Dies Zufügen von Redundanz muß natürlich so geschehen, daß das modulare Multiplizieren zweier Klartextblöcke mit Redundanz keinen dritten Klartextblock mit passender Redundanz ergibt.

Die Redundanz kann beispielsweise erzeugt werden, indem auf den Klartextblock eine kollisionsresistente Hashfunktion angewandt und das Ergebnis an den Klartextblock gehängt wird. Natürlich muß die Hashfunktion dann so gewählt werden, daß sie, falls überhaupt, eine andere multiplikative Struktur hat als das RSA, dessen multiplikative Struktur sie gerade neutralisieren soll. Dies kann beispielsweise geschehen, indem bei Wahl der Permutationen aus §3.5.3 ein vom RSA-Modulus unabhängig gewählter Modulus verwendet wird.

Bild 3-31 veranschaulicht das Gesamtsystem, wenn indeterministische Verschlüsselung und Redundanz mittels kollisionsresistenter Hashfunktion kombiniert werden.

Verschlüsselung eines Klartextblocks erfolgt durch Voranstellen einer Zufallszahl, Anwenden einer kollisionsresistenten Hashfunktion h auf Zufallszahl und Klartextblock, deren Ergebnis dahinter gehängt wird, sowie modulare Exponentiation mit c aller drei Komponenten gemeinsam.
Dies ergibt für Klartextblock m und Zufallszahl z: $(z,m, \mathrm{h}(z,m))^c \bmod n$.

Entschlüsselung erfolgt durch modulare Exponentiation mit d. Sie ergibt drei Komponenten. Danach wird geprüft, ob die Hashfunktion h, angewandt auf die ersten beiden Komponenten, die dritte Komponente ergibt. Nur dann wird die zweite Komponente ausgegeben.
Dies ergibt für Schlüsseltextblock $(z,m, y)^c \bmod n$: $((z,m, y)^c)^d \bmod n =: z,m, y$. h($z,m$) = y ? Gegebenenfalls Ausgabe: m.

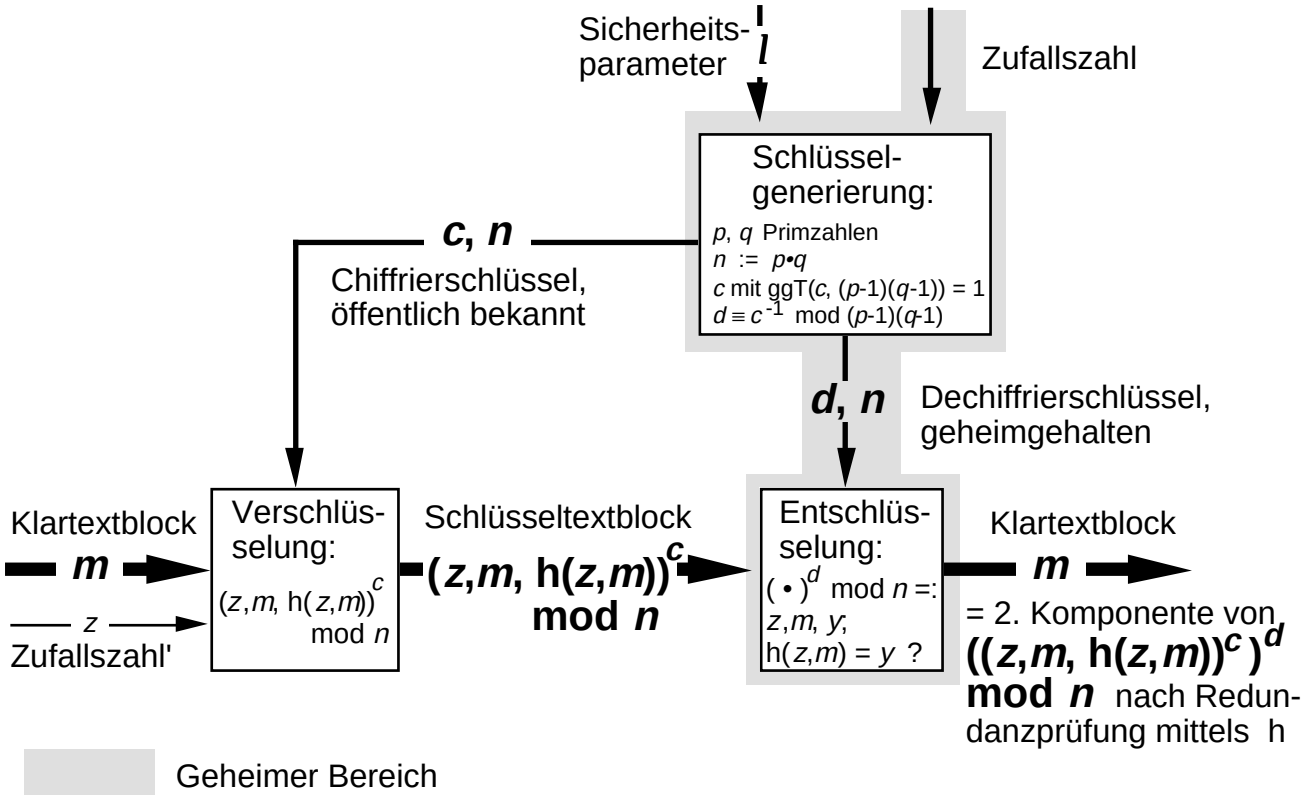


Bild 3-31: Sicherer Einsatz von RSA als asymmetrisches Konzelationssystem: Redundanzprüfung mittels kollisionsresistenter Hashfunktion h sowie indeterministische Verschlüsselung

Kompliziertere und dafür in einem gewissen Sinn beweisbar sichere Konstruktionen zur Kombination von Redundanz mit indeterministischer Verschlüsselung sind in [BeRo_95] beschrieben. Diese Konstruktionen gelten nicht nur für RSA, sondern für jedes asymmetrische Kryptosystem, das eine Permutation (sprich: längentreue Blockchiffre, vgl. §3.8.2 und §3.8.2.7) ist.

3.6.4.2 RSA als digitales Signatursystem

Um das Rückwärtsrechnen zu vereiteln und auch die multiplikative Struktur von RSA zu neutralisieren, wird auf den Textblock vor der modularen Exponentiation eine **kollisionsresistente Hashfunktion** angewandt. Wenn die Hashfunktion Argumente beliebiger Länge als Argument akzeptiert, hat dies zusätzlich den Vorteil, daß dann auch Texte beliebiger Länge in einem Stück signiert werden können. Das Blockungsproblem ist dann für das digitale Signatursystem gelöst.

Ist die Hashfunktion zusätzlich schneller zu berechnen als RSA, bringt ihre Anwendung bei längeren Nachrichten auch einen Geschwindigkeitsvorteil des Gesamtsystems gegenüber der in §3.6.2.2 beschriebenen Anwendung von RSA.

Signieren erfolgt durch Anwenden der Hashfunktion und modulare Exponentiation des Hashwertes mit s.
Dies ergibt für Textblock m: $(h(m))^s \bmod n$.

Testen erfolgt durch modulare Exponentiation der Signatur mit t und anschließendem Vergleich des Ergebnisses mit dem Hashwert des zugehörigen Textblocks.
Dies ergibt für Textblock m mit Signatur $(h(m))^s$: $((h(m))^s)^t \bmod n =: y$, $h(m) = y$?

Bild 3-32 veranschaulicht das Gesamtsystem.

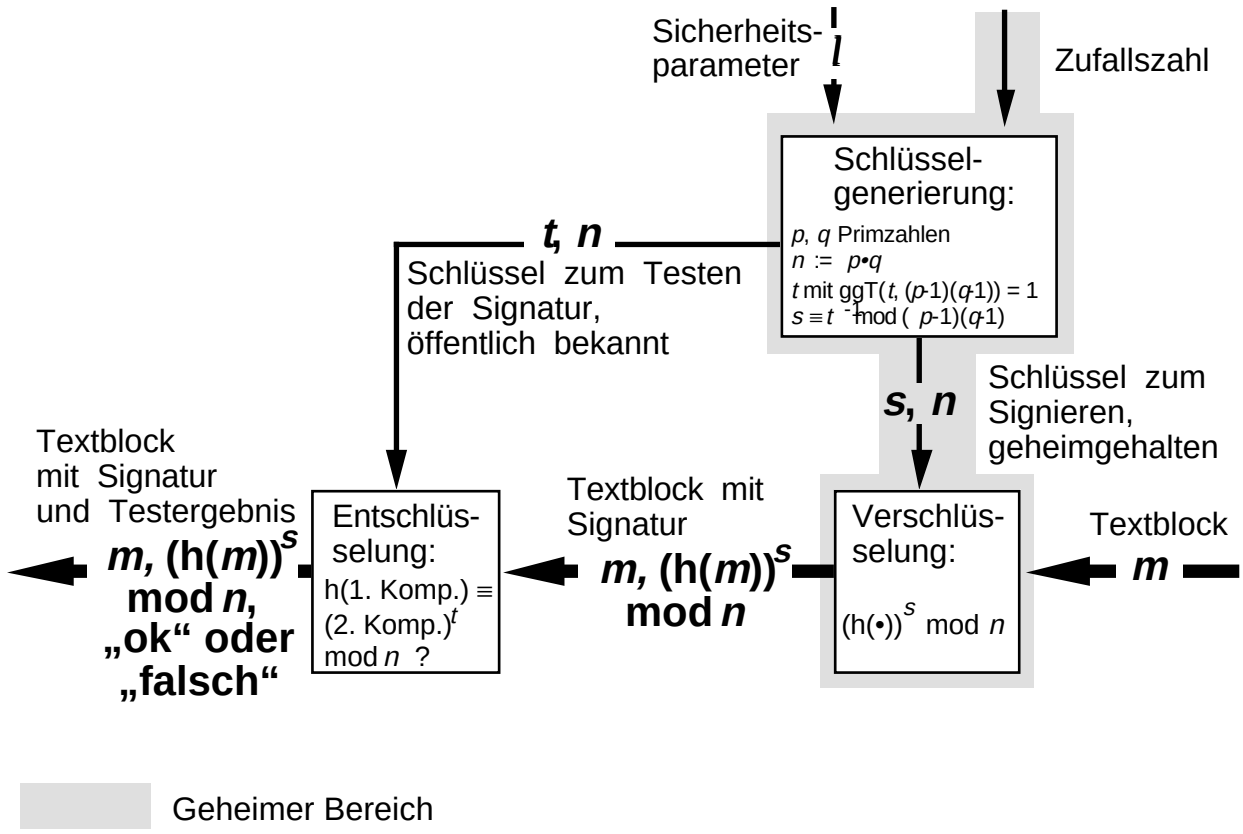


Bild 3-32: Sicherer Einsatz von RSA als digitales Signatursystem mit kollisionsresistenter Hashfunktion h

3.6.5 Effiziente Implementierung von RSA

Durch geschickte Implementierung kann der Aufwand für RSA erheblich gesenkt werden. Zuerst wird dies für die Anwender eines öffentlichen Schlüssels gezeigt, danach für den Inhaber eines geheimen Schlüssels.

3.6.5.1 Öffentlicher Exponent mit fast nur Nullen

Bis auf die Bedingung, daß der öffentliche Exponent teilerfremd zu $(p–1)(q–1)$ sein muß, kann er frei gewählt werden, vgl. §3.6.1. Dies legt nahe, ihn – bis auf führende Nullen – möglichst kurz zu wählen, damit zur Exponentiation möglichst wenige modulare Multiplikations- und Quadrierungsschritte nötig sind. Enthält er auch nach der führenden Eins möglichst nur Nullen, spart man auch für diese Stellen jeweils noch eine modulare Multiplikation, vgl. „square-and-multiply“ in §3.4.1.1. p und q müssen dann natürlich so gewählt werden, daß $p–1$ und $q–1$ zum öffentlichen Exponenten teilerfremd sind.

Da $(p–1)(q–1)$ gerade ist, ist die kleinstmögliche Wahl des öffentlichen Exponenten der Wert 3. Für RSA als digitales Signatursystem ist kein Argument bekannt, das gegen diese Wahl spricht. Für RSA als Konzelationssystem gibt es folgenden *passiven* Angriff von Johan Håstad, der RSA *nachrichtenbezogen* bricht [Hast_86, Hast_88]:

Ein Teilnehmer sendet die gleiche Nachricht m an drei andere, deren öffentliche Exponenten jeweils 3 und deren Moduli n_1, n_2 und n_3 sind. Die Moduli sind paarweise teilerfremd, sonst könnte der Angreifer zumindest einen faktorisieren und so m erhalten.

Der Angreifer beobachtet also: $m^3 \bmod n_1$, $m^3 \bmod n_2$ und $m^3 \bmod n_3$.

Mittels des CRA kann er $m^3 \bmod n_1 \cdot n_2 \cdot n_3$ bestimmen.

Da $m^3 < n_1 \bullet n_2 \bullet n_3$ gilt, ist $m^3 \bmod n_1 \bullet n_2 \bullet n_3 = m^3$. Der Angreifer kann aus $m^3 \bmod n_1 \bullet n_2 \bullet n_3$ also in $\mathbb{Z}$ normal die dritte Wurzel ziehen und erhält so m.

Der Angriff ist leicht auf zweierlei Weise zu verallgemeinern:

Da keine Suche durch den Klartextraum erfolgt und es sich um einen *passiven Angriff* handelt, vereitelt die in §3.6.4.1 beschriebene Anwendung von RSA (indeterministische Verschlüsselung sowie Redundanzbildung und -Prüfung) diesen Angriff nicht, wenn zur indeterministischen Verschlüsselung der 3 Nachrichten jeweils die gleiche Zufallszahl verwendet wird.

Ist der öffentliche Exponent c, so genügen c Verschlüsselungen derselben Nachricht für den entsprechenden Angriff. Allerdings muß der Angreifer dann auch mit c mal so langen Zahlen rechnen.

Als guter Kompromiß zwischen der Vereitelung dieses Angriffs und dem nötigen Rechenaufwand wird an vielen Stellen der Wert $2^{16}+1$ für den öffentlichen Exponenten vorgeschlagen [DaPr_89 Seite 235].

Es spricht nichts dagegen, daß alle Teilnehmer dies als ihre(n) öffentliche(n) Exponenten wählen. Dann brauchen öffentliche Exponenten nicht mehr mitgeteilt zu werden. Ein öffentlicher RSA-Schlüssel besteht dann nur noch aus dem Modulus.

3.6.5.2 Inhaber des geheimen Schlüssels rechnet modulo Faktoren

Für Situationen, in denen es mehr auf die Effizienz des Inhabers des geheimen Schlüssels ankommt, ist man versucht, statt des *öffentlichen* den *geheimen* Exponenten „kurz“ zu wählen (vgl. §3.6.5.1). Dies ist möglich, da sich öffentlicher und geheimer Exponent in den Formeln aus §3.6.1 vollkommen symmetrisch verhalten. Selbst wenn man hierbei nicht den plumpen Fehler begeht, den geheimen Exponenten als 3, $2^{16}+1$ oder sonst eine kleine Zahl zu wählen, die man durch Durchprobieren finden kann, entsteht für geheime Exponenten, die kürzer als ein Viertel der Moduluslänge sind, in aller Regel ein unsicheres System [Wien_90]. Vorsicht also vor dieser Art der Effizienzverbesserung!

Ohne die Wahl der Schlüssel im geringsten zu ändern – und folglich auch ohne jede Änderung der Sicherheit – kann der Inhaber des geheimen Schlüssels nahezu drei Viertel seines Rechenaufwands einsparen: Statt modulo n, rechnet er getrennt modulo der beiden Faktoren p und q. Danach berechnet er aus den beiden Ergebnissen den gesuchten Wert modulo n mittels des CRA. Im einzelnen geht dies folgendermaßen:

Um c-te Wurzeln aus Schlüsseltextblöcken S effizient berechnen zu können, d.h. $S^d \equiv w \bmod n$, bestimmt der Inhaber des geheimen Schlüssels ein für allemal

$d_p := c^{-1} \bmod p-1$, so daß gilt: $(S^{d_p})^c \equiv S \bmod p$ und
$d_q := c^{-1} \bmod q-1$, so daß gilt: $(S^{d_q})^c \equiv S \bmod q$.

Um aus einem konkreten S eine c-te Wurzel zu ziehen, berechnet er:

$S^{d_p} \bmod p$, $S^{d_q} \bmod q$ und dann $w := \mathrm{CRA}(S^{d_p} \bmod p, S^{d_q} \bmod q)$.

Dann gilt sowohl $w^c \equiv (S^{d_p})^c \equiv S \bmod p$ als auch $w^c \equiv (S^{d_q})^c \equiv S \bmod q$.

Da p und q teilerfremd sind, folgt: $w^c \equiv S \bmod n$.

Um wie viel schneller ist dies nun? Im interessierenden Zahlbereich des Sicherheitsparameters l (der Länge von p und q) gilt:

Der Aufwand einer modularen Exponentiation wächst kubisch, der einer modularen Multiplikation quadratisch und der einer modularen Addition linear mit der Länge des Modulus.

Da $|n| = 2 \bullet l$, ist der Aufwand einer Exponentiation modulo n also proportional zu $(2 \bullet l)^3 = 8 \bullet l^3$.

Der Aufwand von 2 Exponentiationen halber Länge (nämlich modulo p bzw. q) ist proportional zu $2 \bullet l^3$.

Der Aufwand des CRA besteht in 2 Multiplikationen und einer Addition modulo n, ist also proportional zu $2 \bullet (2 \bullet l)^2 + 2 \bullet l = 8 \bullet l^2 + 2 \bullet l$.

Da die Proportionalitätskonstanten[78] näherungsweise gleich sind, liegt der Aufwand des CRA bereits für den kleinsten in Betracht zu ziehenden Wert von $l = 300$ deutlich unterhalb von 2% des Aufwands der 2 Exponentiationen halber Länge. Also reduziert Rechnen modulo p und q den Aufwand näherungsweise um den Faktor

$$\frac{8 \bullet l^3}{2 \bullet l^3} = 4.$$

3.6.5.3 Verschlüsselungsleistung

In [Sedl_88] wird eine Hardwareimplementierung von RSA gemäß §3.6.2 mit einer Entschlüsselungsgeschwindigkeit von ca. 200 kbit/s bei einer Moduluslänge von 660 bit angegeben. Letzteres erscheint für die Sicherheit von RSA zur Zeit als ausreichend. Die Geschwindigkeit wird durch Verwendung des in §3.6.5.2 beschriebenen Verfahrens erzielt. Erfolgt die Verschlüsselung mit dem Exponenten $2^{16}+1$, vgl. §3.6.5.1, so wird sogar eine Geschwindigkeit von 3 Mbit/s erzielt.

Eine entsprechende Software-Implementierung auf dem Apple Macintosh IIfx (MC68030, 40 MHz, 32 KByte cache board, 80 ns RAM) erreicht bei einer Moduluslänge von 512 bit eine Entschlüsselungsgeschwindigkeit von 2,5 kbit/s.

3.6.6 Sicherheit und Einsatz von RSA

Egal wie man RSA als asymmetrisches kryptographisches System einsetzt, es kann nicht sicherer sein als die Faktorisierung des Modulus schwer ist – und einen Beweis, daß RSA zu brechen so schwer wie den Modulus zu faktorisieren ist, gibt es (bisher) nicht, vgl. §3.6.1.

Wo also liegen die sinnvollen Einsatzbereiche von RSA?

Hilfreich für die folgende Diskussion ist, zu Bild 3-12 in §3.1.3.5 zurückzuschlagen. Dort sehen wir das wichtige Feld 3 nur mit einem System gefüllt, das nicht äquivalent zu einer reinrassigen Standardannahme ist. D.h. effiziente asymmetrische **Konzelationssysteme**, die gegen aktive Angriffe relativ zu einer reinrassigen Standardannahme kryptographisch stark sind, sind bisher nicht bekannt.

Zwar ist RSA nicht als kryptographisch stark bewiesen (hier also: so sicher wie Faktorisierung schwer), aber für den in §3.6.4.1 beschriebenen Einsatz als indeterministisches asymmetrisches Konzelationssystem sind immerhin keine erfolgreichen Angriffe bekannt – und das schon seit vielen Jahren. Benötigt man ein asymmetrisches Konzelationssystem und kann man aktive Angriffe nicht ausschließen, sollte man RSA (oder CS) einsetzen. Kann man aktive Angriffe ausschließen, ist der s^2-mod-n-Generator vorzuziehen: Zum einen ist er als kryptographisch stark bewiesen, d.h. wer ihn bricht, kann faktorisieren und damit auch RSA brechen. Zum andern ist für den s^2-mod-n-Generator sogar bewiesen, daß ein passiver Angreifer keinerlei Information über den Klartext erhält. Auch solch einen Beweis gibt es für RSA (bisher?) nicht.

Bezüglich des Rechenaufwands sind s^2-mod-n-Generator und RSA vergleichbar. Dies liefert also kein hartes Unterscheidungskriterium.

Für den Einsatz von RSA statt GMR als digitales **Signatursystem** spricht unter Sicherheitsgesichtspunkten nichts. Im Gegenteil: Wer GMR brechen kann, kann – wie bewiesen wurde – faktorisieren und damit auch RSA brechen. Man sollte, wo immer möglich, also GMR den Vorzug geben.

Bezüglich des Rechenaufwands verhält es sich leider umgekehrt: Wird für RSA gemäß §3.6.4.2 eine genauso effiziente Hashfunktion wie bei GMR verwendet (eine aufwendigere zu verwenden,

---

78 für den kubischen Aufwand bei der Exponentiation, den quadratischen Aufwand bei der Multiplikation und den linearen Aufwand bei der Addition

wäre unnötiger Aufwand, der keine Sicherheit bringen kann), sind beide in etwa gleich aufwendig, wobei RSA etwas günstiger liegt, da kein Referenzenbaum erzeugt bzw. geprüft zu werden braucht. Allerdings ist es möglich, bei RSA eine wesentlich effizientere Hashfunktion zu verwenden, da diese *nie* umgekehrt werden muß. Dies ist bei GMR nur bzgl. der N-Signaturen nicht nötig – bei R- und K-Signaturen aber unvermeidbar, vgl. Bild 3-27. Der Rechenaufwand für die Erzeugung und Prüfung der R- und K-Signaturen kann bei GMR also nicht in gleicher Weise gesenkt werden.

Die Verwendung einer wesentlich effizienteren Hashfunktion, die von niemand umgekehrt werden kann, kann einen enormen Effizienzvorteil haben, erfordert aber eine weitere kryptographische Annahme, nämlich die, daß die verwendete schnelle Hashfunktion kollisionsresistent ist. Je mehr unbewiesene Annahmen man benötigt, desto größer wird die Wahrscheinlichkeit, daß mindestens eine falsch ist und damit die Sicherheit pure Illusion. Bezüglich digitalen Signatursystemen kann (und muß) man sich also zwischen Sicherheit (GMR) und Effizienz (RSA mit schneller Hashfunktion) entscheiden.

RSA ist für die USA, aber nirgends sonst patentiert. Das Patent läuft am 20. September 2000 aus [Schn_96 Seite 474].

3.7 DES: Das bekannteste System für symmetrische Konzelation und Authentikation

DES ist das erste standardisierte (daher sein Name als Abkürzung von: Data Encryption Standard) und (deshalb) das am meisten eingesetzte kryptographische System. DES ist ein symmetrisches kryptographisches System, das einen nur 56 Bit langen Schlüssel hat und Blöcke von 64 Bit Länge verschlüsselt.

Erstmals veröffentlicht wurde der DES-Algorithmus in [DES_77]. Unter anderem abgedruckt ist er in [MeMa_82].

3.7.1 DES im Überblick

Der DES-Algorithmus setzt sich zusammen aus einer – kryptographisch unbedeutenden – **Eingangspermutation IP** (Initial Permutation), die u.a. den Eingabeblock in die beiden 32-Bit-Blöcke L_0 und R_0 aufteilt, **16 Iterationsrunden**, in denen die eigentliche Verschlüsselung vorgenommen wird, und einer zur Eingangspermutation inversen **Ausgangspermutation IP^{-1}**, vor deren Ausführung L_{16} und R_{16}, die beiden Blöcke, die man nach der Iterationsrunde 16 erhält, nochmals vertauscht werden.

Aus dem Schlüssel werden die 16 **Teilschlüssel** K_1 bis K_{16} erzeugt – für jede Iterationsrunde einer.

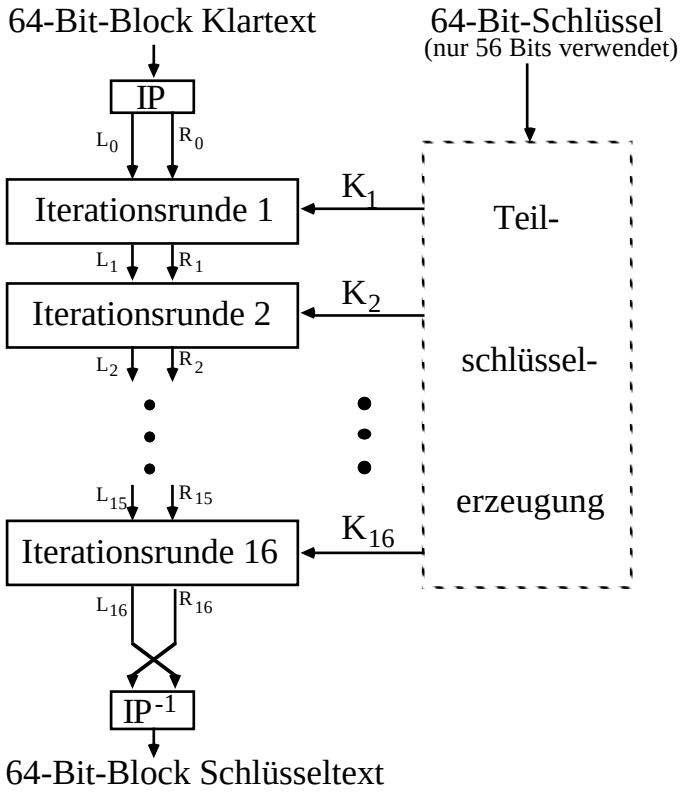


Bild 3-33: Ablaufstruktur von DES

3.7.2 Eine Iterationsrunde

Jede Iterationsrunde *i* (s. Bild 3-34) erhält als Eingabe die Blöcke L_{i-1} und R_{i-1} und liefert die Blöcke L_i und R_i. Dabei erhält man L_i direkt aus R_{i-1}, R_i durch Addition von L_{i-1} und $f(R_{i-1}, K_i)$ (mit Addition ist jeweils die bitweise Addition modulo 2 gemeint):

$$L_i := R_{i-1} \quad (1)$$

$$R_i := L_{i-1} \oplus f(R_{i-1}, K_i) \quad (2)$$

f bezeichnet dabei die **Verschlüsselungsfunktion**.

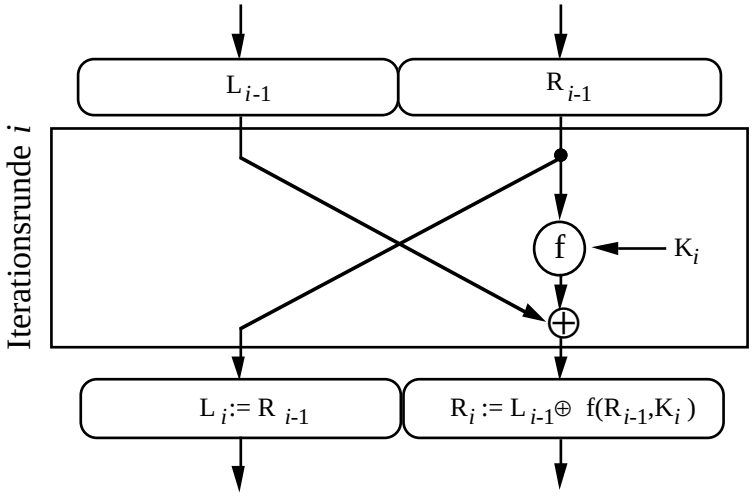


Bild 3-34: Eine Iterationsrunde von DES

3.7.3 Das Entschlüsselungsprinzip von DES

In Bild 3-34 wurde dargestellt, wie in DES eine Iterationsrunde abläuft. Wesentlich für die Entschlüsselung in DES ist nun, daß dieser Prozeß umgekehrt werden kann. Als Ergebnis der Iterationsrunde *i* erhält man:

$L_i := R_{i-1}$ (1)

$R_i := L_{i-1} \oplus f(R_{i-1}, K_i)$ (2)

Man vertauscht nun die beiden Blöcke L_i und R_i miteinander. Dies geschieht in Bild 3-35, indem die Bezeichner für die Werte der linken Bildhälfte an deren Stelle in der rechten Bildhälfte geschrieben sind, obwohl bei Wahl von neuen Bezeichnern gemäß der Regel L = links und R = rechts bei allen Bezeichnern in der rechten Bildhälfte L durch R und umgekehrt ersetzt werden müßte. Außerdem wird rechts der Index der Iterationsrunde rückwärts statt vorwärts gezählt. Führt man dann dieselbe Iteration nochmals aus, so gilt:

$L_i =: R_{i-1}$ (3)

und

$L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(L_i, K_i) = L_{i-1} \oplus f(L_i, K_i) \oplus f(L_i, K_i) =: L_{i-1}$ (4)

Die notwendige Vertauschung der Blöcke wird im DES-Algorithmus nach der 16. Iterationsrunde vorgenommen. Diese Vertauschung wird also vor Beginn der Entschlüsselung durchgeführt. Sie bleibt bei den einzelnen Entschlüsselungsrunden erhalten und wird nach der letzten Entschlüsselungsrunde durch eine abermalige Vertauschung rückgängig gemacht. So kann man mit einem einzigen Algorithmus Daten sowohl ver- als auch entschlüsseln. Man muß lediglich die Reihenfolge der Iterationsrunden umkehren, was man durch eine Umkehrung der Reihenfolge der Teilschlüssel erreicht.

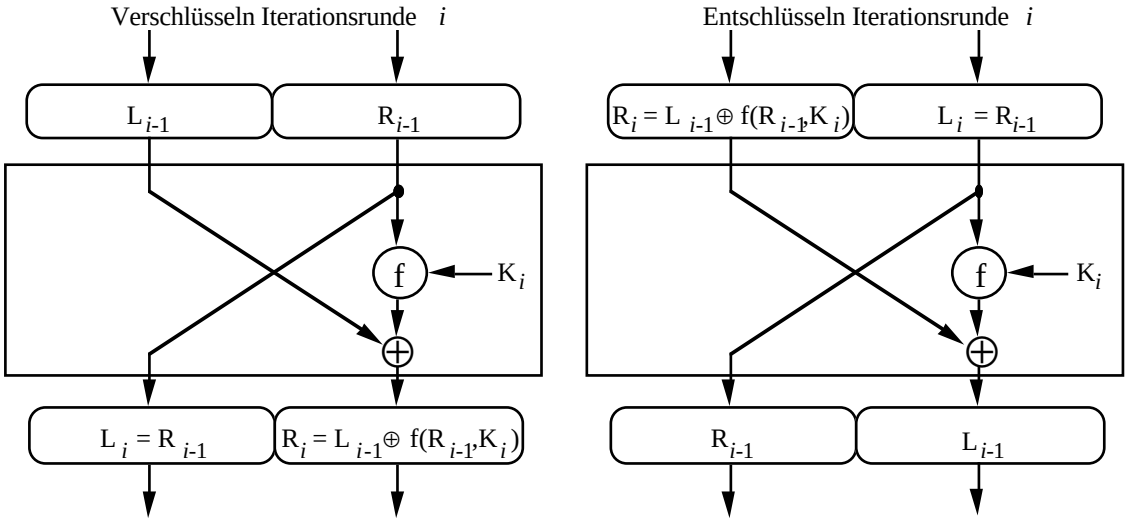


Bild 3-35: Entschlüsselungsprinzip von DES

Man erkennt, daß das Entschlüsselungsprinzip unabhängig ist von der Länge der Blöcke und der Wahl der Eingangspermutation IP (und damit auch von IP^{-1}). Auch die Verschlüsselungsfunktion f kann frei gewählt werden.[79]

---

79 Mit den im nächsten Abschnitt eingeführten Bezeichnern erklärt bedeutet dies, daß für die Expansionspermutation E, die Substitutionsboxen S_i (i=1,..,8) und die Permutation P unter Beibehaltung der Grundstruktur selbstdefinierte Tabellen eingesetzt werden können und auch die Teilschlüsselerzeugung und die Art der Verknüpfung der Teilschlüssel mit dem Block R_{i-1} variabel sind. Zusätzlich sei darauf hingewiesen, daß zur Entschlüsselung f selbst in keiner Weise umgekehrt werden muß.

Zwar ist die Wahl der Verschlüsselungsfunktion f für die Umkehrbarkeit bei Kenntnis des Schlüssels ohne Belang. Für die Sicherheit, d.h. die Nichtumkehrbarkeit für alle, die den Schlüssel nicht kennen, aber essentiell. Deshalb wird ihr grober Aufbau kurz beschrieben.

3.7.4 Die Verschlüsselungsfunktion

Der erste Parameter der in Bild 3-36 dargestellten Verschlüsselungsfunktion, ein 32-Bit-Block, wird durch die **Expansionspermutation E** auf 48 Bit erweitert. (Eine Permutation vertauscht einzelne Bits, ohne ihre Werte oder Anzahl zu ändern. Bei einer Expansionspermutation hingegen treten manche Eingabebits in der Ausgabe mehrfach auf.)

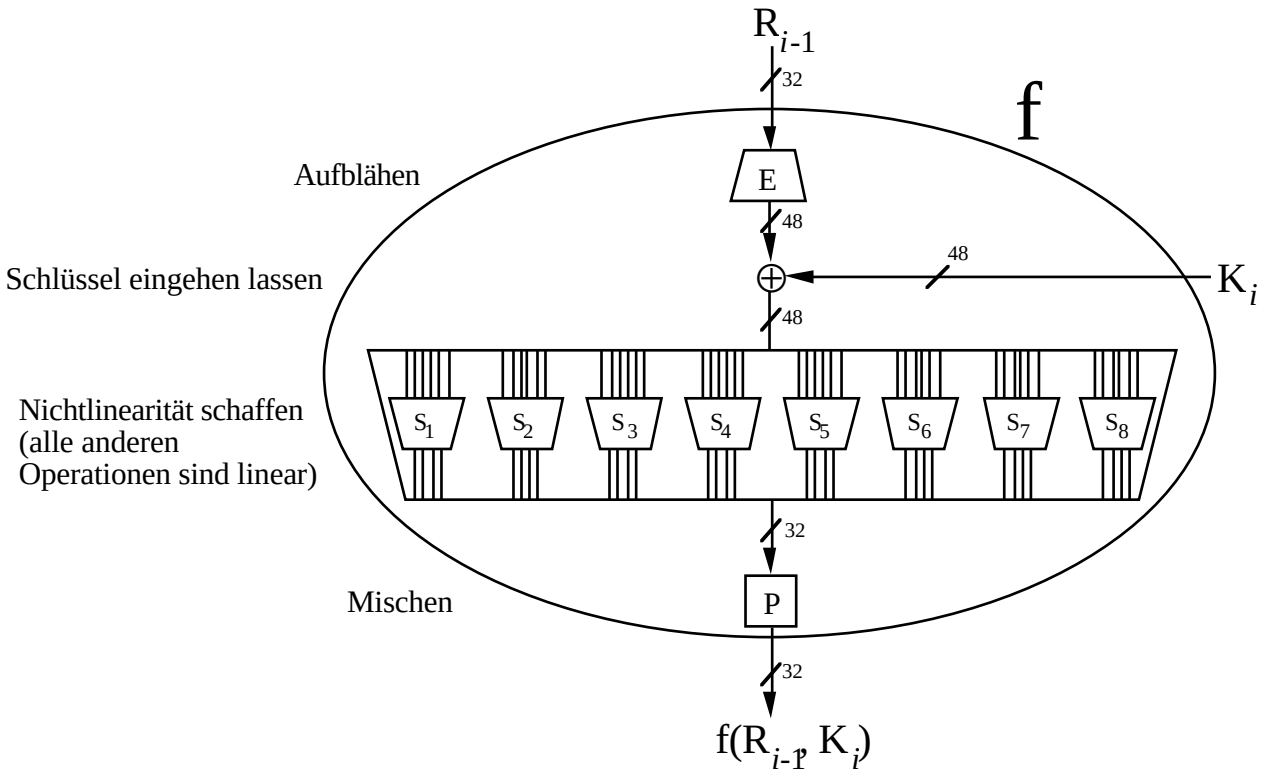


Bild 3-36: Verschlüsselungsfunktion f von DES

Anschließend wird der zweite Parameter, ein 48-Bit-Block (nämlich einer der Teilschlüssel) **hinzuaddiert** (bitweise modulo 2).

Die Summe wird in acht 6-Bit-Blöcke aufgeteilt, mit denen jeweils ein 4-Bit-Wert aus einer **Substitutionsbox S_j**, (j = 1,..., 8) ausgewählt wird. (Die Substitutionsboxen arbeiten nicht bitweise, sondern ordnen Werten von Bit-Blöcken Werte von Bit-Blöcken zu. Sie schaffen so Nichtlinearität, während bei Permutationen und Addieren-modulo-2 jeweils Invertieren eines Eingabebits ein inverses Ausgabebit bewirkt. Wäre DES linear, könnte z.B. bei einem Klartext-Schlüsseltext-Angriff der Wert des Schlüssels Bit für Bit bestimmt werden, vgl. Aufgabe 3-17 f).)

Die acht 4-Bit-Werte werden zu einem 32-Bit-Block zusammengefaßt. Auf diesen wird die **Permutation P** angewandt, wodurch man das Funktionsergebnis erhält. P mischt die Ausgaben der Substitutionsboxen, damit in der nächsten Iterationsrunde die Ausgaben jeder Substitutionsbox in mehrere Substitutionsboxen eingehen. So wird mit acht kleinen Substitutionsboxen mit je 2^6 Einträgen von je 4 Bit, insgesamt also einem Speicheraufwand von $8 \cdot 2^6 \cdot 4$ Bit = 2^8 Byte, näherungsweise eine große Substitution mit 2^{48} Einträgen von je 32 Bit realisiert, für die insgesamt ein Speicheraufwand von $2^{48} \cdot 32$ Bit = 2^{50} Byte nötig wäre.

3.7.5 Die Teilschlüsselerzeugung

Wie Bild 3-37 zeigt, werden aus dem 64-Bit-Schlüssel durch die permutierende Auswahlfunktion PC-1 (PC: permuted choice) 56 Bits ausgewählt und auf die beiden 28-Bit-Blöcke C_0 und D_0 aufgeteilt. Man erhält C_i und D_i (i = 1, .., 16), indem man die Bits in C_{i-1} und D_{i-1} um 1 oder 2 Stellen (abhängig von i) für die Verschlüsselung nach links rotiert (zyklischer Linksshift LS_i). Die Teilschlüssel K_i gewinnt man dann, indem man C_i und D_i konkateniert und mit PC-2 daraus 48 Bits auswählt.

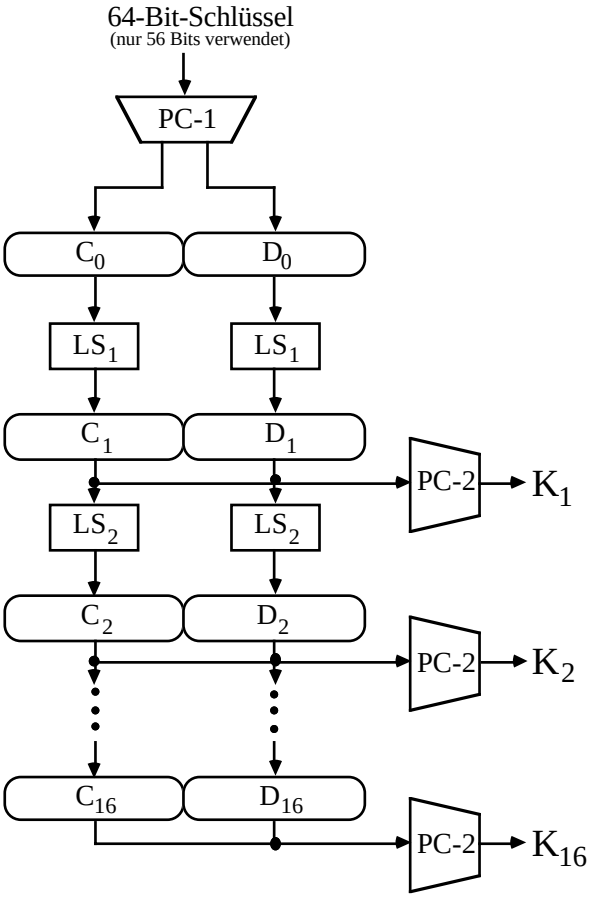


Bild 3-37: Teilschlüsselerzeugung von DES

Bei DES wird für die Verschlüsselung in 12 der 16 Iterationsrunden um 2 Bit nach links rotiert, in den anderen 4 Iterationsrunden um 1 Bit, so daß insgesamt um 28 Bit rotiert wird und damit am Schluß der gleiche Zustand wie am Anfang vorliegt, d.h. $C_0 = C_{16}$ und $D_0 = D_{16}$. Folglich besteht eine Möglichkeit, die Teilschlüssel für die Entschlüsselung zu generieren darin, einfach um die entsprechenden (d.h. in ihrer Reihenfolge umgedrehten) Bitanzahlen nach rechts zu rotieren.

3.7.6 Komplementaritätseigenschaft von DES

Bei Wahl dieser Verschlüsselungsfunktion und Teilschlüsselerzeugung gibt es bei DES eine einzige einfache Regelmäßigkeit, die sogenannte **Komplementaritätseigenschaft**:

$$\mathrm{DES}(\overline{k}, \overline{x}) = \overline{\mathrm{DES}(k, x)}$$

Werden Klartextblock und Schlüssel bitweise komplementiert (jede 0 durch eine 1 ersetzt und umgekehrt), so ergibt sich der komplementäre Schlüsseltext, denn

Permutationen erhalten die Komplementarität, insbesondere also PC-1 und PC-2 sowie IP und IP^{-1},

Rotationen erhalten die Komplementarität, so daß bei komplementärem Schlüssel auch die Teilschlüssel komplementär sind,

jede Iterationsrunde erhält die Komplementarität,

da nach der Expansionspermutation E zum komplementären Wert der komplementäre Teilschlüssel modulo 2 addiert wird, so daß sich vor den Substitutionen die Komplementarität weghebt (da $0 \oplus 0 = 1 \oplus 1$ und $1 \oplus 0 = 0 \oplus 1$), nach den Substitutionen aber durch die Addition modulo 2 von L_{i-1} wieder hergestellt wird, und

da das Kopieren von R_{i-1} nach L_i trivialerweise die Komplementarität erhält,

Vertauschen von linker und rechter Hälfte erhält die Komplementarität.

Die Bilder 3-38 und 3-39 verdeutlichen zusammen den Erhalt der Komplementarität in jeder Iterationsrunde.

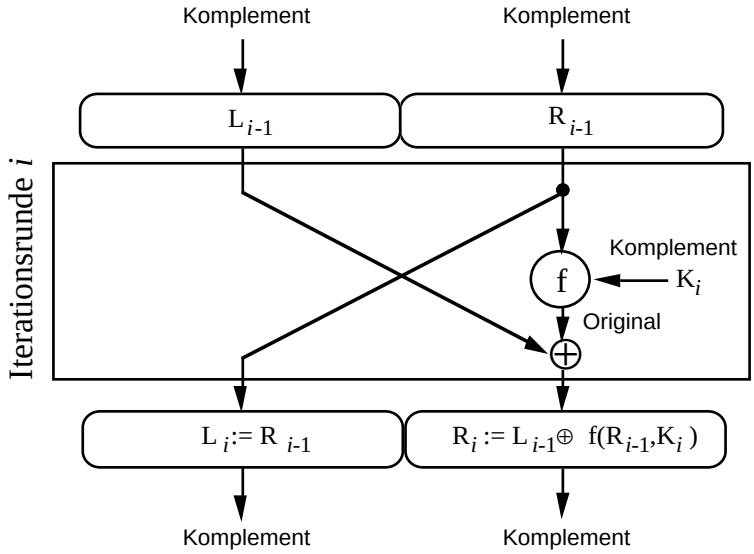


Bild 3-38: Komplementarität in jeder Iterationsrunde von DES

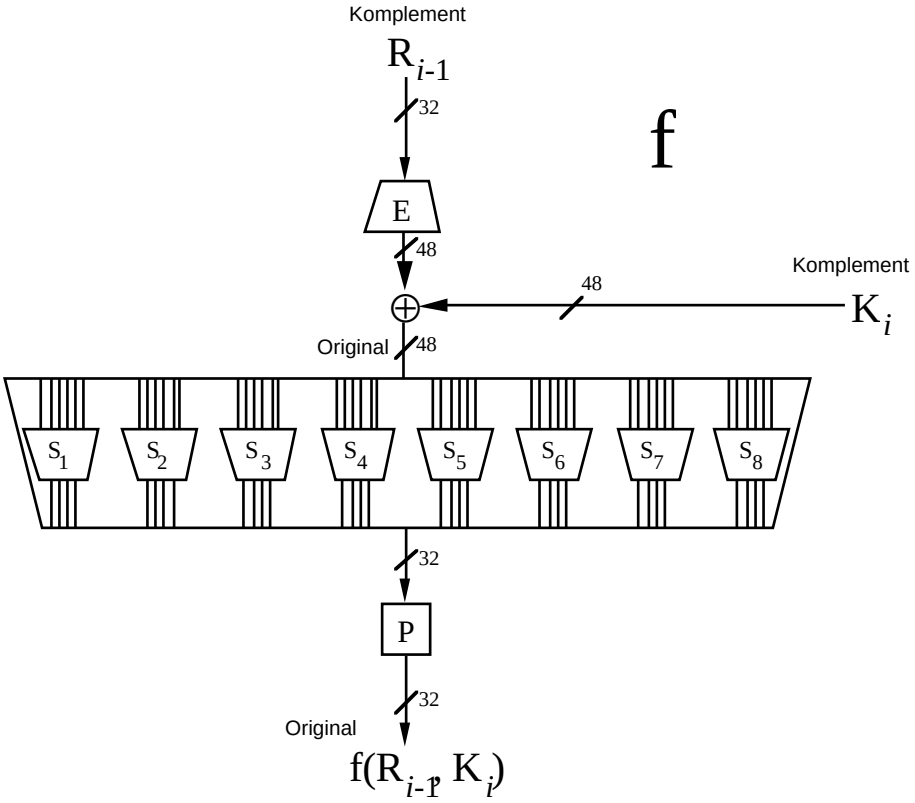


Bild 3-39: Komplementarität in der Verschlüsselungsfunktion f von DES

Diese Regelmäßigkeit kann genutzt werden, um den Aufwand der Suche durch den Schlüsselraum nahezu zu halbieren: Liegen zu zwei komplementären Klartextblöcken die zugehörigen Schlüsseltextblöcke vor (etwa als Ergebnis eines gewählten Klartext-Schlüsseltext-Angriffs), dann wird einer der Klartextblöcke solange mit anderen Schlüsseln verschlüsselt, bis entweder der zugehörige Schlüsseltextblock oder das Komplement des anderen Schlüsseltextblocks entsteht. Im 1. Fall wurde der Schlüssel gefunden. Im 2. Fall wurde das Komplement des Schlüssels gefunden, da Komplement des Klartextblocks mit Komplement des Schlüssels verschlüsselt das Komplement des Schlüsseltextes ergibt.

In beiden Fällen kann noch mit sehr geringer Wahrscheinlichkeit die Situation eingetreten sein, daß zu einem einzelnen Klartext-Schlüsseltext-Paar mehrere Schlüssel passen, so daß die Richtigkeit des Schlüssels in beiden Fällen noch an weiteren Klartext-Schlüsseltext-Paaren geprüft werden muß. Dies gilt – unabhängig von der Komplementaritätseigenschaft – allgemein für DES.

3.7.7 Verallgemeinerung von DES: G-DES

Die Sicherheit von DES wird seit langem aus zwei Gründen kritisiert:

- Der Schlüssel ist mit 56 Bit Länge inzwischen für Anwendungen, die Sicherheit auch bei massiven Angriffen (etwa durch Geheimdienste) erfordern, zu kurz:

  Mit einem Klartext-Schlüsseltext-Angriff wurde DES im Januar/Februar 1998 vollständig gebrochen. Während ihrer Leerlaufzeit durchsuchten 22000 Rechner über das Internet koordiniert brute-force den Schlüsselraum. Innerhalb von 39 Tagen wurden dabei etwa 85% aller möglichen Schlüssel ausprobiert, bis der richtige gefunden wurde [CZ_98]. Alternativ zu diesem software-basierten, nahezu kostenlosen Angriff wird der Bau von Spezialmaschinen diskutiert [Schn_96 Seite 152-157]: Bei der 1995 verfügbaren Technologie sei das Produkt aus Kosten der Maschine in US$ und durchschnittlicher Zeit in

Sekunden zur Ermittlung eines DES-Schlüssels etwa 10^{10}. Für 10^5 US$ kann man also eine Maschine bauen, die in 35 Stunden einen Schlüssel ermittelt, für 10^6 US$ dauert es nur noch 3,5 Stunden, usw. In der Vergangenheit verdoppelte sich die Rechenleistung bei gleichen Kosten etwa alle 18 Monate. Obiges Produkt ist also vermutlich auch zukünftig etwa alle 18 Monate zu halbieren und beträgt im Jahr 2000 dann etwa 10^9. Im Juli 1998 veröffentlichte Electronic Frontier Foundation (EFF), daß sie innerhalb eines Jahres mit einem Gesamtaufwand von weniger als 250000 US$ eine Spezialmaschine zum Brechen von DES gebaut haben, die DES-Schlüssel in weniger als 3 Tagen ermittelt [EFF_98]. Unter Verwendung dieser oberen Schranken liegt das tatsächlich erzielte Produkt von $250000 \cdot 3 \cdot 24 \cdot 60 \cdot 60 = 6{,}48 \cdot 10^{10}$ leicht oberhalb der am Schreibtisch ermittelten Angaben für die 1995 verfügbare Technologie.

- Die Entwurfskriterien für die speziellen Werte der Substitutionen und Permutationen (hier nicht beschrieben, bei Interesse im Standard nachzulesen) werden weiterhin geheimgehalten. In [Copp_92, Copp2_94] findet sich allerdings die weitgehende Aufdeckung des Rätsels.

Unter kryptographischen und Implementierungsgesichtspunkten bieten sich folgende Verallgemeinerungen an, die diesen zwei Kritikpunkten Rechnung tragen:

1. Die je 48 Schlüsselbits der 16 Teilschlüssel werden direkt eingegeben statt aus 56 Bit erzeugt. Der Schlüssel ist damit $16 \cdot 48 = 768$ Bit statt 56 Bit lang.
2. Die Inhalte der Substitutionsboxen S_j (j = 1,..., 8) werden variabel. Dies verlängert den Schlüssel um maximal $8 \cdot 2^6 \cdot 4 = 2048$ Bits.
3. Die Permutationen P und IP werden variabel. Dies verlängert den Schlüssel um maximal $32 \cdot \lceil \mathrm{ld}\ 32 \rceil + 64 \cdot \lceil \mathrm{ld}\ 64 \rceil = 160 + 384 = 544$ Bits.
4. Die Expansionspermutation E wird variabel. Dies verlängert den Schlüssel um maximal $48 \cdot \lceil \mathrm{ld}\ 32 \rceil = 240$ Bits.

Zusammen ergibt dies maximal 3600 Bit. Bezüglich des Aufwandes der Speicherung ist dies bereits heute weniger als 56 Bit im Jahre 1977. Während wie bei DES bei der 1. Verallgemeinerung eine zufällige Wahl des Schlüssels angebracht ist, gilt dies bei der 2., 3. und 4. Verallgemeinerung nicht unbedingt [BiSh_93].

Damit hier keine falschen Erwartungen an den Sicherheitsgewinn entstehen: Während der „normale“ DES mittels eines gewählten Klartext-Schlüsseltext-Angriffs, genannt differentielle Kryptoanalyse (differential cryptanalysis), mittels 2^{47} Klartext-Schlüsseltext-Blöcken gebrochen werden kann, sind hierfür bei der 1. Verallgemeinerung 2^{61} Klartext-Schlüsseltext-Blöcke nötig [BiSh3_91, BiSh4_91, BiSh_93]. Dies ist zwar vom naiv erhofften Aufwand von 2^{768}-1 weit entfernt, aber schon sehr nahe an der optimalen Schwierigkeit von 2^{64}-1: *Jede* permutierende Blockchiffre mit Blocklänge b kann mittels aktivem Angriff mit Aufwand 2^b-1 gebrochen werden – der Angreifer läßt sich einfach alle anderen Blöcke als den, der ihn interessiert, ver- bzw. entschlüsseln.

3.7.8 Verschlüsselungsleistung

Mittels einer Software-Implementierung werden auf dem Apple Macintosh Classic (MC68000, 8 MHz) mit einem gemäß 1., 2. und 3. verallgemeinerten DES über 100 kbit/s ver-/entschlüsselt, auf dem PowerBook 180 (MC 68030, 33 MHz) über 900 kbit/s [PfAß1_90, PfAß_93].

Mittels einer Hardware-Implementierung (Standardzellen-Entwurf, 7155 Gatteräquivalente, 2 µm Technologie) werden mit einem gemäß 1. verallgemeinerten DES auf dem Chip 40 Mbit/s ver-/entschlüsselt. Da die Ein-/Ausgabe aber nur byteweise und asynchron (2-Weg-Handshake) erfolgt, können nur 10 Mbit/s ein-/ausgegeben werden [KFBG_90].

3.8 Zum praktischen Betrieb kryptographischer Systeme

Zum praktischen Betrieb kryptographischer Systeme ist es nützlich,

- Begriffe wie Blockchiffre vs. Stromchiffre, synchron vs. selbstsynchronisierend,
- Betriebsarten von Blockchiffren, d.h. die wichtigen Konstruktionen von selbstsynchronisierenden und synchronen Stromchiffren aus Blockchiffren, und
- wenigstens eine effiziente Konstruktion von kollisionsresistenten Hashfunktionen aus Blockchiffren

zu kennen. Dies wird jeweils in den folgenden Unterkapiteln behandelt.

3.8.1 Blockchiffre, Stromchiffre

Bei einem gegebenen endlichen Alphabet ver-/entschlüsselt eine

- **Blockchiffre** nur Zeichenketten *fester* Länge, während eine
- **Stromchiffre** Zeichenketten *variabler* Länge bearbeiten kann.

Legt man das Alphabet {0, 1} zugrunde, sind die Vernam-Chiffre (one-time pad), der s^2-mod-n-Generator und GMR Stromchiffren, während RSA und DES dann Blockchiffren sind. Legt man jedoch als Alphabet {0, 1, 2, 3, ..., 2^{64}-1} zugrunde und unterstellt, daß mehrere Zeichen jeweils unabhängig voneinander verschlüsselt werden, dann ist DES nach obiger Definition eine Stromchiffre. Die Unterscheidung Blockchiffre-Stromchiffre nach obiger Definition hängt also wesentlich vom unterstellten Alphabet ab und ist deshalb in sich nur relativ. Bei vielen Anwendungen ist das Alphabet aber vorgegeben (oder zumindest kanonisch) und damit die Unterscheidung präzise.

(Anmerkung: Manche Autoren definieren die Klasse der Stromchiffre restriktiver: Bei ihnen ist bei jeder Stromchiffre ein Schlüsselstrom entweder vorgegeben (one-time pad) oder wird generiert (pseudo one-time pad, z.B. s^2-mod-n-Generator) und zum Klartext- bzw. Schlüsseltextstrom addiert bzw. subtrahiert. Mir erscheint diese restriktivere Definition für das folgende jedoch unnötig.)

Bei einer Stromchiffre (im Deutschen auch als kontinuierliche Chiffre [Hors_85] oder Stromsystem [HeKW_85], im Englischen als stream cipher bezeichnet) werden Nachrichten als eine Folge von *Zeichen* codiert, so daß einzelne Zeichen des Alphabets verschlüsselt werden. Diese Zeichen werden jedoch nicht notwendigerweise unabhängig voneinander verschlüsselt, sondern ihre Verschlüsselung kann auch entweder

1. von ihrer Position innerhalb der Nachrichten oder allgemeiner von allen vorhergehenden Klartext- und/oder Schlüsseltextzeichen abhängen oder
2. nur von einer beschränkten Anzahl direkt vorhergehender Schlüsseltextzeichen.

Im ersten Fall spricht man von ***synchronen*** *Stromchiffren* (synchronous stream ciphers), da Ver- und Entschlüsselung streng synchron erfolgen muß: bei Verlust oder Hinzufügen eines Schlüsseltextzeichens, d.h. bei Verlust der Synchronisation, kann nicht mehr ohne weiteres entschlüsselt werden, Ver- und Entschlüsseler müssen sich neu synchronisieren. Sofern Ver- und Entschlüsselung nicht nur von der Position innerhalb der Nachrichten abhängt, sondern auch von allen vorhergehenden Klartext und/oder Schlüsseltextzeichen, müssen Ver- und Entschlüsseler sich auch bei Verfälschung eines Schlüsseltextzeichens neu synchronisieren.

Im zweiten Fall spricht man von ***selbstsynchronisierenden*** *Stromchiffren* (self-synchronous stream ciphers), da sich bei ihnen der Entschlüsseler auch bei Verlust oder Hinzufügen beliebig vieler zusätzlicher Schlüsseltextzeichen spätestens nach Entschlüsselung der oben erwähnten beschränkten Anzahl Schlüsseltextzeichen wieder auf den Verschlüsseler synchronisiert hat.

(Anmerkung: Nach obigen Definitionen ist jede Stromchiffre entweder synchron oder selbstsynchronisierend.[80] Manche Autoren definieren synchron jedoch restriktiver, z.B. [Denn_82 Seite 136]: Der Schlüsselstrom – siehe obige Anmerkung – wird unabhängig vom Klartextstrom erzeugt. Dann gibt es Stromchiffren, die weder synchron noch selbstsynchronisierend sind, z.B. Schlüsseltext- und Ergebnisrückführung in Bild 3-50. Mir erscheint diese feinere Aufteilung in drei Klassen für das folgende jedoch unnötig.)

Für jede symmetrische bzw. asymmetrische deterministische Stromchiffre und beliebige nichtleere Texte x_1, x_2 und Schlüsselpaare (c,d) bzw. Schlüssel k gilt demnach:

Werden zwei Texte separat, aber direkt hintereinander verschlüsselt, so ist das Gesamtergebnis das gleiche, wie wenn die zwei Texte erst konkateniert und dann verschlüsselt werden. In der eingeführten Notation und bei kollateraler (d.h. paralleler, sich gegenseitig nicht beeinflussender) Auswertung beider Seiten der Gleichungen jeweils von links lautet dies:

$k(x_1),k(x_2) = k(x_1,x_2)$ bzw. $c(x_1),c(x_2) = c(x_1,x_2)$.

Werden zwischen den separat verschlüsselten Texten noch weitere verschlüsselt oder wird nach den beiden separaten Verschlüsselungen die der konkatenierten Texte als dritte ausgeführt (also nicht kollateral!), so gelten die obigen Gleichungen mit sehr großer Wahrscheinlichkeit nicht.

3.8.2 Betriebsarten für Blockchiffren

Um Zusammenhänge zwischen Block- und Stromchiffren aufzuzeigen und da sie für viele Anwendungen praktisch sind[81], werden im folgenden die in der Literatur beschriebenen und teilweise genormten wichtigen Betriebsarten für Blockchiffren (sprich: Konstruktionen von selbstsynchronisierenden oder synchronen Stromchiffren aus Blockchiffren) angegeben. Auch wenn heute allgemein davon ausgegangen wird, daß in allen Konstruktionen die Verwendung einer sicheren Blockchiffre üblicherweise den Erhalt einer sicheren Stromchiffre impliziert, erscheinen mir zwei einschränkende Bemerkungen notwendig:

1. Selbst unter einer sehr starken Definition, was eine sichere Blockchiffre ist, sind mir weder (Reduktions-)Beweise für die Sicherheit der erhaltenen Stromchiffren noch eine Quantifizierung des „üblicherweise“ bekannt.
2. Zumindest für manche dieser Konstruktionen konnten pathologische (d.h. extrem gebildete, krankhafte) Gegenbeispiele hergeleitet werden, die ich im folgenden beschreibe. Ich überlasse es dem Leser zu entscheiden, ob die hergeleitete Blockchiffre von ihm als „sicher“ betrachtet wird.

Zunächst werden die standardisierten Betriebsarten ECB, CBC, CFB und OFB beschrieben. Danach zwei Verallgemeinerungen, nämlich PCBC und OCFB:

Aus jeder symmetrischen oder asymmetrischen deterministischen Blockchiffre kann mittels der Konstruktionen

- *Elektronisches Codebuch* (electronic codebook, abgekürzt ECB [DaPr_89]),
- *Blockchiffre mit Blockverkettung* (cipher block chaining, abgekürzt CBC [DaPr_89]) oder
- *Schlüsseltextrückführung* (cipher feedback, abgekürzt CFB [DaPr_89])

---

80 Um auch die Möglichkeit zuzulassen, daß jeweils zeichenweise unabhängig verschlüsselt wird, kann im zweiten Fall die Anzahl direkt vorhergehender Schlüsseltextzeichen, von denen die Verschlüsselung abhängt, auf 0 beschränkt werden. Dann ist auch das erwähnte Beispiel abgedeckt, bei dem DES als Stromchiffre auf dem Alphabet $\{0, 1, 2, 3, ..., 2^{64}-1\}$ betrachtet wird.

81 Bis hierher ist unklar, ob man beispielsweise mit DES auch Nachrichten konzelieren kann, die nicht genau 64 Bit lang sind. Ebenso ist bis hierher unklar, ob mit DES Nachrichten auch authentisiert werden können.

eine *selbstsynchronisierende* Stromchiffre gewonnen werden. Hierbei wird bei ECB und CBC das den Begriffen Block- bzw. Stromchiffre zugrundeliegende Alphabet gewechselt – bei CFB kann es gewechselt werden.

Aus jeder symmetrischen oder asymmetrischen deterministischen Blockchiffre kann mittels der Konstruktionen

- *Ergebnisrückführung* (output feedback, abgekürzt OFB [DaPr_89]),
- *Blockchiffre mit Blockverkettung über Schlüssel- und Klartext* (hier plain cipher block chaining, abgekürzt PCBC, genannt, in der Literatur plaintext-ciphertext feedback oder block chaining using plaintext and ciphertext feedback, [EMMT_78 Seite 110, 111, MeMa_82 Seite 69]) oder
- *Schlüsseltext- und Ergebnisrückführung* (output cipher feedback, abgekürzt OCFB)

eine *synchrone* Stromchiffre gewonnen werden. Hierbei wird bei PCBC das den Begriffen Block- bzw. Stromchiffre zugrundeliegende Alphabet gewechselt – bei OFB und OCFB kann es gewechselt werden.

Jeder der 6 Konstruktionen ist ein eigenes Unterkapitel gewidmet.

An diese schließt sich ein Unterkapitel mit Anmerkungen zur Verwendung expandierender Blockchiffren (in den Unterkapiteln vorher wird nur der Fall behandelt, daß die Blockchiffre längentreu ist) und zur Initialisierung der Speicherglieder an.

Im letzten Unterkapitel wird ein Überblick über die wesentlichen Eigenschaften aller beschriebenen Konstruktionen gegeben.

3.8.2.1 Elektronisches Codebuch (ECB)

Die einfachste Betriebsart einer Blockchiffre ist, lange Nachrichten so in Blöcke aufzuspalten, daß jeder Block *unabhängig* von allen andern mit der Blockchiffre ver- und entschlüsselt werden kann, vgl. Bild 3-40. Diese Betriebsart wird **elektronisches Codebuch** genannt (electronic codebook, abgekürzt ECB).

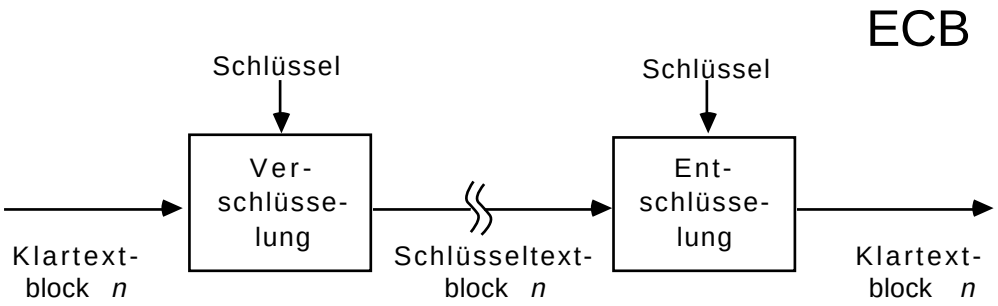
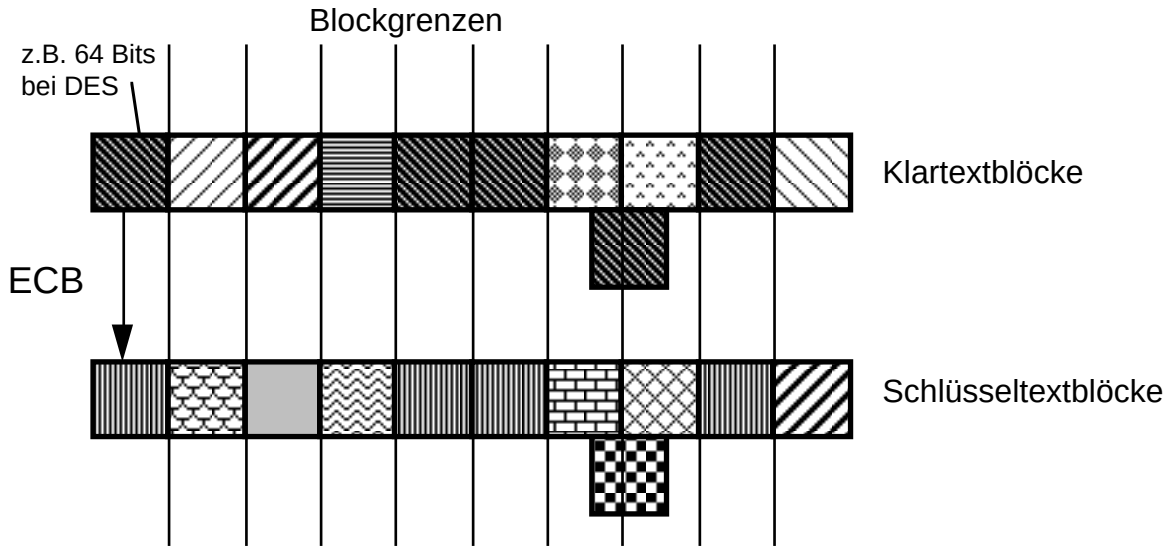


Bild 3-40: Betriebsart elektronisches Codebuch

Ihre Eigenschaften sind kanonisch und deshalb nur kurz in Bild 3-51 tabelliert. Nachteilig ist, daß bei deterministischen Blockchiffren Muster im Klartext, die ganze Blöcke umfassen, auf entsprechende Muster im Schlüsseltext abgebildet werden[82], vgl. Bild 3-41. Selbst bei indeterministischen Block-

82 In einem gewissen Sinne gibt es bei nichtkontextbezogener (d.h. nicht von der Vorgeschichte abhängiger) deterministischer Verschlüsselung semantischer Einheiten immer folgendes **Brechen durch Lernen**: Ein Angreifer beobachtet Nachrichten (Schlüsseltexte) und was dann in der realen Welt passiert. Wenn die gleiche Nachricht, d.h. der gleiche Schlüsseltext noch mal auftritt, liegt die Vermutung nahe, daß in der realen Welt auch das gleiche wie damals passieren wird. Dies sei an einem kleinen Beispiel veranschaulicht: Der Admiral eines Flottenverbandes gebe

chiffren werden Muster im Schlüsseltext auf entsprechende Muster im Klartext abgebildet, aber dies dürfte in der Praxis so gut wie nie vorkommen, während regelmäßige Muster in nicht komprimierten Klartexten durchaus häufig sind und deshalb ins Kalkül gezogen werden müssen.[83]



Bei der Betriebsart ECB werden gleiche Klartextblöcke auf gleiche Schlüsseltextblöcke abgebildet. Muster, die Blockgrenzen überschreiten, werden i.allg. auf andere Muster im Schlüsseltext abgebildet, wie dies am Beispiel der Blöcke unterhalb der Blockfolgen gezeigt ist.

Bild 3-41: Blockmuster bei ECB

3.8.2.2 Blockchiffre mit Blockverkettung (CBC)

Blockchiffre mit Blockverkettung ist in Bild 3-42 gezeigt: Vor dem Verschlüsseln jedes (außer des ersten) Blocks wird zu seinem Klartext der Schlüsseltext des vorherigen modular addiert und entsprechend nach dem Entschlüsseln jedes Blocks der Schlüsseltext des vorherigen von seinem „Klartext“ modular subtrahiert.

Kommandos an alle Schiffe wie: "Schwenk nach Steuerbord", "volle Kraft voraus", "Maschinen stoppen", etc. Werden diese Kommandos jeweils unabhängig von der Vorgeschichte, d.h. ohne Seriennummer, Zeitstempel etc. deterministisch verschlüsselt, entspricht jedem dieser Kommandos genau ein Schlüsseltext. Nach kurzer Zeit kann ein Angreifer die Bedeutung der Schlüsseltexte für die reale Welt lernen – ganz egal wie supersicher das Konzelationssystem ist und ob es symmetrisch oder asymmetrisch arbeitet. Um dieses nichtkryptographische Brechen durch Lernen zu verhindern, müssen semantische Einheiten zumindest kontextabhängig, besser aber noch indeterministisch verschlüsselt werden.

83 Nehmen wir an, Telefax wird mit einer 64-Bit-Blockchiffre mit ECB verschlüsselt. Wir nehmen an, daß die häufigste Kombination von Pixeln „alle weiß“ bedeutet und ordnen also dem häufigsten Schlüsseltextblock 64 weiße Pixel und allen anderen Schlüsseltextblöcken 64 schwarze Pixel zu. Nehmen wir an, unser Faxgerät hat eine Auflösung von 300 dpi, also rund 12 Pixel pro mm. Dann vergröbert Verschlüsselung mit 64-Bit-ECB die horizontale Auflösung des Faxes lediglich von 0,08 auf 5,3 mm – zumindest große Schriften dürften problemlos lesbar bleiben. Werden die Pixel nicht horizontal, sondern etwa quadratweise mit 8•8 Pixel codiert, dann ergibt sich für das Beispiel eine Vergröberung der Auflösung des Faxes horizontal wie vertikal von 0,08 auf 0,67 mm. Da dürften auch kleinere Schriften noch zu entziffern sein.

Alle Linien führen der Blocklänge entsprechend viele Alphabetzeichen
⊕ Addition, z.B. bitweise modulo 2
⊖ Subtraktion, z.B. bitweise modulo 2

CBC

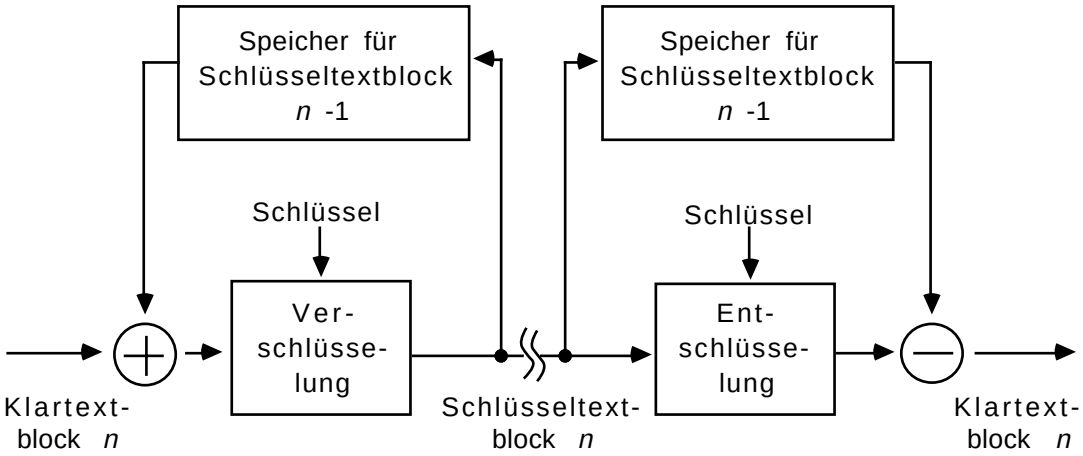


Bild 3-42: Konstruktion einer symmetrischen bzw. asymmetrischen selbstsynchronisierenden Stromchiffre aus einer symmetrischen bzw. asymmetrischen Blockchiffre: Blockchiffre mit Blockverkettung

Diese Konstruktion kann sowohl zum Zwecke der *Konzelation* als auch zum Zwecke der *Authentikation* verwendet werden. Sie hat folgende Vor- und Nachteile bzw. ambivalente Eigenschaften bei der Konzelation:

+ Die Verwendung einer *indeterministischen Blockchiffre* ist möglich. Da bei einer indeterministischen Blockchiffre Schlüsseltextblöcke länger als Klartextblöcke sind, muß dann vor der Addition bzw. Subtraktion eine geeignete Auswahl getroffen werden.
+ Wird eine *asymmetrische* Blockchiffre verwendet, so ist die entstehende Stromchiffre ebenfalls *asymmetrisch.*
– Die Länge der verschlüsselbaren Einheiten ist durch die *Blocklänge* der verwendeten Blockchiffre bestimmt und kann deshalb nicht einfach auf die Einheiten des Übertragungs- oder Speichersystems abgestimmt werden. Deshalb müssen die Blockgrenzen für die Selbstsynchronisation ggf. gesondert kenntlich gemacht werden.
* Bei einer noch so kleinen Verfälschung einer einem Block entsprechenden Einheit des Schlüsseltextstromes (oder auch einem transienten Fehler im Ver- oder Entschlüsseler) sind alle Zeichen des Klartextes dieser Einheit jeweils mit der Wahrscheinlichkeit

$$\frac{\text{Alphabetgröße - 1}}{\text{Alphabetgröße}}$$

gestört. Andernfalls taugt die verwendete Blockchiffre nichts. Zusätzlich ist die der Verfälschung entsprechende Stelle im nächsten Klartextblock gestört, da die Verfälschung im Schlüsseltextstrom gespeichert und in der folgenden Runde nochmals, allerdings anders, verwendet wird. Danach kann sich die Verfälschung nicht weiter auswirken, die folgenden Klartextblöcke werden richtig entschlüsselt, der Entschlüsseler hat sich aufgrund seiner beschränkten Speichertiefe automatisch auf den Verschlüsseler synchronisiert.

Dies ist auch dann der Fall, wenn es sich nicht um einen Übermittlungsfehler außerhalb des Verschlüsselungsprozesses handelt, sondern um einen transienten Fehler bei der Verschlüsselung. Dies ist deshalb bemerkenswert, weil dann ab der Fehlerstelle ein ganz anderer Schlüsseltext erzeugt wird: Die Fehlerauswirkungen beim Verschlüsseler werden von Schlüs-

seltextblock zu Schlüsseltextblock gespeichert und führen durch die Addition auf den nächsten Klartextblock wieder zu einer verfälschten Eingabe an die Blockchiffre und damit zu einem völlig unterschiedlichen Schlüsseltextblock. Trotzdem erhält der Entschlüsseler bereits einen Block nach Ende des transienten Fehlers wieder den richtigen Klartext. Entsprechendes gilt bei transienten Fehlern im Entschlüsseler. Sowohl bei transienten Fehlern im Ver- wie im Entschlüsseler ist in obigen Beispielen natürlich jeweils unterstellt, daß der Schlüssel der Blockchiffre durch den transienten Fehler nicht verfälscht wird. In diesem Fall wäre der beim Entschlüsseler ab diesem Zeitpunkt erhaltene Klartext jeweils pseudozufällig und damit unbrauchbar.

Da es für das Ergebnis der Addition in Bild 3-42 egal ist, ob der Klar- oder Schlüsseltextblock verfälscht ist, gilt Gleiches (wie gerade beschrieben) für den Fall einer noch so kleinen Verfälschung des Klartextstromes. Dies motiviert die Verwendung des linken Teiles der **Konstruktion zur Authentikation** (Bild 3-43):

+ Der einen Klartext Authentizierende verschlüsselt ihn mit CBC und hängt lediglich den letzten Schlüsseltextblock an den Klartext, d.h. der letzte Schlüsseltextblock ist der MAC gemäß Bild 3-6. Der die Authentizität Prüfende verschlüsselt den Klartext ebenfalls mit CBC und vergleicht den von ihm berechneten letzten Schlüsseltextblock mit dem übertragenen. Bei Übereinstimmung ist der Klartext authentisch. Hält man einen kürzeren Authentikator als einen ganzen Schlüsseltextblock für hinreichend, kann man Anhängen und Vergleich auf einen Teil des letzten Schlüsseltextblocks beschränken.
* Die verwendete Blockchiffre muß *deterministisch* sein.
– Die Länge der authentizierbaren Einheiten ist durch die *Blocklänge* der verwendeten Blockchiffre bestimmt.

Dies Verfahren zur Authentikation ist in [ISO8731-1_87] genormt. Dort ist auch beschrieben, daß nur die linksten 32 Bit als Authentikator genommen werden sollen und daß unvollständige letzte Klartextblöcke mit binären Nullen aufgefüllt werden sollen.

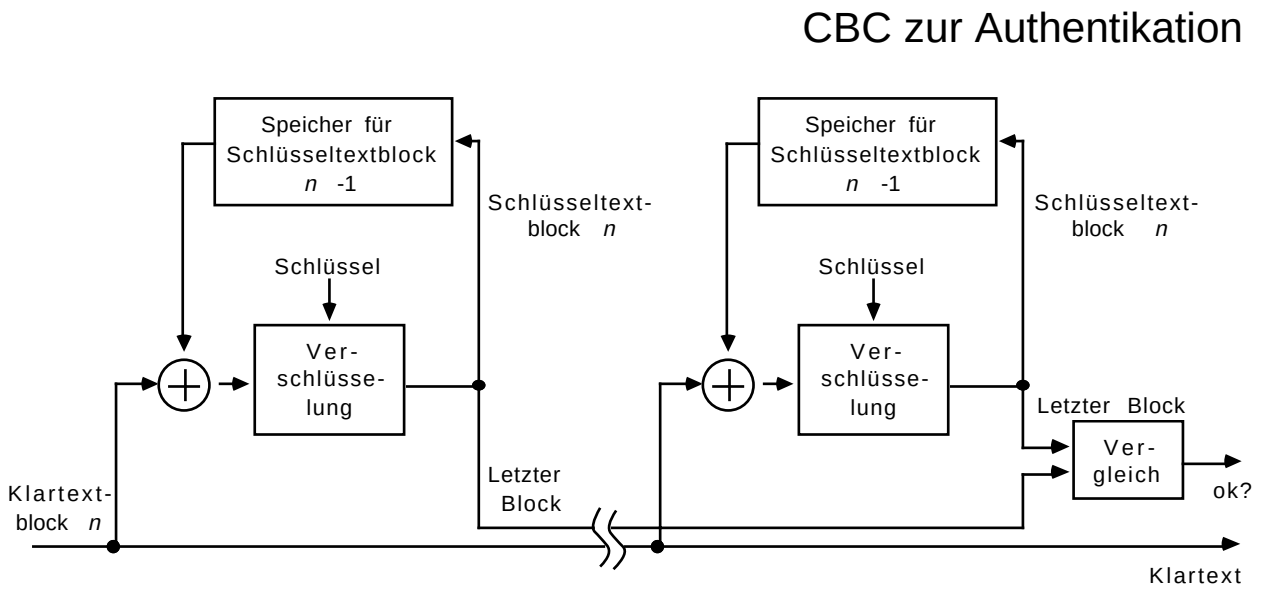


Bild 3-43: Blockchiffre mit Blockverkettung zur Authentikation: Konstruktion aus einer deterministischen Blockchiffre

Wie bisher beschrieben kann ein Angreifer bei der Konzelation noch erkennen, wenn zwei Klartexte mit den gleichen Blöcken anfangen: Auch die Schlüsseltexte fangen dann gleich an.

Das kann man vermeiden, indem man einen zufällig gewählten Klartextblock vor den eigentlichen Klartext hängt[84]: Wir sahen ja gerade, daß alle weiteren Schlüsseltextblöcke dann auch jedesmal verschieden sind. Man kann noch etwas Zeit sparen: Statt diesen Block als Klartext zu wählen und zu verschlüsseln, wodurch sich ein zufälliger Schlüsseltextblock ergibt, kann man gleich einen zufälligen Schlüsseltextblock wählen (also eine Vorbesetzung für den Speicher). Auch diesen muß man dem Entschlüsseler natürlich mitteilen – etwa als ersten Schlüsseltextblock, dessen zugehöriger Klartextblock dann ebenfalls eine Zufallszahl ist und deshalb vom Entschlüsseler weggeworfen wird.

Pathologisches Gegenbeispiel zu Konzelation mittels *Blockchiffre mit Blockverkettung*: Addition und Subtraktion erfolge modulo 2. Die Blockchiffre verschlüssele gerade Blöcke, d.h. letztes Bit 0, sicher auf ungerade Blöcke, d.h. letztes Bit 1, und ungerade Blöcke unsicher auf gerade Blöcke, vgl. Bild 3-44. Bei Beschränkung des Klartextraumes auf gerade Blöcke, d.h. Betrachtung der Blockchiffre als um ein Bit expandierende Blockchiffre (vgl. Bild 3-45), handelt es sich also um eine sichere Blockchiffre.

Die resultierende Stromchiffre ist aber auch für den beschränkten Klartextraum unsicher: Da für das letzte Bit des Blocks gilt $0 \oplus 1 = 1$, Verschlüsselung ergibt 0, $0 \oplus 0 = 0$, Verschlüsselung ergibt 1, usw., ist jeder zweite Eingabeblock in die Blockchiffre ungerade. Der Angreifer kann diese ungeraden Eingabeblöcke aus den geraden Schlüsseltextblöcken errechnen und durch Subtraktion der vorher beobachteten Schlüsseltextblöcke die zugehörigen Klartextblöcke errechnen. Die Stromchiffre ist also bezüglich jedes zweiten Klartextblocks und damit insgesamt unsicher.

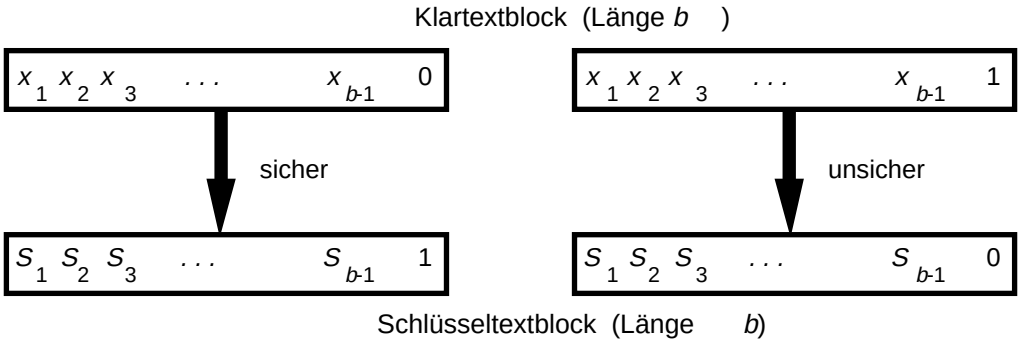


Bild 3-44: Pathologische Blockchiffre (nur sicher auf Klartextblöcken, die rechts eine 0 enthalten)

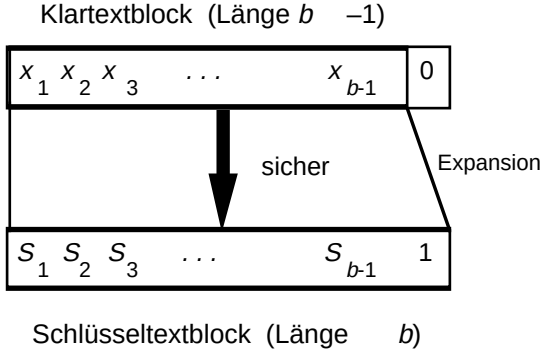


Bild 3-45: Um ein Bit expandierende Blockchiffre

84 Dadurch wird die Verschlüsselung indeterministisch. So etwas hatten wir bei asymmetrischen Konzelationssystemen schon in §3.1.1.1 Anmerkung 2 und in §3.4.4.

3.8.2.3 Schlüsseltextrückführung (CFB)

Schlüsseltextrückführung ist in Bild 3-46 gezeigt: Es wird nicht der Klartext, sondern der Inhalt eines Schieberegisters mit der Blockchiffre verschlüsselt und ein Teil des Ergebnisses vom Verschlüsseler zum Klartext modular addiert (wodurch der Schlüsseltext entsteht) und vom Entschlüsseler vom Schlüsseltext modular subtrahiert wird (wodurch wiederum der Klartext entsteht). In die Schieberegister wird jeweils der Schlüsseltext geschoben, also rückgeführt, weshalb diese Konstruktion *Schlüsseltextrückführung* genannt wird. Werden die Schieberegister gleich initialisiert und wird der Schlüsseltext richtig übermittelt, enthalten die Schieberegister beim Ver- und Entschlüsseler jeweils dieselben Werte. Dann (und nur dann) ergibt sich beim Entschlüsseler wieder der ursprüngliche Klartext.

In Bild 3-46 ist der allgemeine Fall gezeigt, daß der Schlüsseltext nicht direkt in das Schieberegister übernommen, sondern einer Auswahl unterworfen oder gar um feste Werte ergänzt wird. Letzteres ist zwar in der Norm [DINISO8372_87] vorgesehen, erscheint mir aber bezüglich der kryptographischen Sicherheit nicht sinnvoll, da dadurch die Zahl möglicher Werte im Schieberegister verkleinert wird. Da die Funktion „Wähle aus oder ergänze“ öffentlich festgelegt sein dürfte, kann ein Angreifer bezüglich Konzelation gleiche Werte in den Schieberegistern erkennen und durch Differenzbildung der Schlüsseltexte die Differenz zweier Klartexte erhalten. Bezüglich Authentikation ist insbesondere die Auswahl problematisch[85], da dann einzelne Zeichen des Schlüsseltextstromes (und damit auch des Klartextstromes) geändert werden können, ohne daß dies in das Schieberegister und damit den weiteren Verschlüsselungsprozeß eingeht. Damit hat diese Änderung auch keinen Einfluß auf die Bildung des weiter unten beschriebenen Authentikators und wird deshalb bei Prüfung der Authentizität nicht erkannt.

Schlüsseltextrückführung hat gegenüber Blockchiffre mit Blockverkettung folgende Vor- und Nachteile bzw. ambivalente Eigenschaften bei der Konzelation:

+ Es können *kleinere Einheiten* als die durch die Blocklänge der verwendeten Blockchiffre bestimmten ver- und entschlüsselt werden. Wird die elementare Einheit des Übertragungs- oder Speichersystems als Verschlüsselungseinheit verwendet, so ist Selbstsynchronisation immer, insbesondere auch ohne Kenntlichmachen der „Blockgrenzen“, gegeben.
– Die verwendete Blockchiffre muß *deterministisch* sein.
– Unabhängig davon, ob eine symmetrische oder asymmetrische Blockchiffre verwendet wird, entsteht eine *symmetrische* Stromchiffre, da die bei einer asymmetrischen Blockchiffre von der Verschlüsselungsfunktion verschiedene Entschlüsselungsfunktion bei der Konstruktion überhaupt nicht verwendet wird.
* Bei einer noch so kleinen Verfälschung einer Einheit des Schlüsseltextstromes (oder auch einem transienten Fehler im Verschlüsseler) sind alle Zeichen des Klartextes dieser und der folgenden

$$\left\lceil \frac{\text{Blocklänge} - \delta}{\text{Länge der Rückkopplungseinheit}} \right\rceil$$

Rückkopplungseinheiten gestört (δ bezeichnet den Abstand des Fehlers vom rechten Rand der Rückkopplungseinheit; $\lceil x \rceil$ bezeichnet die kleinste ganze Zahl z mit $z \geq x$). Die erste Einheit ist genau an der Störungsstelle gestört, bei letzteren sind alle Zeichen jeweils mit der Wahrscheinlichkeit

$$\frac{\text{Alphabetgröße} - 1}{\text{Alphabetgröße}}$$

gestört.

---

85 weshalb in Bild 3-47 die ganze Funktion „Wähle aus oder ergänze“ weggelassen wurde – denn auch hier ist Ergänzung wegen der Verkleinerung der Zahl möglicher Werte im Schieberegister nicht sinnvoll.

b Blocklänge

a Länge der Ausgabeeinheit, a ≤ b

r Länge der Rückkopplungseinheit, r ≤ b

⊕ Addition bezüglich passend gewähltem Modulus

⊖ Subtraktion bezüglich passend gewähltem Modulus

CFB

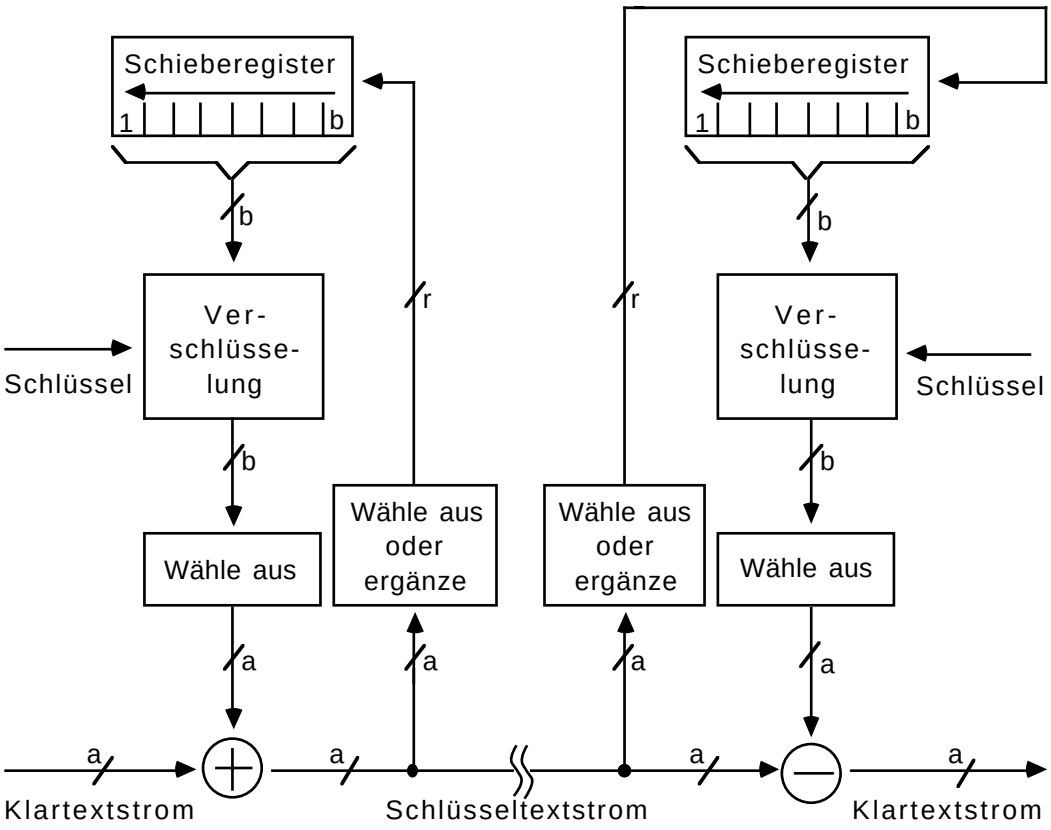


Bild 3-46: Konstruktion einer symmetrischen selbstsynchronisierenden Stromchiffre aus einer deterministischen Blockchiffre: Schlüsseltextrückführung

Aus den bei CBC erläuterten Gründen kann der linke Teil von CFB **zur Authentikation** verwendet werden, vgl. Bild 3-47:

+ Der einen Klartext Authentizierende verschlüsselt ihn mit CFB. Den Inhalt des Schieberegisters verschlüsselt er noch einmal zusätzlich mit der Blockchiffre[86] und hängt die Ausgabe der Blockchiffre an den Klartext. Der die Authentizität Prüfende verschlüsselt in gleicher Weise und vergleicht. Bei Übereinstimmung ist der Klartext authentisch. Hält man einen kürzeren Authentikator als die ganze Ausgabe der Blockchiffre für hinreichend, kann man Anhängen und Vergleich auf einen Teil beschränken.
* Die verwendete Blockchiffre muß *deterministisch* sein.
+ Es können *kleinere Einheiten* als die durch die Blocklänge der verwendeten Blockchiffre bestimmten authentiziert werden.

86 Würde der Inhalt des Scheiberegisters nicht noch einmal zusätzlich mit der Blockchiffre verschlüsselt, sondern direkt als Authentikator an die Nachricht gehängt, könnte ein Angreifer die letzte Ausgabeeinheit unerkannt ändern, indem er zum Authentikator die Differenz zwischen gefälschter Einheit und Originaleinheit hinzuaddiert.

b Blocklänge
a Länge der Ausgabeeinheit, a ≤ b
⊕ Addition bezüglich passend gewähltem Modulus

CFB zur Authentikation

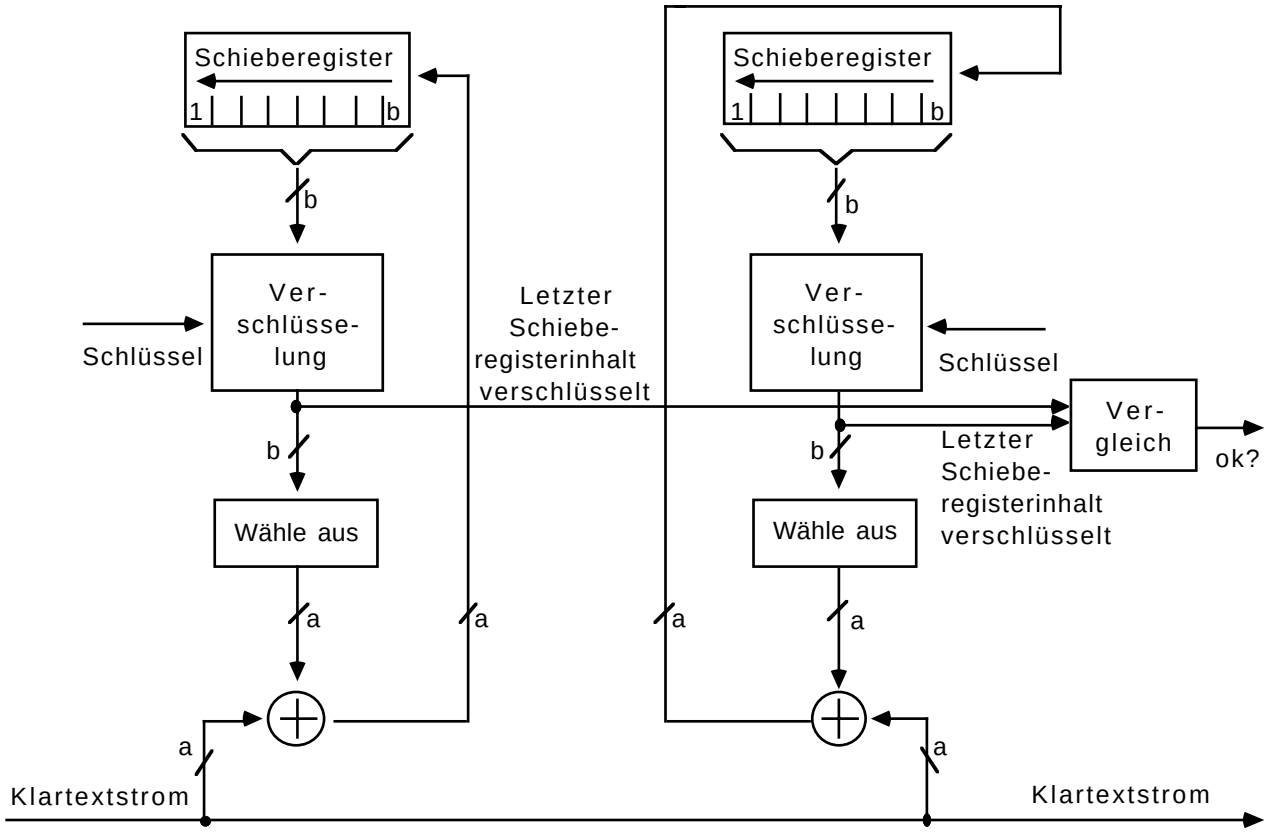


Bild 3-47: Schlüsseltextrückführung zur Authentikation: Konstruktion aus einer deterministischen Blockchiffre

3.8.2.4 Ergebnisrückführung (OFB)

Ergebnisrückführung ist in Bild 3-48 gezeigt: Im Gegensatz zur Schlüsseltextrückführung wird nicht der Schlüsseltext, sondern das Ergebnis (output) der Blockverschlüsselung in das Schieberegister rückgeführt. Entsprechend heißt diese Konstruktion *Ergebnisrückführung* (output feedback, abgekürzt OFB).[87]

Wie in Bild 3-46 ist auch in Bild 3-48 der allgemeine Fall gezeigt, daß das Ergebnis der Blockverschlüsselung erst einer Auswahl unterworfen oder, kombiniert damit, um feste Werte ergänzt wird. Letzteres ist zwar möglich, erscheint aber bezüglich der kryptographischen Sicherheit nicht sinnvoll, da dadurch die Zahl möglicher Werte im Schieberegister und damit die Periode des Pseudozufallszahlengenerators verkleinert wird. Ist die Funktion „Wähle aus oder ergänze“ wie in [DaPa_83] empfohlen und in [DINISO8372_87] vorgesehen die Identität, so kann statt einem Schieberegister ein normaler Speicher verwendet werden.

Diese Konstruktion hat folgende Vor- und Nachteile bzw. ambivalente Eigenschaften:

87 Eine andere Erklärung der Konstruktion OFB wäre: Im oberen Teil wird aus einer Blockchiffre ein *Pseudozufallsgenerator* konstruiert, dessen *Pseudozufallsfolge* im unteren Teil zum Klartextstrom addiert bzw. vom Schlüsseltextstrom subtrahiert wird. In §3.4.2 haben wir dies Pseudo-one-time-pad genannt.

+ Die Länge der ver- und entschlüsselbaren Einheiten ist nicht durch die Blocklänge der verwendeten Blockchiffre bestimmt und kann deshalb einfach auf die Einheiten des Übertragungs- oder Speichersystems abgestimmt werden.
– Die verwendete Blockchiffre muß *deterministisch* sein.
– Unabhängig davon, ob eine symmetrische oder asymmetrische Blockchiffre verwendet wird, entsteht eine *symmetrische* Stromchiffre, da die bei einer asymmetrischen Blockchiffre von der Verschlüsselungsfunktion verschiedene Entschlüsselungsfunktion bei der Konstruktion überhaupt nicht verwendet wird.
* Bei Verfälschung von Zeichen des Schlüsseltextstromes ist immer nur das entsprechende Zeichen des Klartextes gestört, es findet also *keine Fehlererweiterung* (error extension [DaPr_89]) statt. Je nach Anwendung kann dies günstig, z.B. bezüglich Konzelation, oder ungünstig, z.B. bezüglich Integrität, sein. Um einem falschen Eindruck vorzubeugen, sei an dieser Stelle noch an eine generelle Eigenschaft von synchronen Stromchiffren (und damit auch von Ergebnisrückführung) erinnert: Nur bei Verfälschung von Zeichen des Schlüsseltextstromes findet keine Fehlererweiterung statt. Bei verlorenen oder hinzugefügten Schlüsseltextstromzeichen (oder auch nur einem transienten Fehler in der Rückkopplungsschleife des Ver- oder Entschlüsselers) sind bis zur Wiederherstellung der Synchronisation alle folgenden Klartextzeichen jeweils mit der Wahrscheinlichkeit

$$\frac{\text{Alphabetgröße} - 1}{\text{Alphabetgröße}}$$

gestört.

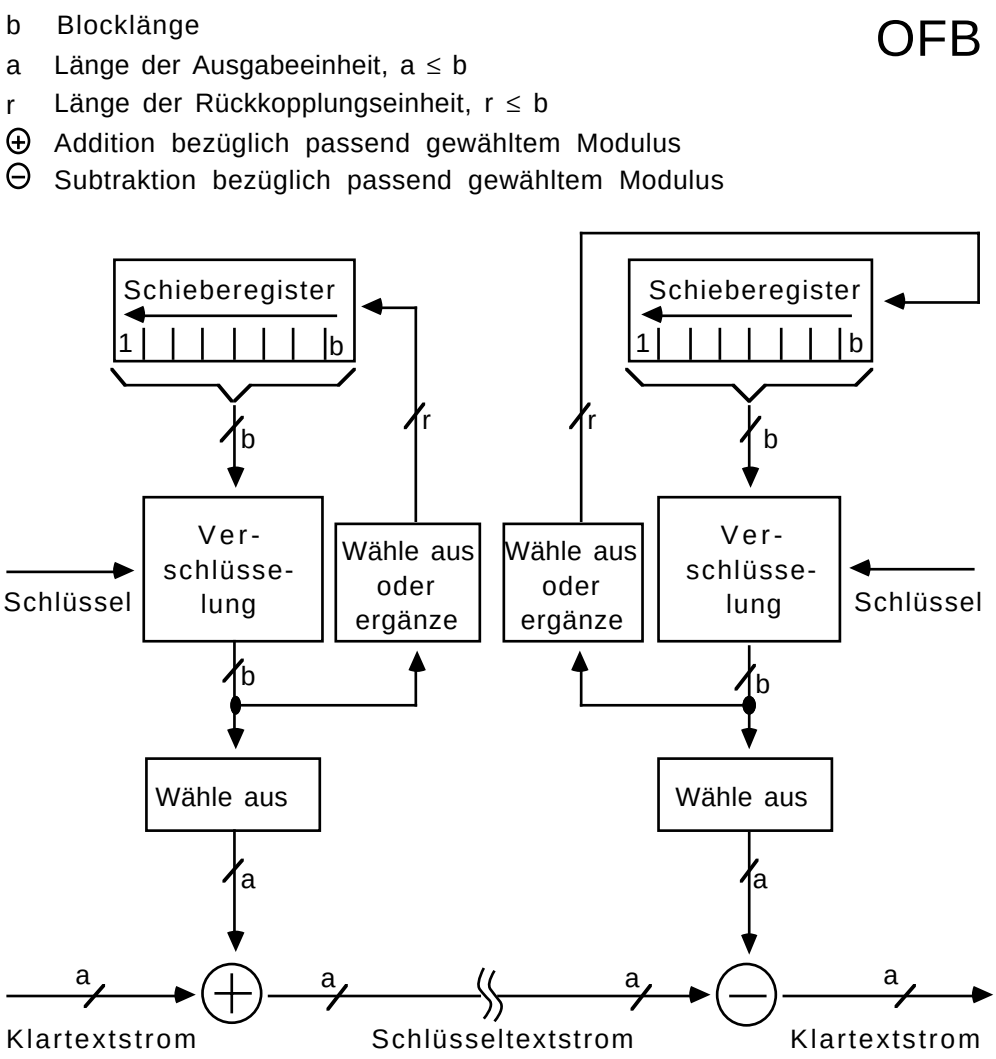


Bild 3-48: Konstruktion einer symmetrischen synchronen Stromchiffre aus einer deterministischen Blockchiffre: Ergebnisrückführung

Pathologisches Gegenbeispiel zu *Ergebnisrückführung*: Es finde keine Auswahl oder Ergänzung statt. Die Blockchiffre (Verschlüsselung in Bild 3-48) bilde Klartext*blöcke* in Schlüsseltext*blöcke* gleicher Länge *b* ab und werde wie folgt definiert: Wähle einen Klartextblock *K*, dem noch kein Schlüsseltextblock zugeordnet ist, aus und ordne ihm als Verschlüsselung zufällig einen Schlüsseltextblock *S* zu, der bisher noch keinem Klartextblock zugeordnet wurde. Ordne danach dem Klartextblock *S* als Verschlüsselung den Schlüsseltextblock *K* zu. Wiederhole beide Schritte, bis allen Klartextblöcken Schlüsseltextblöcke und allen Schlüsseltextblöcken Klartextblöcke zugeordnet sind. Diese Konstruktion erzeugt eine sichere Blockchiffre, da die Zuordnung Klartextblöcke – Schlüsseltextblöcke „zufällig“ ist.

Man beachte, daß sich die gerade erwähnten Klartext*blöcke* in Bild 3-48 im Schieberegister befinden und mit der geheimzuhaltenden Nachricht, in Bild 3-48 Klartext*strom* genannt, nichts zu tun haben.

Die aus Ergebnisrückführung resultierende Stromchiffre ist aber trotz der sicheren Blockchiffre unsicher: Sie wurde so konstruiert, daß von ihr abwechselnd immer dieselben zwei *Blöcke* verschlüsselt werden, so daß zum Klartext*strom* abwechselnd immer dasselbe addiert wird.

3.8.2.5 Blockchiffre mit Blockverkettung über Schlüssel- und Klartext (PCBC)

Erfordert eine Anwendung, daß anders als bei Ergebnisrückführung auch ab einem verfälschten Schlüsseltextstromzeichen alle folgenden Klartextzeichen mit der oben angegebenen Wahrscheinlichkeit gestört sind, so kann man bei der Blockchiffre mit Blockverkettung (Bild 3-42) zusätzlich zum Schlüsseltext des vorherigen Blocks auch über dessen Klartext verketten (Bild 3-49) und erhält so eine **Blockchiffre mit Blockverkettung über Schlüssel- und Klartext** (plain cipher block chaining, abgekürzt PCBC). Es ist bemerkenswert, daß zur Entschlüsselung eine Invertierung der Funktion *h*, die den vorherigen Schlüssel- und Klartextblock verknüpft, nicht nötig ist. Damit diese Konstruktion aus einer sicheren symmetrischen oder asymmetrischen Blockchiffre eine sichere symmetrische oder asymmetrische *synchrone* Stromchiffre gewinnt, muß die Funktion *h* in Abhängigkeit vom für die Addition bzw. Subtraktion gewählten Modulus geeignet gewählt werden, vgl. [MeMa_82 Seite 69-71]. Insbesondere ist es wichtig, daß *h* nicht einfach gleich der verwendeten Additions- bzw. Subtraktionsoperation gewählt wird[88]. Die in Bild 3-49 als Beispiel angegebene Wahl wird in [MeMa_82] empfohlen.

Diese Konstruktion hat folgende Vor- und Nachteile bzw. ambivalente Eigenschaften:

+ Die Verwendung einer *indeterministischen Blockchiffre* ist möglich. Da bei einer indeterministischen Blockchiffre Schlüsseltextblöcke länger als Klartextblöcke sind, muß dann vor der Addition bzw. Subtraktion eine geeignete Auswahl getroffen werden.
+ Wird eine *asymmetrische* Blockchiffre verwendet, so ist die entstehende Stromchiffre ebenfalls *asymmetrisch.*
– Die Länge der verschlüsselbaren Einheiten ist durch die *Blocklänge* der verwendeten Blockchiffre bestimmt und kann deshalb nicht einfach auf die Einheiten des Übertragungs- oder Speichersystems abgestimmt werden.
* Bei geeigneter Wahl der Funktion *h*, z.B. Addition mod $2^{\text{Blocklänge}}$, sind ab einer noch so kleinen Verfälschung einer einem Block entsprechenden Einheit des Schlüsseltextstromes ab diesem Block einschließlich alle Zeichen jeweils mit der Wahrscheinlichkeit

$$\frac{\text{Alphabetgröße - 1}}{\text{Alphabetgröße}}$$

gestört. Fehler pflanzen sich also *unbegrenzt* fort.

Die unbegrenzte Fehlererweiterung kann dazu verwendet werden, in einem Verschlüsselungsdurchgang *sowohl Konzelation als auch Authentikation* zu erreichen, indem an den Klartext ein Klartextblock festen Inhalts, z.B. alle Bits 0, gehängt wird. Beim Entschlüsseln wird dann geprüft, ob als letzter Klartextblock einer dieses Inhalts entsteht.

+ Da im allgemeinen der Aufwand für die Blockchiffre groß gegenüber Addition, Subtraktion und *h* ist, spart PCBC gegenüber separater Konzelation und Authentikation (etwa mit CBC),

---

88 Der Angriff bezieht sich auf Authentikation durch PCBC, die im Haupttext wenige Absätze weiter noch genauer erklärt wird.
Es sei *h* gleich ⊕ gewählt. Der Initialisierungswert *IK* für den Speicher für Klartextblock *n*-1 sowie der erwartete Wert *RK* des Redundanzblocks am Ende des Klartextes seien nicht fest vereinbart, sondern es sei lediglich abgesprochen, daß sie gleich sein sollen. Dann kann ein Angreifer den Empfänger einer Klartextnachricht N dazu bekommen, daß er eine andere Klartextnachricht als authentisch akzeptiert: Sei N aus den Blöcken N1, N2, N3, ... zusammengesetzt. Der Angreifer möchte alle Klartextblöcke um den Wert *W* ändern, den er beliebig wählen kann. Der Angreifer fängt *IK* und den Schlüsseltext ab, ersetzt *IK* durch *IK*⊕*W* und schickt beides weiter an den Empfänger. Der erhält beim Entschlüsseln die Blöcke N1 ⊖ *W*, N2⊕*W*, N3 ⊖ *W*, usw. Bei geradzahliger Blockanzahl (inkl. *RK*) hat *RK* den „richtigen“ Wert *IK*⊕*W*. Wird als Operation die bitweise Addition mod 2 verwendet, klappt dieser Angriff auch bei ungeradzahliger Blockanzahl. (Angriff leicht verallgemeinert nach [MeMa_82 Seite 416].)

wo jeder Block in jedem der zwei Durchgänge jeweils mit der Blockchiffre transformiert werden muß, etwa die Hälfte des Aufwands ein.

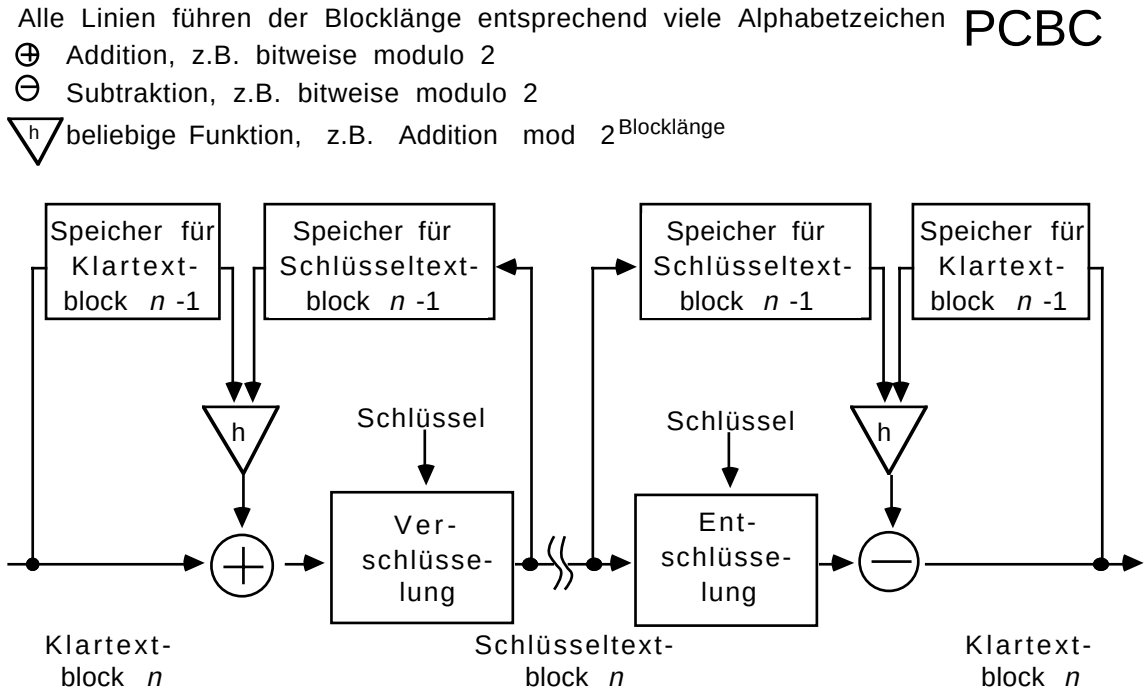


Bild 3-49: Konstruktion einer symmetrischen bzw. asymmetrischen synchronen Stromchiffre aus einer symmetrischen bzw. asymmetrischen Blockchiffre: Blockchiffre mit Blockverkettung über Schlüssel- und Klartext

Vergleicht man „Blockchiffre mit Blockverkettung“ mit „Blockchiffre mit Blockverkettung über Schlüssel- und Klartext“ so fällt zweierlei auf:

- Die Konstruktionen sind sehr ähnlich – beide verwenden Ver- *und* Entschlüsselung der Blockchiffre und operieren invertierbar auf dem „Klartext“. Genauer gesagt umfaßt die letztere Konstruktion die erstere (man wähle die Funktion *h* so, daß sie den eingegebenen Schlüsseltextblock ausgibt und den Klartextblock ignoriert).
- Aus dieser Ähnlichkeit resultieren gleiche Vor- und Nachteile. Die ambivalente Eigenschaft der Fehlererweiterung unterscheidet sich darin, ob sie begrenzt (selbstsynchronisierende Stromchiffre) oder potentiell unbegrenzt ist (synchrone Stromchiffre). Die Fehlererweiterung kann dadurch potentiell unbegrenzt sein, daß die Funktion *h* den vorherigen Klartextblock verwenden kann, der beim Entschlüsseler wiederum von allen vorherigen Schlüsseltextblöcken abhängen kann.

Bild 3-49 ist so gezeichnet, daß PCBC und CBC möglichst leicht vergleichbar sind. Für eine Implementierung würde sinnvollerweise nicht sowohl Klartext- und Schlüsseltextblock *n*-1 gespeichert, sondern stattdessen das Ergebnis der auf diese Werte angewendeten Funktion *h*. Dies spart einerseits Speicherplatz und andererseits Übertragungsaufwand bei der Initialisierung der Speicherglieder.

3.8.2.6 Schlüsseltext- und Ergebnisrückführung (OCFB)

Entsprechendes wie für „Blockchiffre mit Blockverkettung“ und „Blockchiffre mit Blockverkettung über Schlüssel- und Klartext“ gilt für „Schlüsseltextrückführung“ und „Ergebnisrückführung“:

- Die Konstruktionen sind sehr ähnlich – beide verwenden nur die Verschlüsselung der Blockchiffre zur Erzeugung eines pseudozufälligen Zeichenstromes mit dem durch modulare Addition verschlüsselt und durch modulare Subtraktion entschlüsselt wird. In Bild 3-50 ist eine allgemeine Konstruktion **Schlüsseltext- und Ergebnisrückführung** angegeben (output cipher feedback, abgekürzt OCFB), die durch geeignete Wahl der Funktion *h* „Schlüsseltextrückführung“ und „Ergebnisrückführung“ umfaßt. Wie in Bild 3-49 ist für die Entschlüsselung eine Invertierung der Funktion *h* nicht nötig.
- Aus dieser Ähnlichkeit resultieren gleiche Vor- und Nachteile. Die ambivalente Eigenschaft der Fehlererweiterung unterscheidet sich darin, ob sie begrenzt (selbstsynchronisierende Stromchiffre) oder potentiell unbegrenzt ist (synchrone Stromchiffre). Die Fehlererweiterung kann dadurch potentiell unbegrenzt sein, daß die Funktion *h* das Ergebnis der Blockverschlüsselung verwenden kann, das wiederum vom gesamten vorherigen Schlüsseltextstrom abhängen kann.

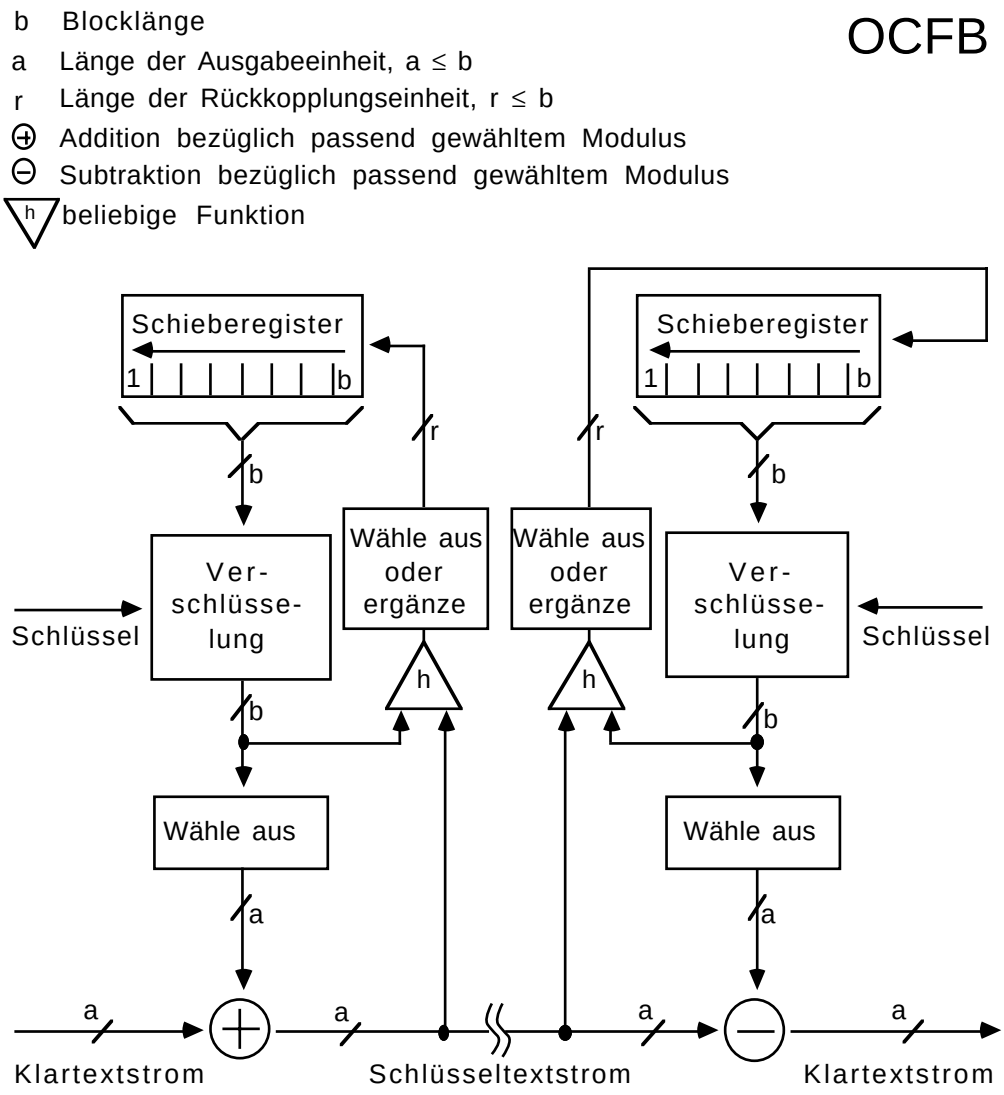


Bild 3-50: Konstruktion einer synchronen symmetrischen Stromchiffre aus einer deterministischen Blockchiffre: Schlüsseltext- und Ergebnisrückführung[89]

89 Die Konstruktion würde dadurch, daß man *h* als Eingabe auch noch den Klartext gibt, nicht allgemeiner, sondern nur unübersichtlicher: Die Funktion *h* kann aus ihren beiden Parametern jeweils den Klartext berechnen.

3.8.2.7 Anmerkungen zu expandierenden Blockchiffren und zur Initialisierung der Speicherglieder

In den Bildern 3-42, 3-46, 3-48, 3-49 und 3-50 sowie den zugehörigen Verfahrensbeschreibungen ist jeweils nur der übliche Fall dargestellt, daß die Blockchiffre Klartextblöcke in Schlüsseltextblöcke gleicher Länge abbildet. Ist dies nicht der Fall, d.h. sind die *Schlüsseltextblöcke länger*, so muß in Bild 3-42 jeweils – geschickterweise vor dem „Speicher für Schlüsseltextblock *n*-1“ – eine Auswahl getroffen werden. Entsprechendes gilt für Bild 3-49. In den Bildern 3-46, 3-48 und 3-50 ist jeweils nur die vorhandene Auswahl zu modifizieren.

Ebenfalls nicht eingegangen wird auf die Notwendigkeit geeigneter, d.h. insbesondere gleicher, *Initialisierung der Speicherglieder*. Die „Speicher für Schlüsseltextblock *n*-1“ in Bild 3-42 sollten geeignet initialisiert werden, die „Speicher für Schlüsseltextblock *n*-1“ und die „Speicher für Klartextblock *n*-1“ in Bild 3-49 müssen geeignet initialisiert werden. Die Schieberegister in Bild 3-46 sollten, die in den Bildern 3-48 und 3-50 müssen geeignet initialisiert werden [MeMa_82, DaPr_89]. Diese Aussagen beziehen sich jeweils auf die Verwendung zum Zweck der Konzelation.

Bei Verwendung zum Zweck der Authentikation müssen beim Generieren und Prüfen des Authentikators jeweils dieselben Initialisierungen verwendet werden, die aber für alle Teilnehmer und alle Nachrichten gleich sein können.

Genaueres zum Thema Initialisierung der Speicherglieder wird in Aufgabe 3-18 d) erarbeitet.

3.8.2.8 Zusammenfassung der Eigenschaften der Betriebsarten

Bild 3-51 gibt eine Zusammenfassung der Eigenschaften der beschriebenen Betriebsarten für Blockchiffren. Während die Beschreibung im Text sich vom Einfachen (und Standardisierten, nämlich ECB, CBC, CFB und OFB) zum Komplizierten, oder mit anderen Worten vom Speziellen zum Allgemeinen vorarbeitete, sind in Bild 3-51 die Betriebsarten nach ihrer Verwandtschaft in zwei Gruppen geordnet.

In der linken Gruppe werden Ver- und Entschlüsselungsfunktion der Blockchiffre benutzt, und die Blockchiffre liegt direkt im Transformationsweg vom Klartext über den Schlüsseltext zum Klartext.

In der rechten Gruppe wird die Verschlüsselungsfunktion der Blockchiffre zur Generierung einer Pseudozufallsfolge benutzt. Die Entschlüsselungsfunktion der Blockchiffre wird nicht verwendet. Im direkten Transformationsweg vom Klartext über den Schlüsseltext zum Klartext liegen dann nur die von der Vernam-Chiffre wohlbekannten Additions- und Subtraktionsoperatoren.

Diese Gruppenspezifika bedingen die ersten drei Eigenschaften. Die in den Gruppen jeweils rechts stehende Betriebsart umfaßt jeweils die beiden links von ihr stehenden. Deshalb ist die Fehlerausbreitung der jeweils rechts stehenden Betriebsart jeweils nur *potentiell* unbegrenzt und sie sind jeweils zur Authentikation im selben Durchgang *geeignet*, d.h. es muß jeweils eine passende Funktion *h* verwendet werden. Entsprechendes gilt für die Frage, ob PCBC bzw. OCFB synchrone Stromchiffren sind. Bei geeigneter Wahl von *h* sind sie es.

Zusätzlich zu diesen Eigenschaften sind für den praktischen Betrieb, d.h. Nutzung oder Implementierung, folgende 3 Eigenschaften der Betriebsarten interessant und nützlich:

Wahlfreier Zugriff: Kann wahlfrei auf einzelne Teile des Schlüsseltextes und/oder Klartextes zugegriffen werden, d.h. kann aus wahlfrei herausgegriffenen Teilen des Klartextes der entsprechende Teil des Schlüsseltextes errechnet werden und/oder aus wahlfrei herausgegriffenen Teilen des Schlüsseltextes der entsprechende Teil des Klartextes? Oder muß Schlüsseltext bzw. Klartext jeweils von Beginn der Nachricht an berechnet werden?

Parallelisierbarkeit: Kann die Ent- und/oder Verschlüsselung parallelisiert werden? Oder muß sie streng sequentiell durchgeführt werden?

Vorausberechnung der Blockchiffre: Kann die Blockchiffre vorausberechnet werden oder kann auch dieser, bei weitem rechenaufwendigste Teil erst ausgeführt werden, wenn der Klartext bzw. Schlüsseltext vorliegt?

	ECB	CBC	PCBC	CFB	OFB	OCFB
Verwendung indeterministischer Blockchiffren	+ möglich			– nicht möglich		
Bei asymmetrischer Blockchiffre entsteht	+ asymmetrische Stromchiffre			– symmetrische Stromchiffre		
Länge der verschlüsselbaren Einheiten	– durch Blocklänge der Blockchiffre bestimmt			+ beliebig		
Fehlererweiterung	nur innerhalb eines Blocks	2 Blöcke	potentiell unbegrenzt	$1 + \lceil b/r \rceil$ Blöcke, wenn Fehler ganz rechts, sonst evtl. einer weniger	keine bei Verfälschung	potentiell unbegrenzt
auch zur Authentikation geeignet?	bei Redundanz innerhalb jedes Blocks: ja	bei deterministischer Blockchiffre: ja	ja, sogar Konzelation im selben Durchgang	ja	bei geeigneter Redundanz: ja	ja, sogar Konzelation im selben Durchgang
synchron oder selbstsynchronisierend?	selbstsynchronisierend, solange Blockgrenzen erkannt	selbstsynchronisierend, solange Blockgrenzen erkannt	potentiell synchron	selbstsynchronisierend	synchron	potentiell synchron
wahlfreier Zugriff	ja	nur bei Entschlüsselung	potentiell nein	nur bei Entschlüsselung	nein	potentiell nein
Parallelisierbarkeit	ja	nur bei Entschlüsselung	potentiell nein	nur bei Entschlüsselung	nein	potentiell nein
Vorausberechnung der Blockchiffre	nein			jeweils für einen Block	ja	potentiell ja, zumindest jeweils für einen Block

Bild 3-51: Eigenschaften der Betriebsarten

Zwischen den bisher diskutierten Eigenschaften der Betriebsarten und diesen 3 weiteren bestehen folgende Zusammenhänge:

Aus *selbstsynchronisierend* folgt *wahlfreier Zugriff bei der Entschlüsselung*, denn zumindest nach dem für die Selbstsynchronisation nötigen Stück Schlüsseltext kann der folgende Teil entschlüsselt werden.

Aus wahlfreiem Zugriff folgt Parallelisierbarkeit, denn wenn Teile unabhängig vom Rest bearbeitet werden können, dann kann dies auch parallel geschehen.

Liegt die Blockchiffre direkt im Transformationsweg vom Klartext über den Schlüsseltext zum Klartext, ist keine Vorausberechnung der Blockchiffre möglich. Andernfalls kann zumindest jeweils ein Block vorausberechnet werden.

Insgesamt ergeben sich die Einträge der unteren 3 Zeilen von Bild 3-51.

3.8.3 Kollisionsresistente Hashfunktion aus Blockchiffre

In diesem Abschnitt wird beschrieben, wie aus deterministischen Blockchiffren, deren Schlüssel nicht (wesentlich) länger als die Blocklänge ist, kollisionsresistente Hashfunktionen konstruiert werden können.

Wie in §3.5.1 für kollisionsresistente Permutationenpaare bereits erklärt wurde, kann das Finden von Kollisionen allenfalls komplexitätstheoretisch schwer sein – informationstheoretisch ist es leicht, denn ein Angreifer könnte immer alle Möglichkeiten durchprobieren. Entsprechendes gilt für kollisionsresistente Hashfunktionen: Da sie beliebig lange Argumente auf Werte fester Länge abbilden, gibt es beliebig viele Kollisionen, d.h. verschiedene Argumente, die auf denselben Wert abgebildet werden. Ein Angreifer muß also nur genügend ausdauernd suchen (können), um Kollisionen zu finden.

Das Beste, was man sich wünschen kann, wäre eine *effiziente* Konstruktion, die aus *beliebigen* Blockchiffren kollisionsresistente Hashfunktionen gewinnt, die *bewiesenermaßen* so sicher wie die verwendete Blockchiffre sind. Leider ist eine solche Konstruktion nicht bekannt – unter den zahlreichen Vorschlägen gibt es nicht einmal solche, die zwei der drei kursiv hervorgehobenen Eigenschaften haben. Da in §3.6.4 bereits (unter der Faktorisierungsannahme) bewiesenermaßen sichere kollisionsresistente Hashfunktionen angegeben wurden, geht es hier vor allem um größere Effizienz. Deshalb wird eine gemäß Bild 3-12 „wohluntersuchte“ Konstruktion angegeben, wie aus deterministischen Blockchiffren, deren Schlüssel aus Sicherheitsgründen hier nicht (wesentlich) länger als die Blocklänge sein dürfen [LuRa_89], sehr effizient kollisionsresistente Hashfunktionen konstruiert werden können.

In [DaPr_84, DaPr_89 Seite 265] wird folgende Konstruktion einer kollisionsresistenten Hashfunktion aus einer deterministischen Blockchiffre (z.B. DES) vorgeschlagen:

Der zu hashende Klartext wird in Blöcke der Schlüssellänge der Blockchiffre unterteilt. In jeder Runde wird einer der Klartextblöcke als Schlüssel der Blockchiffre verwendet.

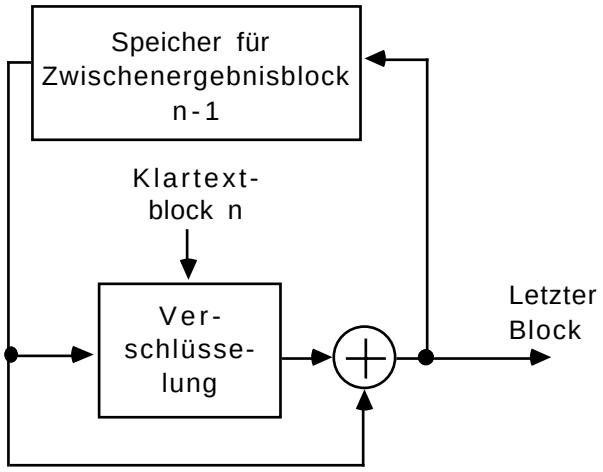
Die Eingabe der Blockchiffre ist für die erste Runde ein Initialisierungswert, für die zweite und alle folgenden Runden das Ergebnis der vorherigen Runde. (In Bild 3-52 wird der Initialisierungswert sinnvollerweise als Zwischenergebnisblock Nummer 0 aufgefaßt.)

In jeder Runde wird nicht nur verschlüsselt, sondern auch die Eingabe der Blockchiffre modulo 2 zur Ausgabe addiert. Dies verhindert, daß ein Angreifer rückwärts rechnen kann, indem er einen Schlüssel (sprich: Klartextblock) wählt, dann bezüglich des gewählten Schlüssels die Blockchiffre invertiert (sprich: entschlüsselt) und so die Eingabe dieser Runde erhält. Auf diese Art könnte er, wenn der Initialisierungswert frei wählbar ist, Kollisionen berechnen, indem er das Beschriebene mit zwei unterschiedlichen Schlüsseln ausführt.

Das Ergebnis der letzten Runde ist die Ausgabe der Hashfunktion. (Damit dieser Hashwert von anderen überprüft werden kann, muß ihnen auch der Initialisierungswert bekannt sein. Ist der Initialisierungwert nicht global festgelegt, so muß er ihnen ggf. mitgeteilt werden.)

Alle Linien führen der Blocklänge entsprechend viele Alphabetzeichen
Klartextblöcke haben die Länge des Schlüssels der Blockchiffre

⊕ Addition mod 2



Aus Sicherheitsgründen sollte gelten:
Schlüssellänge der Blockchiffre nicht wesentlich länger als ihre Blocklänge.

Bild 3-52: Konstruktion einer kollisionsresistenten Hashfunktion aus einer deterministischen Blockchiffre

Bild 3-52 ist in der gleichen Anordnung wie Bild 3-42 gezeichnet. Deshalb fällt auf, daß die Klartextblöcke an einer unüblichen Stelle in die Blockchiffre eingespeist werden – nämlich als Schlüssel.

Wichtig ist – ganz egal wie die Hashfunktion konstruiert wird –, daß der **Initialisierungswert festgelegt** oder die **Nachricht postfixfrei**[90] **codiert** ist. Sonst gibt es folgende trivialen Kollisionen: Bei der rundenweisen Berechnung des Hashwertes einer Nachricht entstehen mit den Zwischenergebnisblöcken jeweils passende Initialisierungswerte für die um die bisher abgearbeiteten Klartextblöcke verkürzte Nachricht.

Um diese mögliche Schwäche zu vermeiden und auch um beim Auffüllen der Nachricht auf ein Vielfaches der Schlüssellänge der Blockchiffre zu wissen, wie viele Bits aufgefüllt wurden, sollte der letzte Klartextblock die **Länge der Nachricht in Bit** enthalten [LaMa_92 Seite 57].

Außerdem wird für jede Hashfunktion, die Ergebnisse der Länge b liefert (also z.B. auch die angegebene Konstruktion bei Verwendung einer Blockchiffre der Blocklänge b), durch Durchprobieren des Nachrichtenraumes im Mittel nach etwa $2^{b/2}$ Versuchen eine Kollision gefunden. Dies liegt am sogenannten **Geburtstagsparadox**: Damit mit Wahrscheinlichkeit 1/2 mindestens zwei Menschen am gleichen Tag Geburtstag haben, werden nicht etwa 365/2, sondern nur etwa $\sqrt{365}$ Menschen benötigt [DaPr_89 Seite 279-281].

Für viele Anwendungen wird bei der angegebenen Konstruktion der Einsatz von DES als Blockchiffre also nicht ausreichen, da 2^{32} Versuche von vielen möglichen Angreifern durchaus durchführbar sind. Selbst eine Blocklänge von 128 Bit dürfte für die überschaubare Zukunft nicht genügen, da demnächst auch 2^{64} Versuche fähigen Angreifern möglich sein werden. Deshalb sollten kollisionsresistente Hashfunktionen Ergebnisse von wenigstens 160 Bit Länge liefern.

---

90 Kein Postfix, d.h. Endstück, einer postfixfrei abgebildeten Nachricht darf gleichzeitig postfixfreie Abbildung einer anderen Nachricht sein.

Eine genaue Klassifikation von kollisionsresistenten Hashfunktionen danach, welchen Angriffstypen (in Analogie zu §3.1.3.1 und §3.1.3.2) sie widerstehen, und wenige Sätze, wie die Kollisionsresistenz der Hashfunktion von der Kollisionsresistenz einer einzelnen Runde abhängt, sind in [LaMa_92] zu finden.

3.9 Skizzen weiterer Systeme

3.9.1 Diffie-Hellman Schlüsselaustausch

Bisher wurden nur asymmetrische Kryptosysteme vorgestellt, deren Sicherheit auf der Faktorisierungsannahme beruht, vgl. §3.4.1. Das folgende Verfahren zur Berechnung eines gemeinsamen geheimen Schlüssels allein auf der Basis öffentlicher Information (Diffie-Hellman Schlüsselaustausch, Diffie-Hellman-Key-Exchange [DiHe_76]) beruht auf der Diskreter-Logarithmus-Annahme, ist aber in den letzten Jahren praktisch wie theoretisch wichtig geworden:

Da im Gegensatz zu RSA das zugehörige Patent abgelaufen ist, verwendet Pretty Good Privacy – PGP ab Version 5 statt RSA eine Variante des Diffie-Hellmann Schlüsselaustauschs (**ElGamal-Konzelationssystem** mit individueller Parameterwahl, vgl. Aufgabe 3-23 c)) in einem hybriden kryptographischen System, vgl. §3.1.4.

Mittels Diffie-Hellmann Schlüsselaustausch ist Steganographie mit öffentlichen Schlüsseln möglich, vgl. §4.1.2.

Zuerst einige Grundlagen, insbesondere was sind und wie erhalten wir Generatoren von $\mathbb{Z}_p^*$:

Ein Element g einer multiplikativ geschriebenen Gruppe G heißt **Generator** von G, wenn die Potenzen von g die Gruppe G erzeugen. Dies bedeutet, daß es für jedes Element h von G eine ganze Zahl i mit $0 \le i < |G|$ gibt, so daß $g^i = h$. Eine Gruppe, in der es mindestens einen Generator gibt, heißt *zyklisch*.

Für jede Primzahl p ist $\mathbb{Z}_p^*$ eine zyklische Gruppe. Es ist leicht, $\mathbb{Z}_p^*$ und einen Generator g zufällig zu wählen [MeOV_97 Seite 163]:

1. Wähle eine Primzahl p zufällig (vgl. §3.4.1).
2. Faktorisiere $|\mathbb{Z}_p^*| = p\text{-}1$. Es sei $p\text{-}1 = p_1^{e1} \cdot p_2^{e2} \cdot \ldots \cdot p_k^{ek}$, wobei $p_1 \ldots p_k$ paarweise verschiedene Primzahlen sind. Wenn die Faktorisierung nicht in akzeptabler Zeit gelingt, gehe zu 1.
3. Wähle ein zufälliges Element g in $\mathbb{Z}_p^*$.
4. Für i von 1 bis k
   Berechne $b := g^{(p-1)/p_i}$
   Wenn $b = 1$, gehe zu 3.

Die Schritte 1. und 2. wählen eine Gruppe $\mathbb{Z}_p^*$, bei der die Primfaktorzerlegung ihrer Elementanzahl bekannt ist. Einerseits, um Rechenzeit beim Faktorisieren im 2. Schritt zu sparen, und andererseits, um sicherzustellen, daß p-1 mindestens einen großen Primfaktor hat, was für die Sicherheit gegen das Ziehen diskreter Logarithmen (s.u.) wünschenswert ist, kann statt der Schritte 1. und 2. folgendermaßen vorgegangen werden:

Generiere eine Primzahl p_1 der Länge *l-const*, wobei *l* der Sicherheitsparameter (vgl. §3.1.3.1 und §3.1.3.4) und *const* eine kleine Konstante, beispielsweise 10, ist.

Suche unter den Zahlen $p := f \cdot p_1 + 1$ mit $|f| = const$ nach Primzahlen.

Faktorisiere f. (Dies ist einfach, da f nicht allzu groß ist.) Sei $f = p_2^{e2} \cdot \ldots \cdot p_k^{ek}$.

Dies Vorgehen ändert die Verteilung der Wahl der Primzahl p. Das ist aber, nach allem was darüber bekannt ist, nicht schlimm.

Die Schritte 3. und 4. wählen innerhalb einer Gruppe $\mathbb{Z}_p^*$, bei der die Primfaktorzerlegung ihrer Elementanzahl bekannt ist, einen Generator:

Sei h ein beliebiges Element der multiplikativ geschriebenen Gruppe G.

Die Potenzen von h bilden eine Untergruppe von G, denn $h^x \bullet h^y = h^{x+y}$ und $(h^x)^{-1} = h^{-x}$.

Nach dem Satz von Lagrange gilt: Wenn H Untergruppe von G, dann ist $|H|$ Teiler von $|G|$.[91]

Ist H eine echte Untergruppe von G, d.h. $|H| < |G|$, und gilt $|G| = p_1^{e1} \bullet p_2^{e2} \bullet ... \bullet p_k^{ek}$, dann gilt für mindestens ein i, $1 \le i \le k$: $|H|$ teilt $|G|/p_i$.

Für dieses i gilt: Nach §3.4.1.2 ist $h^{|H|} = 1$, insbesondere also auch $h^{|G|/p_i} = 1$.

Ist h ein Generator von G, dann muß $h^{|G|/p_i}$ immer einen Wert $\neq 1$ haben, denn sonst würde h nicht eine Gruppe mit $|G|$ Elementen, sondern höchstens eine Gruppe mit $h^{|G|/p_i}$ Elementen erzeugen – denn ab dem Wert 1 wiederholen sich die erzeugten Elemente nur noch, wenn der Exponent weiter erhöht wird.

Also erkennt der 4. Schritt in obigem Algorithmus, ob g ein Generator von $\mathbb{Z}_p^*$ ist oder nicht.

Der Diffie-Hellman Schlüsselaustausch beruht auf der Schwierigkeit, **diskrete Logarithmen**, d.h. Logarithmen modulo einer Primzahl zu ziehen. Sei p eine Primzahl, g ein Generator von $\mathbb{Z}_p^*$, der multiplikativen Gruppe mod p. Sei

$$g^x = h \mod p.$$

Dann heißt x diskreter Logarithmus von h zur Basis g modulo p:

$$x = \log_g(h) \mod p.$$

Zur Diskreter-Logarithmus-Annahme: Entsprechend §3.1.3.4 kann man zur Zeit nicht beweisen, daß diskrete Logarithmen ziehen schwer ist, sondern man muß eine Annahme treffen. Diese besagt, daß für jeden schnellen (Diskrete-Logarithmen-)Algorithmus $\mathcal{L}$ die Wahrscheinlichkeit, daß $\mathcal{L}$ einen diskreten Logarithmus tatsächlich ziehen kann, schnell immer kleiner wird, je größer die Länge l des Modulus wird.

Annahme: Für jeden (probabilistischen) polynomialen Algorithmus $\mathcal{L}$ und jedes Polynom Q gilt: Es existiert ein L, so daß für alle $l \ge L$ gilt: Wenn p als zufällige Primzahl der Länge l gewählt wird und danach g zufällig innerhalb der Menge der Generatoren von $\mathbb{Z}_p^*$ und x zufällig in $\mathbb{Z}_p^*$ gewählt werden und $g^x = h \mod p$

$$W(\mathcal{L}(p,g,h) = x) \le \frac{1}{Q(l)}.$$

Diffie-Hellman Schlüsselaustausch: Fest vereinbart und allgemein bekannt sind p und g. Teilnehmer wählen als privaten Schlüssel jeweils zufällig ein Element von $\mathbb{Z}_p^*$, berechnen g hoch dieses Element mod p und veröffentlichen das Ergebnis[92]. Jeder Teilnehmer kann mit einem anderen einen Schlüssel „austauschen“, indem er dessen öffentlichen Schlüssel mit seinem eigenen privaten Schlüssel mod p exponenziert. Da die Exponentiation, und damit auch die modulare Exponentiation, kommutativ ist, erhalten beide Teilnehmer den gleichen geheimen Schlüssel, vgl. Bild 3-53.

Die Hoffnung ist, daß außer diesen beiden Teilnehmern niemand deren geheimen Schlüssel $k = g^{xy} \mod p$ berechnen kann. Dies ist die **Diffie-Hellman-Annahme**. Sie ist eine stärkere Annahme als die Diskreter-Logarithmus-Annahme: Denn kann jemand diskrete Logarithmen ziehen, so gewinnt er entweder x aus $g^x \mod p$ oder y aus $g^y \mod p$. Damit kann er dann aus $g^y \mod p$ bzw. $g^x \mod p$ bequem $g^{xy} \mod p$ berechnen. Umgekehrt konnte bisher nicht gezeigt werden, daß aus $g^x \mod p$, $g^y \mod p$ und $g^{xy} \mod p$ entweder x oder y bestimmt werden kann.

---

91 Dies wurde schon in §3.4.1.2 zum Beweis des kleinen Fermatschen Satzes verwendet.

92 Die Veröffentlichung muß authentisch erfolgen, vgl. Aufgabe 3-4 c). Gesichert werden kann dies beispielsweise durch die (mehrfache) Zertifizierung der öffentlichen Schlüssel, vgl. §3.1.1.2.

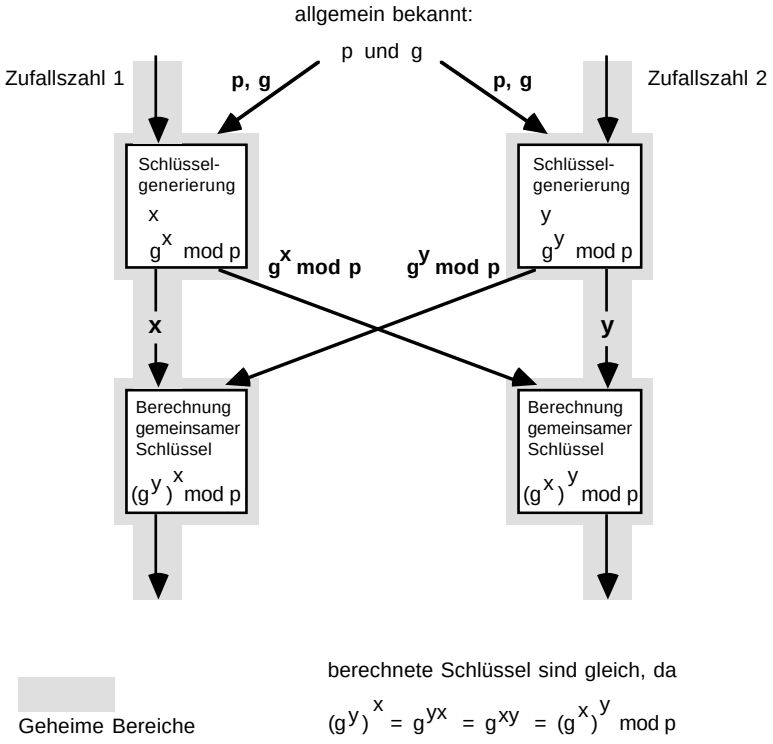


Bild 3-53: Diffie-Hellman Schlüsselaustausch

Das Bemerkenswerte ist, daß hier kein Schlüssel über das Netz übertragen werden muß, sondern der gemeinsame jeweils lokal berechnet werden kann. Dies ist wichtig für die Anwendung von Steganographie und führt zu Steganographie mit öffentlichen Schlüsseln, vgl. §4.1.2.

Bei Diffie-Hellman Schlüsselaustausch stehen im öffentlichen Schlüsselregister statt öffentlicher Chiffrierschlüssel die öffentlichen Schlüssel der Teilnehmer. Ansonsten funktioniert alles wie in Bild 3-5 gezeigt und in Aufgabe 3-4 geübt.

3.9.2 Für den Unterzeichner unbedingt sichere Signaturen

Die Sicherheit eines digitalen Signatursystems ist auch „asymmetrisch“: bei einem üblichen Signatursystem (Bild 3-54) ist sie:

- unbedingt sicher für den Empfänger der Signatur. Hat der Empfänger ein Paar (Text, Signatur), das den Test besteht, so kann er jeden überzeugen, daß der Inhaber des Schlüsselpaares (s, t) den Text signiert hat.
- nur kryptographisch für den Unterzeichner. Denn es kann einem Angreifer gelingen, zu einem Text die durch den öffentlichen Schlüssel t eindeutig bestimmte Signatur $s(x)$ zu berechnen, vgl. Bild 3-54. Wie soll der arme Unterzeichner beweisen, daß er $s(x)$ nicht berechnet hat, obwohl er dies könnte?

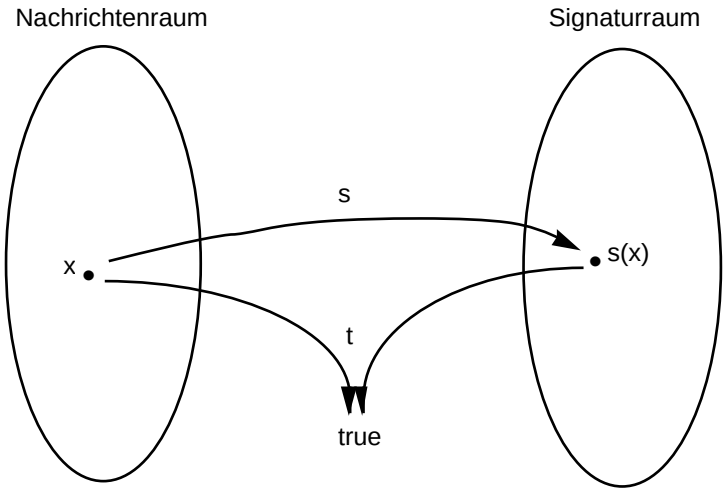


Bild 3-54: Übliches digitales Signatursystem: Zu einem Text *x* und einem Testschlüssel *t* (und ggf. der Signaturumgebung, vgl. GMR) gibt es genau eine Signatur *s*(*x*)

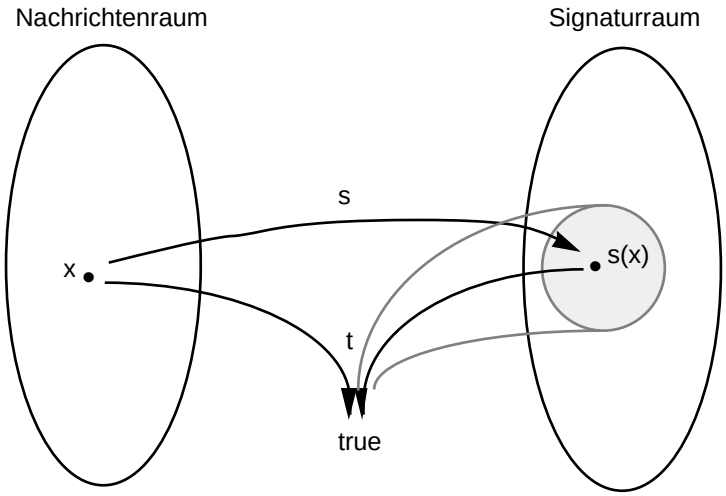


Bild 3-55: Für den Unterzeichner unbedingt sicheres digitales Signatursystem: Zu einem Text *x* und einem Testschlüssel *t* gibt es sehr viele Signaturen (grauer Bereich), die allerdings im Signaturraum dünn liegen

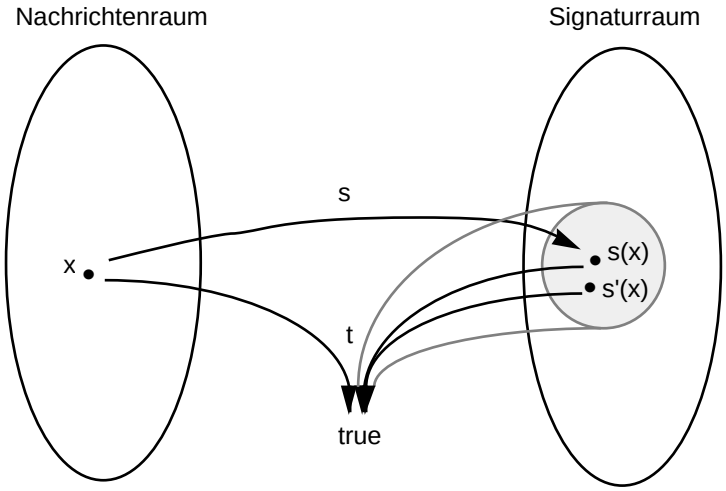


Bild 3-56: Kollision als Fälschungsbeweis in einem für den Unterzeichner unbedingt sicheren digitalen Signatursystem

Die Sicherheit kann auch anders herum „asymmetrisch“ verteilt werden: Bei einem für den Unterzeichner unbedingt sicheren Signatursystem (Bild 3-55 und 3-56) ist sie:

- unbedingt sicher für den Unterzeichner. Zu jedem öffentlichen Testschlüssel gibt es sehr viele geheime Signierschlüssel, die wiederum dünn im Raum aller Signierschlüssel liegen. Ein komplexitätstheoretisch beschränkter Unterzeichner kennt von diesen vielen Signierschlüsseln nur einen, nämlich s in Bild 3-55. Mit ihm kann er zu einem Text x genau eine Signatur $s(x)$ erzeugen. Erzeugt ein komplexitätstheoretisch unbeschränkter Angreifer eine Signatur, so ist sie mit der Wahrscheinlichkeit

  $$\frac{1}{\text{Anzahl aller möglichen Signaturen}}$$

  von $s(x)$ verschieden. Erfährt ein Unterzeichner, daß er zu x die Signatur $s'(x)$ geleistet haben soll, so prüft er mittels des nur ihm bekannten Schlüssels s, ob $s'(x) = s(x)$, vgl. Bild 3-57. Falls nein, d.h. wenn $s \neq s'$, hat der Unterzeichner einen Fälschungsbeweis und kann nachweisen, daß die komplexitätstheoretische Annahme verletzt ist. Man sollte sich dann auf diese Annahme nicht mehr verlassen, z.B. also längere Primzahlen verwenden. Außerdem kann man die Verantwortung so definieren, daß der Empfänger der Signatur einen auftretenden Schaden trägt.
- nur kryptographisch für den Empfänger. Denn es kann einem Angreifer oder auch dem wirklichen Unterzeichner gelingen, einen Fälschungsbeweis zu finden.

Hauptvorteil eines solchen Signatursystems ist, daß man einen Beweis erhält, wenn das Signatursystem gebrochen ist (**Fail-Stop-Signatursystem**), und deshalb die Verantwortung für den Schaden beliebig verteilen kann: Jeder komplexitätstheoretisch beschränkte Teilnehmer (inkl. dem legitimen Unterzeichner, d.h. dem Generierer des Schlüsselpaares (t, s)) kann nur mit vernachlässigbarer Wahrscheinlichkeit einen Fälschungsbeweis erzeugen.

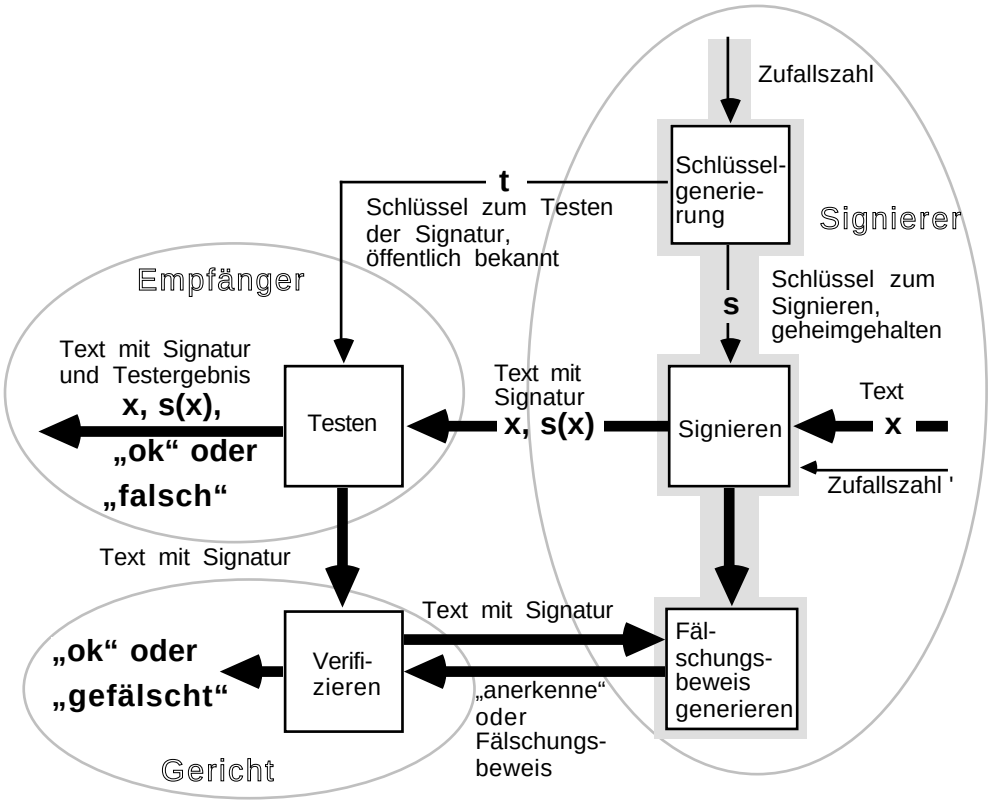


Bild 3-57: Fail-Stop-Signatursystem
(≈ Glasvitrine mit Schloß, es gibt nur einen Schlüssel, um etwas hineinzutun. Das Glas kann eingeschlagen, nicht aber danach unerkennbar ersetzt werden.)

Mehr über diese Sorte Signatursystem steht in [PfWa_91, Pfit8_96] sowie in [WaPf_89, WaPf1_89, Pfit_89, Bleu_90, BlPW_91, HePe_93].

3.9.3 Unbedingt sichere Pseudosignaturen

Es gibt mittlerweile auch ein System, das fast die Eigenschaften von digitalen Signaturen hat und informationstheoretisch sicher ist, d.h. wo Signaturen überhaupt nicht gefälscht werden können. (Bis auf die exponentiell kleine Wahrscheinlichkeit, daß ein Angreifer richtig rät, die alle Authentikationssysteme haben.) Es ist aber leider bisher ziemlich unpraktikabel.

Mehr über diese Sorte Signatursystem steht in [ChRo_91].

3.9.4 Nicht herumzeigbare Signaturen (Undeniable Signatures)

Übliche digitale Signaturen können leicht perfekt kopiert und deshalb unbeschränkt herumgezeigt werden. Abhilfe schafft hier die Idee, das Testen einer Signatur nicht als vom Empfänger autonom durchführbaren deterministischen Algorithmus vorzusehen, sondern als ein interaktives Protokoll, zu dem man denjenigen braucht, der die Signatur angeblich geleistet hat, vgl. Bild 3-58. So erfährt er, wenn ein Dritter seine Signatur erhalten soll. („Erhalten“ bedeutet hier nicht nur, die entsprechende Bitkette zu erfahren, sondern auch Gewißheit zu erlangen, daß es sich um eine gültige Signatur handelt.) Wichtig bei der Gestaltung des Protokolls ist, daß der Unterzeichner eine echte Signatur nicht ableugnen kann (deshalb die Bezeichnung von David Chaum: undeniable): Betrügt er beim Protokoll, so wird er mit einer Wahrscheinlichkeit exponentiell nahe bei 1 erwischt. Macht er beim Protokoll nicht mit, obwohl er dazu verpflichtet ist, gilt dies als Beweis für die Echtheit der Signatur.

Mehr über diese Sorte Signatursystem steht in [ChAn_90].

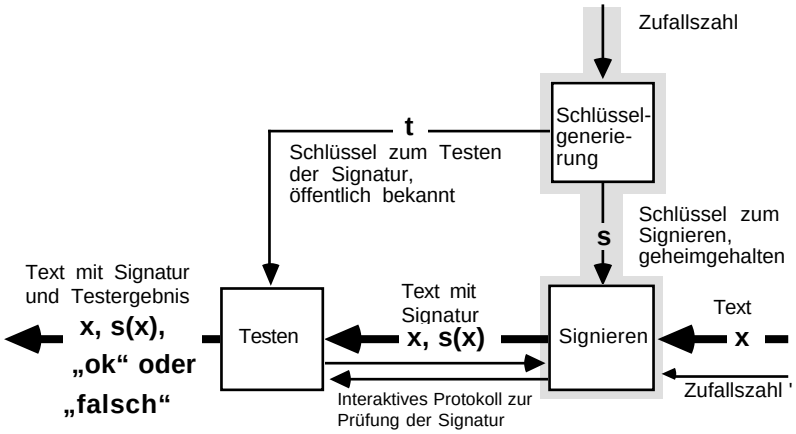


Bild 3-58: Signatursystem mit nicht herumzeigbaren Signaturen (undeniable Signatures)
(≈ undurchsichtige Vitrine mit Guckloch und Schloß; es gibt nur einen Schlüssel, um etwas hineinzutun, und der Besitzer des Schlüssels kontrolliert den Zugang zum Guckloch)

3.9.5 Blind geleistete Signaturen (Blind Signatures)

Ziel dieses Signatursystem-Typs ist, daß Signierer Signaturen leisten können, ohne dabei das Geringste darüber zu erfahren, *was* sie signieren. Zuerst mag dies vollkommen unsinnig wirken, denn der Signierer sollte daran interessiert sein, was er signiert. Für manche Anwendungen ist aber das Gegenteil der Fall: Bei ihnen kommt es nur darauf an, daß der Signierer weiß, *daß* er signiert. Es interessiert ihn nicht – und darf ihn auch nicht interessieren – was er signiert.

Ein Beispiel hierfür ist ein digitales Zahlungssystem, bei dem der Signierer Banknoten von bestimmtem Wert schafft, indem er mit dem passenden Signierschlüssel eine Zufallszahl (passender Länge und Redundanz) signiert. Er muß dabei nur merken und sicher sein, daß er genau eine Banknote eines solchen Wertes schafft. Dann kann er hierfür die passende Menge Geld verlangen. Derjenige, der sich die Banknote ausstellen, d.h. blind signieren läßt, hat das Interesse, daß der Aussteller die Banknote dabei nicht sieht, genauer: nichts über sie erfährt. Dann kann der Aussteller auch später keine Zahlungsflüsse im digitalen Zahlungssystem analysieren. Mehr über diese Anwendungen steht in §6.4.

In Bild 3-59 ist der **Ablauf des blinden Signierens** dargestellt:

Wie üblich wird zunächst vom Signierer ein Schlüsselpaar (t,s) erzeugt und t bekannt gemacht.

Anders als bisher üblich wird der (blind) zu signierende Text x vom Empfänger der Signatur (und nicht etwa vom Signierer) erzeugt – ggf. mit Redundanz versehen, was im Bild und der Notation nicht aufgeführt ist – und mittels einer Zufallszahl' (z') geblendet, d.h. in gewissem Sinne verschlüsselt.

Der geblendete Text $z'(x)$ wird dem Signierer zur Signatur übermittelt.

Der Signierer signiert $z'(x)$ mit s und schickt $z'(x)$, $s(z'(x))$ an den Empfänger der Signatur.

Nur dieser kennt die Zufallszahl z', die zum Entblenden der Signatur nötig ist – was Abhörer auf dem Übermittlungsweg, ja sogar den Signierer am Ermitteln und Nutzen der blind geleisteten Signatur hindert. Der Empfänger entblendet also $z'(x)$, $s(z'(x))$ mit z' und erhält so die Signatur $s(x)$. Danach kann er mittels t testen, ob die Signatur $s(x)$ zu x paßt.

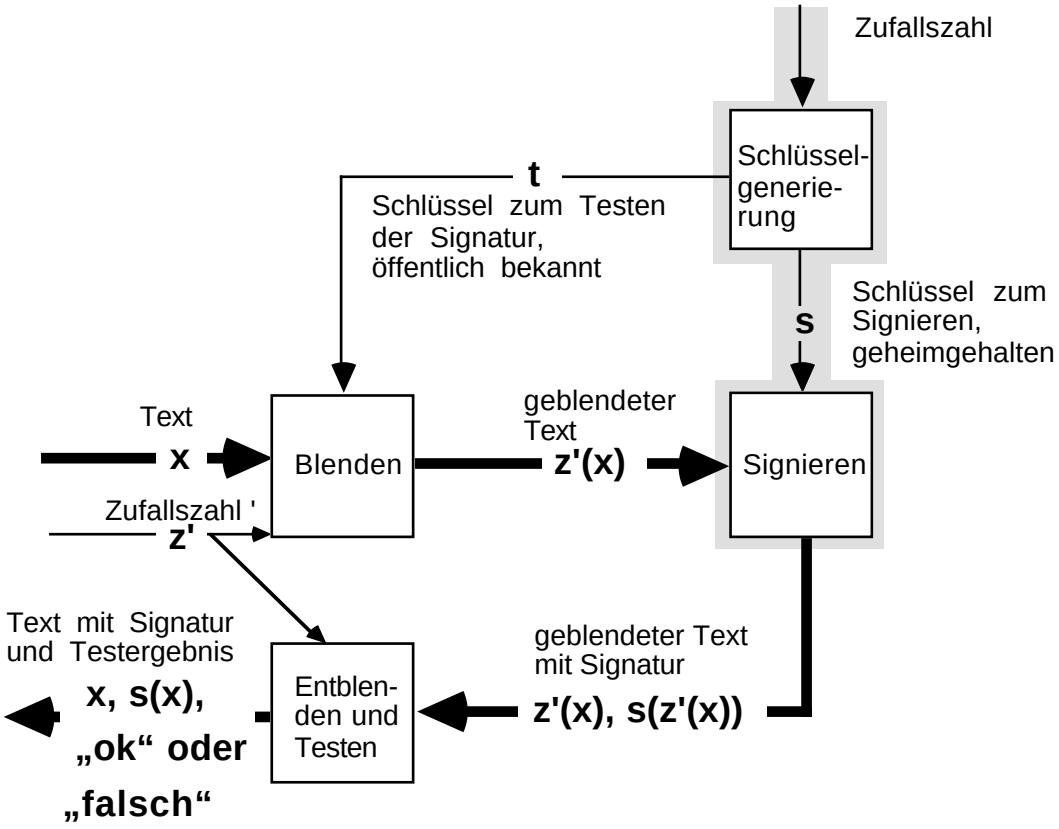


Bild 3-59: Signatursystem zum blinden Leisten von Signaturen (blind Signatures)

Blind geleistete Signaturen mit RSA:

Hier wird der Angriff von Davida in der Form von Judy Moore, vgl. §3.6.3.1, für einen guten Zweck genutzt:

Schlüsselgenerierung, das Versehen von Texten mit Redundanz (muß vor dem Blenden erfolgen), Signieren und Testen erfolgt exakt wie in §3.6.4.2 beschrieben.

Die Zufallszahl' z' wird in $\mathbb{Z}_n^*$ gleichverteilt zufällig gewählt.

Das Blenden mittels Zufallszahl' z' erfolgt, indem der (redundante) Text x modular mit z'^t multipliziert wird. Als geblendeter Text entsteht: $x \bullet z'^t \bmod n$.

(Anmerkung: Ist $x \in \mathbb{Z}_n^*$, so erfährt der Signierer dadurch im informationstheoretischen Sinne nichts über x, da z'^t gleichverteilt in $\mathbb{Z}_n^*$ ist. Andernfalls benötigt der Generierer von x die Hilfe des Signierers überhaupt nicht: ggT(x,n) ergibt einen nichttrivialen Faktor von n. Damit hat der Generierer von x RSA vollständig gebrochen, vgl. §3.1.3.1 und §3.6.1, so daß er sich x selbst signieren kann.)

Als geblendeter Text mit Signatur entsteht mittels modularer Exponentiation mit s: $x \bullet z'^t \bmod n$, $(x \bullet z'^t)^s \bmod n$.

Das Entblenden erfolgt, indem durch z' modular dividiert wird: $(x \bullet z'^t)^s \bullet z'^{-1} \bmod n = x^s \bullet (z'^t)^s \bullet z'^{-1} \bmod n = x^s \bullet z' \bullet z'^{-1} \bmod n = x^s$.

Der blind signierte Text mit Signatur lautet dann: x, x^s.

Mehr über diese Sorte Signatursystem steht in [Chau_83, Cha1_83, Cha1_88, Cha8_85, Chau_89, Schn_96].

3.9.6 Schwellwertschema

Mittels eines Schwellwertschemas (engl. *threshold scheme* oder auch *secret sharing scheme* genannt)) kann der Eigentümer eines Geheimnisses G es so in n Teile zerlegen, daß die Kenntnis von mindestens k Teilen die effiziente Rekonstruktion des Geheimnisses erlaubt, k–1 Teile aber keine Information über das Geheimnis liefern.

Adi Shamirs Polynominterpolation:

Das Geheimnis G sei Element von $\mathbb{Z}_p$, p sei Primzahl.[93]
Der Eigentümer wählt ein Polynom $q(x)$ des Grades k-1 folgendermaßen:
Er wählt die Koeffizienten $a_1, a_2, ..., a_{k-1}$ zufällig, gleichmäßig und unabhängig in $\mathbb{Z}_p$.
Er definiert $q(x) := G + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1}$, wobei alle Operationen in $\mathbb{Z}_p$ erfolgen.
Er berechnet in $\mathbb{Z}_p$ die n Teile $(i, q(i))$ mit $1 \le i \le n$, wobei $n < p$ gelten muß.

Aus k Teilen $(x_j, q(x_j))$ $(j = 1,..,k)$ kann das Polynom $q(x)$ folgendermaßen berechnet werden (Lagrange-Polynominterpolation [Denn_82 Seite 180]):

$$q(x) = \sum_{j=1}^{k} q(x_j) \prod_{m=1,\, m\neq j}^{k} \frac{(x - x_m)}{(x_j - x_m)} \mod p$$

Das Geheimnis G erhält man dann als $q(0)$.

Beweisskizze:

1. Das Schwellwertschema liefert bei Kenntnis von lediglich k-1 Teilen $(j, q(j))$ keinerlei Information über G, da es für *jeden* Wert von G immer noch *genau ein* Polynom vom Grad k-1 gibt, das durch die insgesamt k Punkte $(j, q(j))$ und $(0, G)$ geht.

2. Obige Formel berechnet das richtige Polynom, da es sowohl den richtigen Grad k-1 hat (jeder Produktteil hat den Grad k-1 und durch Summation bleibt dies erhalten) und für jedes Argument x_j den Wert $q(x_j)$ liefert: Setzt man im Produktteil für x ein x_j ein, so ist das Produkt genau 1, wenn der Summationsindex den Wert j hat (denn dann hat jeder einzelne Quotient den Wert 1, also ist ihr Produkt 1) und 0 sonst (denn dann ist ein Faktor 0). Diese 1 multipliziert mit $q(x_j)$ ergibt als Gesamtwert der Summe $q(x_j)$.

Anmerkung: Es ist nicht zwingend, daß die Teile als Werte des Polynoms an den Stellen 1 bis n gewählt werden. Jede andere Wahl von n verschiedenen Stellen ($\neq 0$) tut es auch.

Ist das Geheimnis groß und möchte man nicht modulo einem großen p rechnen, so zerschneidet man das Geheimnis in mehrere Stücke und führt obiges Schema mit jedem der Stücke durch.

Oftmals sollen Geheimnisse nicht nur vertraulich bleiben, sondern sie müssen auch für ihren Eigentümer über lange Zeit verfügbar sein. Möchte sich der Eigentümer eines Geheimnisses G vor seinem Verlust etwa durch höhere Gewalt schützen, andererseits es aber niemand anderem allein zur Aufbewahrung anvertrauen, so gibt der Eigentümer des Geheimnisses n Bekannten jeweils ein unterschiedliches Teil. Verliert er sein Geheimnis, so bittet er k Bekannte, ihm Kopien ihrer Teile zu geben. Aus

---

93 Vorsicht Falle: p darf nicht in starker Abhängigkeit von G gewählt werden. Also auf keinen Fall: Wähle zu G die nächst größere Primzahl p. Sonst erhält ein Angreifer durch Kenntnis des öffentlich werdenden p schon sehr, sehr viel Information über G.

ihnen kann er dann *G* effizient berechnen (im wesentlichen quadratischer Aufwand in *k*). Arbeiten von den *n* Bekannten nur höchstens *k*–1 in böser Absicht zusammen, so können sie *G* nicht hinter dem Rücken seines Eigentümers heimlich rekonstruieren, wie viel auch immer sie rechnen. Wählt der Eigentümer *k* fast so groß wie *n*, kann sein Geheimnis durch Verlust von wenigen Teilen bei seinen Bekannten (sei es durch höhere Gewalt oder Bosheit) unrekonstruierbar werden, andererseits kann bei solcher Wahl von *k* in Bezug auf *n* sein Geheimnis fast nicht unbefugt rekonstruiert werden. Wählt der Eigentümer *k* sehr viel kleiner als *n*, kann sein Geheimnis nur durch Verlust von vielen Teilen bei seinen Bekannten unrekonstruierbar werden, andererseits kann bei solcher Wahl von *k* in Bezug auf *n* sein Geheimnis wesentlich leichter unbefugt rekonstruiert werden. Durch geeignete Wahl von *k* und *n* kann der Eigentümer die Zerlegung seines Geheimnisses also an seine Bedürfnisse anpassen.

Beweise finden Sie in [Sham_79], eine Erweiterung zum informationstheoretisch sicheren Erkennen von Verfälschungen des Geheimnisses durch Verfälschung von Teilen in [ToWo_88].

Um Störungen der Integrität des Geheimnisses komplexitätstheoretisch sicher zu verhindern oder zumindest zu erkennen, ist es sinnvoll, daß der Eigentümer des Geheimnisses die *n* Teile (i, q(i)) einzeln digital signiert. Bevor ein Teil zur Rekonstruktion des Geheimnisses verwendet wird, wird die Signatur geprüft. Dann kann, sofern die Signaturen nicht gefälscht werden können, niemand für die Rekonstruktion eines falschen Geheimnisses sorgen oder die Rekonstruktion des Geheimnisses sehr aufwendig machen. Ersteres wäre der Fall, wenn nur *k* Teile zur Verfügung stehen und eines davon verfälscht ist. Letzteres wäre der Fall, wenn $k+1+v$ Teile zur Verfügung stehen, wobei $k+1$ Teile richtig und *v* Teile verfälscht wären. Da nicht bekannt wäre, welche Teile verfälscht sind, bliebe nichts anderes übrig, als aus allen Mengen mit jeweils *k* Teilen ein Geheimnis zu rekonstruieren und das zu nehmen, was am häufigsten rekonstruiert wird. Leider gibt es sehr viele Mengen mit *k* Teilen, ausgewählt aus den zur Verfügung stehenden $k+1+v$.

Alles bisher über Schwellwertschemata Gesagte ging davon aus, daß der Eigentümer des Geheimnisses gutwillig ist, d.h. die *n* Teile so generiert, daß sich aus ihnen sein Geheimnis rekonstruieren läßt. Lediglich seine Umwelt, insbesondere auch manche Aufbewahrer der *n* Teile wurden als Angreifer betrachtet. Betrachtet man auch den Generierer der Teile als potentiellen Angreifer, dann kann man an ein Schwellwertschema zusätzliche Anforderungen stellen:

1. konsistentes Ergebnis
2. vorgegebenes Ergebnis

Leseempfehlungen

Geschichte der Kryptographie

Ein Überblick von 1900 bis 1999 aus deutscher Sicht: Leib_99

Lehrbücher der verwendeten mathematischen Grundlagen:

Eine echte Einführung in Algebra, aber speziell für Informatiker bzw. mit Hinblick auf Implementierungen und relativ verständlich geschrieben: Lips_81

Sehr elementare Darstellung der Grundlagen der elementaren Zahlentheorie, v.a. für euklidschen Algorithmus u.ä.: Lüne_88

Kurz und nett zu lesen, etwas veraltete Terminologie: ScSc_73

Wenn man die schnellen Algorithmen gern genau wüßte: Knut_97

in enger Verbindung mit Kryptographie: Kran_86, MeOV_97

Lehrbücher der Kryptographie:

ausführlicher klassischer Aufbau: Denn_82, DaPr_89, Hors_85
moderner Aufbau, aber sehr knapp: Bras_88
moderner Aufbau, sehr ausführlich und praxisnah: Schn_96
moderner Aufbau, sehr ausführlich mit theoretischer Fundierung: MeOV_97

Aktuelle Forschungsliteratur:

Journal of Cryptology, Springer-Verlag Heidelberg
Tagungsserien Crypto, Eurocrypt, Asiacrypt, Tagungsbände erscheinen in der Serie LNCS des Springer-Verlags Heidelberg
The Theory of Cryptography Library: access via URL http://philby.ucsd.edu/cryptolib.html, email should be sent to cryptolib@philby.ucsd.edu

Hinweise auf Standards, Patente, Produkte, Kommentare zu aktuellen Entwicklungen:

http://www.rsa.com
http://www.rsa.com/rsalabs
http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/ oder
abonnieren unter cipher-request@itd.nrl.navy.mil

Hinweise auf die rechtliche Situation bzgl. der Benutzung sowie des Exports bzw. Imports von Kryptographie:

http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm

4 Steganographische Grundlagen

Steganographie ist die alte Kunst und die junge Wissenschaft vom Verstecken geheimzuhaltender Daten in umfangreicheren, harmlos wirkenden Daten.

Bild 4-1 veranschaulicht am Schutzziel Vertraulichkeit die Unterschiede zur Kryptographie. Während der Angreifer bei Verwendung von guter Kryptographie merkt, daß er den Schlüsseltext nicht versteht und folglich vermuten wird, daß gerade vertraulich kommuniziert wird, hält der Angreifer bei Verwendung von guter Steganographie den Stegotext für eine plausible und von ihm voll verstandene Nachricht. Er merkt nicht, daß vertraulich kommuniziert wird. Im dargestellten Beispiel ist die Hülle (das Original-Urlaubsbild Ballon über Savanne) genauso plausibel wie der Stegotext (das leicht modifizierte Urlaubsbild: ein genauso schöner Ballon über einer genauso stimmungsvollen Savanne in gleicher photographischer Qualität).

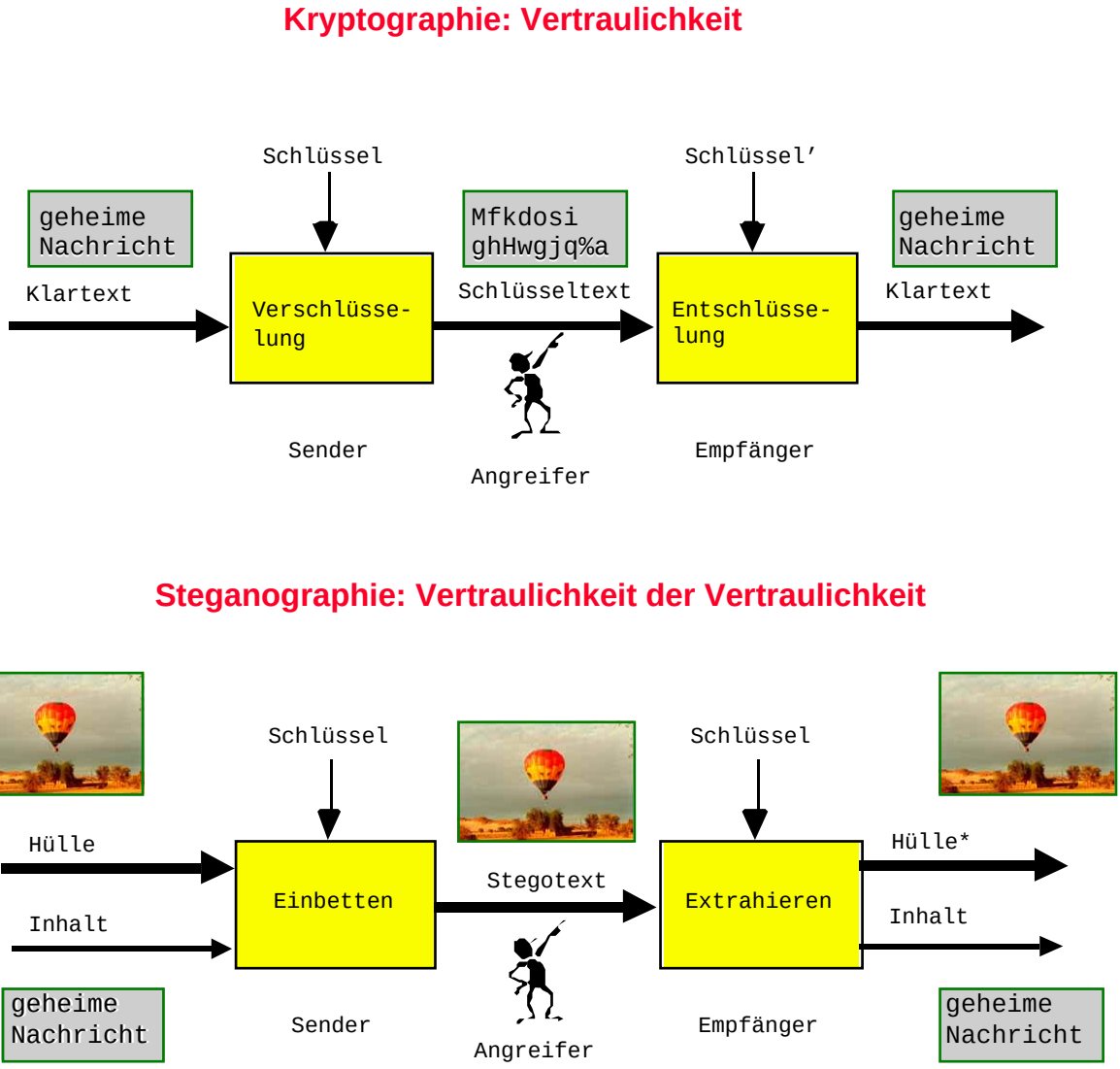


Bild 4-1: Kryptographie vs. Steganographie

Oftmals wird in Bild 4-1 gelten: *Hülle* = Stegotext*, d.h. der Empfänger will und kann den Wert von *Hülle* vor dem Einbetten nicht rekonstruieren und nimmt *Stegotext* als *Hülle** oder ignoriert diesen Ausgabeparameter des **Stegosystems**[94] vollkommen.

Die alte Kunst Steganographie besteht beispielsweise in der Verwendung von

- Geheimtinten, die nach dem Schreiben unsichtbar sind, vom kundigen Empfänger aber chemisch oder thermisch sichtbar gemacht werden können,
- Microdots, das sind auf Punktgröße verkleinerte Photographien, die statt i-Punkten in harmlose Texte eingeklebt werden und unter dem Mikroskop gelesen werden können, sowie – aus Gründen der Menschenrechte und Realzeitanforderungen moderner Telekommunikation nicht mehr brauchbare –
- Sklaven, denen der Schädel rasiert, die geheim zu übermittelnde Botschaft auf die Kopfhaut geschrieben, genügend Zeit für das Nachwachsen der Haare gegeben und dann der Empfänger gesagt sowie die Anweisung mitgegeben wurde, ihm zu sagen: „Herr, schere mir die Haare."

Alternativ wurden geheim zu übermittelnde Botschaften

- vom Boten geschluckt[95] oder an Tiere verfüttert[96] sowie
- auf die Unterseite von Wachstafeln geschrieben, so daß der eingeweihte Empfänger nur das Wachs entfernen mußte.

Einen guten historischen Überblick gibt [Kahn_96].

Die junge Wissenschaft Steganographie bedient sich Rechner, um geheimzuhaltende Daten beispielsweise in digitalisierte Bilder, Video- oder Audiosignale einzubetten. Dies kann zwei Zielen dienen:

Steganographische Konzelationssysteme haben das Ziel, nicht nur die geheimzuhaltende Nachricht, sondern sogar ihre Existenz zu verbergen, vgl. Bild 4-2. Dies ist nützlich, wenn vertrauliche Kommunikation verdächtig, unerwünscht oder gar verboten ist.[97]

Steganographische Authentikationssysteme haben das Ziel, Informationen über Urheber und/oder Nutzungsrechteinhaber so in digital(isiert)e Werke hineinzucodieren, daß sie nicht ohne Zerstörung des Werkes entfernt werden und so zur Wahrung von Urheberrechten benutzt werden können. Bei starken Änderungen des Werkes werden die eingebetteten Urheberinformationen natürlich nicht fehlerfrei extrahiert, vgl. Bild 4-3.

---

94 Ähnlich wie statt *kryptographisches Konzelationssystem* kürzer *Kryptosystem* gesagt wird, so wird statt *steganographisches Konzelationssystem* oft *Stegosystem* gesagt. Puristen meinen, es müßte *Steganosystem* heißen und haben damit zweifellos recht. Aber *Stegosystem* ist halt noch kürzer und wird deshalb allgemein verwendet.

95 Bezüglich geheimer Kommunikation ist auch dies aus der Mode gekommen, aber wohl bzgl. Drogentransport nach wie vor gebräuchlich. Hoffentlich reißt die Verpackung nicht, sonst ist im ersteren Fall die Botschaft, im zweiten Fall auch der Bote hops.

96 Dies hat gegenüber dem Verfüttern an Boten Zeitvorteile, da man durch Schlachten der Tiere schnell an die Botschaft kommt.

97 Die Entwicklung steganographischer Konzelationssysteme erhielt enormen Schub durch die seit 1993 öffentlich geführte Diskussion, die Benutzung von Kryptographie für vertrauliche Kommunikation einzuschränken oder gar zu verbieten, um die Überwachung potentieller Straftäter zu erleichtern. Die Existenz sicherer steganographischer Konzelationssysteme ist ein Argument gegen diese sogenannte Kryptoregulierung, vgl. §9.

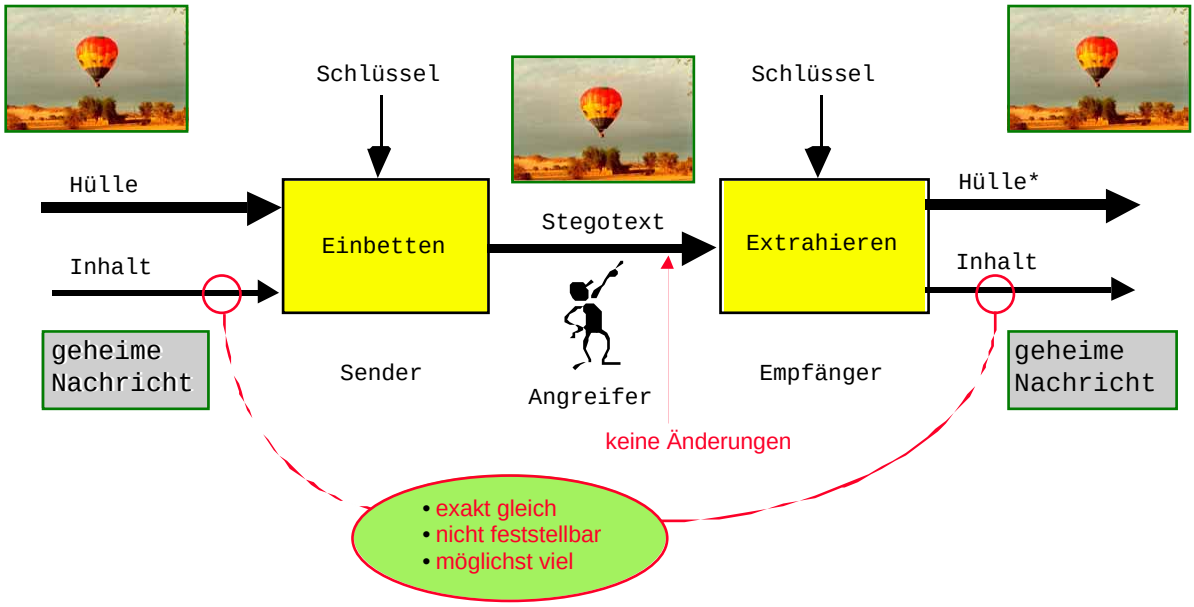


Bild 4-2: Steganographisches Konzelationssystem

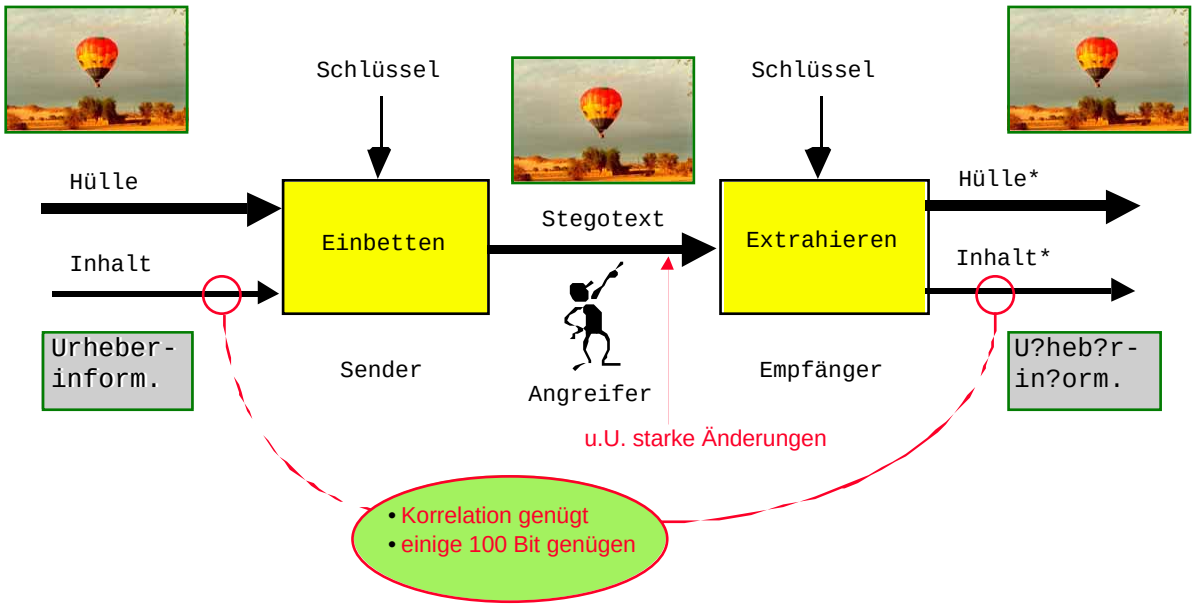


Bild 4-3: Steganographisches Authentikationssystem

In §4.1 wird eine Systematik steganographischer Systeme entwickelt. §4.2 leitet notwendige Bedingungen für informationstheoretisch sichere Steganographie her. In §4.3 werden zwei gegensätzliche Stegoparadigmen diskutiert. §4.4 erläutert die Paritätscodierung als Mittel zur Verstärkung der Verborgenheit.

Die folgenden Abschnitte geben Beispiele dafür, wie in digitale Telefongespräche (§4.5), Bilder (§4.6) und Bewegtbilder (§4.7) eingebettet werden kann.

4.1 Systematik

Zuerst wird in §4.1.1 beschrieben, was unter einem steganographischen System verstanden wird. Danach folgt in §4.1.2 eine Anmerkung zum Austausch steganographischer Schlüssel. Grundsätzliches über steganographische Sicherheit bringt §4.1.3.

4.1.1 Steganographische Systeme

Moderne steganographische Systeme haben mit modernen kryptographischen Systemen gemein, daß ihre Sicherheit nicht auf der Geheimhaltung des Verfahrens beruht, sondern allein auf der Geheimhaltung von Schlüsseln, die das Verfahren parametrisieren. Wo immer eine Verwechslungsmöglichkeit besteht, sprechen wir von **Stegoschlüsseln** im Gegensatz zu **Kryptoschlüsseln**.

Bisher sind lediglich *symmetrische* steganographische Systeme bekannt, d.h. zum Einbetten und Extrahieren wird die gleiche Information, insbesondere also der gleiche Stegoschlüssel verwendet [FePf_97].

4.1.1.1 Steganographische Konzelationssysteme, Überblick

ZFPW_97
Hülle (engl. cover)

Ziel „Verbergen der Existenz der geheimzuhaltenden Nachricht“ (plakativ ausgedrückt: Vertraulichkeit der Vertraulichkeit)

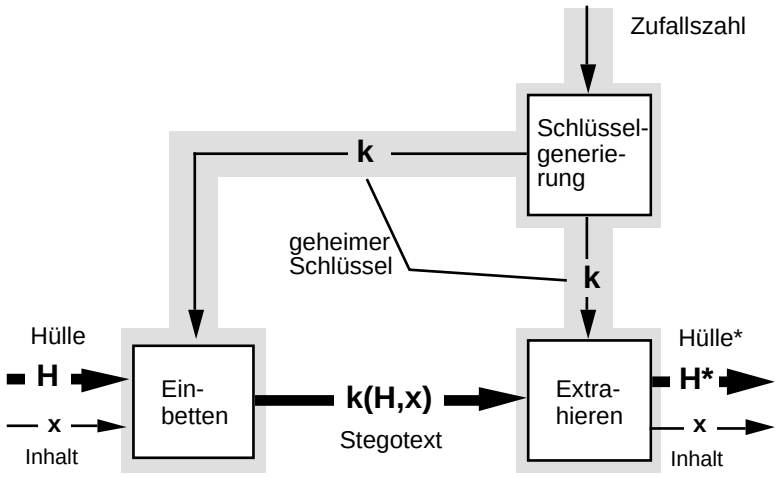


Bild 4-4: Steganographisches Konzelationssystem gemäß Einbettungsmodell

Synthesemodell erklären

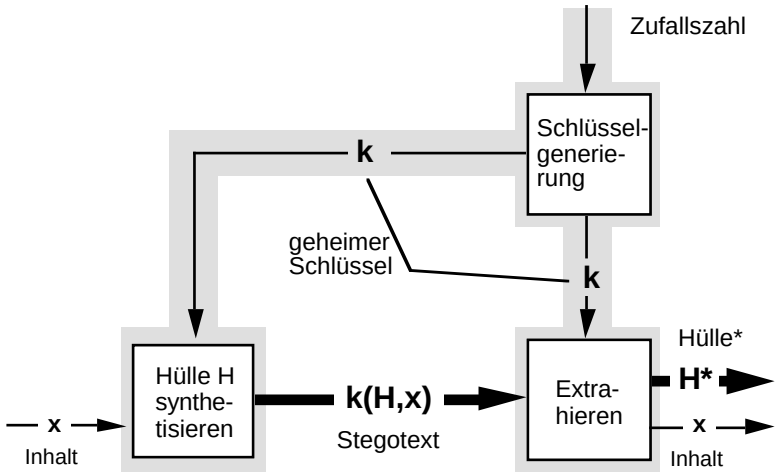


Bild 4-5: Steganographisches Konzelationssystem gemäß Synthesemodell

Da beim Einbettungsmodell die Einbettungsfunktion die Eingabe *Hülle* ignorieren und eine andere Hülle synthetisieren und verwenden kann und damit das Einbettungsmodell das Synthesemodell umfaßt, ist das Einbettungsmodell das allgemeinere Modell. Deshalb wird es im folgenden als *das* Modell für steganographische Konzelation verwendet [ZFPW_97 Kap. 2.1].

4.1.1.2 Steganographische Authentikationssysteme, Überblick

Watermarking und Fingerprinting:
Im Gegensatz zu steganographischen Konzelationssystemen ist bei steganographischen Authentikationssystemen oftmals nur gefordert, daß ihre Verwendung für den Menschen nicht wahrnehmbar ist, damit beispielsweise Watermarking und Fingerprinting von Kunstwerken diese nicht störend verändert. Dies ist eine deutlich schwächere Forderung als die, daß Watermarking und Fingerprinting ohne Kenntnis des Stegokeys nicht nachweisbar sein dürfen.

Allerdings ist für Watermarking und Fingerprinting davon auszugehen, daß der Stegotext stark und vor allem gezielt verändert wird, um die eingebettete Urheberinformation zu entfernen. Hiergegen muß das steganographische Authentikationssystem **robust** sein, d.h. die eingebettete Urheberinformation muß auch nach solchen Angriffen erkennbar sein, etwa durch eine genügend starke Korrelation zwischen eingebetteter und extrahierter Information. In dem für offene Netze einzig realistischen Szenario, daß der Angreifer alle Details des steganographischen Authentikationssystems kennt, wird Robustheit bisher von keinem einzigen steganographischen Authentikationssystem erreicht. Meine persönliche Vermutung ist, daß dies auch so bleiben wird.

Linartz Papier vom IHW98 als Grenze von Watermarking

4.1.1.3 Vergleich

In der folgenden Tabelle werden die steganographischen Systeme miteinander sowie mit den kryptographischen Systemen verglichen.

Steganographische	
Konzelationssysteme	**Authentikationssysteme**
Existenz der Einbettung soll verborgen werden	Existenz der Einbettung braucht nicht verborgen zu sein

Die Hülle (engl.: cover) darf drastisch geändert, im Extremfall synthetisch erzeugt werden	Einbettung soll nicht stören, d.h. Veränderungen der Hülle (dies ist das zu schützende Werk) sollen sinnlich nicht wahrnehmbar sein
möglichst viel einbetten	gegebene Menge (100 bis 1000 Bit) möglichst robust einbetten
Ziel Angreifer:	
erkennen, ob eingebettet wurde	watermark bzw. fingerprint gezielt ändern, z.B. austauschen, zerstören, generieren
Parameter:	
stego unverändert	stego kann vom Angeifer deutlich, allerdings nicht zu deutlich (sonst Verlust des Werkes) verändert werden das watermark bzw. der fingerprint kann bei *beiden* Parteien Eingabeparameter sein ebenso kann das unveränderte Werk (dies kann sowohl das Werk ohne wie auch mit eingebetteter Urheberinformation oder auch beides sein) bei *beiden* Parteien Eingabeparameter sein
Überdeckung der Funktionalität des kryptographischen Systems:	
ja, da mit der Vertraulichkeit der Existenz der Nachricht immer auch die Vertraulichkeit ihres Inhalts gegeben ist	nein, da zwar Urheberschaft erkennbar, nicht aber kleine Veränderungen[98]

4.1.2 Eine Anmerkung zum Schlüsselaustausch: Steganographie mit öffentlichen Schlüsseln

symmetrische Steganographie + Diffie-Hellman Schlüsselaustausch, vgl. §3.9.1

Bei Verwendung von PGP ab Version 5 sollten Diffie-Hellman Schlüssel mit der Einstellung „Faster key generation“ erzeugt werden. Dann wird für alle Teilnehmer ein gemeinsames *p* und *g* verwendet.

Sollten die Teilnehmer kein gemeinsames *p* und *g* verwenden, so ist Diffie-Hellman Schlüsselaustausch für Steganographie nicht brauchbar, während er für Kryptographie durchaus auch so brauchbar ist, vgl. Aufgabe 3-23 c).

Übrig bleibt für Steganographie dann nur das, was in [Ande4_96, AnPe_98 Seite 480] Steganographie mit öffentlichen Schlüsseln genannt wird (vgl. Aufgabe 4-5): Zunächst wird ohne Verwendung eines steganographischen Schlüssels der mit dem öffentlichen kryptographischen Konzelationsschlüssels des Partner verschlüsselte steganographische Sitzungsschlüssel eingebettet und übermittelt.

Für diese Verschlüsselung eines steganographischen Sitzungsschlüssels kann dann auch der Diffie-Hellman Schlüsselaustausch ohne gemeinsames *p* und *g* (vgl. Aufgabe 3-23 c)) verwendet werden: Der Sender generiert unter Verwendung von *p* und *g* des Empfängers für die Sitzung seinen öffentlichen Parameter und bettet diesen ohne Verwendung eines steganographischen Schlüssels ein.

---

98 Manche Autoren definieren sogenannte *zerbrechliche* Watermarks, d.h. Watermarks ohne Robustheit, zu genau dem Zweck, Änderungen des Werkes zu erkennen. Da dieses Schutzziel aber auch *ohne* Änderung des Werkes durch kryptographische Authentikationssysteme zu erreichen ist, halte ich zerbrechliche Watermarks für ziemlich überflüssig.

Danach kann dann der geheime Diffie-Hellman Sitzungsschlüssel auch vom Empfänger berechnet und als steganograhischer Schlüssel benutzt werden.

4.1.3 Grundsätzliches über Sicherheit

ZFPW_97

4.1.3.1 Angriffsziele und -Erfolge

4.1.3.2 Angriffstypen

4.1.3.3 Stegoanalyse führt zu verbesserter Steganographie

Wissenschaftlich interessant (und für die Regulierbarkeit von Steganographie bedeutsam, vgl. §9) ist, daß jedes Verfahren zur Entdeckung von Steganographie unmittelbar zur Verbesserung des Stegosystems verwendet werden kann. Dieser bereits in [MöPS_94 Kap. 4.3.2] angedeutete Sachverhalt wird nun genauer dargestellt und untersucht. Hierbei werden hauptsächlich steganographische Konzelationssysteme betrachtet.

Zuerst wird eine Konstruktion für das Verbergensmodell, d.h. für S und S^{-1} angeben. Dabei wird die Umkehroperation nicht geändert – denn sie hat schon beim nicht verbesserten Stegosystem die Aufgabe, nur dann ein *emb* nach außen zu geben, wenn tatsächlich eins eingebettet wurde. Um dies mit genügend großer Wahrscheinlichkeit richtig zu entscheiden, benötigt die Umkehroperation genügend Redundanz.

Danach werden Konstruktionen für das Einbettungsmodell, d.h. für E und E^{-1} entwickelt. Hier muß jeweils auch die Umkehroperation geändert werden.

4.1.3.3.1 Konstruktion für S und S^{-1}

Seien S der Algorithmus „Verbergen“ und S^{-1} die Umkehroperation zu S, d.h. ein Algorithmus, der die eingebetteten Daten *emb* extrahiert. Sofern nichts eingebettet wurde, liefert S^{-1} das Ergebnis *leer*. Sei R ein Ratealgorithmus, der bei Eingabe von *stego* mit überwältigender Wahrscheinlichkeit *semi-richtig* rät, ob in *stego* etwas verborgen ist oder nicht. Hierbei heiße **semi-richtig**: Wenn R „etwas_verborgen“ rät, stimmt dies. Aber „nichts_verborgen“ kann auch bedeuten, daß etwas verborgen ist, aber R dies nicht erkennt.[99]

Dann kann hieraus ein verbesserter Algorithmus „Verbergen“ S' und seine Umkehroperation S'^{-1} folgendermaßen gewonnen werden:

```
function S'(in: Source, emb; out: stego);
stego := S(Source, emb);
if R(stego) = etwas_verborgen then stego := S(Source, leer);
```

Die Umkehroperation S'^{-1} sei exakt S^{-1}.

In der Notation wurde der Stegoschlüssel weggelassen, da es für die Konstruktion auf ihn nicht ankommt, ja nicht einmal darauf, ob symmetrische oder – falls es sie denn geben sollte – asymmetrische Steganographie verwendet wird. Ebenso ändert sich nichts, wenn dem Ratealgorithmus R

99 Solch ein semi-richtiger Ratealgorithmus ist genau das, was vor Gericht vorgelegt werden müßte, um die Verwendung von verbotener Steganographie zu beweisen. Er muß Steganographie nicht unbedingt erkennen – wenn er aber behauptet, Steganographie sei verwendet worden, dann muß diese Behauptung stimmen. Andernfalls wird kein Richter diesem Algorithmus vertrauen.

noch *Source* als zusätzlicher Eingabe-Parameter gegeben wird, von einem *stego*-Angriff also zu einem *Source-stego*-Angriff übergegangen wird. Ebenso ändert es nichts, wenn R nicht nur das aktuelle *stego*, sondern alle bisherigen *stego*s (sowie ggf. auch alle bisherigen *Source*s) als Eingabe-Parameter erhält.

Für die verborgene Übertragung von Information wird das Stegosystem (also S und S^{-1} bzw. S' und S'^{-1}) in einer Schleife solange aufgerufen, bis alle verborgen zu übertragende Information tatsächlich übertragen wurde. Hierfür muß S' noch einen zusätzlichen Ausgabeparameter erhalten, der anzeigt, ob die Übertragung geklappt hat oder ob es mit demselben Wert für *emb* (aber einem anderen für *Source*) noch mal versucht werden muß.

Beh. Für das verbesserte Stegosystem S' und S'^{-1} gilt folgendes:

1. Der Empfänger empfängt nur die Daten *emb*, die der Sender sendet.
2. Die Daten *emb* werden besser versteckt als durch das ursprüngliche Stegosystem.
3. Die Nutzbandbreite des verbesserten Stegosystems ist kleiner als die des ursprünglichen.
4. Ist die Nutzbandbreite des ursprünglichen Stegosystems größer 0 und umfaßt es Nichtdeterminismus[100], so ist auch die Nutzbandbreite des verbesserten Stegosystems größer 0 und es umfaßt den gleichen Nichtdeterminismus. Dann erhält der Empfänger die Daten *emb*, die der Sender senden will, nach endlich vielen Versuchen.
5. Sofern der Aufwand für das ursprüngliche Stegosystem und für R akzeptabel ist, ist es auch der Aufwand für das verbesserte Stegosystem.

Bew.

1. Hat das ursprüngliche Stegosystem diese Eigenschaft, so hat sie auch das verbesserte: Der Sender verhält sich entweder wie beim ursprünglichen Stegosystem, dann ist Empfangen genau gleich. Oder der Sender sendet ein *stego*, in das nichts eingebettet wurde. Auch das muß das ursprüngliche Stegosystem verkraften.
2. Da *stego* nur vom ursprünglichen Stegosystem gebildet wird, versteckt das verbesserte mindestens so gut. Da angenommen war, daß das ursprüngliche Stegosystem noch hin und wieder ein *stego* generierte, bei dem die Einbettung von R erkannt werden konnte, dies beim verbesserten Stegosystem aber nicht mehr vorkommt, versteckt es die Daten besser.
3. Dies gilt nicht für jeden Einzelfall, aber wohl im Mittel.
4.
5.

Kritische Anmerkungen zu den einzelnen Punkten:

2. Es ist alles ganz prima, solange R auf einzelnen Teilen von *stego* arbeitet, so daß iteriert werden kann. Ist dies nicht der Fall, etwa weil auch Abhängigkeiten zwischen unterschiedlichen Teilen betrachtet werden müssen, dann ist nicht mal mehr ein „mindestens“ zu beweisen, geschweige denn ein „besser“. Trotzdem glaube ich, daß die Aussagen „im Mittel“ stimmen.
3. Entsprechendes gilt für die Bandbreite.

---

100 Nichtdeterminismus aus Sicht des Angreifers ist notwendig, damit für ihn unerkennbar geändert und damit ihm gegenüber verborgen werden kann. Umfaßt das Stegosystem Nichtdeterminismus, dann existiert insbesondere auch aus Sicht des Angreifers Nichtdeterminismus und damit die Möglichkeit, unerkennbar zu ändern und zu verbergen. Kurzum: Das Stegosystem ist dann nicht zu dumm gemacht.

Forschungsfragen:

Erreicht man, indem man die Konstruktion mit einem Stegosystem beginnt, das allen Indeterminismus nutzt, durch Iteration der Konstruktion ein sicheres Stegosystem maximaler Kapazität? (Hierbei darf die Iteration des Aufrufs des Stegosystems zur Übertragung von Information und die Iteration der Konstruktion zur Verbesserung des Stegosystems nicht verwechselt werden.)

4.1.3.3.2 Konstruktionen für E und E^{-1}

Damit auch einzelne Bits eingebettet werden können, ist das Ziel von E und E^{-1} schwächer als das von S und S^{-1}, wo S^{-1} entscheiden soll, ob etwas verborgen *wurde* und nur dann das *emb* nach außen geben. Dazu muß *emb* von S so redundant codiert werden, daß es mindestens einige 10 Bit umfaßt.

Das Ziel von E und E^{-1} ist:
E soll *emb* einbetten, wenn dies unerkennbar geschehen kann. E^{-1} soll erkennen, ob etwas eingebettet werden *konnte*. Falls etwas eingebettet werden konnte, soll das potentiell eingebettete *emb* nach außen gegeben werden.

Diese Schnittstelle von E und E^{-1} kann effizient genutzt werden: Wenn der Sender eine Nachricht geheim zu übertragen hat, dann bettet er sie sukzessive, d.h. wenn immer möglich, ein – sonst tut er nichts. Der Empfänger erhält folglich jede geheim übertragene Nachricht als kompakten Bitstring, d.h. ohne Unterbrechungen durch bedeutungslose Bitstrings. Hat der Sender nicht fortlaufend Nachrichten geheim zu übertragen, findet der Empfänger zwischen Nachrichten bedeutungslose Bitstrings.

Konkret betrachten wir im folgenden den Fall, daß *emb* jeweils ein Bit ist. Damit können sowohl Sender wie auch Empfänger effizient prüfen, ob jeweils *alle* möglichen Werte von *emb* eingebettet – und damit *alle* Nachrichten damit kompakt übertragen – werden können. Wie in §4.1.3.3.1 sei R ein Ratealgorithmus, der bei Eingabe von *stego* mit überwältigender Wahrscheinlichkeit *semi-richtig* rät, ob in *stego* etwas eingebettet ist oder nicht. E' sei der verbesserte Algorithmus „Einbetten“ und E'^{-1} seine Umkehrfunktion.

Die naheliegende Konstruktion (korrekt nur für Einbettungsposition unabhängig vom einzubettenden Bit):

Der Sender prüft, ob E(*cover*, 0) und E(*cover*, 1) jeweils **akzeptable**, d.h. aus seiner Sicht für einen Stegoanalytiker die Einbettung nicht verratende Werte für Stegotext sind und bettet nur dann ein, d.h. sendet nur dann *stego* := E(*cover*, *emb*) statt *cover*.

Eine Implementierung des verbesserten Einbettungsalgorithmusses E' mit möglichst wenig Aufrufen von E und R lautet:

```
function E'(in: cover, emb; out: stego);
stego := E(cover, emb);
stego_alternativ := E(cover, not(emb));
if R(stego) = etwas_verborgen or R(stego_alternativ) = etwas_verborgen
then stego := cover;
```

Der Empfänger prüft, ob das empfangene *stego* ein akzeptabler Wert ist[101] und berechnet ggf. *emb* := E^{-1}(*stego*). Nun muß der Empfänger noch prüfen, ob auch der andere mögliche Wert des Bits, not(*emb*), hätte eingebettet werden können. Leider kennt der Empfänger *cover* nicht und kann folglich nicht E(*cover*, not(*emb*)) berechnen. Stattdessen berechnet er E(*stego*, not(*emb*)). Gilt für das Stegosystem die Gleichung

---

[101] Wenn das Stegosystem gut ist, sollte dies *immer* der Fall sein – unabhängig davon, ob eingebettet wurde oder nicht.

$$E(cover, not(emb)) = E(stego, not(emb)),$$

welche durch Einsetzen von *stego* = E(*cover*, *emb*) äquivalent umgeformt werden kann in

$$E(cover, not(emb)) = E(E(cover, emb), not(emb))^{102},$$

dann kann der Empfänger so auch prüfen, ob beide möglichen Werte des Bits hätten eingebettet werden können und ob der Sender folglich einbetten konnte. Der Empfänger gibt das errechnete *emb* also nur nach außen, wenn E(*stego*, not(*emb*)) ein akzeptabler Wert für den Stegotext ist.

Eine effiziente Implementierung des verbesserten Extraktionsalgorithmusses E'^{-1} lautet:

```
function invE'(in: stego; out: emb);
emb := invE(stego);
stego_alternativ := E(stego, not(emb));
if R(stego_alternativ) = etwas_verborgen
then emb := leer;
```

Es stellt sich nun die Frage: Ist obiges Ziel für E und E^{-1} nur zu erreichen, wenn das Stegosystem die obigen Gleichungen erfüllt? Die Antwort ist ein konstruktives Nein.

Die ineffiziente Konstruktion:

Die Idee besteht darin, den Sender wie den Empfänger prüfen zu lassen, ob auch in *stego* beide möglichen Werte eines Bits eingebettet werden können. Nur dann gilt das Bit als übertragen. Im einzelnen:

Der Sender prüft, ob E(*cover*, 0) und E(*cover*, 1) jeweils **akzeptable**, d.h. aus seiner Sicht für einen Stegoanalytiker die Einbettung nicht verratende Werte für Stegotext sind und bettet nur dann ein, d.h. sendet nur dann *stego* := E(*cover*, *emb*) statt *cover*. Wenn sowohl E(*stego*, 0) wie auch E(*stego*, 1) akzeptable Werte sind, betrachtet der Sender *emb* als übertragen, ansonsten sendet er *emb* noch mal.

Eine Implementierung des verbesserten Einbettungsalgorithmusses E' mit möglichst wenig Aufrufen von E und R lautet:

```
function E'(in: cover, emb; out: stego);
stego := E(cover, emb);
stego_alternativ := E(cover, not(emb));
if R(stego) = etwas_verborgen or R(stego_alternativ) = etwas_verborgen
then begin stego := cover; E'(next cover, emb) end
else
if R(E(stego, 0)) = etwas_verborgen or R(E(stego, 1)) = etwas_verborgen
then E'(next cover, emb);
```

Der Empfänger erhält *stego'* (dies kann *stego* oder *cover* sein) und berechnet *emb'* := E^{-1}(*stego'*). Der Empfänger prüft, ob E(*stego'*, 0) und E(*stego'*, 1) jeweils akzeptable Werte für Stegotext sind und empfängt ggf. *emb'*, ansonsten nichts.

Warum funktioniert dies?

- Fall 1: Der Sender sendet *stego*, d.h. *stego'* = *stego*. Der Empfänger empfängt also *emb'* = *emb* genau dann, wenn der Sender *emb* als übertragen betrachtet.
- Fall 2: Der Sender sendet *cover*, d.h. *stego'* = *cover*. Dann ist entweder E(*cover*, 0) oder E(*cover*, 1) nicht akzeptabel – oder beide. Wegen *stego'* = *cover* gilt Gleiches also für die vom Empfänger getesteten Werte E(*stego'*, 0) und E(*stego'*, 1).

Eine effiziente Implementierung des verbesserten Extraktionsalgorithmusses E'^{-1} lautet:

---

102 Diese Gleichung besagt: Wenn nacheinander zwei inverse Bits eingebettet werden, dann überschreibt die zweite Einbettung die erste. D.h. die Einbettungsposition ist nicht vom Wert des Bits abhängig.

```
function invE'(in: stego; out: emb);
if R(E(stego, 0)) = etwas_verborgen or R(E(stego, 1)) = etwas_verborgen
then emb := leer else emb := invE(stego);
```

Diese ineffiziente Konstruktion läßt Sendemöglichkeiten aus, wo E(*cover*, *emb*) akzeptabel ist, E(*cover*, not(*emb*)) aber nicht. Die folgende effiziente Konstruktion ist also günstiger.

Die effiziente Konstruktion:

Der Sender prüft, ob E(*cover*, *emb*) **akzeptabel**, d.h. aus seiner Sicht für einen Stegoanalytiker die Einbettung nicht verratender Werte für Stegotext ist und bettet nur dann ein, d.h. sendet nur dann *stego* := E(*cover*, *emb*) statt *cover*. Wenn sowohl E(*stego*, 0) wie auch E(*stego*, 1) akzeptable Werte sind, betrachtet der Sender *emb* als übertragen, ansonsten sendet er *emb* noch mal.

Eine Implementierung des verbesserten Einbettungsalgorithmusses E' mit möglichst wenig Aufrufen von E und R lautet:

```
function E'(in: cover, emb; out: stego);
stego := E(cover, emb);
if R(stego) = etwas_verborgen
then begin stego := cover; E'(next cover, emb) end
else
if R(E(stego, 0)) = etwas_verborgen or R(E(stego, 1)) = etwas_verborgen
then E'(next cover, emb);
```

Der Empfänger erhält *stego'* (dies kann *stego* oder *cover* sein) und berechnet *emb'* := E^{-1}(*stego'*). Der Empfänger prüft, ob E(*stego'*, 0) und E(*stego'*, 1) jeweils akzeptable Werte für Stegotext sind und empfängt ggf. *emb'*, ansonsten nichts. (Dies ist exakt gleich wie oben, so daß die obige Implementierung übernommen werden kann.)

Warum funktioniert dies?

Fall 1: Der Sender sendet *stego*, d.h. *stego'* = *stego*. Der Empfänger empfängt also *emb'* = *emb* genau dann, wenn der Sender *emb* als übertragen betrachtet.

Fall 2: Der Sender sendet *cover*, d.h. *stego'* = *cover*. Dann ist entweder E(*cover*, 0) oder E(*cover*, 1) nicht akzeptabel (denn eins von beiden ist E(*cover*, *emb*), was der Sender als nicht akzeptabel verwirft) – oder beide. Der Empfänger empfängt also nichts, da wegen *stego'* = *cover* entweder E(*stego'*, 0) oder E(*stego'*, 1) (oder beide) ein nicht akzeptabler Werte für Stegotext sind.

4.2 Informationstheoretisch sichere Steganographie

4.2.1 Informationstheoretisch sichere steganographische Konzelation

KlPi_97

4.2.2 Informationstheoretisch sichere steganographische Authentikation

Arbeitspapier von Herbert Klimant und Rudi Piotraschke

4.3 Zwei gegensätzliche Stegoparadigmen

4.3.1 Möglichst wenig ändern

Least Significant Bit (LSB)

4.3.2 Normalen Prozeß nachbilden

[FrPf_98]

4.4 Paritätscodierung zur Verstärkung der Verborgenheit

Statt mit dem Stegoschlüssel die einzelnen Bits auszuwählen, an deren Position eingebettet wird, schlägt Ross Anderson vor, mit dem Stegoschlüssel Mengen von Bits auszuwählen und jedes einzubettende Bit jeweils als deren Parität zu codieren [Ande4_96, AnPe_98]. Dies gibt dem Einbettungsprozeß mehr Freiheit und verstärkt in diesem Sinne die Verborgenheit der Einbettung.

Je größer die Mengen von Bits sind, über die die Parität gebildet wird, desto gleichverteilter ist die Parität und desto unauffälliger kann eingebettet werden. Umgekehrt treffen dann Störungen bei der Übertragung des Stegotextes – seien sie zufällig oder auch zum Zwecke der Erschwerung von Steganographie bewußt herbeigeführt – mit immer größerer Wahrscheinlichkeit das Eingebettete.

4.5 Steganographie in digitalen Telefongesprächen

MöPS_94, FJMPS_96

4.6 Steganographie in Bildern

[FrPf_98]

4.7 Steganographie in einer Videokonferenz

West_97

Leseempfehlungen

Lehrbuchartige Artikelsammlung:

KaPe_00

Aktuelle Forschungsliteratur:

Tagungsserie Information Hiding, Tagungsbände erscheinen in der Serie LNCS des Springer-Verlags Heidelberg

Themenheft „Copyright and Privacy Protection“, IEEE Journal on Selected Areas in Communications 16/4 (Mai 1998)

Themenheft „Identification and Protection of Multimedia Information“, Proceedings of the IEEE 87/7 (Juli 1999)

5 Sicherheit in Kommunikationsnetzen

5.1 Problemstellung

5.1.1 Gesellschaftliche Problemstellung

Bei allen Kommunikationsnetzen sollte man sich fragen, wie gut ihre Teilnehmer vor Schaden geschützt sind. Die resultierenden Schutzziele (und Schutzmechanismen) lassen sich grob in zwei Kategorien einteilen (vgl. §1.2.4).

Erwünschtes leisten, d.h. erwünschte Aktionen des Systems sollen stattfinden: Ein Teilnehmer kann geschädigt werden, indem eine Diensterbringung für ihn verhindert, verzögert oder verändert wird, oder indem eine Kommunikationsbeziehung unter seinem Namen oder auf seine Kosten, jedoch ohne sein Wissen oder seine Billigung aufgebaut wird. Ebenso kann es ihm schaden, wenn er im Konfliktfall nicht beweisen kann, eine bestimmte Nachricht gesendet oder empfangen zu haben. Dies sind – in anderen Worten – die in §1.3 unter *Integrität* und *Verfügbarkeit* aufgeführten Schutzziele.

Damit dieser Schaden tatsächlich entsteht, sind *verändernde Angriffe* nötig. Diese können prinzipiell im IT-System erkannt werden, wie bereits in §1.2.5 angemerkt wurde. Oder anders herum gesagt: Daß das Erwünschte geleistet wurde, ist auch im nachhinein überprüfbar.

Unerwünschtes verhindern, d.h. unerwünschte Aktionen des Systems sollen verhindert werden: Ein Teilnehmer kann auch durch Beobachten seiner Kommunikation Schaden erleiden. Die Beobachtung kann sich hierbei sowohl auf die *Kommunikationsinhalte* als auch auf die *Kommunikationsumstände* (mit wem, wann, von wo?) beziehen. Dies wurde in §1.3 unter dem Schutzziel *Vertraulichkeit* bereits formuliert.

Damit möglicher Schaden tatsächlich entsteht, sind nur *beobachtende Angriffe* nötig. Diese können im IT-System prinzipiell nicht erkannt werden, wie bereits in §1.2.5 angemerkt wurde. Die Einhaltung des Schutzzieles Vertraulichkeit ist hier also im Nachhinein nicht überprüfbar – also sind vorbeugende Maßnahmen unumgänglich.

Die für „Erwünschtes leisten“ notwendigen Maßnahmen sind ausführlich in [VoKe_83] beschrieben und von denjenigen zu „Unerwünschtes verhindern“ größtenteils unabhängig. Sie sollen im folgenden nur kurz betrachtet werden.

Somit bleibt zunächst die Frage zu beantworten, welche Auswirkung die oben geschilderte Entwicklung der Netze auf „Unerwünschtes verhindern“ haben wird.

In Bild 5-1 ist die Endsituation dieser Entwicklung dargestellt: Alle Dienste, z.B. Fernsehen, Radio, Telefon, Bildschirmtext, werden über eine Glasfaser von der Vermittlungszentrale zum Netzabschluß eines Teilnehmers vermittelt.

Die Glasfaser ist diesem Teilnehmer (bzw. seiner Familie, seiner Firma o.ä.) eindeutig zugeordnet, und über sie wird, da es sich um ein Vermittlungsnetz handelt, nur übertragen, was von ihm oder speziell für ihn bestimmt ist. Folglich stellt die physikalische Netzadresse eine Art Personenkennzeichen (Kennzeichen für natürliche oder juristische Personen) dar, unter dem Daten über diesen Teilnehmer gesammelt werden können.

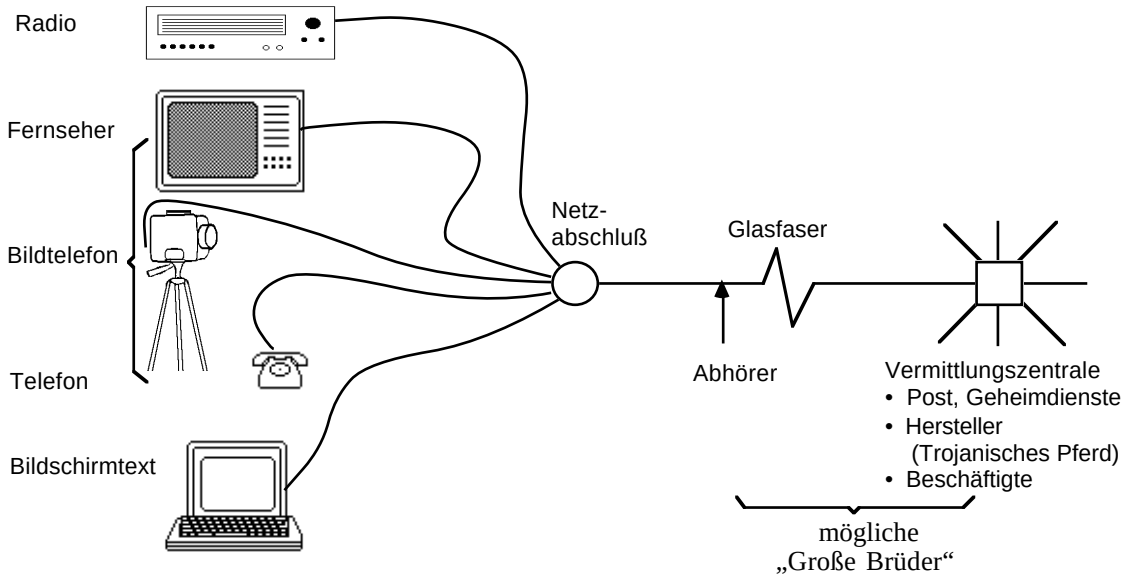


Bild 5-1: Beobachtbarkeit des Benutzers im sternförmigen, alle Dienste vermittelnden IBFN

Die dafür auf der Glasfaser und in den Vermittlungszentralen anfallenden Informationen bestehen *technisch* gesehen aus

- den transportierten **Nutzdaten** (Bild, Ton, Text) und
- den **Vermittlungsdaten** (Adressen und Absender der Kommunikationspartner – bei mobilen Teilnehmern noch zusätzlich deren momentaner Ort –, Datenumfang, Dienstart, Zeit).

Die daraus zu gewinnenden personenbezogenen Daten des Teilnehmers kann man *inhaltlich* gesehen einteilen in:

- **Inhaltsdaten**, d.h. Inhalte vertraulicher (persönlicher oder geschäftlicher) Nachrichten, z.B. von Telefongesprächen oder elektronischer Post.
- **Interessensdaten**, d.h. Informationen über das Interesse des Teilnehmers an Nachrichten, deren Inhalt nicht vertraulich ist. Hierzu zählen die Beobachtungen, welche Zeitungsartikel sich der Teilnehmer schicken läßt, welche Auskünfte er aus Datenbanken, z.B. Bildschirmtext, einholt und was genau er im Fernsehen sieht bzw. im Radio hört:

  Will der Teilnehmer z.B. das Fernsehprogramm wechseln, teilt er dies über seinen Fernseher und seinen Netzabschluß der Vermittlungszentrale mit. Diese überträgt dann statt des bisher gesehenen das angeforderte Fernsehprogramm über die Glasfaser.

  Interessensdaten charakterisieren nicht nur natürliche Personen, sondern auch Firmen:

  Welche Fachaufsätze und Patente eine Firma anfordert, gibt z.B. Hinweise auf die von ihr gerade durchgeführten Forschungs- und Entwicklungsarbeiten.

  Interessensdaten waren vor der Einrichtung öffentlich zugänglicher Datenbanken, insbesondere Bildschirmtext, in Kommunikationsnetzen überhaupt nicht zu gewinnen.
- **Verkehrsdaten**, also z.B. wann der Teilnehmer wie lange von wo mit wem kommuniziert. Diese können aus den Vermittlungsdaten gewonnen werden.
  Bei natürlichen Personen ergeben die Verkehrsdaten bereits interessante Bausteine für ein Persönlichkeitsbild, z.B. Konsumgewohnheiten, Freundeskreis, Tagesablauf, Kontakte mit Polizei und Gesundheitsamt.

Verkehrsdaten können aber nicht nur Persönlichkeitsrechte natürlicher Personen, sondern auch Geschäftsinteressen juristischer Personen verletzen. Steigt z.B. das Verkehrsaufkommen zwischen zwei konkurrierenden Firmen sprunghaft an, so legt dies Vermutungen über eine gemeinsame Produktentwicklung nahe.

Eine Möglichkeit, an diese Daten zu gelangen, ist das **Abhören** der den Teilnehmer mit seiner Ortsvermittlungsstelle verbindenden **Leitung** (bzw. bei mehreren dienstespezifischen Netzen: Leitungen) oder von Leitungen zwischen Vermittlungsstellen. Letzteres erlaubt zwar bei geeigneter Gestaltung des Protokolls zwischen Ortsvermittlungsstelle und „Lieferanten“ von Radio- und Fernsehprogrammen nicht die Überwachung der Massenkommunikation [Kais_82] (Radio- und Fernsehempfang) sowie keine Überwachung von Individualkommunikation zwischen an dieselbe Ortsvermittlungsstelle angeschlossenen Teilnehmern. Es erlaubt aber die gleichzeitige Überwachung vieler Teilnehmer bezüglich der über den Bereich ihrer Ortsvermittlungsstelle hinausgehenden Individualkommunikation.

5.1.2 Informatische Problemstellung und Lösungsskizzen

Die Schutzziele aus §1.3, bzgl. Vertraulichkeit um einen weiteren Punkt ergänzt sowie zur leichteren Bezugnahme mit Abkürzungen versehen, lauten also:

Unerwünschtes verhindern: Unerwünschte Aktionen des Systems sollen verhindert werden.

Schutzziel Vertraulichkeit (confidentiality)

- c1 *Nachrichteninhalte* sollen vor allen Instanzen außer dem Kommunikationspartner vertraulich bleiben.
- c2 *Sender* und/oder *Empfänger* von Nachrichten sollen voreinander *anonym* bleiben können, und *Unbeteiligte* (inkl. Netzbetreiber) sollen *nicht in der Lage* sein, sie *zu beobachten*.
- c3 Weder potentielle Kommunikationspartner noch Unbeteiligte (inkl. Netzbetreiber) sollen ohne Einwilligung den *momentanen Ort* einer mobilen Teilnehmerstation bzw. des sie benutzenden Teilnehmers ermitteln können.

Erwünschtes leisten: Erwünschte Aktionen des Systems sollen stattfinden.

Schutzziel Integrität (integrity)

- i1 Fälschungen von *Nachrichteninhalten* (inkl. des *Absenders*) sollen erkannt werden.
- i2 Der Empfänger soll gegenüber Dritten *nachweisen* können, daß Instanz *x* die Nachricht *y gesendet hat* (und ggf. auch den Zeitpunkt, an dem dies geschah).
- i3 Der Absender soll das *Absenden* einer Nachricht mit korrektem Inhalt *beweisen* können, möglichst sogar den Empfang der Nachricht (und ggf. auch den Zeitpunkt, an dem dies geschah).
- i4 Niemand kann dem Netzbetreiber *Entgelte* für erbrachte Dienstleistungen vorenthalten[103]. Umgekehrt kann der Netzbetreiber nur für korrekt erbrachte Dienstleistungen Entgelte fordern.

Schutzziel Verfügbarkeit (availability)

- a1 Das Netz ermöglicht Kommunikation zwischen allen Partnern, die dies *wünschen* (und denen es nicht verboten ist).

Um einen Überblick zu geben und das Leichte gleich abzuhandeln, seien diesen Schutzzielen die **zugehörigen Schutzmechanismen** gegenübergestellt:

---

103 Diese Forderung ist stärker als das, was ein Kommunikationsnetz unbedingt leisten muß. Es würde reichen, daß der Netzbetreiber die korrekte Erbringung von Dienstleistungen *beweisen* kann. Ob und wie unter Vorlage dieser Beweise dann Geldflüsse außerhalb des Kommunikationsnetzes erzwungen und realisiert werden, müßte hier eigentlich nicht betrachtet werden. Da aber die Vertraulichkeitsforderung c2 das Eintreiben von Entgeltforderungen zu erschweren scheint, wird hier die stärkere Forderung erhoben. In §6 wird dann gezeigt, wie auch sie erfüllt werden kann.

Unerwünschtes verhindern

Schutzmechanismen für Vertraulichkeit

c1 Verwendung eines oder mehrerer Konzelationssysteme zwischen Sender und Kommunikationspartner: Ende-zu-Ende-Verschlüsselung, vgl. §5.2.1.2.

c2 Verfahren innerhalb des Kommunikationsnetzes zum Schutz der Verkehrs- und Interessensdaten, §5.3 bis §5.6.1.

c3 Verfahren innerhalb des Kommunikationsnetzes zum Schutz des Mobilitätsverhaltens von Teilnehmer(statione)n, §5.7.

Erwünschtes leisten

Schutzmechanismen für Integrität

i1 Verwendung eines oder mehrerer Authentikationssysteme.

i2 Nachrichten werden vom Urheber digital signiert. Testschlüssel sind öffentlich bekannt. Empfänger prüft Signaturen, bevor er Nachrichten akzeptiert.

Soll sichergestellt werden, daß Signaturen auch dann gültig bleiben, wenn der geheime Signierschlüssel des Teilnehmers bekannt wird (was dieser, um der Verbindlichkeit seiner Signaturen zu entgehen, ggf. auch selbst herbeiführen könnte), sollten Signaturen von einer oder mehreren zusätzlichen Instanzen (**elektronische Notare**) mit einem Zeitstempel versehen und mit deren Signierschlüssel nochmals signiert, d.h. notariell beurkundet, werden. Dann kann man festlegen, daß durch das Bekanntwerden des geheimen Signierschlüssels eines Teilnehmers die Gültigkeit seiner alten Signaturen unberührt bleibt und die elektronischen Notare lediglich zukünftig keine mit diesem Signierschlüssel geleisteten Signaturen mehr beurkunden.

Was zu tun ist, wenn das zukünftige Brechen des bisher verwendeten Signatursystems absehbar wird, wird in Aufgabe 3-11 b) diskutiert.

Unabhängig vom Schutz der Gültigkeit digitaler Signaturen sind Zeitstempel natürlich auch geeignet nachzuweisen, daß eine Nachricht spätestens zum Zeitpunkt des Zeitstempels gesendet wurde.

i3 Das Kommunikationsnetz (und bei Kooperation auch der Kommunikationspartner) erteilt digital signierte Empfangsquittungen für zum Transport übergebene (erhaltene) Nachrichten. Ggf. müssen die signierten Empfangsquittungen Zeitstempel enthalten, damit Zeitpunkte nachgewiesen werden können.

i4 Mittels eines digitalen Zahlungssystems (§6) werden Entgelte während der Diensterbringung bezahlt.

Schutzmechanismen für Verfügbarkeit

a1 Diversitäre Netze, d.h. unterschiedliche Übertragungs- und Vermittlungssysteme; faire Aufteilung von Betriebsmitteln.

Das **Ziel** der zur Verhinderung von Unerwünschtem zu entwickelnden Verfahren ist es, einem *Angreifer* das Erfassen sensitiver Daten unmöglich zu machen: Die Kommunikation sollte gegenüber Unbeteiligten weitgehend *unbeobachtbar* und gegenüber Beteiligten (z.B. Kommunikationspartnern) üblicherweise *anonym* erfolgen. Damit sich nicht doch nach und nach unspezifiziertes Wissen über Personen ansammeln kann, sollten einzelne Verkehrsereignisse auch durch Beteiligte üblicherweise *unverkettbar* sein.

Sofern durch die Verfahren zur Verhinderung von Unerwünschtem das Leisten des Erwünschten (vgl. §5.1.1) erschwert wird, ist zu zeigen, wie auch dies weiterhin gewährleistet werden kann.

Wegen ihrer großen Bedeutung für das Verständnis möchte ich die gerade nebenbei eingeführten Begriffe etwas genauer erläutern. Die dabei verwendeten, hier teilweise noch undefinierten Sachverhalte werden bei der Anwendung der Begriffe jeweils konkretisiert.

Da Schutz vor einem allmächtigen Angreifer, der alle Leitungen, alle Vermittlungszentralen, alle Teilnehmerstationen außer der des Angegriffenen und den Kommunikationspartner kontrolliert, durch Maßnahmen innerhalb des Kommunikationsnetzes nicht möglich ist, sind alle folgenden Maßnahmen nur Annäherungen an den perfekten Schutz der Teilnehmer vor jedem möglichen Angreifer. Die Annäherung wird im allgemeinen durch Angabe der **Stärke eines Angreifers** in der Form eines **Angreifermodells** bestimmt: *wie weit verbreitet* ist der Angreifer (wie viele und welche Leitungen und/oder Stationen kann er maximal beobachten und/oder gar aktiv verändernd kontrollieren) und mit *wieviel Rechenkapazität ausgestattet* ist er (vgl. §1.2.5 und §3)?

(Um Mißverständnissen bei den folgenden Diskussionen der Stärke von Schutzmaßnahmen vorzubeugen sei hier ein für alle mal darauf hingewiesen, daß das In-Betracht-Ziehen einer Organisation oder Personengruppe als Angreifer ausschließlich der technisch-naturwissenschaftlichen Klärung der Stärke der Schutzmaßnahme dient und alle Organisationen lediglich als Rollenträger bezüglich des Kommunikationsnetzes betrachtet werden. Es wird im folgenden also lediglich diskutiert, ob eine Organisation oder Personengruppe mit Erfolg angreifen *könnte*, nicht ob sie dies tat, tut oder tun wird!)

Ein **Ereignis** *E* (z.B. Senden einer Nachricht, Abwickeln eines Geschäftes) heißt **unbeobachtbar** bezüglich eines Angreifers *A*, wenn die Wahrscheinlichkeit des Auftretens von *E* nach jeder für *A* möglichen Beobachtung *B* sowohl echt größer 0 als auch echt kleiner 1 ist. Dies kann natürlich nur für an *E unbeteiligte* Angreifer der Fall sein und lautet in der üblichen wahrscheinlichkeitstheoretischen Schreibweise: Für *A* gilt für alle *B*: $0 < W(E|B) < 1$.

Das Ereignis *E* heißt *perfekt* unbeobachtbar bezüglich eines Angreifers *A*, wenn die Wahrscheinlichkeit des Auftretens von *E* vor und nach jeder für *A* möglichen Beobachtung *B* gleich ist, d.h. für *A* gilt für alle *B*: $W(E) = W(E|B)$. Dies bedeutet nach [Sha1_49], daß die Beobachtung *A* keinerlei zusätzliche Information über *E* liefern kann.

Eine **Instanz** (z.B. Person) heißt in einer Rolle *R* **anonym** bezüglich eines Ereignisses *E* (z.B. als Sender einer Nachricht, Käufer in einem Geschäft) und eines Angreifers *A*, wenn für jede mit *A* nicht kooperierende Instanz die Wahrscheinlichkeit, daß sie bei *E* die Rolle *R* wahrnimmt, nach jeder für *A* möglichen Beobachtung sowohl echt größer 0 als auch echt kleiner 1 ist. Dies kann auch für an *E beteiligte* Angreifer (z.B. den Empfänger der Nachricht bzw. den Verkäufer) der Fall sein.

Eine Instanz heißt in einer Rolle *R* bezüglich eines Ereignisses *E* und eines Angreifers *A perfekt* anonym, wenn für jede mit *A* nicht kooperierende Instanz die Wahrscheinlichkeit, daß sie bei *E* die Rolle *R* wahrnimmt, vor und nach jeder für *A* möglichen Beobachtung gleich ist. Letzteres bedeutet nach [Sha1_49], daß die Beobachtung dem Angreifer keinerlei zusätzliche Information darüber liefert, wer bei *E* die Rolle *R* wahrnimmt.

Zwei **Ereignisse** *E* und *F* heißen bezüglich eines Merkmals *M* (z.B. zwei Nachrichten bezüglich ihres Senders oder bezüglich der Transaktion, zu der sie gehören) und bezüglich eines Angreifers *A* **unverkettbar**, wenn die Wahrscheinlichkeit, daß sie in *M* übereinstimmen, nach jeder für *A* möglichen Beobachtung sowohl echt größer 0 als auch echt kleiner 1 ist. Dies kann auch für an *E beteiligte* Angreifer der Fall sein.

Zwei Ereignisse *E* und *F* heißen bezüglich eines Merkmals *M* und bezüglich eines Angreifers *A perfekt* unverkettbar, wenn die Wahrscheinlichkeit, daß sie in *M* übereinstimmen, vor und nach jeder für *A* möglichen Beobachtung gleich ist. Letzteres bedeutet nach [Sha1_49], daß die Beobachtung dem Angreifer keinerlei zusätzliche Information darüber liefert, ob diese Ereignisse in *M* übereinstimmen.

Von den drei Definitionen der Unbeobachtbarkeit, Anonymität und Unverkettbarkeit ist die Unverkettbarkeit die allgemeinste (und damit tiefliegendste):

Unbeobachtbarkeit von Ereignissen kann als Unverkettbarkeit von Beobachtungen und sich dahinter verbergenden Ereignissen aufgefaßt werden.

Anonymität kann als Unverkettbarkeit zwischen Instanzen und Ereignissen aufgefaßt werden.

Die **Vertrauenswürdigkeit der Unverkettbarkeit, Unbeobachtbarkeit bzw. Anonymität** wird dadurch bestimmt, ein wie *starker* Angreifer den vorherigen Definitionen unterlegt wird.

Unbeobachtbarkeit und Anonymität kann man zusätzlich noch dadurch parametrisieren, daß man sie nur innerhalb gewisser Klassen von Ereignissen (z.B. Nachrichtentypen) bzw. Instanzen (z.B. denen eines bestimmten Teilnetzes eines größeren Kommunikationsnetzes, vgl. §5.5) verlangt. In den obigen Definitionen wurden keinerlei Klassen definiert, sie definieren daher den maximal erreichbaren **Grad an Unbeobachtbarkeit bzw. Anonymität**. Mit einer **Klasseneinteilung** parametrisierte Definitionen ergeben sich kanonischerweise. Als Beispiel wird die für unbeobachtbar angegeben:

Ein **Ereignis** *E* heißt **unbeobachtbar** (bzw. *perfekt* unbeobachtbar) bezüglich eines Angreifers *A* und einer gegebenen Klasseneinteilung von Ereignissen, wenn die bedingte Wahrscheinlichkeit des Auftretens von *E*, gegeben daß ein Ereignis seiner Klasse auftritt, nach jeder für *A* möglichen Beobachtung *B* sowohl echt größer 0 als auch echt kleiner 1 (bzw. vor und nach den Beobachtungen gleich) ist.

Beides kann natürlich nur für an *E unbeteiligte* Angreifer der Fall sein. Es ist sowohl im Falle der Unbeobachtbarkeit als auch im Falle der perfekten Unbeobachtbarkeit möglich, daß sich für den Angreifer durch die Beobachtung die Wahrscheinlichkeiten des Auftretens von Ereignissen aus bestimmten Klassen ändern. Der Angreifer gewinnt also möglicherweise Information über diese Klassen, im zweiten Fall aber nicht über einzelne Ereignisse innerhalb einzelner Klassen.

5.2 Einsatz und Grenzen von Verschlüsselung in Kommunikationsnetzen

5.2.1 Einsatz von Verschlüsselung in Kommunikationsnetzen

Für den Einsatz eines kryptographischen Systems zum Schutz der Kommunikation (vor allem zum Zweck der Konzelation, aber auch zum Zweck der Integrität oder Authentifikation) hat man zwei Strategien zur Auswahl, die leider beide Nachteile haben: Verbindungs-Verschlüsselung und Ende-zu-Ende-Verschlüsselung.

5.2.1.1 Verbindungs-Verschlüsselung

Die erste Strategie besteht darin, *alle* Daten jeweils zwischen benachbarten Netzknoten, d.h. Teilnehmerstationen und Vermittlungszentralen, zu verschlüsseln (**Verbindungs-Verschlüsselung**, link-by-link encryption) [Bara_64, Denn_82, VoKe_83, DaPr_89].

Es sollte ein *gleichmäßiger Zeichenstrom* übertragen werden, damit ein Abhörer nicht beobachten kann, wann keine Nachrichten übertragen werden. Ein gleichmäßiger Zeichenstrom ist bei allen Übertragungsstrecken, die den verbundenen Netzknoten *statisch* zugeordnet sind, z.B. Punkt-zu-Punkt-Leitungen und Richtfunkstrecken, ohne Mehraufwand möglich.

Aus den in §3 dargelegten Gründen, sollte und kann eine sichere *Stromchiffre* verwendet werden: Würde eine Blockchiffre verwendet, so könnte, solange der Schlüssel nicht gewechselt würde, der Abhörer zumindest manchmal beobachten, daß sich gewisse Nachrichten(fragmente) wiederholen. Der Abhörer könnte also manche Nachrichten *verketten* (vgl. §5.1.2).

Wird ein gleichmäßiger, mit einer sicheren Stromchiffre verschlüsselter Zeichenstrom übertragen, erhält ein Angreifer durch Abhören der Übertragungsstrecken, d.h. von Leitungen (Glasfasern, Koaxialkabel, Kupferdoppeladern), Richtfunk- oder Satellitenstrecken keine Information mehr. Ob und ggf. welche Nachrichten übertragen werden, ist für ihn *perfekt unbeobachtbar* (vgl. §5.1.2).

Die Nachteile von Verbindungs-Verschlüsselung sind:

- In den Vermittlungszentralen liegen alle Daten unverschlüsselt vor. Von den möglichen Angreifern werden also nur diejenigen abgewehrt, die Übertragungsstrecken abhören.
- In der in Bild 5-2 gezeigten Endsituation der geplanten Entwicklung der Kommunikationsnetze werden auf der den Netzabschluß des Teilnehmers mit der Vermittlungszentrale verbindenden Leitung, einer Glasfaser, mindestens 560 Mbit/s übertragen. Dies ist eher oberhalb dessen, was heute mit Kryptogeräten, die auf einem für halbwegs sicher gehaltenen kryptographischen System beruhen und halbwegs preiswert sind, verschlüsselt werden kann, vgl. §3. Es wäre zumindest von Vorteil, wenn man Fernsehen, insbesondere hochauflösendes Fernsehen (High Definition TV, HDTV), als breitbandigen Dienst, bei dem es nicht um den Schutz von Inhalts-, sondern von Interessensdaten geht, nicht verschlüsseln müßte. Wieviel hierdurch eingespart werden kann, verdeutlichen die folgenden Zahlen: für Fernsehen heutiger Qualität (PAL) werden ohne bzw. mit Redundanzreduktion etwa 140 Mbit/s bzw. 34 Mbit/s benötigt, für hochauflösendes Fernsehen etwa viermal soviel.

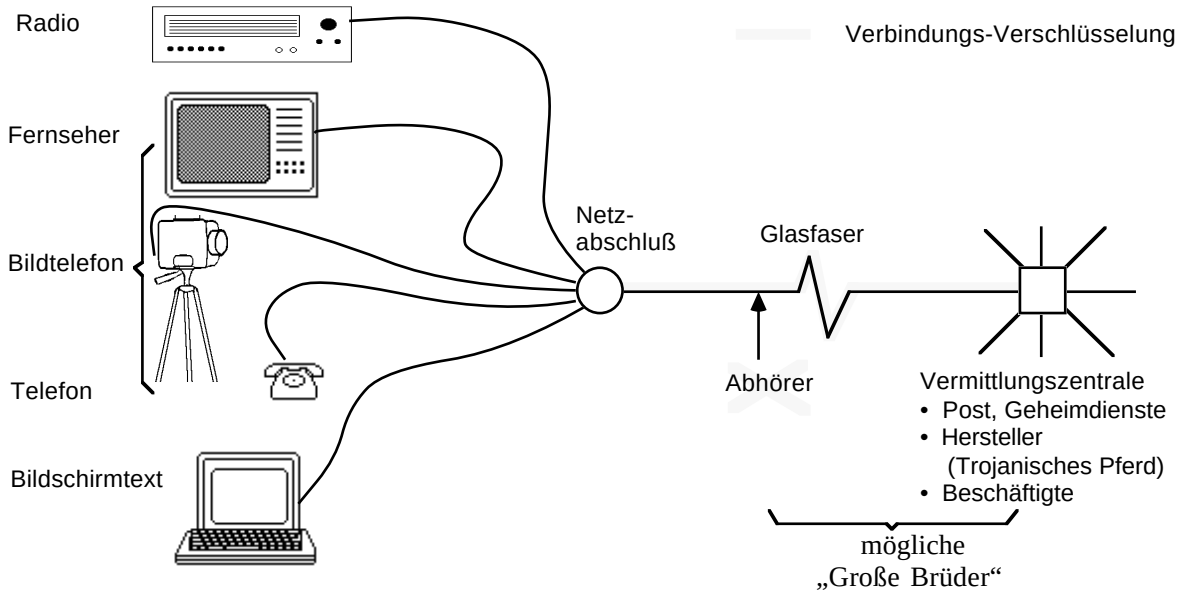


Bild 5-2: Verbindungs-Verschlüsselung zwischen Netzabschluß und Vermittlungszentrale

In jedem Fall benötigen benachbarte Netzknoten jeweils zueinander passende und leistungsfähige, also direkt in Hardware implementierte kryptographische Systeme, zweckmäßigerweise jeweils selbstsynchronisierende Stromchiffren. Da die Nachbarschaft von Netzknoten vergleichsweise statisch ist, können Schlüssel einer Verbindung zugeordnet werden, so daß ein symmetrisches kryptographisches System den Anforderungen vollauf genügt.

5.2.1.2 Ende-zu-Ende-Verschlüsselung

Die zweite Strategie ist, die Daten zwischen Teilnehmerstationen verschlüsselt zu übertragen (**Ende-zu-Ende-Verschlüsselung**, end-to-end encryption) [Bara_64, Denn_82, VoKe_83, DaPr_89],

damit sie in der (bzw. bei erweiterter Betrachtung: den) Vermittlungszentrale(n) nicht interpretiert werden können (Bild 5-3).

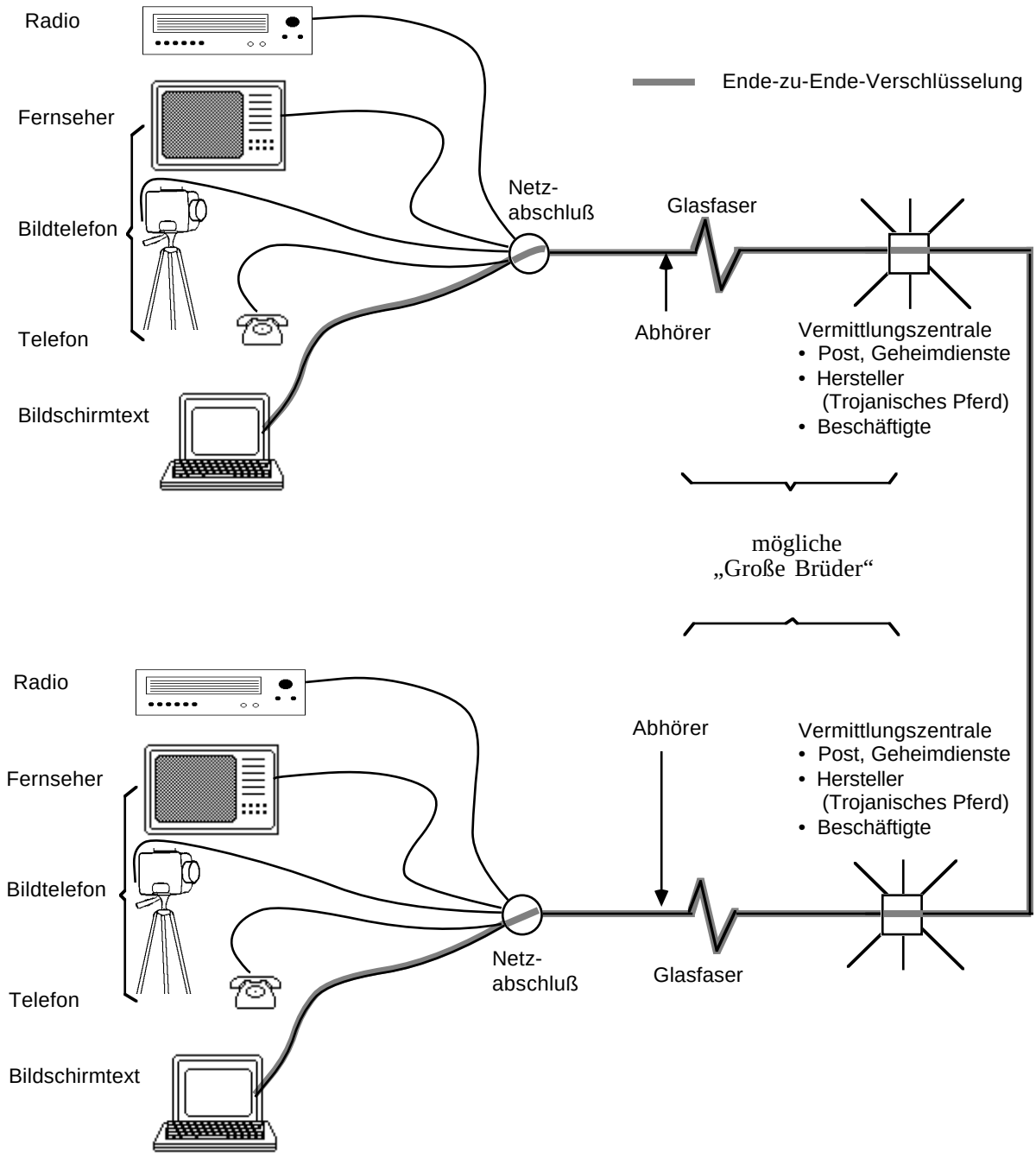


Bild 5-3: Ende-zu-Ende-Verschlüsselung zwischen Teilnehmerstationen

Aus den schon bei Verbindungs-Verschlüsselung dargelegten Gründen sollte auch für Ende-zu-Ende-Verschlüsselung möglichst ein gleichmäßiger Zeichenstrom (zumindest bei Kanalvermittlung für die Dauer des Kanals) und in jedem Fall eine Stromchiffre verwendet werden.

Die Nachteile von Ende-zu-Ende-Verschlüsselung sind:

– Durch Ende-zu-Ende-Verschlüsselung können nur die Nutzdaten, nicht die Vermittlungsdaten und damit auch nicht die *Verkehrsdaten* geschützt werden.

– Vor jemandem, der die Nutzdaten schon vorher kannte und nun die Vermittlungsdaten erhält, sind damit auch die *Interessensdaten* (vgl. §5.1.1) ungeschützt, was zu weiteren Verkettungen von Verkehrsdaten und dann wiederum zur Gewinnung von weiteren Interessensdaten usw. verwendet werden kann.

  Insbesondere kann dieses Verfahren also nicht als Schutz vor dem Netzbetreiber und anderen, die Zugriff auf die Rechner des Netzbetreibers haben, dienen, wenn diese gleichzeitig Kommunikationspartner sind oder wenn sich die Nutzdaten in Form einer Datenbank in einem Rechner des Netzbetreibers (z.B. Bildschirmtext-Zentrale in Ulm) befinden. Die Nutzdaten können zwar in verschlüsselter Form in der Datenbank abgespeichert werden. Dies nützt jedoch nur etwas, wenn der Angreifer (Netzbetreiber bzw. andere mit Zugriff auf die Rechner des Netzbetreibers) die zugehörigen Schlüssel nicht kennt und auch nicht in Erfahrung bringen kann. Letzteres erscheint außer bei kleinen geschlossenen Benutzergruppen unrealistisch, da der Angreifer in große geschlossene Benutzergruppen einen Strohmann einschleusen und bei offenen Benutzergruppen als normaler Nutzdateninteressent auftreten kann.
  Daneben ist auch eine Zusammenarbeit von jemandem, der an Daten in der Vermittlungszentrale gelangen kann, und dem Kommunikationspartner denkbar. Dies könnte z.B. von einem Geheimdienst ausgehen, der über den Netzbetreiber die Vermittlungsdaten erhält und Kommunikationspartner, z.B. Datenbanken, Zeitungsverlage, veranlaßt, ihm die Nutzdaten offenzulegen, aber auch vom Kommunikationspartner, der über Mitarbeiter o.ä. Zugang zu den Vermittlungsdaten erhält.

  Außerdem ist es trivialerweise sinnlos, sich mit Ende-zu-Ende-Verschlüsselung vor Kommunikationspartnern schützen zu wollen.

– Wie bei Verbindungs-Verschlüsselung ist es auch hier recht umständlich, auch Fernsehen, insbesondere hochauflösendes Fernsehen, verschlüsseln zu müssen, um die Interessensdaten vor den Angreifern in den Vermittlungszentralen zu schützen.

Soll Ende-zu-Ende-Verschlüsselung zwischen beliebigen Teilnehmerstationen möglich sein, benötigen sie jeweils paarweise zueinander passende kryptographische Systeme. Soll Ende-zu-Ende-Verschlüsselung auch für breitbandigere Kommunikation möglich sein, muß zumindest ein kryptographisches System direkt in Hardware implementiert sein (vgl. §3). Wegen der dynamischen Natur der Kommunikationsbeziehungen zwischen Teilnehmern kann man diesen Beziehungen nicht statisch Schlüssel zuordnen, da es bei großen Teilnehmerzahlen zu viele potentielle Beziehungen, nämlich $n \cdot (n-1)/2$ bei n Teilnehmern, gibt. Aus all diesen Gründen ist eine Standardisierung, oder besser noch eine formelle Normung, eines asymmetrischen kryptographischen Systems zum Schlüsselaustausch und zur Authentifikation sowie eines schnellen symmetrischen Konzelationssystems zur Ver- und Entschlüsselung der Nutzdaten dringend notwendig, soll das Kommunikationsnetz bezüglich Datenschutz (oder auch Rechtssicherheit, vgl. [PWP_90, WaPP_87]) ein **offenes** System bilden, wie dies insbesondere für das ISDN geplant ist. Denn nur auf der Basis von Standards oder Normen können Implementierungen von kryptographischen Systemen so gestaltet werden, daß sie effizient sind und jeweils *beliebige* Paare zueinander passen.

Ohne Standardisierung und Normung von geeigneten kryptographischen Systemen ist Vertraulichkeit (ebenso wie Integrität) nur innerhalb **geschlossener** Benutzergruppen, die sich jeweils auf ein kryptographisches System geeinigt und eine genügend leistungsfähige Implementierung haben, erreichbar. An dieser Stelle sei noch einmal unterstrichen, welch verheerende Folgen die Pläne der

NSA[104] (bzw. ZSI[105]) zur Verwendung geheimer Konzelationssysteme im öffentlichen Bereich haben können – und sei es „nur“ eine langjährige Verzögerung der Einführung standardisierter oder genormter, öffentlich validierter Konzelationssysteme und der entsprechenden Implementierungen.

5.2.2 Grenzen von Verschlüsselung in Kommunikationsnetzen

Selbst wenn, wie in Bild 5-4 gezeigt, Verbindungs- und Ende-zu-Ende-Verschlüsselung eingesetzt werden [Bara_64, Denn_82, VoKe_83, DaPr_89], bleiben im wesentlichen alle in §5.2.1.2 für Ende-zu-Ende-Verschlüsselung aufgezählten Nachteile erhalten. Lediglich der erste Nachteil wird in der Form eingeschränkt, daß (wie bei Verbindungs-Verschlüsselung allein) Abhörer der Übertragungsstrecken bei Übertragung eines gleichmäßigen, mit einer sicheren Stromchiffre verschlüsselten Zeichenstroms bei der Verbindungs-Verschlüsselung keine Information mehr erhalten.

Auch die bekannten kryptographischen Techniken erlauben also auf der heute üblichen und auch für die Zukunft geplanten, für Kommunikation in beiden Richtungen vorgesehenen Netzstruktur, nämlich der reiner Vermittlungsnetze, keinen ausreichenden und mit vernünftigem Aufwand **überprüfbaren** Schutz für Verkehrs- und Interessensdaten vor vielen möglichen Angreifern in den Vermittlungszentralen und Kommunikationspartnern. Dies ist bei dem als reines Vermittlungsnetz geplanten, dienstintegrierenden Kommunikationsnetz eine besonders schwerwiegende und bei Realisierung der Planung langandauernde Beeinträchtigung des Rechtes auf informationelle Selbstbestimmung.

Deshalb wird im folgenden untersucht, wie der noch fehlende Schutz durch andere Maßnahmen erreicht werden kann. Dabei muß darauf geachtet werden, daß nicht jemand, der bisher als möglicher Angreifer gar nicht auftrat, z.B. Nachbarn, plötzlich Beobachtungsmöglichkeiten erhält.

Technisch gesehen ist dabei das Hauptziel, die Verkehrsdaten vor dem Betreiber der Vermittlungseinrichtungen zu schützen und sich auch gegenüber dem Kommunikationspartner nicht identifizieren zu müssen. Damit sind Verkehrs- und Interessensdaten nicht nur vor diesen geschützt, sondern (und das ohne zusätzliche Verschlüsselung von nicht vertraulichen Nutzdaten wie Fernsehen) erst recht vor anderen Angreifern in den Vermittlungseinrichtungen oder Abhörern, da alle diese höchstens genausoviel Information erhalten können wie der Betreiber selbst. Der Schutz der Inhaltsdaten wird dann zusätzlich durch Ende-zu-Ende-Verschlüsselung sensitiver Nutzdaten erreicht. Obwohl also durch Verschlüsselung allein die Schutzziele im Bereich Vertraulichkeit in einem Kommunikationsnetz nicht erreicht werden können, so bildet Verschlüsselung doch die Basis vieler der im folgenden noch zu beschreibenden technischen Schutzmaßnahmen für Vertraulichkeit.

In den folgenden zwei Abschnitten werden grundlegende Verfahren zum Schutz von Verkehrs- und Interessensdaten vor Angreifern in Vermittlungszentralen und Kommunikationspartnern beschrieben und wo angebracht, wird ihre Wirksamkeit bewiesen:

In §5.3 werden Schutzmaßnahmen beschrieben, die *außerhalb* des Kommunikationsnetzes angesiedelt sind, d.h. die jeder Benutzer für sich trifft. Diese werden sich als nicht ausreichend erweisen.

Danach werden in §5.4 solche Schutzmaßnahmen beschrieben, die *innerhalb* des Kommunikationsnetzes angesiedelt sind, d.h. die Benutzung des Netzes nicht verändern, jedoch den Transport innerhalb des Netzes.

---

104 National Security Agency, der mit Kryptographie und internationalen elektronischen Lauschoperationen befaßte Geheimdienst der USA.

105 Zentralstelle für Sicherheit in der Informationstechnik (früher ZfCh, Zentralstelle für das Chiffrierwesen, damals dem Bundeskanzleramt der Bundesrepublik Deutschland zugeordnet; heute BSI, Bundesamt für Sicherheit in der Informationstechnik, dem Innenministerium unterstellt).

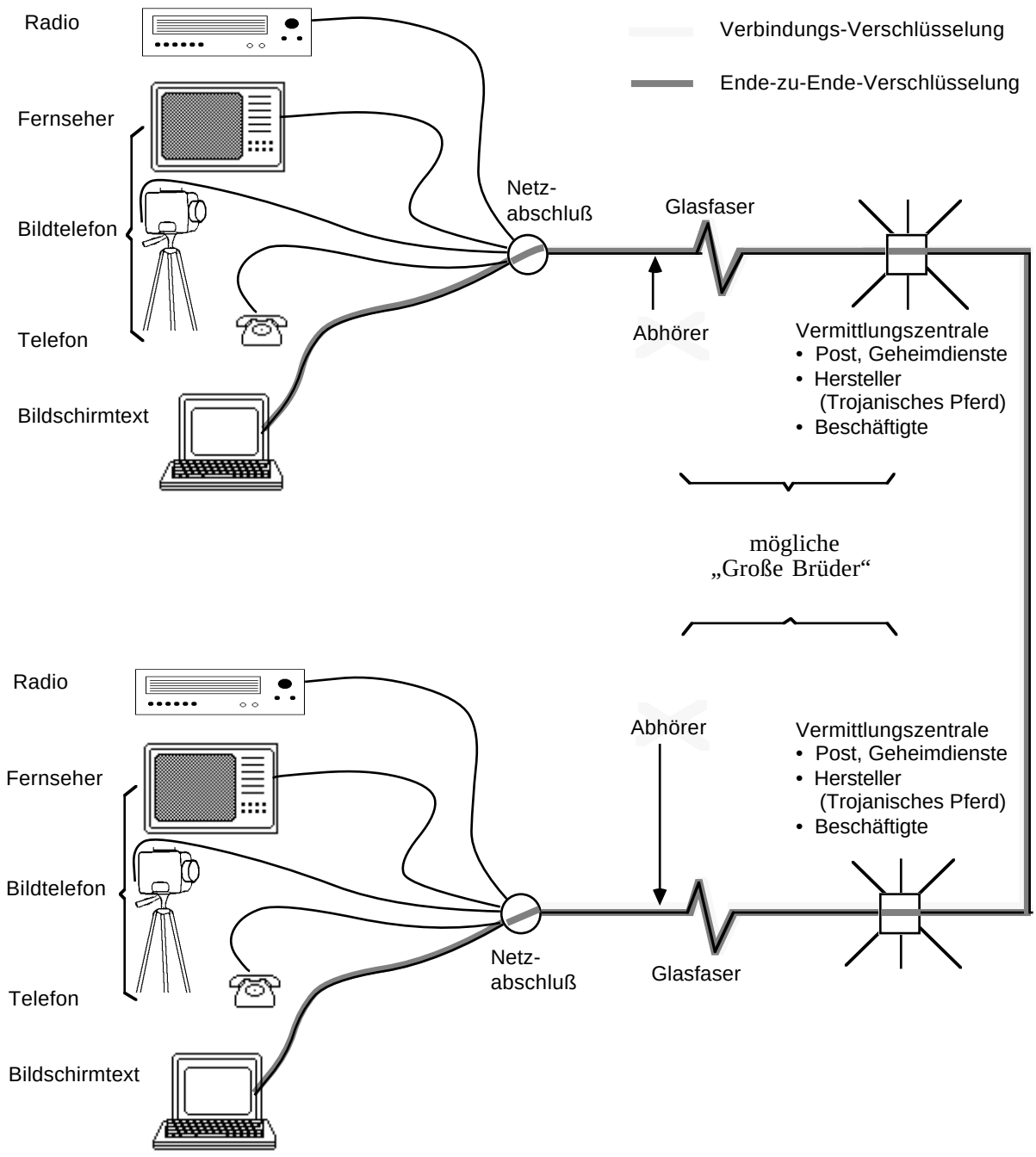


Bild 5-4: Ende-zu-Ende-Verschlüsselung zwischen Teilnehmerstationen und Verbindungs-Verschlüsselung zwischen Netzabschlüssen und Vermittlungszentralen sowie zwischen Vermittlungszentralen

5.3 Grundverfahren außerhalb des Kommunikationsnetzes zum Schutz der Verkehrs- und Interessensdaten

Bei Schutzmaßnahmen außerhalb des Kommunikationsnetzes sind Ziel- und Herkunftsadresse einer Nachricht weiterhin im Netz als Vermittlungsdaten sichtbar. Dazu sind natürlich die Zeit, zu der eine Nachricht im Netz ist, und zumindest für den Kommunikationspartner die Nutzdaten sichtbar. Man muß verhindern, daß daraus Schlüsse auf Verkehrs- und Interessensdaten gezogen werden können.

5.3.1 Öffentliche Anschlüsse

Gibt es für das Kommunikationsnetz öffentliche Anschlüsse, z.B. Telefonzellen für das Fernsprechnetz, oder ist das Kommunikationsnetz zumindest von einem anderen erreichbar, für das es öffentliche Anschlüsse gibt, so ist die folgende Schutzmaßnahme praktizierbar:

Die Herkunftsadresse und Zieladresse einer Nachricht werden weitgehend bedeutungslos, wenn man **verschiedene öffentliche Anschlüsse benutzt**. Dies gilt natürlich nur, wenn man sich nicht zu Zwecken der Zugangskontrolle oder der Zahlung von Entgelten identifizieren muß, d.h. anonym bleibt [Pfit_85].

Die Anwendung und Wirkung der Schutzmaßnahme „Benutzung verschiedener öffentlicher Anschlüsse“ ist stark eingeschränkt:

Wer etwa will für jedes einzelne Telefonat die Wohnung oder das Büro verlassen bzw., um auch eine zeitliche Verkettung zu erschweren (vgl. §5.3.2), den öffentlichen Anschluß wechseln? Wer will sich gar zu vorher vereinbarten Zeitpunkten in eine Telefonzelle begeben, um einen Rückruf unter nur erhoffter (denn es ist ja vorher klar, wo er sich physisch befinden wird: konventionelle Observierung!) Wahrung seiner Anonymität entgegennehmen zu können? Wer will in (Bild-)Telefonzellen noch so hochauflösendes Fernsehen genießen?

Selbst wenn alle diese Fragen positiv zu beantworten wären, bliebe die Wirkung dieser Maßnahme eingeschränkt: Selbst wenn jeder zwischen verschiedenen Telefonaten die Telefonzelle wechselt, bleibt (aus Gründen des Energie- und Zeitaufwandes) eine starke Lokalität erhalten. Selbst wenn der *einzelne* so unter einigen tausend Leuten anonym wäre, gäben seine *Kommunikationsbeziehungen* doch eine Möglichkeit, seine Sende- oder Empfangsereignisse zu verketten und ihn somit im Laufe der Zeit doch nach und nach zu identifizieren – und sei es nur mit Hilfe der Hypothese, daß sein Wohn- bzw. Arbeitsort ganz in der Nähe des Schwerpunktes des durch die von ihm benutzten Telefonzellen gebildeten, mit der Häufigkeit ihrer Benutzung in den entsprechenden Zeiträumen gewichteten Telefonzellenpolygons liegt.

5.3.2 Zeitlich entkoppelte Verarbeitung

Die **Zeit**, zu der sich eine Nachricht im Kommunikationsnetz befindet, wird weitgehend bedeutungslos, wenn man Teilnehmerstationen (z.B. Personal Computer) Informationen nicht erst dann **anfordern** läßt, wenn der Teilnehmer sie benötigt, sondern **zu einem beliebigen Zeitpunkt vorher** [Pfi1_83 Seite 67, 68, Pfit_84]:

> *Will man eine Zeitung lesen, so kann die Teilnehmerstation sie bereits zum Erscheinungszeitpunkt oder einem Zeitpunkt mit besonders geringen Übertragungskosten (z.B. Nachttarif) anfordern und zum späteren Lesen abspeichern.*

Diese Schutzmaßnahme verhindert, daß der Netzbetreiber allein aus dem Vermittlungsdatum „Zeit“ und Kontextwissen schließen kann, wer mit wem kommuniziert:

> *Wenn tagsüber eine Zeitung angefordert wird und 10 Leute als Anforderer in Frage kommen, von denen bekanntlich 9 in Tag- und einer in Nachtschicht arbeiten, wäre ohne zeitliche Entkopplung sehr wahrscheinlich, daß sie der Nachtarbeiter liest.*

Entsprechend kann die Teilnehmerstation Information **zu einem beliebigen Zeitpunkt nach ihrer Entstehung senden**.

Kann der Netzbetreiber auf andere Art erkennen, wer mit wem kommuniziert, so erschwert diese Maßnahme doch zumindest das Erstellen von Persönlichkeitsbildern über den Tagesablauf.

Natürlich greift diese Maßnahme nicht bei Kommunikationsformen mit Realzeitanforderungen, z.B. Telefon. Bei allen anderen erschwert sie aber die Verkettung vom zeitlich entkoppelten Verkehrsereignis mit vom Teilnehmer später veranlaßten Folge-Verkehrsereignissen (vgl. §5.1.2).

5.3.3 Lokale Auswahl

Um Interessensdaten zu schützen, kann man **Information in großen Einheiten anfordern und lokal auswählen (lassen), was einen wirklich interessiert**:

> *Bestellt man sich statt eines bestimmten Zeitungsartikels mehrere Zeitungen verschiedener politischer Richtungen, so können aus der Bestellung keinerlei Rückschlüsse auf die politischen Interessen und Meinungen des Bestellers gezogen werden.*
> *Zusätzlich ließe sich durch „intelligente" Teilnehmerstationen die Dienstleistung sogenannter Ausschnittsbüros, die einem Kunden auf Wunsch zu einem bestimmten Thema Artikel aus mehreren Zeitungen zusammenstellen, jedem Teilnehmer bieten.*

Durch die lokale Auswahl verschleiert man selbst gegenüber dem Kommunikationspartner, was einen wirklich interessiert, wie man (unter anderem die genaue Bedienung der Teilnehmerstation) lernt und reagiert, sowie vieles mehr.

Das Anfordern von großen Informationseinheiten vom Kommunikationspartner ist folglich als Schutz bei solchen Informationsarten sinnvoll, die unverschlüsselt übertragen werden oder bei denen Netzbetreiber und Kommunikationspartner identisch sind oder als zusammenarbeitend angenommen werden können.

In zukünftigen Kommunikationsnetzen, in denen (zumindest für schmalbandiges Senden) reichlich Bandbreite zur Verfügung steht, verursacht diese Maßnahme bei Diensten, bei denen der Benutzer nur bereits bereitgestellte Information abruft, real beinahe keine Kosten. Es ist jedoch eine Frage des Abrechnungsmodus, ob dies auch dem Anwender dieser Maßnahme zugutekommt (vgl. §5.6.2). Selbst wenn kein akzeptabler Abrechnungsmodus gefunden werden kann, der es erlaubt, z.B. mehrere Zeitungen verschiedener politischer Richtungen und damit vermutlich auch aus verschiedenen Verlagen zum Preise einer zu beziehen, so sollten doch zumindest wie heutzutage z.B. ganze Zeitungen verkauft und übertragen werden. Eine auf kleine Bildschirmseiten orientierte Struktur wie die des Bildschirmtext-Systems, die damit begründet wurde, die übliche Teilnehmerstation müsse billig und deshalb *zwangsweise* ohne lokale Verarbeitungs- und Speicherfähigkeit realisiert werden, ist zumindest für die Zukunft technisch obsolet und könnte nur damit gerechtfertigt werden, weiterhin das zur Teilnehmerbeobachtung technisch optimal geeignete System behalten zu wollen.

Lokale Auswahl wurde (mindestens) dreimal unabhängig voneinander erfunden: zuerst wird sie bezüglich *eines* Informationsanbieters und ohne generelle Einordnung lediglich im Kontext des Bildschirmtext-Systems in [BGJK_81 Seite 153, 159] vorgeschlagen, danach in [Pfit_83] und schließlich zusammen mit zeitlich entkoppelter Verarbeitung in [GiLB_85].

5.4 Grundverfahren innerhalb des Kommunikationsnetzes zum Schutz der Verkehrs- und Interessensdaten

Im Gegensatz zu den in §5.3 genannten Schutzmaßnahmen sollen in diesem Abschnitt Maßnahmen vorgestellt werden, die die im Kommunikationsnetz anfallenden Vermittlungsdaten (vgl. §5.1.1) selbst verringern.

Dadurch soll im Interesse jedes Teilnehmers Senden und Empfangen, zumindest aber die Kommunikationsbeziehungen zwischen Teilnehmern vor möglichen Angreifern (bösen Nachbarn, dem

Netzbetreiber, den Herstellern der verwendeten Soft- und Hardware, großen Organisationen und Konzernen usw.) verborgen werden. Dadurch wird das Erfassen von Verkehrs- oder Interessensdaten unmöglich.

Gestaltungsziele sind:

- Die Netzbenutzung soll sich für den Teilnehmer nicht ändern oder gar komplizierter werden.
- Jeder Teilnehmer soll die Möglichkeit haben, die erhaltene Vertraulichkeit selbst zu überprüfen.

Da Schutz vor einem allmächtigen Angreifer, der alle Leitungen, alle Vermittlungszentralen und alle Teilnehmerstationen außer der eigenen kontrolliert, nicht möglich ist, sind alle folgenden Maßnahmen nur Annäherungen an den perfekten Schutz der Teilnehmer vor jedem möglichen Angreifer. Die Annäherung wird – wie schon in §5.1.2 erwähnt – im allgemeinen durch Angabe des unterstellten *Angreifermodelles* (Was kann er maximal beobachten oder gar aktiv verändernd kontrollieren; mit wieviel Rechenkapazität ist er ausgestattet?) beschrieben.

Verbirgt ein Kommunikationsnetz für alle Kommunikationsereignisse wer *sendet* oder *empfängt* oder zumindest die *Kommunikationsbeziehung* vor dem unterstellten Angreifer, so wird es *anonym* genannt. Es wird dann von **anonymer Kommunikation** gesprochen.

Die meisten der heute bekannten und in den folgenden Unterabschnitten beschriebenen Grundverfahren innerhalb des Kommunikationsnetzes zum Schutz der Verkehrs- und Interessensdaten schützen für alle realistischerweise in Betracht gezogenen Angreifer entweder

- die Kommunikationsbeziehung oder
- wer sendet oder
- wer empfängt.

Gegen wesentlich schwächere Angreifer bieten diese Grundverfahren vollständigen Schutz: Es sind Sender *und* Empfänger und damit auch die Kommunikationsbeziehung geschützt. Gegen wesentlich stärkere Angreifer bieten sie jedoch keinen Schutz. Wegen dieser kanonischen, d.h. von den Details einer realistischen Angreiferdefinition unabhängigen Zuordnung der Grundverfahren zu Schutzzielen sind die Unterabschnitte entsprechend diesen drei Möglichkeiten benannt.

5.4.1 Verteilung: Schutz des Empfängers

Indem das Kommunikationsnetz alle Informationen an alle Teilnehmer sendet (**Verteilung**, broadcast), kann man den Empfänger der Information vor dem Kommunikationsnetz und dem Kommunikationspartner schützen [FaLa_75].

Verteilung ermöglicht das Analogon zu perfekter informationstheoretischer Konzelation (vgl. §3): *perfekte informationstheoretische Anonymität* des Empfängers. Dies bedeutet nach der Definition von perfekter Anonymität in §5.1.2 und dem in §3 über die informationstheoretische Modellwelt Gesagtem, daß für einen alle Leitungen und ggf. einige Stationen kontrollierenden Angreifer für alle Nachrichten die Wahrscheinlichkeit, daß eine Station die Nachricht empfangen hat, für alle von ihm nicht kontrollierten Stationen vor und nach seiner Beobachtung gleich ist. Der Angreifer gewinnt durch seine Beobachtung also keine Information darüber, welche von ihm nicht kontrollierte Station welche Nachricht empfängt. Es ist für den Angreifer sogar nicht einmal beobachtbar, ob die Nachricht überhaupt empfangen wird. Verteilung garantiert also auch *perfekte informationstheoretische Unbeobachtbarkeit* des Empfangens. Ebenso verkettet Verteilung keine Verkehrsereignisse, so daß auch *perfekte informationstheoretische Unverkettbarkeit* gegeben ist.

5.4.1.1 Verändernde Angriffe auf die Verteilung

Die Realisierung von Verteilung stellt keinerlei konzeptionelles Problem dar, solange nur *beobachtende Angriffe* betrachtet werden. Ansonsten ist dies nicht der Fall:

1) Ein verändernder Angreifer kann die *Diensterbringung verhindern.*
   Hiergegen helfen die üblichen Fehlertoleranztechniken, gegebenenfalls kombiniert mit einem Wechsel der Betreiberschaft, falls „zufällig" häufig alle redundanten Subsysteme ausfallen.
2) Ein verändernder Angreifer kann *durch Störung der Konsistenz der Verteilung manche Empfänger zumindest teilweise deanonymisieren*.
   Ist der verändernde Angreifer beispielsweise der Sender einer Nachricht *N*, auf die der Empfänger antworten wird, und möchte er den Empfänger deanonymisieren, so kann er dies bei unsicher implementierter Verteilung folgendermaßen tun: Er sendet *N* und verhindert die Verteilung von *N* an alle anderen Teilnehmerstationen außer einer. Dann wartet er, bis der Empfänger geantwortet haben müßte. Erhält der Angreifer keine Antwort, so wiederholt er seinen Angriff, indem er eine andere Teilnehmerstation *N* empfangen läßt. Ist auf *N* nicht nur *eine* Antwort zu erwarten, sondern ist dieser einfachstmögliche Dialog in einen längeren eingebettet, so kann die Länge dieses Angriffs, bei dem durchschnittlich die Hälfte aller anderen Teilnehmerstationen durchprobiert werden muß, auf den Logarithmus der Anzahl der anderen Teilnehmerstationen „gedrückt" werden: die Verteilung wird nur für die Hälfte der anderen Teilnehmerstationen gestört. Erhält der Angreifer Antwort, halbiert er für den nächsten Schritt die Gruppe, die die Nachricht erhielt, anderenfalls halbiert er deren Komplement zum vorherigen Schritt [Waid_90, WaPf1_89].
   Diese Angriffsart kann entweder durch analoge oder digitale Maßnahmen vereitelt werden. Eine **analoge Maßnahme** könnte in der Verwendung eines Mediums bestehen, daß *Konsistenz der Verteilung garantiert* – viele schreiben *nichtzellularen Funk- und Satellitennetzen* diese Eigenschaft zu. Da solche Eigenschaften schwer nachgewiesen werden können (vgl. das in §2 über unmanipulierbare Gehäuse Gesagte) und Funknetze anfällig für Angriffe auf die Diensterbringung sowie in der verfügbaren Bandbreite beschränkt sind, da das elektromagnetische Spektrum im wesentlichen nur einmal genutzt werden kann, ist es wichtig, digitale Maßnahmen zur Verfügung zu haben. **Digitale Maßnahmen** können *inkonsistente Verteilung natürlich nur erkennen*, aber nicht verhindern, denn ein physisch genügend verbreiteter Angreifer kann die Signalausbreitung zu einzelnen Teilnehmerstationen immer mit Durchtrennen von Leitungen, Errichtung Faradayscher Käfige oder – für den Angreifer meist einfacher – Beschädigung der Funkantenne unterbinden. Die Erkennung inkonsistenter Verteilung kann dabei entweder *komplexitätstheoretisch* realisiert werden, indem Nachrichten mit Sequenznummern, Uhrzeit etc. versehen und von einer bezüglich eines Angriffes auf die Verteilung als nichtkooperierend angenommenen Gruppe von Stationen digital signiert werden [PfPW1_89]. Sie kann sogar *informationstheoretisch* erreicht werden, indem das in §5.4.5 beschriebene überlagernde Senden durch geeignetes Einbeziehen der verteilten Nachrichten in die dortige Schlüsselgenerierung erweitert wird, vgl. §5.4.5.3 und [Waid_90, WaPf1_89].

5.4.1.2 Implizite Adressierung

Will man Verteilung auch bei bisher vermittelten Diensten einsetzen, so muß jede Teilnehmerstation anhand eines **implizite Adresse** genannten Merkmals entscheiden können, welche Nachrichten wirklich für sie bestimmt sind. Im Gegensatz zu einer **expliziten Adresse** enthält eine implizite keinerlei Information, wo sich der Empfänger befindet und ggf. auf welchem Weg er zu erreichen ist.

Kann eine Adresse nur vom Empfänger ausgewertet werden, so spricht man von **verdeckter Adressierung**. Kann eine Adresse von jedem ausgewertet, d.h. auf Gleichheit mit anderen Adres-

sen getestet werden, so nennt man dies **offene Adressierung** [Waid_85]. Bei offener Adressierung sind Nachrichten prinzipiell immer dann, wenn gleiche Adressen auftreten, über ihre Adresse, d.h. den Empfänger *verkettbar*.

5.4.1.2.1 Implementierung von verdeckter impliziter Adressierung

Die übliche Implementierung von verdeckter impliziter Adressierung verwendet Redundanz innerhalb des Nachrichteninhalts und ein *asymmetrisches Konzelationssystem* (vgl. §3). Jede Nachricht wird ganz oder teilweise mit dem Chiffrierschlüssel des adressierten Teilnehmers verschlüsselt (wodurch bei der ersten Möglichkeit gleichzeitig Ende-zu-Ende-verschlüsselt wird). Nach der Entschlüsselung mit dem zugehörigen Dechiffrierschlüssel kann die Teilnehmerstation des adressierten Teilnehmers anhand der Redundanz feststellen, daß die Nachricht für sie bestimmt ist. Da die Implementierungen von asymmetrischen Konzelationssystemen aufwendig und damit bei begrenztem Implementierungsaufwand langsam sind (vgl. §3) und jede Teilnehmerstation alle Nachrichten entschlüsseln muß, ist dieses Verfahren im allgemeinen viel zu aufwendig [Ken1_81]. Es kann durch Austausch eines geheimen Schlüssels und Verwendung eines schnelleren *symmetrischen Konzelationssystems* [Karg_77 Seite 111, 112] (statt eines asymmetrischen) nach Aufnahme der Kommunikationsbeziehung jedoch etwas effizienter gestaltet werden: Beginnt jede Nachricht mit einem Bit, das angibt, ob das asymmetrische oder symmetrische Konzelationssystem zur Adressierung verwendet wurde, so kann durch Verwendung von Letzterem bei langandauernden Kommunikationsbeziehungen asymptotisch soviel Aufwand gespart werden, wie das symmetrische Konzelationssystem effizienter als das asymmetrische implementierbar ist.

Verwendet ein Empfänger mehrere verdeckte implizite Adressen, so muß er alle Nachrichten mit allen seinen Dechiffrierschlüsseln entschlüsseln. Während Teilnehmer bei Verwendung eines asymmetrischen Konzelationssystems für die verdeckte implizite Aressierung nur wenige Dechiffrierschlüssel haben dürften, dürfte deren Anzahl bei Verwendung eines symmetrischen Konzelaationssystems drastisch größer sein – was den Effizienzvorteil der symmetrischen Konzelation abschwächt, ja sogar in Mehraufwand resultieren kann.

Die Unverkettbarkeit von Nachrichten über die verdeckte implizite Adressierung ist natürlich nur so sicher, wie das unsicherste zur Adressierung verwendete Konzelationssystem (bei der gegeben Öffentlichkeit der Schlüssel, vgl. übernächster Unterabschnitt über öffentliche und private Adressen).

Statt eines symmetrischen Konzelationssystems kann auch ein *symmetrisches Authentikationssystem* verwendet werden: Statt Nachrichten redundant zu codieren und danach zu verschlüsseln und nach der Entschlüsselung die Redundanz zu testen, werden Nachrichten symmetrisch authentisiert, d.h. das jeweils angehängte MAC stellt die Redundanz dar. Die potentiellen Adressaten prüfen jeweils, ob aus ihrer Sicht die Nachricht richtig authentisiert ist. Wenn ja, ist die Nachricht für sie.

5.4.1.2.2 Äquivalenz von verdeckter impliziter Adressierung und Konzelation

Daß die verdeckte Adressierung der ersten Nachricht zwischen zwei Parteien, die noch keinen geheimen Schlüssel ausgetauscht haben, nicht unverkettbarer und in Bezug auf die verwendete Adresse anonymer als mit dem sichersten asymmetrischen Konzelationssystem möglich ist, kann man sich wie folgt überlegen: Mit jeder Möglichkeit zur verdeckten Adressierung kann man die Funktion eines asymmetrischen Konzelationssystems erbringen, indem die ein Geheimnis übermitteln wollende Partei der anderen die Nachricht *„Die geheime Nachricht ist in der Form codiert, daß jede an dich adressierte Nachricht eine 1, jede nicht an dich adressierte Nachricht eine 0 bedeutet. <adressierte Nachricht 1>, <adressierte Nachricht 2>, ... <adressierte Nachricht n>“* unverschlüsselt zusendet [Pfi1_85 Seite 9, PfWa_86]. Dieses auf Empfänger-Anonymität basierende Konzelations-Protokoll

entspricht dem auf Sender-Anonymität basierenden Konzelations-Protokoll von Alpern und Schneider [AlSc_83].

Wird in diesem Beweis unterstellt, daß beide Parteien einen geheimen Schlüssel ausgetauscht haben, so folgt, daß mit jeder Möglichkeit zur verdeckten Adressierung die Funktion eines symmetrischen Konzelationssystems erbracht werden kann. Beide Beweise zusammen und die oben beschriebenen Implementierungen verdeckter impliziter Adressierung mittels Konzelationssystemen ergeben, daß verdeckte Adressierung und Konzelation sowohl im asymmetrischen wie im symmetrischen Fall äquivalent sind.

Aus den Beweisen resultieren in natürlicher Weise Schranken für die mögliche Effizienz, d.h. Konzelationssysteme und verdeckte Adressierung können sowohl im asymmetrischen wie auch im symmetrischen Fall jeweils „ähnlich“ effizient sein.

5.4.1.2.3 Implementierung von offener impliziter Adressierung

Offene Adressierung läßt sich einfacher realisieren (als verdeckte), indem man z.B. Nachrichten mit einem Adreßfeld versieht und die Teilnehmer(stationen) beliebige Zahlen als Adressen erzeugen. Eine Teilnehmerstation muß dann nur bei allen erhaltenen Nachrichten dieses Adreßfeld mit ihren Adressen vergleichen. Dies kann sie z.B. mittels eines *Assoziativspeichers*, in den sie alle ihre Adressen schreibt und an den dann alle Adreßfelder angelegt werden, sehr schnell und effizient tun, selbst wenn sie (momentan) eine große Zahl von verschiedenen Adressen besitzt. Adreßerkennung ist bei offener Adressierung also wesentlich effizienter als bei verdeckter möglich.

5.4.1.2.4 Adreßverwaltung

Hinsichtlich der Adreßverwaltung kann man bei beiden Formen der impliziten Adressierung öffentliche und private Adressen unterscheiden: **Öffentliche Adressen** stehen in allgemein zugänglichen Adreßverzeichnissen (z.B. in einem „Telefonbuch“) und dienen meist einer ersten Kontaktaufnahme. **Private Adressen** werden an einzelne Kommunikationspartner gegeben. Dies kann entweder außerhalb des Kommunikationsnetzes oder innerhalb als Absenderangabe in Nachrichten geschehen. Es sei angemerkt, daß die Kategorien „öffentliche Adresse“ und „private Adresse“ noch nichts über den Personenbezug aussagen: Öffentliche Adressen können (zumindest vor ihrem Gebrauch) informationstheoretisch anonym vor allen Instanzen außer dem Adressaten sein. Private Adressen können für den Benutzer durchaus einer Person zuordenbar sein.

			Adreßverwaltung	
			öffentliche Adresse (z.B. in Telefonbuch)	private Adresse (nur an bestimmte Kommunikationspartner)
Adressierungsart	implizite Adresse	verdeckt	sehr aufwendig, für Kontaktaufnahme nötig	aufwendig, kein sinnvoller Anwendungsbereich
		offen	effizient, abzuraten	effizient, nach Kontaktaufnahme ständig wechseln
	explizite Adresse		spart Bandbreite, sehr abzuraten	spart Bandbreite, sehr abzuraten

Bild 5-5: Effizienz- und Unbeobachtbarkeits-Bewertung der Kombinationen von Adressierungsart und Adreßverwaltung

Bei offener Adressierung, im Gegensatz zur verdeckten, kann das Netz – wie bereits oben erwähnt – prinzipiell Informationen über die Empfänger von Nachrichten gewinnen. Dies kann bei Verwendung privater Adressen geschehen, wenn man dieselbe Adresse mehrfach verwendet, weil dann selbst für ansonsten unverkettbare Nachrichten erkennbar wird, daß sie an denselben Empfänger gerichtet sind. Die mehrmalige Verwendung offener Adressen muß also durch fortlaufendes Generieren und Mitübertragen oder durch Vereinbarung eines Generieralgorithmus vermieden werden. Der Generieralgorithmus „Verwende die nächsten *n* Zeichen eines rein zufällig generierten, nur einmal benutzten und in perfekte informationstheoretische Konzelation und Integrität garantierender Weise vorher verteilten Schlüssels“ erlaubt *perfekte informationstheoretische Unverkettbarkeit und Anonymität* vor Unbeteiligten. Mit den bis hierher geschilderten Verfahren ist dann allerdings aus organisatorischen Gründen keine wirkliche Anonymität zwischen den Beteiligten möglich (vgl. das in §3 über perfekte informationstheoretische Konzelation Gesagte). In §5.4.5.4 wird dann mit dem paarweisen überlagernden Empfangen eine auf Senderanonymität basierende Methode zum Austausch eines Schlüssels zwischen anonymen Beteiligten beschrieben. Ist die Senderanonymität informationstheoretisch, so erfolgt der Schlüsselaustausch (ebenfalls in der informationstheoretischen Modellwelt) in Konzelation und Integrität garantierender Weise. Da informationstheoretische Senderanonymität möglich, aber mit sehr, sehr großem Aufwand verbunden ist, ist das angedeutete Schlüsselaustausch-Verfahren zwar möglich, aber praktisch kaum anwendbar. Erzeugt man hingegen den nur einmal benutzten Schlüssel mit einem kryptographisch starken Pseudozufallsbitfolgengenerator (vgl. §3.4.3), so kann nur *perfekte komplexitätstheoretische Unverkettbarkeit und Anonymität* gewahrt werden – dafür ist dann Anonymität zwischen den Beteiligten auch praktisch kein Problem mehr, da der Startwert des Pseudozufallsbitfolgengenerators mit einem perfekten komplexitätstheoretischen asymmetrischen Konzelationssystem ausgetauscht werden kann.

Bei Verwendung öffentlicher offener Adressen kann sogar festgestellt werden, unter welcher Bezeichnung der Empfänger im Adreßverzeichnis eingetragen ist. Die Verwendung solcher Adressen sollte also möglichst vermieden werden.

Die Schutz- und Aufwandseigenschaften der Kombinationen von Adressierungsart und Adreßverwaltung sind in Bild 5-5 zusammengefaßt.

Durch Verteilung und geeignete implizite Adressierung kann der Empfänger einer Nachricht also vollständig geschützt werden.

Leider sind nicht alle Typen von Kommunikationsnetzen für diese Maßnahme gleichermaßen geeignet. Dies wird in §5.4.9 diskutiert.

5.4.1.3 Tolerierung von Fehlern und verändernden Angriffen bei Verteilung

Zwischen Verteilung einerseits und Tolerierung von Fehlern und verändernden Angriffen andererseits gibt es folgende relevanten Interaktionen:

Empfänger, die *Informationseinheiten fehlerhaft oder gar nicht erhalten*, sollten unabhängig davon, ob sie sie überhaupt benötigen, auf einen fehlerfreien Empfang – etwa mittels nochmaliger Übertragung – bestehen. Anderenfalls könnte eine Reaktion auf fehlerhafte oder gar nicht erhaltene Informationseinheiten den bzw. die Empfänger verraten. Bei der praktischen Durchführung dieser Regel stößt man möglicherweise an zwei Grenzen:

Einerseits können Teilnehmerstationen möglicherweise nicht die gesamte ihnen zufließende Information gleichzeitig empfangen und deshalb manche fehlerhaften Informationseinheiten gar nicht erkennen.

Andererseits könnte es keine Instanz geben, von der eine fehlerfreie Kopie angefordert werden kann. Letzteres kann durch eine kurzzeitige Speicherung aller verteilten Informationseinheiten in einem (oder mehreren) global bekannten Speicher(n) ohne irgendwelche Einschränkungen der Vertraulichkeit gelöst werden.

Bezüglich der Erkennung fehlerhafter Verteilung bzw. verändernder Angriffe auf die Verteilung sei an das in §5.4.1.1 Gesagte erinnert.

Für das Bestehen auf fehlerfreien Empfang ist ein Senden der Teilnehmerstation nötig. Bei Funknetzen kann die Mobilstation hierdurch peil- und identifizierbar werden, vgl. [Pfit_93, ThFe_95]. Hier ist dann ggf. eine Güterabwägung nötig.

Ähnlich wie bei Verschlüsselung muß bei *impliziter Adressierung* darauf geachtet werden, daß implizite Adressen auch bei Verfälschung oder Verlust von vorangehenden Informationseinheiten richtig erkannt werden. Dies kann durch Verwendung einer Blockchiffre bei verdeckter impliziter Adressierung perfekt erreicht werden. Bei offener impliziter Adressierung mit jeweils nur einmal verwendeten Adressen, die zum Zwecke des Adreßvergleichs in einen Assoziativspeicher geschrieben und nach Erhalt einer entsprechend adressierten Informationseinheit durch die folgende Adresse ersetzt werden (vgl. §5.4.1.2), ist dies nicht ohne weiteres möglich. Zwar können bei einer Fehlervorgabe von maximal k aufeinanderfolgenden verlorenen oder verfälschten Informationseinheiten statt der nächsten immer die k+1 nächsten Adressen jeder Adreßfolge in einen entsprechend um den Faktor k+1 größeren Assoziativspeicher geschrieben werden, was gegen kurzzeitige Störungen hilft. Bei längeren Störungen müssen Sender und Empfänger aber – etwa mittels Nachrichten mit verdeckter impliziter Adressierung – neu synchronisiert werden.

5.4.2 Abfragen und Überlagern: Schutz des Empfängers

Durch Verteilung der Nachrichten an alle (Broadcast) sind die Empfänger bestmöglich geschützt. Da jeder alle Nachrichten erhält, kann niemand verfolgen, wer an welchen Daten Interesse zeigt. Verteilung ist jedoch in vielen Kommunikationsnetzen für Individualkommunikation zu aufwendig.

In [CoBi_95] wird das Verfahren **Abfragen und Überlagern** beschrieben, welches ebenfalls die Interessensdaten schützt: Den Teilnehmern eines Kommunikationsnetzes wird ermöglicht, allen zugängliche Nachrichten von einem verteilt implementierten **Nachrichten-Service** abzufragen, ohne daß ein anderer erfährt, welche Nachricht gerade abgefragt wird, an welchen Daten also Interesse besteht.

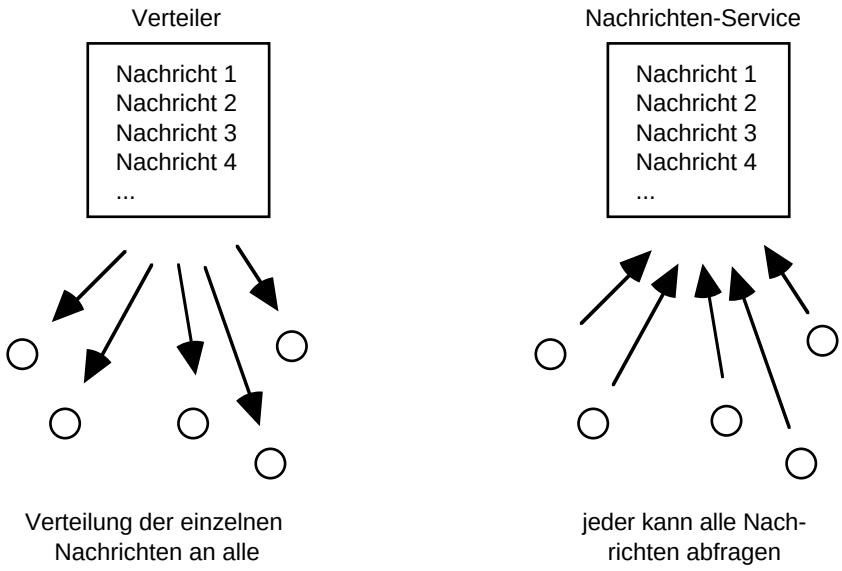


Bild 5-6: Verteilung vs. Abfragen

Genauer können die Teilnehmer aus Servern Nachrichten *überlagert* abfragen. Andere sind nicht in der Lage herauszufinden, welche Informationen abgefragt wurden, da die aus mehreren Servern überlagert abgefragten Nachrichten vom Lesenden *lokal überlagert* werden und er so die ihn interessierende Nachricht erhält.

Ganz genau funktioniert Abfragen und Überlagern folgendermaßen: Das System bilden m Server mit je n Speicherzellen (entsprechend der Anzahl von Nachrichten, die gespeichert werden können). Alle Server enthalten die gleichen Nachrichten in derselben Reihenfolge.

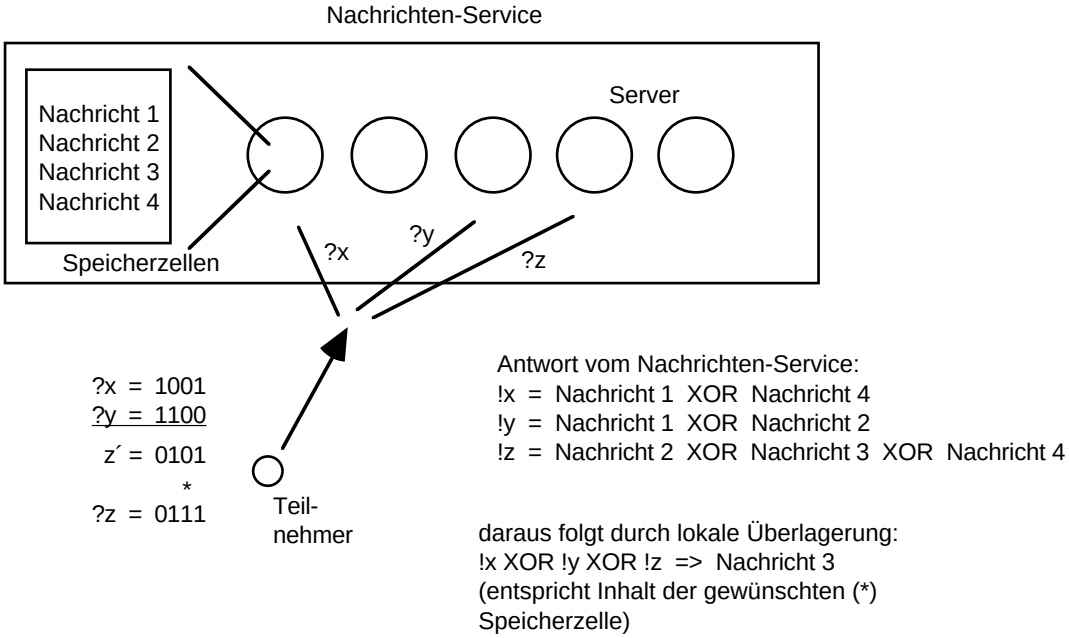


Bild 5-7: Beispiel für Nachrichten-Service mit $m = 5$ und $s = 3$, zu lesen ist Nachricht 3

Um zu verdecken, an welcher Nachricht ein Teilnehmer interessiert ist, fordert er von s Servern ($1 < s \leq m$) unterschiedliche Informationen an. Dazu bildet er s-1 Abfragevektoren, die zufällige Bitketten der Länge n sind. Der s-te Abfragevektor wird durch XOR aller s-1 Abfragevektoren gebildet. Für

das Auslesen der den Nutzer interessierenden Nachricht aus der *t*-ten Speicherzelle ($1 \le t \le n$) wird das *t* -te Bit des berechneten Vektors negiert.

An *s* der *m* Server wird je einer der Abfragevektoren gesendet. Die gewünschten Nachrichten, d.h. die Nachrichten, an deren Position im Abfragevektor eine "1" steht, werden in den Servern aus den Speicherzellen gelesen, mit XOR verknüpft und an den Teilnehmer geschickt. Der Teilnehmer kann durch XOR Verknüpfung der so übermittelten Nachrichten die ihn interessierende Nachricht bilden.

Zwischen Teilnehmer und den Servern wird jeweils verschlüsselt kommuniziert. Deshalb brauchen externe Angreifer nicht betrachtet zu werden. Aber selbst bei Zusammenarbeit von *s*-1 Servern sind die Interessensdaten des Teilnehmers geschützt:

Beh. Beliebige *s*-1 Abfragevektoren enthalten auch zusammen keinerlei Information über die Nachricht, die den Teilnehmer interessiert.

Bew. Vor der Negation des *t* -ten Bits des berechneten *s*-ten Abfragevektors gilt dies sogar für alle *s* Abfragevektoren zusammen, von denen jeweils beliebige *s*-1 zusammen zufällige und unabhängige Bitketten sind. Jeweils beliebige *s*-1 Abfragevektoren können folglich als die in der Verfahrensbeschreibung ersten *s*-1 Abfragevektoren betrachtet werden, so daß die zu schützende Information nur in den *s*-ten Abfragevektor eingeht. □

Also gilt:

> Jeder der vom Teilnehmer benutzten Server kann die Interessensdaten des Teilnehmers schützen.

In manchen Netzumgebungen kann dieses Verfahren effizienter sein, als einmal alle Nachrichten anzufordern. Allerdings muß der intendierte Empfänger einer Nachricht noch irgendwie erfahren, daß für ihn eine Nachricht vorliegt und in welcher Speicherzelle sie sich befindet. Dies kann geschehen, indem Nachrichten implizite Adressen (vgl. §5.4.1.2) haben und der Nachrichten-Service die impliziten Adressen gefolgt von den zugehörigen Nummern der Speicherzellen entweder an alle Teilnehmer verteilt oder in Speicherzellen, deren Nummern allen Teilnehmern bekannt sind, ablegt und die Teilnehmer diese Speicherzellen abfragen. Effizient ist beides natürlich nur, wenn die Nachrichten wesentlich länger als die Nummern der Speicherzellen sind.

[CoBi_95, CoBi1_95] beschreibt die Anwendung des Verfahrens in der Mobilkommunikation.

5.4.3 Bedeutungslose Nachrichten: Schwacher Schutz des Senders und Empfängers

Der Sender einer Nachricht kann sich (bezüglich des Sendens und bei expliziter Adressierung auch den Empfänger bezüglich seines sonstigen Empfangens) schützen, indem er **bedeutungslose Nachrichten** (dummy traffic) sendet. Werden diese im Falle expliziter Adressierung in ihrem verschlüsselten Nachrichteninhaltsteil mit einem speziellen Kennzeichen versehen bzw. im Falle von Verteilung und impliziter Adressierung mit nicht existenten impliziten Adressen versehen, können sie von der explizit adressierten bzw. allen Teilnehmerstationen weggeworfen werden. Werden auch bedeutungslose Nachrichten gesendet, kann das Netz nicht mehr entscheiden, wann genau und wieviele bedeutungsvolle Nachrichten ein Teilnehmer sendet [Chau_81]. (Symmetrisch dazu kann das Netz selbst im Falle expliziter Adressierung nicht mehr entscheiden, wann genau und wieviele bedeutungsvolle Nachrichten ein Teilnehmer empfängt. Da das Wegwerfen von an andere Stationen adressierten Nachrichten jedoch nicht aufwendiger ist als das Wegwerfen von an die eigene Station

adressierten, und ersteres Empfängeranonymität[106] auch vor dem Sender der Nachricht und erhebliche Aufwandsersparnis in geeignet entworfenen Kommunikationsnetzen ermöglicht, wurde das Verfahren der bedeutungslosen Nachrichten in §5.4.1 nicht erwähnt. Mag oder kann man sich vollständige Verteilung (broadcast) nicht leisten, wird man aus der Sicht des Empfängers statt bedeutungsloser Nachrichten immer partielle Verteilung (multicast) vorziehen.)

Das Verfahren der bedeutungslosen Nachrichten verursacht stets einen sehr hohen Aufwand und erfüllt zugleich nicht die Forderung nach Anonymität des Senders auch bei Angriffen, an denen sich der Empfänger bedeutungsvoller Nachrichten beteiligt. In der Diktion von §5.1.2 wird durch dieses Verfahren also lediglich das *Senden* (bzw. symmetrisch dazu: das *Empfangen*) bedeutungsvoller Nachrichten *für Unbeteiligte unbeobachtbar und unverkettbar*. Anonymität des Senders vor Beteiligten ist mit diesem Verfahren allein jedoch bezüglich realistischer Angreifermodelle nicht zu erreichen. Trotz dieser offensichtlichen Schwächen des Verfahrens der bedeutungslosen Nachrichten ist es das einzige zum Schutz der Verkehrs- und Interessensdaten, das auch in der neueren, nicht von David Chaum oder Mitgliedern unserer Forschungsgruppe geschriebenen, weit verbreiteten und zitierten Literatur zum Thema „Datenschutz und Sicherheit in Kommunikationsnetzen“, insbesondere auch im Entwurf einer Ergänzung des ISO Modells für „Offene Kommunikation“ in bezug auf Datenschutz und Sicherheit steht, vgl. [VoKe_83 Seite 157f, VoKe_85 Seite 18, ISO7498-2_87 Seite 17, 24-29, 31-33, 36, 39f, 53, 56, 60f, Rul1_87, AbJe_87 Seite 28, Folt_87 Seite 45] (natürlich gibt es auch neuere weit verbreitete Literatur, die überhaupt keine Lösungsmöglichkeit nennt, vgl. [Brau_87]). Kennen diese weltberühmten Kapazitäten ihr eigenes Gebiet nicht – oder wollen oder dürfen sie es nicht kennen, vgl. das Vorgehen von NSA und ZSI im Bereich Kryptographie?

Wird das Senden (und Empfangen) bedeutungsloser Nachrichten mit einem Verfahren zum Schutz der Kommunikationsbeziehung geeignet kombiniert, so hält es auch Angriffen stand, an denen sich der Netzbetreiber und der Empfänger bedeutungsvoller Nachrichten beteiligen. Soll durch die Kombination der zwei Verfahren der Sender (oder Empfänger) geschützt werden, so muß er mit einer von seinen Sende- und Empfangswünschen unabhängigen Rate Nachrichten senden und empfangen. Dies bedeutet, daß die Bandbreite des Kommunikationsnetzes im wesentlichen *statisch* auf die Teilnehmer aufgeteilt werden muß, was für viele Anwendungen ungeschickt ist. Außerdem kann auf diese Weise höchstens *komplexitätstheoretischer* Schutz des Senders (und Empfängers) erreicht werden. Trotz dieser Einschränkungen kann diese Kombination bei bestimmten äußeren Randbedingungen sinnvoll sein, siehe §5.5.1.

Die in den beiden folgenden Abschnitten beschriebenen Verfahrensklassen zum Schutz des Senders haben die gerade genannten Nachteile nicht. Sie sind in gewisser Weise effizient, indem sie eine *dynamische* Aufteilung der Bandbreite erlauben sowie bedeutungslose Nachrichten und Umcodieren vermeiden, und es ist in der *informationstheoretischen* Modellwelt bewiesen, daß der Sender selbst bei Identität oder Zusammenarbeit von Empfänger und Netzbetreiber anonym ist.

Später wird mit dem überlagernden Senden ein sehr mächtiges, aber auch entsprechend aufwendiges Verfahren zum Realisieren von Senderanonymität auf einem beliebigen Kommunikationsnetz beschrieben.

Davor wird erklärt, wie ein Kommunikationsnetz insbesondere durch Wahl einer geeigneten Leitungstopologie (Ring oder Baum) und digitale Signalregenerierung gleich so – und damit preiswerter als durch „Aufsetzen“ eines zusätzlichen Verfahrens – realisiert werden kann, daß realistische Angreifer nur unter hohem Aufwand das Senden von einzelnen Stationen beobachten können. Ist

106 Eigentlich müßte es Empfänger*pseudonymität* heißen, da der Sender ja mindestens ein Pseudonym des Empfängers kennt. Da sich der Begriff Empfängeranonymität aber eingebürgert hat, verwende ich ihn.

hierbei ihr Aufwand genauso groß wie der anderer Formen der Individualüberwachung, so ist dem in einer Demokratie nötigen Datenschutz Genüge getan.

Die beiden Verfahrensklassen können sinnvoll kombiniert werden.

5.4.4 Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung: Schutz des Senders

Das aufwendige Generieren, ggf. Verteilen und Überlagern von Schlüsseln und Nutzdaten im folgenden §5.4.5 ist nötig, da davon ausgegangen wird, daß ein Angreifer die Ein- und Ausgänge *aller* Teilnehmerstationen abhört bzw. die jeweils angrenzenden Stationen kontrolliert, so daß Verbindungs-Verschlüsselung gegen ihn nicht hilft.

Die Idee für weit weniger aufwendige Verfahren besteht darin, das Kommunikationsnetz bereits physisch (bzw. – wesentlich aufwendiger – mittels Verbindungs-Verschlüsselung) so zu gestalten, daß es für einen Angreifer nicht einfacher ist, alle Ein- und Ausgänge einer Teilnehmerstation abzuhören bzw. die jeweils angrenzenden Stationen zu kontrollieren, als den Teilnehmer bzw. die Teilnehmerstation direkt zu beobachten. Wo immer der Angreifer mittels Abhören von Leitungen oder Kontrolle von Stationen Nachrichten beobachten kann, sollten diese von vielen (am besten von allen von ihm nicht kontrollierten Teilnehmerstationen) stammen und für viele (am besten für alle von ihm nicht kontrollierten Teilnehmerstationen) bestimmt sein können – beides geht natürlich nur, wenn der Angreifer nicht der Sender oder Empfänger der Nachrichten ist.

Eine wesentliche Voraussetzung hierzu ist **digitale Signalregenerierung**: Jede Station regeneriert von ihr empfangene und weitergeleitete Signale (im Falle zweiwertiger Signale: Bits) so, daß sie von entsprechenden (d.h. in der digitalen Welt gleichen) von ihr generierten Signalen (auch anhand analoger Charakteristika) nicht unterschieden werden können. Dies impliziert, daß auch einander entsprechende Signale, die eine Station von unterschiedlichen Stationen empfangen hat, nach ihrer Regenerierung nicht unterschieden werden können.

Nicht gefordert wird hingegen, daß entsprechende Signale, die von *verschiedenen* Stationen gesendet werden, nicht unterschieden werden können. Während letzteres wegen der analogen Charakteristika des Senders und ihrer bei jedem Produktionsprozeß unvermeidbaren Streuung eine praktisch und wegen der Änderung des Signals bei seiner Ausbreitung, z.B. breiten sich verschiedene Frequenzkomponenten verschieden schnell aus (Dispersion), auch theoretisch nicht erfüllbare Forderung wäre [Pfi1_83 Seite 17], ist das Geforderte vergleichsweise einfach zu realisieren.

In den folgenden zwei Unterabschnitten werden zwei Beispiele zu der eben geschilderten und begründeten Idee behandelt.

Das **RING-Netz** ist das Paradebeispiel, da es bei bezüglich aller Verteilnetze (für Empfängeranonymität) minimalem Aufwand die obige maximale Forderung bezüglich eines Angreifers, der eine beliebige Station kontrolliert, erfüllt. Der Aufwand des RING-Netzes ist in dem Sinne minimal, daß es aus Gründen der Empfängeranonymität nötig ist, daß jede Station jede Nachricht mindestens einmal empfängt (diese untere Grenze wird mit den in §5.4.4.1.1 beschriebenen effizienten anonymen Zugriffsverfahren erreicht) und daß es aus Gründen der Senderanonymität nötig ist, daß jede Station mindestens mit der Summenrate aller eigentlichen Senderaten sendet (wobei bei Punkt-zu-Punkt-Duplex-Kanälen beides ohne wesentliche Anonymitätseinbuße halbiert werden kann, wie mit dem paarweisen überlagernden Empfangen auf dem DC-Netz gezeigt werden wird und mit einem speziellen Verfahren zum Schalten von Punkt-zu-Punkt-Duplex-Kanälen auf dem RING-Netz in [Pfit_89 §3.1.4] gezeigt ist). Auch diese untere Grenze wird durch die erwähnten effizienten anonymen Zugriffsverfahren bis auf einen beliebig kleinen Verwaltungsaufwand (overhead) erreicht. Zusätzlich ist günstig, daß es im lokalen Bereich viel Erfahrung mit seiner physischen Struktur gibt

und zumindest in den USA Pläne zum Einsatz seiner Struktur auch als Großstadtnetz (Metropolitan Area Network = MAN, [Sze_85, Roch_87]) bestehen.

Das **BAUM-Netz** ist ein aus pragmatischen Gründen wichtiges Beispiel: Seine physische Struktur ist die der heute üblichen Breitbandkabelverteilnetze, die nach leichten Erweiterungen (und damit in kurzer Zeit) für die anonyme Kommunikation benutzt werden können. Das BAUM-Netz bietet zwar nur bezüglich eines schwächeren Angreifers als das RING-Netz Anonymität, es kann aber wie jenes für *überlagerndes Senden*, was bezüglich jedes Angreifers stärkstmögliche Anonymität bietet, erweitert werden. Wie in §5.4.5.5 gezeigt wird, ist die Struktur des BAUM-Netzes hierfür sogar noch besser geeignet als die des RING-Netzes.

5.4.4.1 Ringförmige Verkabelung (RING-Netz)

Die effizienteste Möglichkeit, ein Kommunikationsnetz bereits physisch so zu gestalten, daß es für einen Angreifer nicht einfacher ist, alle Ein- und Ausgänge einer Teilnehmerstation abzuhören bzw. die jeweils angrenzenden Stationen zu kontrollieren, als den Teilnehmer bzw. die Teilnehmerstation direkt zu beobachten, ist, die Teilnehmerstationen ringförmig anzuordnen, wie dies im Bereich lokaler Netze seit langem praktiziert wird. Da dank digitaler Signalregenerierung in jeder Teilnehmerstation aus den Signalformen keine Information über den ursprünglichen Sender gewonnen werden kann, müssen, um bei ringförmiger Anordnung der Teilnehmerstationen das Senden einer Station zu überwachen, ihre beiden Nachbarn zusammenarbeiten – alternativ kann auch jeweils die verbindende Leitung abgehört werden. Durch bauliche Maßnahmen, z.B. direkte Verkabelung von Wohnungen in Mehrfamilienhäusern sowie direkte Verkabelung aneinandergrenzender privater Grundstücke [Pfi1_83 Seite 18], ergänzt, keineswegs aber allein durch Verwendung eines möglichst schwierig abhörbaren Übertragungsmediums (Glasfaser), kann man erreichen, daß letzteres ebenso aufwendige physikalische Maßnahmen erfordert wie das direkte Abhören des Teilnehmers oder der Teilnehmerstation innerhalb der Wohnung. Damit verspricht es keinen zusätzlichen Vorteil. Versuchen die zwei Ringnachbarn eines Teilnehmers, diesen ohne Auftrag Dritter (d.h. hier ohne Zusammenarbeit mit großen Kommunikationspartnern) zu beobachten, so erfahren sie beinahe nichts, da alle ausgehenden Nachrichten verschlüsselt und die Adressen bei geeigneter Verwendung impliziter Adressierung für sie nicht interpretierbar sind.

Zu erwarten sind also im wesentlichen Angreifer, die nur eine Gruppe von vielen zusammenhängenden Stationen eingekreist haben. Sie können durch Vergleich der ein- und auslaufenden Signalmuster (für preiswerte Ringe: Bitmuster) nur feststellen, welche davon von den Mitgliedern dieser Gruppe gesendet bzw. vom Ring entfernt wurden, nicht aber von welchem Mitglied genau. Es gilt sogar die stärkere Aussage, daß diese Angreifer kein Mitglied der Gruppe als Sender oder Entferner der Signalmuster ausschließen können.

Senden Stationen auf dem gemeinsamen Verteil-Kanal „Ring“ unkoordiniert, so kann es Kollisionen von Nachrichten geben. Im Gegensatz zum in §5.4.5 beschriebenen DC-Netz und dem im folgenden §5.4.4.2 beschriebenen BAUM-Netz ist die Sichtweise von Empfänger und Sender einer Nachricht bezüglich einer eventuellen Kollision im RING-Netz nicht naturgegebenermaßen konsistent – die Kollision kann „nach“ dem Empfänger auftreten, so daß nur der Sender sie bemerkt. (Fehler, die auch beim DC- und BAUM-Netz eine inkonsistente Sicht von Empfänger und Sender bezüglich einer eventuellen Kollision verursachen können, seien hierbei zunächst einmal außer Betracht gelassen – sie treten hoffentlich um Größenordnungen seltener als Kollisionen auf und werden deshalb separat in §5.4.7 behandelt.)

Unter der (realistischen) Annahme, daß nie zwei verschiedene Nachrichten dieselbe Codierung haben, kann der Sender aber feststellen, daß der Empfänger die Nachricht empfangen konnte. Dies ist immer dann der Fall, wenn der Sender seine Nachricht nach einem Ringumlauf unverändert zurückerhält (und er sie mit einer anderen Nachricht oder 0 überschreiben kann). Durch die globale

Verabredung, daß Empfänger Nachrichten, die sie schon erhalten haben, für alle Zukunft ignorieren (durchaus nichts Ungewöhnliches in Rechnernetzen, wenn Nachrichten einen Zeitstempel enthalten), kann sich der Sender durch ggf. wiederholtes Senden Gewißheit über den Empfang seiner Nachricht verschaffen. Auf diese Art kann für Sender und Empfänger eine konsistente Sicht über die Tatsache des Empfangs hergestellt werden, nicht aber über den genauen Zeitpunkt. Sollte dieser wichtig sein oder aber eine stochastische Angriffsmöglichkeit zur Ermittlung des Senders einer Nachricht über ihre Senderate vermieden werden, so sind entweder geschicktere Zugriffsprotokolle oder eine andere Gestaltung des RING-Netzes nötig.

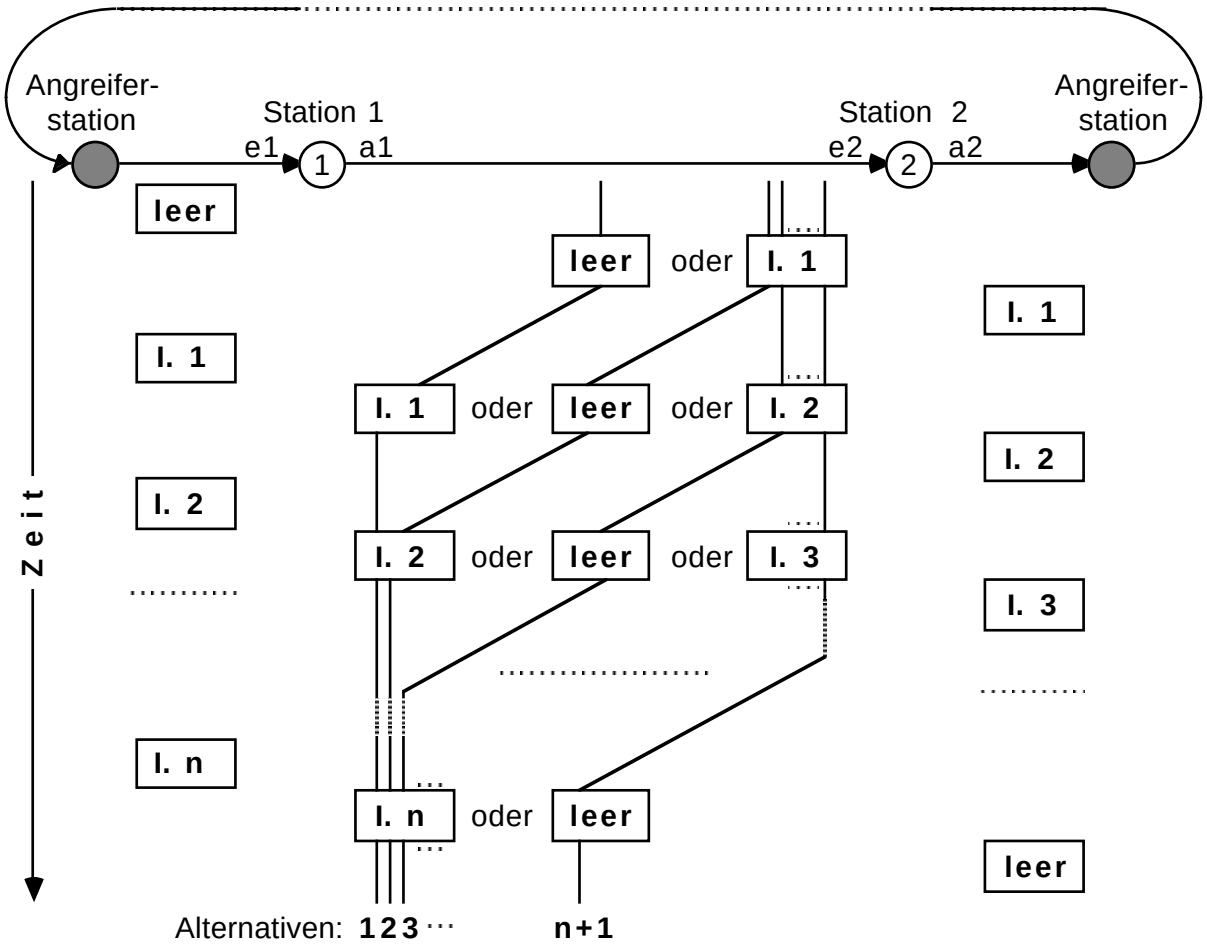
Einerseits bietet ein Ring im Gegensatz zum DC-Netz durch seine Struktur eine effiziente Möglichkeit, durch spezielle Verfahren zum Mehrfachzugriff, nämlich verteiltes und (wie in §5.4.4.1.1 gezeigt wird) anonymes Abfragen (distributed and unobservable polling [Pfi1_85 Seite 19]), Kollisionen vollständig zu vermeiden, was den gerade beschriebenen Nachteil einer möglicherweise inkonsistenten Sicht bezüglich Kollisionen und damit auch bezüglich des Zeitpunktes des Nachrichtenempfangs mehr als kompensiert. Sowohl das Senden von Nachrichten als auch das Schalten von Kanälen ist mit Hilfe dieser Zugriffsverfahren möglich. Bei Punkt-zu-Punkt-Duplex-Kanälen kann man sogar ohne große Anonymitätseinbußen darauf verzichten, daß alle Information einmal ganz um den Ring läuft. Statt dessen nimmt der Empfänger die Information vom Ring und ersetzt sie sofort durch Information für den Kommunikationspartner. Dadurch ist die Kapazität des Rings doppelt so gut nutzbar.

Andererseits bietet ein Ring auch bei unkoordiniertem Senden zwei Möglichkeiten, eine (abgesehen von Fehlern) konsistente Sicht von Empfänger und Sender bezüglich Empfang und Zeitpunkt des Empfangs einer Nachricht herzustellen [Pfit_89 Seite 150f]. Hierzu läuft die Nachricht zweimal um den Ring. Während des ersten Umlaufs können Kollisionen auftreten, wohingegen der zweite Umlauf zur Verteilung des Ergebnisses des ersten ohne Kollisionen abgewickelt werden muß und kann. Die beiden Möglichkeiten unterscheiden sich darin, ob die Umläufe bei einer festen und damit statisch ausgezeichneten Station oder jeweils beim Sender und damit bei einer nur dynamisch ausgezeichneten Station beginnen. Die statisch ausgezeichnete Station darf global bekannt sein, während dynamisch ausgezeichnete Stationen dies aus Gründen der Senderanonymität tunlichst nicht sein sollten.

5.4.4.1.1 Ein effizientes 2-anonymes Ringzugriffsverfahren

Ein Ring mit gegebenem Zugriffsprotokoll heiße ***n*-anonym**, wenn es keine Situation gibt, in der ein Angreifer bei Einkreisung von *n* hintereinander liegenden Stationen *durch beliebig viele Angreiferstationen* eine davon als Sender bzw. Empfänger einer Informationseinheit identifizieren kann.

Geeignet und effizient sind gewisse bekannte Verfahren (Ring mit umlaufenden Übertragungsrahmen = slotted ring, Ring mit umlaufendem Senderecht = token ring), die dahingehend abgewandelt wurden, daß Senderecht zeitlich unbefristet vergeben wird und daß jede Informationseinheit (Paket, Nachricht, Übertragungseinheit eines Kanals) einmal ganz um den Ring läuft. Ersteres bewirkt verteiltes (und wie zu zeigen ist: anonymes) Abfragen. Letzteres, d.h. daß Informationseinheiten nicht vom Empfänger, sondern erst vom Sender wieder vom Ring entfernt werden, realisiert bereits Verteilung, so daß neben dem Sender auch der Empfänger geschützt wird.



Effiziente Ringzugriffsverfahren (token, slotted) reichen ein zeitlich unbefristetes Senderecht von Station zu Station weiter. Wenn ein Angreifer 2 Stationen umzingelt hat, die nach Erhalt des Senderechts insgesamt n Informationseinheiten (I.) senden und dann das Senderecht weiterreichen, gibt es n+1 Alternativen, welche Station welche Informationseinheit gesendet haben kann. Für jede Informationseinheit gibt es mindestens eine Alternative, bei der Station i (i=1, 2) die Informationseinheit sendet.

Das Beispiel kann auf g umzingelte Stationen verallgemeinert werden. Es gibt $\binom{n+g-1}{n}$ Alternativen und zumindest eine Alternative für jede Informationseinheit, bei der Station i (i=1, 2, ...,g) die Informationseinheit sendet.

Bild 5-8: Ein anonymes Zugriffsverfahren für RING-Netze garantiert: ein Angreifer, der eine Folge von Stationen umzingelt hat, kann nicht entscheiden, welche was sendet.

Um die Unsicherheit des Angreifers bezüglich des tatsächlichen Ablaufs zu beweisen, werden für jeden Ablauf, in dem eine Informationseinheit gesendet wird, alternative Abläufe konstruiert. Diese alternativen Abläufe sind in Bild 5-8 dargestellt. Damit ist der Ring mit diesen Ringzugriffsverfahren als 2-anonym bewiesen.

Der in [HöPf_85, Höck_85] enthaltene formale Beweis auf Bitübertragungsebene entspricht völlig dem dargestellten Beweis auf Informationseinheitenebene.

Diese Beweise gelten natürlich nur für einen Angreifer, der nicht den Sender der Informationseinheiten kennt, die im alternativen Ablauf von einer anderen Station gesendet werden müssen. Diese Kenntnis könnte der Angreifer etwa dadurch erhalten, daß er selbst Empfänger einer solchen Informationseinheit ist und sich der Absender in der Informationseinheit zu erkennen gibt. Dies wird hier nicht modelliert, da es sich gemäß §5.4.9 um Einschränkungen möglicher Alternativen auf Schicht 2 durch höhere Schichten handelt.

5.4.4.1.2 Fehlertoleranz beim RING-Netz

Beim RING-Netz ist die Serieneigenschaft sofort ersichtlich: alle Leitungen und Stationen müssen funktionieren, damit Kommunikation zwischen zwei Stationen in beiden „Richtungen“ möglich ist.

Durch das beschriebene Zusammenspiel von ringförmiger Übertragungstopologie, digitaler Signalregenerierung und Verfahren zum anonymen Mehrfachzugriff, um innerhalb des Kommunikationsnetzes Anonymität und Unverkettbarkeit zu schaffen (vgl. §5.4.9), sind spezielle Fehlertoleranz-Maßnahmen erforderlich, die dieses Zusammenspiel nicht (zu sehr) stören.

Das Folgende konzentriert sich auf die Schichten 0, 1 und 2 des ISO OSI Referenzmodells, vgl. Bild 5-40 in §5.4.9.

Transiente Fehler des Zugriffsverfahrens (Schicht 2) können durch die bei Ringen mit umlaufenden Übertragungsrahmen oder umlaufendem Senderecht jeweils üblichen Verfahren zum Neustart des Zugriffsverfahrens toleriert werden.

Permanente Fehler des Zugriffsverfahrens (Schicht 2) implizieren, daß mindestens eine Station im RING-Netz fehlerhaft ist. Diese Station muß (und kann) mit den im folgenden geschilderten Verfahren solange ausgegliedert werden, bis sie repariert ist.

Transiente Fehler im Übertragungssystem des Rings (d.h. auf den Schichten 0 oder 1) betreffen entweder nur den Nutzinhalt übertragener Informationseinheiten oder auch das Zugriffsverfahren (Schicht 2). Ersterer Fehlertyp ist einfach und wird durch das Ende-zu-Ende-Protokoll erkannt und behoben. Schwierig sind diejenigen transienten Fehler, die das Zugriffsverfahren betreffen, z.B. bei einem Ring mit umlaufendem Senderecht das Senderechtszeichen zerstören. Dieser und ähnliche Fehler können bei Ringen mit umlaufenden Übertragungsrahmen oder umlaufendem Senderecht mit den jeweils üblichen Verfahren zum Neustart toleriert werden. Zwei zusätzliche Ideen sind bei der Fehlertoleranz des Zugriffsverfahrens (Schicht 2) wichtig:

1. Um transiente Fehler der zweiten Art auf den Schichten 0 oder 1 (siehe oben) tolerieren zu können, führt man bewußt Indeterminismus ein (*indeterministische Protokolle*, siehe [Mann_85 Seite 89ff]). Dadurch kann in dem deterministischen Angreifermodell (der Angreifer muß deterministisch schließen können, welche Station gesendet bzw. empfangen hat) kein Angreifer sichere Rückschlüsse ziehen. Da es bisher kein befriedigendes formales statistisches Angreifermodell gibt (darin gibt sich ein Angreifer zufrieden, wenn er den Sender bzw. Empfänger mit z.B. der Wahrscheinlichkeit 0,9 kennt, vgl. auch [Höck_85 Seite 60ff]), kann die Auswirkung des Indeterminismus nicht quantifiziert werden.
2. Eine andere Art der Fehlertoleranz basiert auf der Idee, einigen wenigen, ausgezeichneten Stationen keine Anonymität zu garantieren, etwa Protokollumsetzern in einem hierarchischen Kommunikationsnetz. Diese Stationen können nach anderen Protokollen arbeiten als die Stationen von Netzteilnehmern und dadurch gezielt Fehler tolerieren [Mann_85 Seite 99ff]. Daher spricht man von *asymmetrischen Protokollen*. Diese sind leichter zu implementieren und sehr viel leistungsfähiger, schwächen jedoch die Anonymität der Netzteilnehmer ab. Ist das einfache Zugriffsverfahren *n*-anonym, so sind die modifizierten asymmetrischen Protokolle häufig nur noch (n+1)- oder gar nur (n+2)-anonym.

Permanente Fehler im Übertragungssystem des Rings (d.h. auf den Schichten 0 oder 1) führen immer zu einer *Ringrekonfigurierung* mit Neustart (Synchronisation,...) des Verfahrens zum anonymen Mehrfachzugriff (Schicht 2). Da die zu tolerierenden Fehler auf den Schichten 0 (medium) und 1 (physical) angesiedelt sind, sind Fehler leicht zu erkennen und zu lokalisieren (Zeitschranken, Selbsttest sowie Fremdtest durch mehrere Instanzen, damit ein Angreifer, der wenige Stationen kontrolliert, nicht andere Stationen beliebig an- und abschalten kann, u.ä.). Problematisch ist die Fehlerbehebung, da für jede nachgewiesen werden muß, daß die Anonymität nicht verletzt wird. Ziel jeder Fehlerbehebung muß es sein, aus den fehlerfreien Stationen und Leitungen einen Ring (und nicht etwa eine

„Acht“, d.h. zwei durch eine Station verbundene Ringe etc.) zu rekonfigurieren, der alle fehlerfreien Stationen umfaßt.

Die einfachste und am wenigsten aufwendige Art der Ringrekonfigurierung ist die Verwendung einer **By-Pass-Einrichtung** (bypass) an jeder Station. Diese By-Pass-Einrichtung schließt den Ring physisch, wenn die Station bemerkt, daß sie fehlerhaft ist, ausgeschaltet wird oder ihre Versorgungsspannung ausfällt. Allerdings kann mit diesem Verfahren natürlich der Ausfall einer Leitung genausowenig toleriert werden wie Fehler von Stationen, die diese nicht selbst diagnostizieren können. Kann nur die Station selbst ihre By-Pass-Einrichtung schließen, so hat deren Einrichtung keine tieferen problematischen Wechselwirkungen mit der Anonymität oder Unbeobachtbarkeit des RING-Netzes. Ein Angreifer kann natürlich insbesondere das Schließen der By-Pass-Einrichtung angrenzender, von ihm nicht kontrollierter Stationen beobachten, da sich dann analoge Charakteristika seines Eingangssignals ändern (vgl. das in §5.4.4 über digitale Signalregenerierung Gesagte). Dann weiß er, daß jetzt eine kleinere Gruppe von Teilnehmerstationen umzingelt ist, wodurch er entsprechend mehr Information über das Senden dieser Teilnehmer erhält. Kann ein Angreifer aber nur viele Teilnehmerstationen gleichzeitig umzingeln und sind an diesen Teilnehmerstationen jeweils nur wenige By-Pass-Einrichtungen geschlossen, so ist dies nicht schlimm.

Die Verwendung von *Ring-Verkabelungs-Konzentratoren*, d.h. von By-Pass-Einrichtungen, die nicht bei der Teilnehmerstation gelegen sind und auch nicht von ihr kontrolliert werden [BCKK_83, KeMM_83], ist mit den in §5.4.4.1 erläuterten Zielen des RING-Netzes unverträglich. Denn Ring-Verkabelungs-Konzentratoren wären ideale Beobachtungspunkte für einen Angreifer, da sie die vollständige Beobachtung aller angeschlossenen Stationen durch Beobachtung dieses einen Gerätes erlauben.

Eine ebenfalls einfache, aber bereits aufwendige Art der Ringrekonfigurierung ist die Verwendung **mehrerer paralleler Ringe** (statisch erzeugte Redundanz). Fällt eine Leitung eines Ringes aus, so wird ein anderer, noch intakter Ring verwendet (dynamisch aktivierte Redundanz). Dies erlaubt, solange noch mehr als ein Ring intakt ist, einen größeren Durchsatz als das Senden jeder Informationseinheit auf allen Ringen gleichzeitig (statisch aktivierte Redundanz). Allerdings muß dann, sofern kein Ring mehr als Ganzes intakt ist, entweder die Zuordnung von Leitungen zu Ringen gewechselt oder doch auf allen Ringen gleichzeitig dasselbe gesendet werden. Allerdings kann natürlich auch mit diesem Verfahren ein Fehler einer Station, den diese nicht bemerkt, nicht toleriert werden.

Entsprechendes gilt für eine Kombination von By-Pass-Einrichtung und mehreren parallelen Ringen.

Um mindestens einen beliebigen permanenten Einzelfehler im Übertragungssystem des Rings tolerieren zu können, wird ein **geflochtener Ring** (braided ring, siehe Bild 5-9) verwendet. Solange keine permanenten Fehler auftreten, werden die Leitungen, die benachbarte Stationen verbinden, als ein RING-Netz betrieben. Für ungerade Stationsanzahlen können die anderen Leitungen als ein zweites RING-Netz betrieben werden, was die nutzbare Übertragungskapazität verdoppelt.

Im geflochtenen Ring gibt es drei mögliche *Einzelfehler*, nämlich den Ausfall

- einer *Station i*: sie wird mit Leitung $L_{i-1 \rightarrow i+1}$ überbrückt (Bild 5-9, oben rechts).
- einer *inneren Leitung* $L_{i-1 \rightarrow i+1}$: der äußerer Ring bleibt intakt und wird solange ausschließlich benutzt, bis der Ausfall der Leitung behoben ist.
- einer *äußeren Leitung* $L_{i-1 \rightarrow i}$: Station S_{i-2} sendet an die Stationen S_{i-1} und S_i (kopiert den Datenstrom auf zwei Leitungen). Damit Station S_{i+1} die Station S_i nicht beobachten kann, sendet Station S_{i-1} die Hälfte aller Informationseinheiten (z.B. alle Übertragungsrahmen mit gerader Nummer, sofern i-1 gerade) an Station S_{i+1}, entsprechend sendet Station S_i die andere Hälfte (z.B. alle Übertragungsrahmen mit ungerader Nummer, sofern i ungerade) an Station

S_{i+1}. S_{i+1} vereint die beiden (verzahnten) Datenströme wieder (Bild 5-9, unten rechts). Dies ist einer Rekonfigurierung des inneren Rings (Bild 5-9, unten links) vorzuziehen, da auf diese Weise ggf. noch Ausfälle von inneren Leitungen und Stationen toleriert werden können.

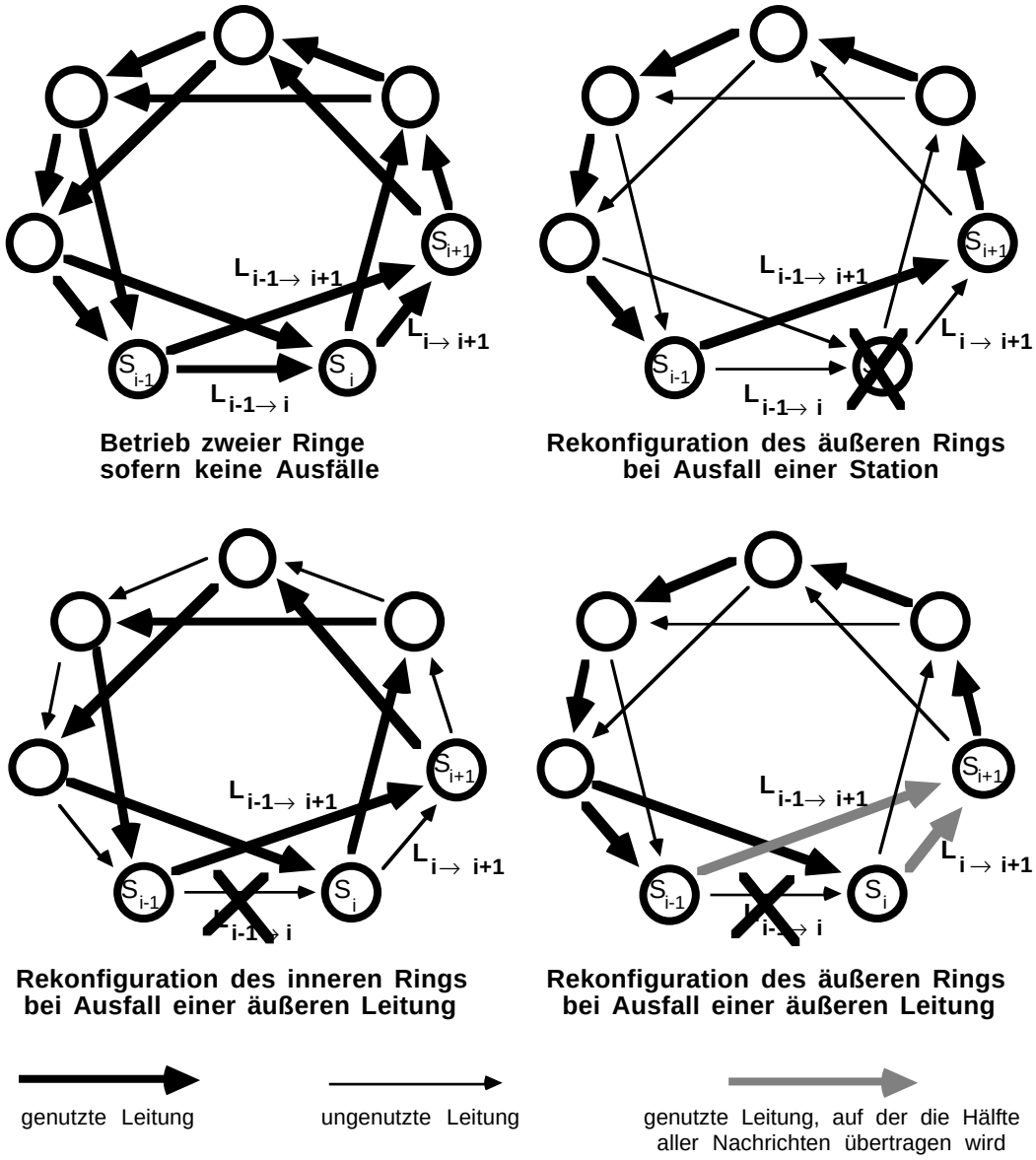


Bild 5-9: Geflochtener Ring kann bei Ausfällen von Stationen oder Leitungen so rekonfiguriert werden, daß die Anonymität der Netzbenutzer gewahrt bleibt.

Basierend auf diesen drei Grundkonzepten zur Tolerierung des Ausfalls kann der geflochtene Ring (auch bei den meisten Mehrfachfehlern) mittels des folgenden, von jeder Station auszuführenden Protokolls so rekonfiguriert werden, daß jede fehlerfreie Station jederzeit in einem rekonfigurierten, fast alle fehlerfreien Stationen umfassenden Ring liegt, d.h. auf der ein- und der auslaufenden Leitung werden Informationseinheiten von fast allen nichtfehlerhaften Stationen übertragen.

Rekonfigurationsprotokoll für den geflochtenen Ring:
Bemerkt eine Station S_i einen permanenten Signalausfall auf einer ihrer Eingangsleitungen, so signalisiert sie das mittels einer an alle anderen Stationen adressierten Nachricht auf ihren beiden Ausgangsleitungen. Solange der Signalausfall nicht behoben ist, ignoriert S_i diese Eingangsleitung.

Sendet die auf dieser Leitung sendende Station nicht auf ihrer anderen Leitung, daß sie in Ordnung ist, so wird ihr Ausfall unterstellt und sie wird mit der entsprechenden inneren Leitung überbrückt. Anderenfalls wird ein Leitungsausfall unterstellt und entsprechend rekonfiguriert.

Da beim Ausfall einer Station oder einer inneren Leitung ein vollständiger Ring aller fehlerfreien Stationen entsteht, lassen sich in beiden Fällen die Beweise für die Anonymität aus dem fehlerfreien Fall direkt übernehmen. Beim Ausfall einer äußeren Leitung entstehen bezüglich der Anonymität des Senders zwei RING-Netze mit je der halben Bandbreite. In Bild 5-9 führt eines der RING-Netze halber Bandbreite durch S_{i-1} und das andere durch S_i. Daß in beiden RING-Netzen halber Bandbreite jeweils noch eine Station (nur) empfangen kann, ist irrelevant, da der Empfänger in den normalen Ringzugriffsverfahren an den empfangenen Informationseinheiten nichts ändert. Wird aber – wie in [Pfit_89 §3.1.4.3] beschrieben – ein Übertragungsrahmen von einer anderen Station als Duplexkanal mit einer der Stationen S_{i-1} oder S_i verwendet, so kann die gerufene Station in diesem Übertragungsrahmen nur mit der Wahrscheinlichkeit 0,5 senden. Die Situation, daß eine nur in einem RING-Netz halber Bandbreite liegende Station von einer anderen in einem Übertragungsrahmen zum Senden aufgefordert wird, in dem sie nicht senden kann, mag ein seltenes Ereignis sein, kann aber die Anonymität gegenüber der rufenden Partei aufheben. Aber dies ist nur ein spezieller Fall des generellen Problems, daß ein Empfänger identifiziert werden kann, wenn er wegen des Ausfalls seiner Teilnehmerstation nicht antworten kann und seine Teilnehmerstation die einzige ausgefallene ist und der Sender dies weiß (vgl. den in §5.4.8 beschriebenen aktiven Verkettungsangriff über Betriebsmittelknappheit).

Außerdem ist (wie beim Verfahren der By-Pass-Einrichtung) zu beachten, daß aus einem berücksichtigten Angreifer im nicht fehlertoleranten Fall im fehlertoleranten Fall ein nicht berücksichtigter Angreifer (gemäß Angreifermodell) wird. Kontrolliert der Angreifer die Stationen S_1 und S_4 (vgl. Bild 5-9 mit i=2), so kann er den Ausfall der Leitung $L_{3\rightarrow4}$ vortäuschen oder abwarten. In beiden Fällen sendet Station S_2 sowohl an Station S_3 als auch an S_4. Damit ist Station S_2 durch den Angreifer eingekreist und daher beobachtbar. Es sei angemerkt, daß bei manchen Ausfällen sogar eine Station allein eine andere beobachten kann. Fällt S_1 so aus, daß sie auf ihren beiden Ausgangsleitungen dasselbe sendet, so kann die Station S_3 die Station S_2 allein beobachten.

Eine genaue Beschreibung von Fehlertoleranz im RING-Netz, sowie viele Beispiele und Protokolle sind in [Mann_85] zu finden. Ebenfalls dort wurde eine genaue Analyse der Zuverlässigkeitsverbesserung durch die beschriebenen Fehlertoleranzverfahren vorgenommen. Die Ergebnisse sind äußerst positiv.

Neuere Formeln zur Berechnung der Zuverlässigkeit sind in [Papa_86] zu finden.

Es ist bemerkenswert, daß das überlagernde Senden auf dem geflochtenen Ring so implementiert werden kann, daß ein auch nach einem beliebigen Einzelfehler mit der ganzen Bandbreite der einzelnen Leitungen arbeitendes DC-Netz entsteht [Pfit_89 Seite 264].

Eine Kombination von By-Pass-Einrichtung und geflochtenem Ring ist möglich und zweckmäßig.

5.4.4.2 Kollisionen verhinderndes Baumnetz (BAUM-Netz)

Fast alle im Teilnehmeranschlußbereich bereits vorhandenen Breitbandkabel sind baumförmig verlegt. Ein aus pragmatischen Gründen, nämlich ihrer Benutzung, wichtiges Beispiel für die Idee „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ ist deshalb ein Kollisionen verhinderndes Baumnetz (BAUM-Netz) [Pfit_86 Seite 357].

Die inneren Knoten des Baumnetzes werden mit **Kollisionen verhindernden Schaltern** (collision-avoidance circuits in [Alba_83], collision-avoidance switches in [SuSY_84]) ausgerüstet. Ein Kollisionen verhindernder Schalter schaltet einen aus Richtung der Blätter kommenden Informa-

tionsstrom nur dann in Richtung der Wurzel durch, wenn der Übertragungskanal in Richtung Wurzel frei ist. Bewerben sich mehrere Informationsströme aus Richtung der Blätter gleichzeitig um den freien Übertragungskanal in Richtung Wurzel, wählt der Kollisionen verhindernde Schalter genau einen zufällig aus und ignoriert die anderen. Sind Teilnehmerstationen auch direkt an Kollisionen verhindernde Schalter angeschlossenen, so daß es auch innere Teilnehmerstationen des BAUM-Netzes gibt, so werden von diesen kommende Informationsströme genauso wie aus Richtung der Blätter kommende behandelt.

Der von der Wurzel des Baumes durchgeschaltete Informationsstrom wird an alle Stationen verteilt, d.h. die Wurzel des Baumes und die Kollisionen verhindernden Schalter senden ihn auf allen Leitungen in Richtung der Blätter.

Um das Senden einer Teilnehmerstation an den Blättern des BAUM-Netzes beobachten zu können, muß eine spezielle Leitung beobachtet werden (beim RING-Netz müssen für alle Stationen je zwei spezielle Leitungen beobachtet werden); um das Senden einer inneren Teilnehmerstation des BAUM-Netzes beobachten zu können, müssen alle ihre Eingänge und ihr Ausgang beobachtet werden. Da üblicherweise fast alle Teilnehmerstationen Blätter des BAUM-Netzes sind und ein Angreifer durch Abhören einer Leitung oder durch Kontrolle eines Kollisionen verhindernden Schalters in jedem Fall das BAUM-Netz bezüglich der Senderanonymität partitioniert, ist die durch das BAUM-Netz realisierte Senderanonymität geringer als die des RING-Netzes. Wie bei diesem gilt auch hier, daß, wer ohne Auftrag Dritter (d.h. hier ohne Zusammenarbeit mit großen Kommunikationspartnern) versucht, seine Nachbarn zu beobachten, so gut wie nichts erfährt, da alle Nachrichten verschlüsselt und die Adressen bei geeigneter Verwendung impliziter Adressierung für ihn nicht interpretierbar sind.

Leistungsbewertungen [Alba_83, SuSY_84] attestieren mit Kollisionen verhindernden Schaltern ausgerüsteten Baumnetzen ein hervorragendes Leistungsverhalten auch bei völlig unkoordiniertem Zugriff.

Soll das BAUM-Netz durch *überlagerndes Senden* (wird in §5.4.5 erklärt) erweitert werden, so müssen die Kollisionen verhindernden Schalter durch modulo-Addierer ersetzt werden. Wie in [Pfit_89 Seite 105] ausführlich erklärt wird, entspricht jeder vom Angreifer nicht beobachteten Leitung ein ausgetauschter Schlüssel bezüglich des überlagernden Sendens, was auch beim BAUM-Netz – wie bei jedem Netz mit digitaler Signalregenerierung und auf die Übertragungstopologie abgestimmter Überlagerungstopologie – gilt.

Bei Verwendung von modulo-Addierern statt Kollisionen verhindernden Schaltern kann mittels überlagerndem Empfangen nicht nur bei allen Verkehrsarten ein gleich guter Durchsatz, sondern bei manchen Verkehrsarten sogar mittels paarweisem überlagerndem Empfangen der doppelte Durchsatz wie mit Kollisionen verhindernden Schaltern erzielt werden, vgl. §5.4.5.4. Dies spricht dafür, Kollisionen verhindernde Schalter als durch das überlagernde Empfangen „überholt“ zu betrachten.

5.4.5 Überlagerndes Senden (DC-Netz): Schutz des Senders

Nach einem ersten Überblick über das verallgemeinerte überlagernde Senden in §5.4.5.1 wird in §5.4.5.2 die Senderanonymität definiert und bewiesen. Danach wird in §5.4.5.3 gezeigt, wie mittels Knack-Schnapp-Verteilung auch die Empfängeranonymität gewährleistet werden kann.

In §5.4.5.4 wird mit dem überlagernden Empfangen ein Verfahren eingeführt, das eine wesentlich effizientere Nutzung des überlagernden Sendens erlaubt.

Schließlich werden in §5.4.5.5 einige Überlegungen zur Optimalität des überlagernden Sendens bezüglich Senderanonymität, sowie zu seinem Aufwand und zu möglichen Implementierungen angestellt. Abschließend werden in §5.4.5.6 Fehlertoleranz-Verfahren für das DC-Netz beschrieben.

5.4.5.1 Ein erster Überblick

In [Cha3_85, Cha8_85, Chau_88] gibt David Chaum eine von ihm DC-Netz genannte Möglichkeit zum anonymen Senden an (DC-network als Abkürzung für Dining Cryptographers network; der Name ist passend zu seinem Einführungsbeispiel gewählt; als zusätzliche Merkhilfe sei erwähnt, daß er David Chaums Initialen darstellt). Diese Möglichkeit zum anonymen Senden (und unbeobachtbaren Empfangen gemäß §5.4.1) wurde in [Pfi1_85 Seite 40, 41] zur im folgenden beschriebenen Möglichkeit verallgemeinert.

Bekanntlich bildet jedes endliche Alphabet bezüglich einer ab 0 beginnenden Numerierung seiner Zeichen und der Addition (modulo der Alphabetgröße) bezüglich dieser Numerierung eine abelsche Gruppe. Wie üblich werde unter der Subtraktion (eines Zeichens) die Addition seines inversen Gruppenelementes verstanden. Das **verallgemeinerte überlagernde Senden** geschieht dann folgendermaßen:

Teilnehmerstationen erzeugen für jedes zu sendende Nutzzeichen ein oder mehrere Schlüsselzeichen zufällig und gemäß einer Gleichverteilung. Jedes dieser Schlüsselzeichen teilen sie genau einer anderen Teilnehmerstation mittels eines (noch zu diskutierenden) Konzelation garantierenden Kanals mit. (Wer mit wem solch einen Konzelation garantierenden Kanal unterhält, muß nicht geheimgehalten werden, sondern sollte sogar, um Angriffen gegen die Diensterbringung leichter begegnen zu können, öffentlich bekannt sein, vgl. §5.4.7.) Jede Teilnehmerstation addiert (modulo Alphabetgröße) lokal alle von ihr erzeugten Schlüsselzeichen, subtrahiert (modulo Alphabetgröße) davon lokal alle ihr mitgeteilten Schlüsselzeichen und addiert (modulo Alphabetgröße), sofern sie ein Nutzzeichen senden will, lokal ihr Nutzzeichen. Dieses Addieren (bzw. Subtrahieren) modulo der Alphabetgröße des verwendeten Alphabets wird *Überlagern* genannt. Jede Teilnehmerstation sendet das Ergebnis ihrer lokalen Überlagerung (daher der den Mechanismus betonende Name *überlagerndes Senden*). Alle gesendeten Zeichen werden global überlagert (modulo Alphabetgröße addiert) und das entstehende Summenzeichen an alle Teilnehmerstationen verteilt.

Da jedes Schlüsselzeichen genau einmal addiert und subtrahiert wurde, sich nach der globalen Überlagerung also alle Schlüsselzeichen gegenseitig wegheben, ist das Summenzeichen die Summe (modulo Alphabetgröße) aller gesendeten Nutzzeichen. Wollte keine Teilnehmerstation senden, ist das Summenzeichen das 0 entsprechende Zeichen, wollte genau eine Teilnehmerstation senden, ist das Summenzeichen gleich dem gesendeten Nutzzeichen.

Wählt man als Alphabet die Binärzeichen 0 und 1, so erhält man den für praktische Zwecke wichtigen, von David Chaum angegebenen Spezialfall des **binären überlagernden Sendens**, bei dem zwischen Addition und Subtraktion von Zeichen nicht zu unterschieden werden braucht (Bild 5-10).

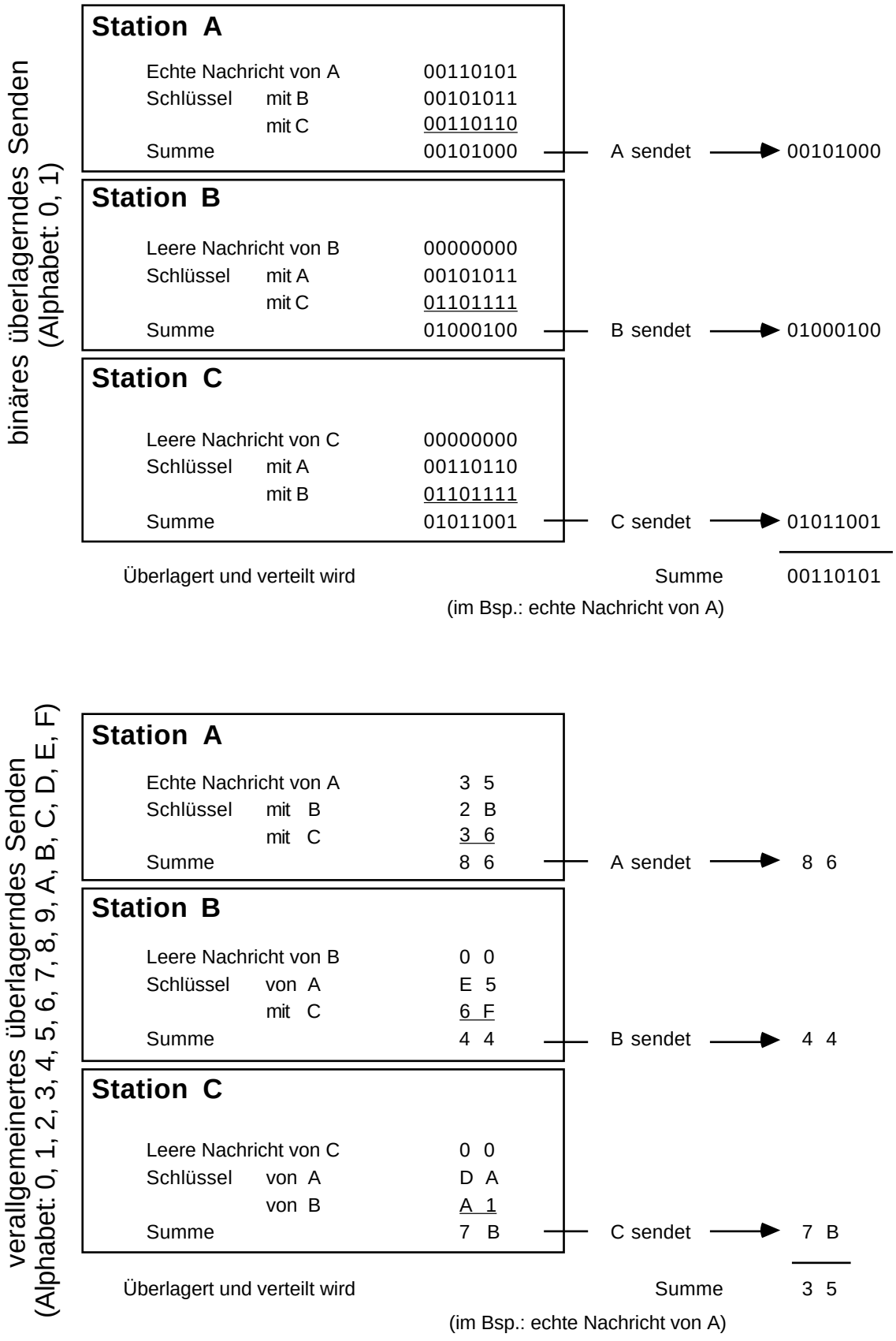


Bild 5-10: Überlagerndes Senden

Natürlich können (digitale) *Überlagerungs-Kollisionen* auftreten, falls mehrere Teilnehmerstationen gleichzeitig senden wollen: alle Stationen erhalten die (wohldefinierte) Summe des gleichzeitig Gesen-

deten. Kollisionen sind ein übliches Problem bei Verteil-Kanälen mit Mehrfachzugriff, zu dessen Lösung es eine große Zahl von Zugriffsverfahren gibt. Alle publizierten Zugriffsverfahren sind auf (analoge) *Übertragungs-Kollisionen*, z.B. in Bussystemen (Bsp. Ethernet), Funk- und Satellitennetzen ausgelegt, bei denen es kein wohldefiniertes „Kollisionsergebnis" gibt. Die „Arbeitsbedingungen" der Zugriffsverfahren sind beim überlagernden Senden in dieser Hinsicht also wesentlich besser als bei üblichen Verteil-Kanälen. Allerdings darf man natürlich nur solche Zugriffsverfahren verwenden, die die Anonymität des Senders und – wenn möglich – auch die Unverkettbarkeit von Sendeereignissen erhalten. Daneben sollten sie bei zu erwartender Verkehrsverteilung den zur Verfügung stehenden Kanal günstig nutzen. Beispiele anonymer und nichtverkettender Zugriffsverfahren sind das einfache, aber nicht sehr effiziente Verfahren slotted ALOHA und eine für Kanäle mit großer Verzögerungszeit entworfene, effiziente Reservierungstechnik [Tane_81 Seite 272, Cha3_85].

Vereinbart man, daß ein Teilnehmer einen Übertragungsrahmen, in dem er ohne Kollision gesendet hat, weiterbenutzen darf und andere Teilnehmer in diesem Rahmen erst wieder senden dürfen, wenn er einmal nicht benutzt wurde, so kann man durch die Verwendung mehrerer Übertragungsrahmen (slots) sehr effizient Kanäle schalten [Höck_85, HöPf_85]. Natürlich ist alles, was über solch einen Kanal gesendet wird, verkettbar – andererseits werden Kanäle typischerweise für solche Dienste verwendet, in deren Natur dies liegt. Dies wird ausführlich in §5.4.9 diskutiert.

5.4.5.2 Definition und Beweis der Senderanonymität

Nach dieser dem Überblick dienenden Kurzbeschreibung des überlagernden Sendens wird zunächst die *Senderanonymität* genau definiert und für den allgemeinst-möglichen Fall der Überlagerung (beliebige abelsche Gruppe) bewiesen.

Danach wird gezeigt, wie mittels *überlagerndem Empfangens* ausgenutzt werden kann, daß Überlagerungs-Kollisionen einen wohldefinierten und für alle am überlagernden Senden beteiligten Stationen wahrnehmbaren Wert haben. Beides im Gegensatz zu Übertragungs-Kollisionen in Bussystemen (Bsp. Ethernet) und Funknetzen und nur ersteres auch im Gegensatz zu Satellitennetzen.

Einige Bemerkungen zur Realisierung des Schlüsselaustausches und zur Implementierung des überlagernden Sendens schließen diesen Abschnitt ab.

David Chaum beweist in [Cha3_85] für binäres überlagerndes Senden (genauer: unter Ausnutzung der Körpereigenschaften von GF(2)), daß, solange eine Gruppe von Teilnehmerstationen Schlüssel nur in der beschriebenen Form weitergibt und solange diese Gruppe bezüglich der ausgetauschten Schlüssel zusammenhängend ist, andere an diesem Verfahren Beteiligte auch durch Zusammentragen all ihrer Information keine Information darüber erhalten, wer innerhalb der Gruppe sendet. „Bezüglich der ausgetauschten Schlüssel zusammenhängend" bedeutet, daß für je zwei beliebige Teilnehmerstationen T_0, T_1 der Gruppe es eine möglicherweise leere Folge von Teilnehmerstationen T_2 bis T_n der Gruppe gibt, so daß für $1 \le i \le n$ gilt: T_i hat mit $T_{(i+1) \bmod (n+1)}$ einen Schlüssel ausgetauscht.

Im folgenden wird ein vereinfachter und auf verallgemeinertes überlagerndes Senden erweiterter Beweis dieses Sachverhaltes gegeben.

Der Beweis verwendet nur, daß die Zeichen und die auf ihnen definierte Addition eine *abelsche Gruppe* bilden, nicht aber, daß die Addition nach der Numerierung der Zeichen modulo der Alphabetgröße durchgeführt wird (mit anderen Worten: daß es sich um eine *zyklische Gruppe* handelt). Die dadurch beim verallgemeinerten überlagernden Senden implizit gegebene *Ringstruktur* wird für den Beweis nicht verwendet, so daß er so allgemein wie irgend wünschenswert ist:

1. Damit eine Station kein Zeichen senden kann, muß es ein *neutrales Element* geben.
2. Damit sich die Schlüssel nach der Überlagerung paarweise wegheben, muß es zu jedem Zeichen ein *inverses* geben.

3. Damit man Freiheiten bei der Organisation der lokalen und insbesondere globalen Überlagerung hat, sollte die Addition kommutativ sein.
   Damit ein Schlüsselaustausch zwischen beliebigen Paaren von Teilnehmerstationen möglich ist, muß die Addition *kommutativ* sein. Denn anderenfalls gäbe es eine durch die globale Überlagerungsreihenfolge gegebene lineare Ordnung der von den Teilnehmerstationen gesendeten Nutzzeichen und damit auch der Teilnehmerstationen. Da Schlüsselzeichen nicht mit Nutzzeichen kommutierten, könnte jede Teilnehmerstation nur mit ein bzw. zwei bezüglich der linearen Ordnung benachbarten Teilnehmerstationen Schlüssel austauschen.

Der Beweis ist also etwas allgemeiner als die Definition des verallgemeinerten überlagernden Sendens – allerdings gibt es für diese größere Allgemeinheit keine Nutzanwendung. Denn der Hauptsatz über endliche abelsche Gruppen (vgl. [Waer_67 Seite 9]) besagt, daß jede endliche abelsche Gruppe als direktes Produkt zyklischer Gruppen geschrieben werden kann. Das direkte Produkt, d.h. die komponentenweise Ausführung der Addition in der jeweiligen zyklischen Gruppe, entspricht aber genau dem kollateralen, d.h. seriellen oder parallelen Betrieb mehrerer verallgemeinerter überlagernder Sendeverfahren.

Beh. Überlagern bezüglich der konzeliert ausgetauschten Schlüssel zusammenhängende Teilnehmerstationen jeweils ihre Nachrichten und die ihnen bekannten Schlüssel und geben nur das Ergebnis aus, so *erfährt* ein alle Ergebnisse beobachtender Angreifer *nur* die Summe aller gesendeten Nachrichten. Letzteres bedeutet, daß für den Angreifer für alle Zufallsvariablen Z gilt: W(Z|alle Ergebnisse) = W(Z|Summe aller gesendeten Nachrichten).

Anm. Je nach Vorwissen des Angreifers kennt er damit möglicherweise auch die die Summe bildenden einzelnen Nachrichten sowie, sofern einzelne Nachrichten mit dem Vorwissen des Angreifers Rückschlüsse auf ihren Sender zulassen, den Sender dieser Nachrichten. Der Angreifer *erhält* aber durch Kenntnis aller Ergebnisse, d.h. der Ausgaben der einzelnen Teilnehmerstationen, keine über die Kenntnis ihrer Summe hinausgehende *zusätzliche* Information. Er kann sich also das Beobachten der Ausgaben der einzelnen Teilnehmerstationen sparen, da ihm das keine zusätzliche Information darüber liefert, welche Teilnehmerstation welche Nachricht gesendet hat. Gemäß der Begriffsbildung in §5.1.2 bleiben die Sender von Nachrichten also anonym bezüglich des Angreifers und des Ereignisses des Sendens.

Bew. O.B.d.A. wird der Beweis im folgenden auf solche Situationen beschränkt, in denen die m Teilnehmerstationen bezüglich der ausgetauschten Schlüssel *minimal* zusammenhängen[107]. Selbiges bedeute, daß es für je zwei beliebige Teilnehmerstationen T_0, T_1 *genau eine* möglicherweise leere Folge von Teilnehmerstationen T_2 bis T_n gibt, so daß für $1 \le i \le n$ gilt: T_i hat mit $T_{(i+1) \bmod (n+1)}$ einen Schlüssel ausgetauscht. Gibt es mehrere Folgen, so werden dem Angreifer weitere Schlüssel mitgeteilt, was ihn nur stärker macht. Er kann dann diese Schlüssel von den Ausgaben der betroffenen Teilnehmerstationen subtrahieren, wodurch er jeweils genau die „Ausgabe“ erhält, die im folgenden durch vollständiges Weglassen dieser Schlüssel entsteht.

Der Beweis selbst wird mit **vollständiger Induktion** über die Anzahl m der Teilnehmerstationen geführt.

---

107 Anschaulich bedeutet dies, daß die Teilnehmerstationen bezüglich der ausgetauschten Schlüssel zusammenhängen, aber kein Schlüssel mehr entfernt werden kann, ohne daß danach mindestens 2 Teilnehmerstationen nicht mehr zusammenhängen. Für den aus den Teilnehmerstationen als Knoten und Schlüsseln als Kanten gebildeten Graphen bedeutet dies: Er ist zyklenfrei.

Unter einer *Nachrichtenkombination* wird die Zuordnung von je einer Nachricht zu jeder Teilnehmerstation, unter einer *Schlüsselkombination* die Zuordnung von Werten zu allen zwischen Teilnehmerstationen konzeliert ausgetauschten und dem Angreifer unbekannten Schlüsseln verstanden. Für jedes m wird bewiesen, daß es zu jeder die Summe aller gesendeten Nachrichten ergebenden Nachrichtenkombination genau eine Schlüsselkombination gibt, so daß Nachrichtenkombination und Schlüsselkombination mit den Ausgaben aller Teilnehmerstationen verträglich sind. Da zusätzlich die Schlüssel und damit auch die Schlüsselkombinationen zufällig und gemäß einer Gleichverteilung generiert werden, liefert die Beobachtung der Ausgaben der Teilnehmerstationen dem Angreifer keine über die Summe aller gesendeten Nachrichten hinausgehende Information gemäß den Shannon'schen Definitionen [Shan_48, Sha1_49].

Ein von T_i generierter und T_j konzeliert mitgeteilter Schlüssel werde mit $S_{i\to j}$ bezeichnet, die von T_i gesendete Nachricht mit N_i und ihre Ausgabe mit A_i, wobei nach der Beschreibung des verallgemeinerten überlagernden Sendens gilt:

$$A_i \;=\; N_i \;+\; \sum_j S_{i\to j} \;-\; \sum_j S_{j\to i}$$

Induktionsanfang: $m = 1$
Die Behauptung gilt trivialerweise, da der Angreifer genau die Ausgabe der einzigen Teilnehmerstation erfährt.

Induktionsschritt von $m - 1$ auf m:
Da die Teilnehmerstationen bezüglich der ausgetauschten Schlüssel *minimal* zusammenhängen, gibt es stets eine Teilnehmerstation, die nur mit einer anderen Teilnehmerstation durch einen ausgetauschten Schlüssel verbunden ist. Erstere werde o.B.d.A. mit T_m, letztere mit T_i mit $1 \le i \le m-1$ und der Schlüssel o.B.d.A. mit $S_{i\to m}$ bezeichnet (vgl. Bild 5-11).
T_i bildet N_i und $A_i = N_i+..+S_{i\to m}$, T_m bildet N_m und $A_m = N_m-S_{i\to m}$.
Der Angreifer beobachtet das Senden von $A_1, A_2,.., A_m$.
Sei $N' = (N'_1, N'_2,.., N'_m)$ eine beliebige Nachrichtenkombination, deren Summe gleich der Summe aller gesendeten Nachrichten ist, d.h. $N'_1+N'_2+..+N'_m = N_1+N_2+..+N_m$ $(= A_1+A_2+..+A_m)$.
Es muß nun gezeigt werden, daß es zur Nachrichtenkombination N' genau eine passende Schlüsselkombination S' gibt. Der zwischen T_i und T_m ausgetauschte passende Schlüssel $S'_{i\to m}$ hat wegen $A_m = N_m-S_{i\to m}$ den Wert $S'_{i\to m} = N'_m-A_m$.
Der Rest der passenden Schlüsselkombination S' wird wie in der Induktionsvoraussetzung bestimmt, wobei hierfür als Ausgabe von T_i der Wert $A_i - S'_{i\to m}$ verwendet wird. Die Induktionsvoraussetzung ist anwendbar, da die restlichen m-1 Teilnehmerstationen bezüglich der restlichen Schlüssel wiederum minimal zusammenhängen. □

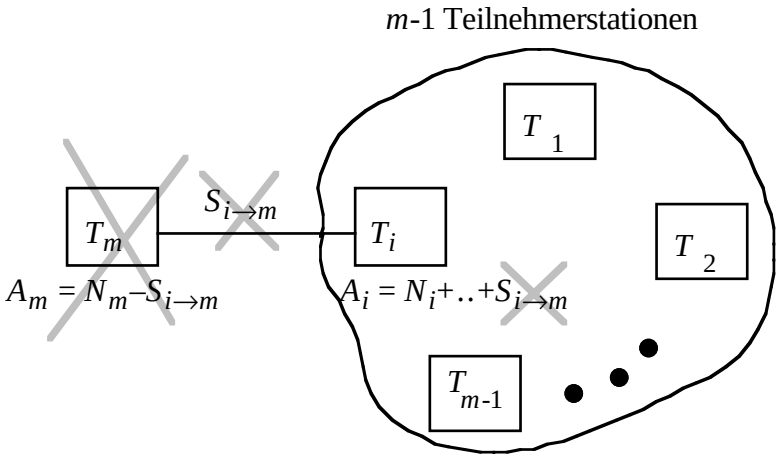


Bild 5-11: Veranschaulichung des Induktionsschrittes

Hiermit ist bewiesen, daß das verallgemeinerte (und damit natürlich auch das binäre) überlagernde Senden *perfekte informationstheoretische Anonymität* des Senders innerhalb der Gruppe schafft, die über rein zufällig gewählte und in perfekter informationstheoretischer Konzelation garantierender Weise ausgetauschte Schlüssel zusammenhängt.

Ebenso sind Nachrichten über das Senden in keiner Weise verkettbar, solange wir Einflüsse des Mehrfachzugriffsverfahrens ignorieren, vgl. §5.4.5.4.1 und §5.4.9. Überlagerndes Senden ermöglicht also auch *perfekte informationstheoretische Unverkettbarkeit* von Sendeereignissen. Erleidet eine Nachricht eine Überlagerungs-Kollision, so muß sie allerdings, bevor sie erneut gesendet wird, neu und damit anders (Ende-zu-Ende-)verschlüsselt werden – anderenfalls könnte ein Angreifer bei genügend langen Nachrichten beispielsweise schließen, daß Nachrichten, deren Summe vorher schon einmal gesendet wurde, wohl nicht von der gleichen Teilnehmerstation gesendet wurden, da diese anderenfalls Übertragungsbandbreite des DC-Netzes verschwenden würde, indem sie Überlagerungs-Kollisionen produziert.

David Chaums Beweis für das binäre und der hier gegebene für das verallgemeinerte überlagernde Senden gelten nicht nur bei *beobachtenden*, sondern auch bei koordinierten *verändernden* Angriffen – im hier gegebenen Beweis z.B. kommt das Verhalten der Angreiferstationen gar nicht vor, ihr Verhalten ist also beliebig. (Nicht protokollgemäßes Verhalten der Angreiferstationen stellt „nur" einen Angriff auf die Diensterbringung, d.h. Verfügbarkeit, dar, was in §5.4.7 behandelt wird.)

Jedoch betrachten beide Beweise jeweils nur die Anonymität des *Senders*. Ist das Senden einer Nachricht N abhängig vom Empfangen vorhergehender Nachrichten N_i, so kann ein verändernder Angriff auf die Anonymität des *Empfängers* einer der N_i (vgl. §5.4.1) trotz noch so perfekten Schutzes des Sendens den Sender von N identifizieren. Wie bereits in §5.4.1 erwähnt kann dieses Problem durch geeignetes Einbeziehen der verteilten Nachrichten in die Schlüsselgenerierung selbst innerhalb der informationstheoretischen Modellwelt perfekt gelöst werden [Waid_90, WaPf1_89].

5.4.5.3 Empfängeranonymität durch Knack-Schnapp-Verteilung

Das Ziel der Knack-Schnapp-Verteilung ist, die Nachrichtenübertragung zu stoppen, sobald zwei nichtangreifende Teilnehmer in einer Runde des DC-Netzes unterschiedliche Zeichen als globales Überlagerungsergebnis erhalten.[108]

---

108 Gemäß der Gliederung von §5.4 nach den Schutzzielen „Schutz des Empfängers", „Schutz des Senders" und „Schutz der Kommunikationsbeziehung" gehört dies nicht unter „Schutz des Senders", sondern unter „Schutz des Empfängers", also in §5.4.1.1. Dort wäre dies Verfahren, da es auf dem Senderanonymitätsverfahren „Überlagerndes Senden" beruht, allerdings unverständlich. Deshalb wird es hier beschrieben.

Entdeckt ein nichtangreifender Teilnehmer T_i eine solche Inkonsistenz, ist das Stoppen der Nachrichtenübertragung leicht: T_i stört einfach das überlagernde Senden in den nächsten Runden, indem er sein Ausgabezeichen zufällig und gleichverteilt wählt. Damit ist das globale Überlagerungsergebnis dieser Runden unabhängig von den Nutzzeichen.

In §5.4.5.3.1 wird die offensichtliche, aber ineffiziente Implementierung dieser Idee mittels eines Vergleichsprotokolls diskutiert.

In §5.4.5.3.2 und §5.4.5.3.3 werden Knack-Schnapp-Schlüsselgenerierungsverfahren beschrieben: Sie generieren Schlüssel für das überlagernde Senden in Abhängigkeit der zuvor erhaltenen globalen Überlagerungsergebnisse und stellen sicher, daß zwei Teilnehmer, die inkonsistente Überlagerungsergebnisse erhalten haben, danach völlig unabhängige Schlüssel benutzen und damit die Nachrichtenübertragung stoppen.

5.4.5.3.1 Vergleich der globalen Überlagerungsergebnisse

Um Inkonsistenzen der Verteilung zu entdecken, können die Teilnehmer die erhaltenen globalen Überlagerungsergebnisse mittels eines zusätzlichen Protokolls explizit vergleichen: Nach jeder Runde des überlagernden Sendens sendet jede Teilnehmerstation T_i ihr globales Überlagerungsergebnis I_i (input) allen anderen Teilnehmerstationen T_j mit $j > i$. Wenn eine nichtangreifende Teilnehmerstation T_j ein globales Überlagerungsergebnis ungleich I_j oder nichts von einer anderen Teilnehmerstation T_i mit $i < j$ erhält, stört sie das überlagernde Senden in allen folgenden Runden.

Solche Testphasen sind von Byzantinischen Übereinstimmungsprotokollen wohlbekannt.

Um diese Tests sicher zu machen, muß die Kommunikation zwischen T_i und T_j mit einem perfekten Authentikationscode geschützt sein [GiMS_74, Sim3_88], d.h. mit einem Code, der einem Angreifer das erfolgreiche Fälschen einer Nachricht höchstens mit der Wahrscheinlichkeit $1 / \sqrt{|F|}$, erlaubt, wenn F der Schlüsselraum ist, vgl. §3.3. Eine zusätzliche Nachricht und ein zusätzlicher Schlüssel sind daher für jeden Test nötig.

Die Zahl der Tests kann abhängig von der unterstellten Stärke des Angreifers gewählt werden: Sei G^* ein ungerichteter Graph, dessen Knoten die Teilnehmerstationen sind. Zwei Teilnehmerstationen T_i und T_j sind in G^* direkt verbunden gdw. T_i und T_j ihre globalen Überlagerungsergebnisse vergleichen. In Analogie zum überlagernden Senden gilt folgendes Lemma 1:

Lemma 1 Sei A die Teilmenge der Teilnehmerstationen, die vom Angreifer kontrolliert werden, und sei $G^* \setminus (T \times A)$ zusammenhängend.

Wenn zwei nichtangreifende Teilnehmerstationen T_i und T_j verschiedene globale Überlagerungsergebnisse I_i und I_j erhalten, dann gibt es ein Paar nichtangreifender Teilnehmerstationen $T_{i'}$ und $T_{j'}$, die in G^* direkt verbunden sind und auch verschiedene globale Überlagerungsergebnisse erhalten.

Deshalb entdeckt entweder $T_{i'}$ oder $T_{j'}$ die Inkonsistenz und stört daraufhin das überlagernde Senden.

Bew. Wegen des Zusammenhangs von $G^* \setminus (T \times A)$ existiert ein Pfad $(T_i = T_{k_1}, \ldots, T_{k_m} = T_j)$ mit $T_{k_z} \notin A$ und $(T_{k_z}, T_{k_{z+1}}) \in G^* \setminus (T \times A)$. Da nach Annahme $I_i \neq I_j$ ist, existiert ein Index z mit $I_{k_z} \neq I_{k_{z+1}}$. Wähle $(i', j') = (k_z, k_{z+1})$. □

Offensichtlich ist der Zusammenhang von $G^* \setminus (T \times A)$ eine notwendige Bedingung.

Das Protokoll erfordert in jeder Runde $|G^*|$ zusätzliche Nachrichten, was üblicherweise in der Größenordnung $O(n^2)$ ist. Falls $G = G^*$, und falls angenommen wird, daß für jede Testnachricht ein Schlüssel aus F zur Authentikation nötig ist, wird die Zahl der Schlüssel, die in Konzelation und

Authentizität garantierender Weise ausgetauscht werden müssen, im Vergleich zu normalem überlagerndem Senden verdoppelt.

Handelt es sich um ein physisches Verteilnetz[109], kann die Zahl der Testnachrichten auf $O(n)$ reduziert werden, indem statt eines Authentikationscodes ein digitales Signatursystem verwendet wird [DiHe_76, GoMR_88]. Das resultierende Schema ist natürlich höchstens komplexitätstheoretisch sicher.

5.4.5.3.2 Deterministische Knack-Schnapp-Schlüsselgenerierung

Eine effizientere Realisierung von Knack-Schnapp-Verteilung wird durch Kombination der beiden Aufgaben erreicht, Inkonsistenzen zu entdecken und das überlagernde Senden zu stoppen: Hängen die für das überlagernde Senden verwendeten aktuellen Schlüssel K_{ij} und K_{ji} vollständig (aber nicht ausschließlich) von den früheren globalen Überlagerungsergebnissen ab, die T_i und T_j erhalten haben, dann stoppt eine Inkonsistenz zwischen I_i und I_j automatisch das überlagernde Senden, indem das globale Überlagerungsergebnis nicht mehr die Summe aller Nutzzeichen, sondern zufällig ist.

Sei $\delta_{ij}^t := K_{ij}^t - K_{ji}^t$ und $\varepsilon_{ij}^t := I_i^t - I_j^t$ für alle i, j, t. (Hierbei ist t jeweils kein Exponent, sondern ein „oben“ geschriebener Index.) Ein Knack-Schnapp-Schlüsselgenerierungsschema für überlagerndes Senden sollte für alle in G direkt verbundenen Teilnehmerstationen T_i und T_j garantieren:

ÜS *Überlagerndes Senden:* Wenn für alle Runden $s = 1, \ldots, t\text{-}1$ die Gleichung $I_i^s = I_j^s$ gilt, sind die Schlüssel K_{ij}^t und K_{ji}^t für Runde t *gleich* und zufällig in F gewählt. Formaler (es bedeute $x \in_R M$, daß das Element x gleichverteilt zufällig und unabhängig von allen andern aus der Menge M gewählt wird):

$$[\forall\, s \in \{1, \ldots, t\text{-}1\}: \varepsilon_{ij}^s = 0] \Rightarrow K_{ij}^t \in_R F \text{ und } \delta_{ij}^t = 0$$

Dann funktioniert überlagerndes Senden wie üblich.

KS *Knack-Schnapp:* Gibt es einen Index $s < t$ mit $I_i^s \neq I_j^s$, dann sind die Schlüssel K_{ij}^t und K_{ji}^t für Runde t *unabhängig* und zufällig in F gewählt. Formaler:

$$[\exists\, s \in \{1, \ldots, t\text{-}1\}: \varepsilon_{ij}^s \neq 0] \Rightarrow K_{ij}^t \in_R F \text{ und } \delta_{ij}^t \in_R F$$

Das überlagernde Senden wird von jedem solchen Schlüsselpaar gestoppt, d.h. das Ergebnis der globalen Überlagerung ist unabhängig von den gesendeten Nutzzeichen. Wegen des Zusammenhangs von $G \setminus (T \times A)$ realisiert dies die Knack-Schnapp-Eigenschaft von Lemma 1 (mit $G = G^*$).

Im Rest dieses Abschnitts wird ein beliebiges, aber festes Schlüsselpaar (K_{ij}, K_{ji}) mit $T_i \notin A$ und $T_j \notin A$ betrachtet. Deshalb werden die Indizes i, j oftmals weggelassen.

Es wird der mächtigste Angreifer unterstellt: er kann die Werte von K_{ij}^t und K_{ji}^t für jede Runde t direkt beobachten und er kann T_i und T_j mit beliebigen globalen Überlagerungsergebnisse I_i^t und I_j^t versorgen. Die Teilnehmerstationen T_i und T_j werden als unsynchronisiert angenommen, so daß der Angreifer auf K_{ij}^{t+1} warten kann, bevor er T_j mit I_j^t versorgt.

Sei $(F, +, \bullet)$ ein endlicher Körper und seien $a^1, a^2, \ldots, a^{t_{max}}$ und $b^1, b^2, \ldots, b^{t_{max}-1}$ zwei Folgen, deren Elemente in F zufällig gewählt und in Konzelation und Authentizität garantierender Weise zwischen T_i und T_j ausgetauscht wurden. Sei für $t = 1, \ldots, t_{max}$ definiert

$$K_{ij}^t := a^t + \sum_{k=1}^{t-1} b^{t-k} \bullet I_i^k \qquad (1)$$

---

109 bei dem die Konsistenz der Verteilung nicht garantiert ist, sonst könnten wir uns den ganzen Zauber sparen

$$K_{ji}^{\,t} := a^t + \sum_{k=1}^{t-1} b^{\,t-k} \bullet I_j^{\,k}$$

Lemma 2 Das durch Gleichung (1) definierte Schlüsselgenerierungsschema erfüllt die zwei obigen Bedingungen ÜS und KS.

Bew. Da $a^t \in_R F$ und da $\sum := K_{ij}^{\,t} - a^t$ unabhängig von a^t ist, ist $K_{ij}^{\,t} \in_R F$.

Sei $\varepsilon_{ij}^{\,s} = 0$ für alle $s < t$. Dann ist offensichtlich $\delta_{ij}^{\,t} = 0$ und die Bedingung ÜS ist erfüllt.

Nun sei s die erste Runde mit $\varepsilon_{ij}^{\,s} \neq 0$. Der Einfachheit halber sei $\varepsilon^u := \varepsilon_{ij}^{\,u}$ und $\delta^u := \delta_{ij}^{\,u}$. Die Differenzen δ^u werden nach dem folgenden System linearer Gleichungen gebildet:

$$\delta^u = 0 \text{ für } u = 1, \ldots, s$$

$$\begin{pmatrix} \delta^{s+1} \\ \delta^{s+2} \\ \ldots \\ \delta^{t-1} \\ \delta^t \end{pmatrix} = \begin{pmatrix} \varepsilon^s & 0 & \ldots & 0 & 0 \\ \varepsilon^{s+1} & \varepsilon^s & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \varepsilon^{t-2} & \varepsilon^{t-3} & \ldots & \varepsilon^s & 0 \\ \varepsilon^{t-1} & \varepsilon^{t-2} & \ldots & \varepsilon^{s+1} & \varepsilon^s \end{pmatrix} \bullet \begin{pmatrix} b^1 \\ b^2 \\ \ldots \\ b^{t-s-1} \\ b^{t-s} \end{pmatrix}$$

Da $\varepsilon^s \neq 0$ ist, ist die Matrix regulär und definiert eine bijektive Abbildung. Da $b^1,\ldots,b^{t-s} \in_R F$, sind $\delta^{s+1},\ldots,\delta^t$ gleichmäßig und unabhängig in F verteilt. Die Unabhängigkeit aller $K_{ij}^{\,1}$, …, $K_{ij}^{\,t}$ und δ^{s+1}, …, δ^t folgt aus der Unabhängigkeit aller a^1, …, a^t und δ^{s+1}, …, δ^t. □

Der zusätzliche Aufwand dieses Schlüsselgenerierungsschemas besteht in

- den insgesamt $2 \bullet t_{max} - 1$ in Konzelation und Authentizität garantierender Weise zwischen jedem in G direkt verbundenen Teilnehmerstationenpaar T_i und T_j ausgetauschten Schlüsselzeichen a^t, b^t (anstatt nur t_{max} für normales überlagerndes Senden),
- der Speicherung aller t_{max}-1 erhaltenen globalen Überlagerungszeichen und
- den jeweils t-1 Additionen und Multiplikationen (im Körper), um die Schlüssel für Runde t zu berechnen.

Aus letzterem folgt, daß das Schema durchschnittlich jeweils $t_{max} / 2$ Additionen und Multiplikationen benötigt. Deshalb ist das Schema praktisch nur von geringer Verwendbarkeit.

Gibt es keine zusätzliche Kommunikation zwischen T_i und T_j über ihren momentanen Zustand, so ist das Schema *optimal* bezüglich der Zahl der ausgetauschten Schlüsselzeichen und des zusätzlich benötigten Speichers. Dies ist in [WaPf_89 Seite 13f] gezeigt.

Das Schema zur deterministischen Knack-Schnapp-Schlüsselgenerierung wird verwendbar, wenn man die Summen in Gleichung (1) jeweils nicht bei 1, sondern für ein festes s jeweils bei t-s beginnen läßt. Hierbei muß s so gewählt werden, daß jede Teilnehmerstation spätestens nach s-1 von anderen gesendeten Nutzzeichen wieder ein eigenes sendet, daß der Angreifer nicht mit genügend großer Wahrscheinlichkeit raten kann. Erhält sie als globales Überlagerungsergebnis ihr Nutzzeichen (was bei einer Störung durch inkonsistente Verteilung in einer der letzten s Runden niemand erfahren hätte), dann gab es in den zurückliegenden s Runden mit entsprechender Wahrscheinlichkeit keine inkonsistente Verteilung. Da bei konsistenter Verteilung der Angreifer nichts über die b^u erfährt, kann man sie nach s Runden jeweils wiederverwenden. Dadurch wird nicht nur die Zahl der Körperoperationen reduziert, sondern auch die Zahl der nötigen Schlüsselzeichen. Man kann also durch geeignete Wahl von s ein leidlich praktikables Schema erhalten.

{Wie der Aufwand der deterministischen Knack-Schnapp-Schlüsselgenerierung näherungsweise halbiert werden kann, steht in [LuPW_91 §2.2].}

5.4.5.3.3 Probabilistische Knack-Schnapp-Schlüsselgenerierung

Eine wesentlich effizientere Realisierung der Knack-Schnapp-Schlüsselgenerierung erhält man, wenn man die Forderung KS etwas abschwächt: Inkonsistente Verteilung muß nicht mit der Wahrscheinlichkeit 1 bis in alle Ewigkeit stören, sondern nur mit der Wahrscheinlichkeit p (z.B. $1 - 10^{-50}$) die nächsten r Runden (z.B. $r = 10^{100}$). Für praktische Zwecke dürfte dies ausreichen.

Wie dies genau geht, steht in [WaPf_89], benötigt werden pro Runde zwei Additionen und zwei Multiplikationen im Körper, der allerdings groß gewählt werden muß. {Wie eine der beiden teuren Multiplikationen durch eine billige Addition ersetzt werden kann, wodurch der Aufwand der probabilistischen Knack-Schnapp-Schlüsselgenerierung näherungsweise halbiert wird, steht in [LuPW_91 §2.1].}

Sendet jede Teilnehmerstation nach spätestens s-1 Zeichen wieder ein eigenes Nutzzeichen (siehe vorangehender Abschnitt), können s Schlüsselzeichen b^u zyklisch verwendet werden.

5.4.5.4 Überlagerndes Empfangen

Die Bandbreite des DC-Netzes kann erheblich besser genutzt werden, wenn die Teilnehmerstationen nutzen, daß wer immer das globale Überlagerungsergebnis $ü$ von n gleichzeitig überlagernd gesendeten Nachrichten sowie n-1 davon einzeln kennt, die ihm noch fehlende Nachricht durch Überlagerung von $ü$ mit den n-1 ihm einzeln bekannten Nachrichten erhalten kann. Dieses **überlagernde Empfangen** kann **global**, d.h. alle Teilnehmerstationen überlagern gleich und empfangen so dasselbe, oder **paarweise**, d.h. genau zwei Teilnehmerstationen können überlagern, geschehen:

Beim *globalen überlagernden Empfangen* speichern alle Teilnehmerstationen nach einer Überlagerungs-Kollision diese (zunächst) nicht verwertbare Nachricht ab, so daß bei n kollidierten Nachrichten nur n-1 noch einmal gesendet werden müssen: die letzte Nachricht ergibt sich durch Subtraktion (modulo Alphabetgröße) der n-1 Nachrichten von der (zunächst) nicht verwertbaren Nachricht.

Empfangen die Teilnehmerstationen global überlagernd, so verschwendet eine Teilnehmerstation, die hin und wieder mehrere Nachrichten gleichzeitig überlagert und alle bis auf eine noch einmal einzeln sendet, keine Bandbreite des DC-Netzes, so daß dies ohne weiteres zulässig und allgemein bekannt sein kann. Dadurch wird der obige Schluß, daß kollidierte Nachrichten nicht von derselben Teilnehmerstation stammen, falsch, so daß das überlagernde Empfangen trotz der Effizienzsteigerung Sendeereignisse bei geeignet gewählten Wahrscheinlichkeiten für das gleichzeitige Überlagern von mehreren Nachrichten informationstheoretisch unverkettbar läßt. Zusätzlich zur besseren Nutzung der Bandbreite des DC-Netzes spart das globale überlagernde Empfangen auch noch das neu und damit anders (Ende-zu-Ende-)Verschlüsseln von kollidierten Nachrichten.

Paarweises überlagerndes Empfangen ist dann möglich, wenn sich zwei möglicherweise vollständig anonyme Teilnehmerstationen (etwa mittels eines anonymen Reservierungsschemas, vgl. §5.4.5.4.3) mit allen anderen am überlagernden Senden Beteiligten darauf geeinigt haben, daß und wann beide exklusives Senderecht haben: Obwohl *beide* gleichzeitig senden, können beide, indem sie je das von ihnen Gesendete mit dem globalen Überlagerungsergebnis überlagern, empfangen, was der andere gesendet hat (Bild 5-12). Dies kann für zwei Zwecke genutzt werden:

Senden beide Teilnehmerstationen je eine exklusiv für die andere Teilnehmerstation bestimmte Nachricht, so wird die Bandbreite des DC-Netzes doppelt so gut wie mit den in [Cha3_85, Chau_88, Pfi1_85] beschriebenen Zugriffsverfahren genutzt, beispielsweise kann in der Bandbreite eines Verteil-Simplex-Kanals ein Punkt-zu-Punkt-Duplex-Kanal untergebracht werden.

Sendet eine der beiden Teilnehmerstationen einen rein zufällig generierten Schlüssel, so kann sie von der anderen eine Nachricht in perfekter informationstheoretischer Konzelation erhalten, obwohl beide Teilnehmerstationen voreinander vollständig anonym sein können, wenn das zugrundeliegende

DC-Netz gegenüber dem Angreifer perfekte informationstheoretische Anonymität innerhalb irgendeiner Gruppe bietet, die beide Teilnehmerstationen enthält. Letzteres ist eine besonders effiziente Implementierung des auf Sender-Anonymität basierenden Konzelations-Protokolls von Alpern und Schneider [AlSc_83], das schon vor der Erfindung von Kommunikationsnetzen mit Sender-Anonymität veröffentlicht wurde. Mit dem von Axel Burandt entworfenen Code [Bura_88] kann erreicht werden, daß beide Partner mit informationstheoretischer Sicherheit überprüfen können, ob Dritte den Nachrichtentransfer störten, so daß *perfekte informationstheoretische Integrität* erreichbar ist.

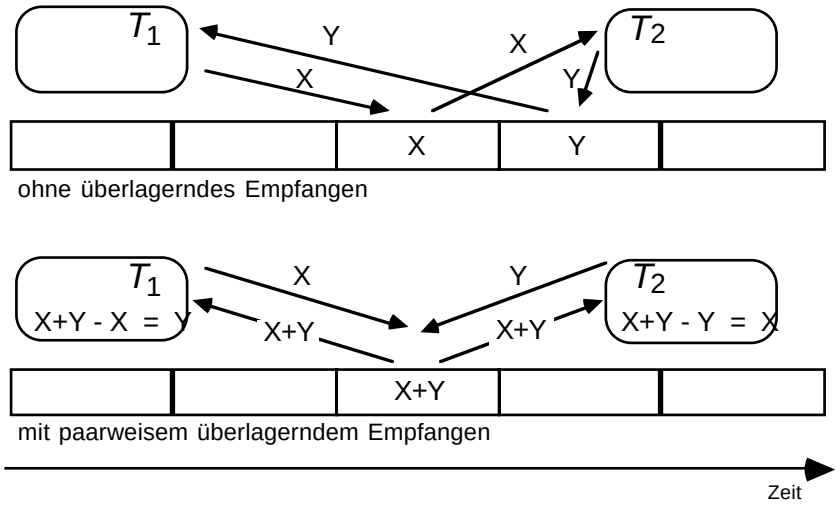


Bild 5-12: Paarweises überlagerndes Empfangen der Teilnehmerstationen T_1 und T_2

5.4.5.4.1 Kriterien für die Erhaltung von Anonymität und Unverkettbarkeit

Mehrfachzugriffsverfahren sollten, um die Anonymität des Senders zu erhalten und Verkehrsereignisse nicht zu verketten,

1. *alle Stationen absolut gleich behandeln bzw. handeln lassen* (beispielsweise sollten Stationen keine eineindeutige Nummer wie in dem Bitleisten-Protokoll (Bit-Map Protocol) in [Tane_81 Seite 296] oder in dem Gruppentest-Protokoll (Group Testing Protocol) in [BMTW_84 Seite 771] haben),
2. *keine Beziehungen zwischen Verkehrsereignissen herstellen* (beispielsweise „Wenn dies von Station *A* gesendet wurde, wurde jenes von Station *B* gesendet." oder „Wenn dies von Station *A* gesendet wurde, wurde jenes nicht von Station *B* gesendet.", wobei in beiden Fällen in der Praxis häufig *A*=*B* vorkommt),
3. jeder Station *erlauben, die gesamte Bandbreite zu nutzen*, wenn keine andere Station etwas sendet.

1. ist notwendig und hinreichend für *perfekte Anonymitätserhaltung* des Senders. Mit dem in §5.4.5.2 für das überlagernde Senden geführten Beweis ist 1. auch hinreichend für perfekte informationstheoretische Anonymität des Senders, sofern dieser sich nicht über den Nachrichteninhalt oder verkettbare Nachrichten explizit identifiziert.

2. ist notwendig und hinreichend für *perfekte Unverkettbarkeitserhaltung*. Mit dem in §5.4.5.2 für das überlagernde Senden geführten Beweis ist 2. auch hinreichend für perfekte informationstheoretische Unverkettbarkeit von Sendeereignissen, sofern diese nicht über den Nachrichteninhalt oder Adressen explizit verkettet sind.

3. ist notwendig für *perfekte Unverkettbarkeitserhaltung*. Dürfte nicht jede Station die gesamte Bandbreite nutzen, wären alle nahezu gleichzeitig stattfindenden Sendeereignisse insoweit verkettbar,

daß sie nicht alle von derselben Station (sofern nur manche Stationen nicht die gesamte Bandbreite nutzen dürfen: ... sie nicht alle von einer dieser Stationen) stammen.

5.4.5.4.2 Kollisionsauflösungsalgorithmus mit Mittelwertbildung und überlagerndem Empfangen

Die modulo-*g*-Addition des überlagernden Sendens wird zur (echten) Addition bis zum Wert *g*-1 verwendet, um mittels dieser Addition den abgerundeten Mittelwert $\lfloor Ø \rfloor$ aller an einer Kollision beteiligten Informationseinheiten zu berechnen. Jede Station läßt die Größe ihrer Informationseinheit im Vergleich zu $\lfloor Ø \rfloor$ darüber entscheiden, wann sie nochmals sendet (**Mittelwertvergleich**). Damit es sich bei der modulo-*g*-Addition um eine (echte) Addition handelt, muß die Summe aller beteiligten Informationseinheiten kleiner *g* sein. Dies ist bei *s* Stationen, von denen jede maximal *m* Informationseinheiten gleichzeitig senden darf, und einer maximalen Größe *G* (im Sinne der Interpretation der Binärcodierung der Informationseinheiten als Dualzahl) von Informationseinheiten mit Sicherheit immer dann der Fall, wenn $g > s \bullet m \bullet G$. Sofern nicht alle kollidierten Informationseinheiten gleich sind, gibt es immer mindestens eine kleiner und eine größer als $\lfloor Ø \rfloor$. Da dann beim Mittelwertvergleich nie alle dasselbe Ergebnis erhalten, senden entsprechend nie sofort alle noch mal oder keiner sofort noch mal. Deshalb wird keinerlei Bandbreite verschwendet.

Bild 5-13 zeigt ein passendes Nachrichtenformat für eine sendende Station. Bei einer nicht sendenden Station sind alle Bits der Nachricht, auch das letzte, 0. Die führenden Nullen in Bild 5-13 sind nötig, um einen Überlauf bei der Addition zu verhindern. Bild 5-14 zeigt ein detaillierteres Beispiel mit dem in Bild 5-13 angegebenen Nachrichtenformat. Weder in Bild 5-13 noch in Bild 5-14 noch im folgenden Text ist dargestellt, daß bei Verwendung dieses Mehrfachzugriffsverfahrens im DC-Netz zusätzlich noch paarweise ausgetauschte Schlüssel überlagert werden. Da deshalb die führenden Nullen im allgemeinen nicht als Nullen übertragen werden, müssen die ihnen entsprechenden Zeichen tatsächlich übertragen werden.

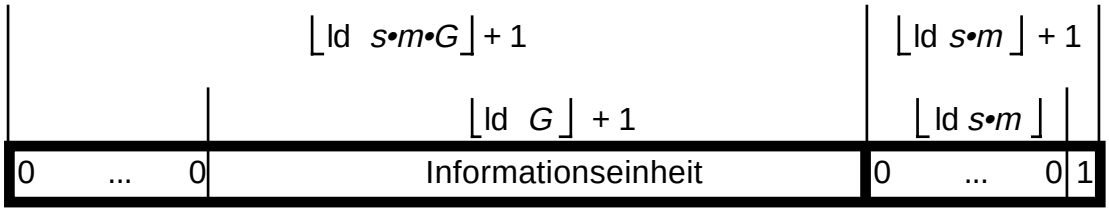
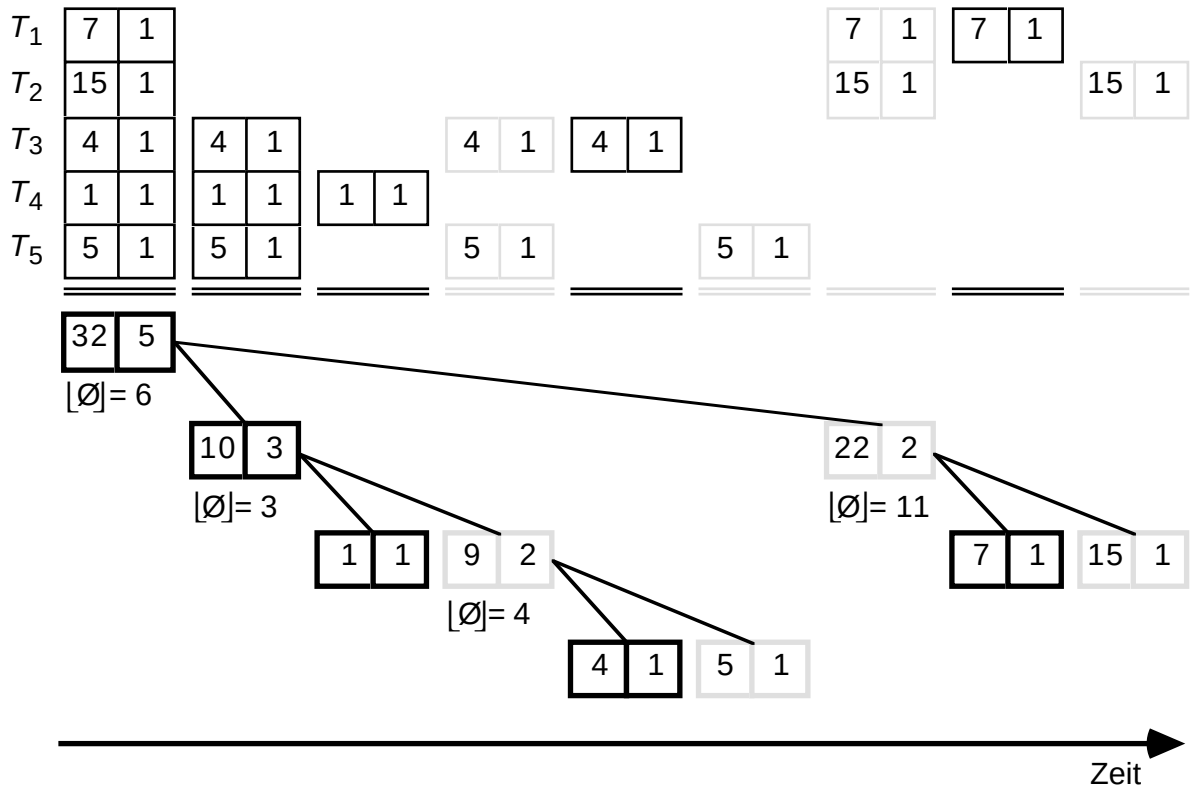


Bild 5-13: Nachrichtenformat für Kollisionsauflösungsalgorithmus mit Mittelwertbildung und überlagerndem Empfangen



Oben ist zeilenweise dargestellt, was die Stationen T_1 bis T_5 senden,
unten jeweils das globale Überlagerungsergebnis, die Summe.
Gilt für eine Station *Informationseinheit* $\leq \lfloor \varnothing \rfloor$, so sendet sie sofort wieder, anderenfalls später.
Das globale überlagernde Empfangen ist durch graue Quadrate symbolisiert.

Bild 5-14: Detailliertes Beispiel zum Kollisionsauflösungsalgorithmus mit Mittelwertbildung und überlagerndem Empfangen

Im Nachrichtenformat kann die Anzahl der führenden Nullen während der Ausführung des Kollisionsauflösungsalgorithmus gesenkt werden, wenn jeweils nur noch genügend wenig Informationseinheiten überlagert gesendet werden. Diese dynamische Verkleinerung spart zwar Übertragungsaufwand bzw. erlaubt bei gegebener Übertragungsrate einen höheren Durchsatz. Es ist jedoch für jede Implementierung abzuwägen, ob sich der Aufwand für diese dynamische Anpassung des Nachrichtenformates und der zur Überlagerung verwendeten zyklischen Gruppe lohnt.

Von eher theoretischem Interesse ist, daß bei Mittelwertvergleich nicht nur „Determinismus“ mit exponentiell kleiner Mißerfolgswahrscheinlichkeit erzielt werden kann, sondern sogar **„echter“ Determinismus**: Wird nach einer Überlagerungs-Kollision im nächsten Schritt nichts oder dasselbe noch mal übertragen, so sind alle kollidierten Informationseinheiten gleich. Da die Zahl der kollidierten Informationseinheiten schon für die Mittelwertbildung allen bekannt sein muß (etwa durch das früher bereits erwähnte Aufaddieren der der 1 entsprechenden Zeichen), können alle das vorherige gespeicherte Kollisionsergebnis durch die Zahl der kollidierten Informationseinheiten teilen und die resultierende Informationseinheit entsprechend oft empfangen.

Unter den üblichen (idealisierenden) Annahmen erzielt dieses Mehrfachzugriffsverfahren also einen Durchsatz von 100%.

Bei Verwendung eines genügend großen Alphabets und des gerade beschriebenen Mittelwertvergleichs kann ein realer Durchsatz von nahezu 100% erzielt werden. Bei binärer Codierung ist der Durchsatz des Mehrfachzugriffsverfahrens

$$\frac{\text{Anzahl der zur Codierung der Informationseinheiten nötigen Bits}}{\text{Anzahl aller zu übertragenden Bits}} \cdot 100\%.$$

Ist die Codierung der Informationseinheiten redundant, so geht der daraus resultierende Verlust an Durchsatz zu Lasten der Quellcodierung oder der binären Kanalcodierung, jedoch nicht zu Lasten des Mehrfachzugriffsverfahrens. Denn dies schränkt die Codierung der Informationseinheiten in keiner Weise ein.

Bei feldweiser binärer Codierung gemäß Bild 5-13 ist der reale Durchsatz des Mehrfachzugriffsverfahrens also mindestens

$$\frac{\lfloor \mathrm{ld}\ G \rfloor + 1}{\lfloor \mathrm{ld}\ s \bullet m \bullet G \rfloor + \lfloor \mathrm{ld}\ s \bullet m \rfloor + 2} \cdot 100\%,$$

da bei binärer Codierung der ganzen Zahlen des geschlossenen Intervalls $[0, x]$ genau $\lfloor \mathrm{ld}\ x \rfloor + 1$ Bits benötigt werden. Hierbei bezeichnet ld (logarithmus dualis) den Logarithmus zur Basis 2 und $\lfloor y \rfloor$ die größte ganze Zahl z mit $z \le y$.

Außerdem ist garantiert, daß jede Station in $s \bullet m$ zur Übertragung einer Informationseinheit benötigten Zeiteinheiten m Informationseinheiten übertragen kann: Jeder Station ist also, sofern sie etwas zu senden hat, ein fairer Anteil an der Bandbreite des DC-Netzes ebenso „echt“ deterministisch garantiert wie eine obere Schranke, nach der sie spätestens erfolgreich senden konnte. Andererseits kann eine einzelne Station, wenn die anderen nichts zu senden haben, die Bandbreite des DC-Netzes vollständig nutzen.

Die mittlere Verzögerungszeit kann minimiert werden, indem immer mit der Teilmenge geringerer Kardinalität fortgefahren wird. Dies kann und sollte mit dem Mittelwertvergleich kombiniert werden.

Die mittlere Verzögerungszeit dieses Mehrfachzugriffsverfahren ist in [Marc_88] genau untersucht.

Ist einem die zur Erfüllung der Bedingung $g > s \bullet m \bullet G$ nötige Alphabetgröße zu groß, so kann bei großem $s \bullet m$ statt $s \bullet m$ in der Ungleichung ein kleinerer Wert k, beispielsweise k=10, verwendet werden. Die Mittelwertaussage ist dann für Kollisionen von mehr als k Informationseinheiten möglicherweise falsch. Ist k geeignet gewählt, so treten die Bandbreite verschwendenden Fälle, daß der falsche, nämlich modulo g berechnete, Mittelwert kleiner oder größer als alle beteiligten Informationseinheiten ist, selten genug auf. Sie sind von allen Beteiligten erkennbar und werden durch Münzwurf aufgelöst.

Eine andere Möglichkeit zur Verwendung einer kleineren Alphabetgröße ist, den Mittelwertvergleich nicht auf ganze Informationseinheiten, sondern nur Teile (etwa den Anfang) zu erstrecken. Hierdurch steigt allerdings die Wahrscheinlichkeit, daß alle an einer Kollision beteiligten Informationseinheiten jeweils gleiche Teile, beispielsweise Anfänge, haben. Haben die verwendeten Teile jeweils mindestens 20 Bit Länge und sind die Werte auch nur näherungsweise gleichverteilt (etwa implizite Adressen oder Ende-zu-Ende-verschlüsselte Teile), so dürfte diese Wahrscheinlichkeit aber hinreichend klein sein, um die nutzbare Leistung nur unmerklich zu senken.

In beiden Fällen ist der Verlust des „echten“ Determinismus der Preis für die kleinere Alphabetgröße.

Der **Kollisionsauflösungsalgorithmus mit Mittelwertbildung und überlagerndem Empfangen** ist, sieht man vom (allerdings geringen, vgl. §5.4.5.5) Aufwand der Realisierung eines sehr großen Alphabets und vom Aufwand für das globale überlagernde Empfangen ab, das in jeder Hinsicht **optimale Mehrfachzugriffsverfahren für das DC-Netz** – es sei denn paarweises überlagerndes Empfangen oder Konferenzschaltung sind möglich, vgl. [Pfit_89 §3.1.2.5 und §3.1.2.6].

Jedes für überlagerndes Senden mit großem Alphabet geeignete Kommunikationsnetz arbeitet mit diesem (und ggf. weiteren) Mehrfachzugriffsverfahren für so viele Dienste so effizient, daß sein Einsatz zumindest im lokalen Bereich auch für Zwecke, bei denen es nicht auf Senderanonymität ankommt, zweckmäßig erscheint – Schlüsselgenerierung und synchronisierte Überlagerung können dann natürlich weggelassen werden.

5.4.5.4.3 Reservierungsschema (Roberts' scheme)

Wie bereits in §5.4.5.1 erwähnt, wurde die Verwendung dieses Anonymität perfekt erhaltenden Mehrfachzugriffsverfahrens, mit dem ein Durchsatz von nahe 100% erzielt werden kann [Tane_81 Seite 272], bereits in [Cha3_85] mit der folgenden Implementierung in der digitalen Welt des binären überlagernden Sendens vorgeschlagen:

> Die Bandbreite wird in Rahmen (frames) eingeteilt, deren Länge zwischen einem minimalen und maximalen Wert liegt. Um die Sprechweise der folgenden Funktionsbeschreibungen zu vereinfachen, bezeichne „der nächste" Rahmen nicht den physisch folgenden, sondern den *n*-ten der physisch folgenden, wobei *n* fest und minimal so gewählt wird, daß vor dessen Senden alle Stationen das Überlagerungsergebnis des „vorherigen" Rahmens (auch bei Rahmen minimaler Länge dazwischen) so rechtzeitig erhalten haben, daß sie zum Ausführen des Mehrfachzugriffsverfahrens in der Lage sind. Jeder Rahmen beginnt mit einem Datenübertragungsteil variabler Länge und endet mit einem Reservierungsteil fester Länge. Das Überlagerungsergebnis des Reservierungsteils eines Rahmens bestimmt die Länge des Datenübertragungsteils des nächsten Rahmens, indem in ihm für jede signalisierte Reservierung Platz (genauer: Zeit) für eine Informationseinheit vorgesehen wird, in der exklusiv diejenige Station senden darf, die diese Reservierung tätigte. Wird in einem Rahmen keine Reservierung getätigt, ist der Datenübertragungsteil des nächsten Rahmens leer. In [Cha3_85, Chau_88] schlägt David Chaum vor, den Reservierungsteil als eine Bitleiste aufzufassen, wobei jede 1 einer Reservierung entspricht. Der Datenübertragungsteil kann also maximal so viele Informationseinheiten enthalten, wie der Reservierungsteil Bits enthält. Um eine (oder mehrere) Reservierungen zu tätigen, sendet eine Station eine (oder mehrere) 1 an zufälligen Stellen des Reservierungsteils. Ist das Überlagerungsergebnis an einer (oder mehreren) dieser Stellen 1, so sendet sie in den entsprechenden Stellen (genauer: Zeiten) des Datenübertragungsteils des nächsten Rahmens.

Leider ist nun nicht garantiert, daß es keine Kollision gibt: Haben 3, 5, 7, ... Stationen an dieser Stelle reserviert, so gehen alle davon aus, daß nur sie an der entsprechenden Stelle des nächsten Rahmens senden. Die Wahrscheinlichkeit dieses Trugschlusses ist bei binärem überlagernden Senden nicht auf 0 zu senken, kann jedoch erheblich verkleinert werden, indem jeweils x Bits für jede Reservierung verwendet werden, von den jeweils y ($y<x$) als 1 gesendet werden, um eine Reservierung zu signalisieren. Durch geeignete Wahl von x und y kann zwar die Wahrscheinlichkeit, daß eine Kollision bei der Reservierung nicht erkannt wird, beliebig gesenkt werden, allerdings werden zur Reservierung auch x mal so viele Bits verwendet. Zu gegebener Verkehrsverteilung die optimalen Werte für x und y zu finden, ist eine lohnende Aufgabe.

Stattdessen sei hier noch einmal darauf hingewiesen, daß verallgemeinertes überlagerndes Senden mit $g > s \bullet m$ dazu verwendet werden kann, die Wahrscheinlichkeit von Kollisionen bei der Übertragung von Informationseinheiten auf 0 zu senken, wie vor der Beschreibung der einzelnen Mehrfachzugriffsverfahrensklassen erklärt wurde. Dieses in Bild 5-15 dargestellte Mehrfachzugriffsverfahren ist die älteste Anwendung von verallgemeinertem überlagerndem Senden [Pfi1_85 Seite 40].

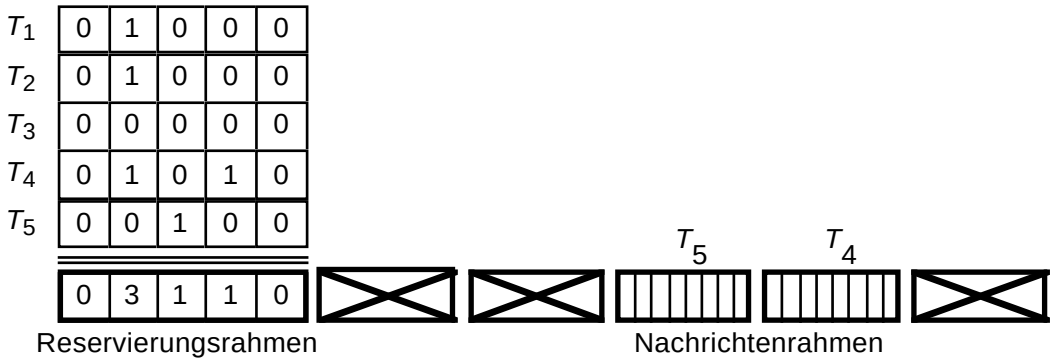


Bild 5-15: Reservierungsschema mit verallgemeinertem überlagerndem Senden und den Teilnehmerstationen T_i

Eine Kombination dieser Reservierungsverfahren mit überlagerndem Empfangen erscheint weitgehend unnötig. Lediglich im Falle, daß bei verallgemeinertem überlagernden Senden eine Reservierung den Wert 2 ergibt, kann ohne Einbuße an Durchsatz, aber mit Verringerung der durchschnittlichen Wartezeit bis zur erfolgreichen Übertragung vereinbart werden, daß an der entsprechenden Stelle des nächsten Rahmens beide ihre Informationseinheit senden, im übernächsten dann nur noch derjenige, der die größere Informationseinheit gesendet hat. Dies ist genau das bereits früher beschriebene Verfahren der deterministischen Kollisionsauflösung bei Kollision von zwei Informationseinheiten.

5.4.5.5 Optimalität, Aufwand und Implementierungen

Da aus Gründen der Empfängeranonymität potentiell alle Teilnehmerstationen die (Ende-zu-Ende-verschlüsselten) Nachrichten erfahren sollen (und dies damit – außer beim paarweisen überlagernden Empfangen – auch jeder realistische Angreifer kann), ist das überlagernde Senden mit Austausch eines Schlüssels zwischen jedem Teilnehmerstationenpaar und neuer Verschlüsselung kollidierter Nachrichten bzw. globalem überlagerndem Empfangen mit geeignet häufigem gleichzeitigem Überlagern von mehreren Nachrichten das bezüglich Senderanonymität *optimale* Verfahren.

Der Einsatz des Verfahrens des überlagernden Sendens ist jedoch sehr, sehr aufwendig, weil große Mengen an Schlüsseln in Konzelation garantierender Weise ausgetauscht werden müssen: *Jedes* Paar von Teilnehmerstationen, das Schlüssel miteinander austauscht, benötigt dazu einen Konzelation garantierenden Kanal mit derselben Bandbreite, wie sie das Kommunikationsnetz allen Benutzern *zusammen* zum Austausch ihrer Nachrichten bereitstellt.

Diesen *Aufwand beim Schlüsselaustausch* kann man reduzieren, indem Pseudozufallszahlen-, bzw. im Falle des binären überlagernden Sendens Pseudozufallsbitfolgengeneratoren verwendet werden. Dann müssen nur relativ kurze Schlüssel geheim ausgetauscht werden, aus denen dann sehr lange Schlüssel, die äußeren Betrachtern zufällig erscheinen, erzeugt werden können.

Das Verfahren ist dann nicht mehr informationstheoretisch sicher, sondern nur noch komplexitätstheoretisch mehr oder weniger sicher; bei Verwendung kryptographisch starker Pseudozufallszahlen-, bzw. -bitfolgengeneratoren schafft das Verfahren des überlagernden Sendens *perfekte komplexitätstheoretische Anonymität* des Senders und *perfekte komplexitätstheoretische Unverkettbarkeit* von Sendeereignissen.

Leider sind die bekannten schnellen Pseudozufallszahlen-, bzw. -bitfolgengeneratoren alle leicht brechbar oder zumindest von zweifelhafter Sicherheit (z.B. rückgekoppelte Schieberegister), während bisher die als kryptographisch stark bewiesenen Pseudozufallszahlen-, bzw. -bitfolgengeneratoren (§3.4.3, [VaVa_85, BlMi_84, BlBS_86]) sehr aufwendig und sehr langsam sind.

David Chaum schlägt in [Cha3_85] vor, das überlagernde Senden *auf einem Ring zu implementieren*. Dadurch wird einem Angreifer das Brechen von schnellen (und vielleicht leicht brechbaren) Pseudozufallszahlen-, bzw. -bitfolgengeneratoren erschwert, weil er die Ausgaben von aufeinanderfolgenden Teilnehmerstationen im Ring nur durch sehr aufwendige physikalische Maßnahmen, also realistischerweise nicht erfährt (vgl. §5.4.4.1).

Bei dieser Implementierung kreist jedes Bit einmal zum Zwecke des Sendens durch sukzessives Überlagern und einmal zum Zwecke des Empfangens um den Ring.

Da diese Implementierung im Mittel nur den vierfachen Übertragungsaufwand verursacht wie ein übliches Sendeverfahren auf einem Ring, während eine Implementierung auf einem sternförmigen Netz bei *n* Teilnehmerstationen den *n*-fachen Übertragungsaufwand gegenüber einem gewöhnlichen Sendeverfahren auf einem Stern verursacht, wirkt sie recht effizient.

Da aber die Übertragungsmenge auf jeder einzelnen Leitung, also die geforderte Bandbreite, bei allen Implementierungen gleich ist, kann die Implementierung auf einem *Stern* (oder allgemeiner: *Baum*) trotzdem effizienter sein. Die Implementierung eines Kanals ist um so besser, je kürzer die Verzögerungszeit ist, also die Zeit, die zwischen Sendeversuch und der Rückmeldung, ob eine Überlagerungs-Kollision auftrat, verstreicht. Für dieses Verfahren können die Knoten von Stern- und Baumnetzen wesentlich einfacher als übliche Vermittlungszentralen sein und so entworfen werden, daß die Summe der Schaltzeiten nur logarithmisch mit der Teilnehmeranzahl wächst. Die reine Laufzeit wächst ungefähr mit der Wurzel der Teilnehmeranzahl, während beide bei Ringnetzen stets proportional zur Teilnehmeranzahl wachsen.

Bisher haben sich folgende Ziele für die Realisierung der Anonymität schaffenden (Teil-)Schicht des DC-Netzes ergeben:

Z1 Die *Verzögerungszeit*, d.h. die Zeit, die vom Senden eines Zeichens bis zum Empfang der Summe (modulo Alphabetgröße) aller gesendeten Zeichen vergeht, soll möglichst gering sein, da dies für alle Kommunikationsdienste günstig und manche Mehrfachzugriffsverfahren notwendig ist. Trivialerweise muß die Verzögerungszeit kleiner als das Minimum aller bei den abzuwickelnden Diensten zulässigen Verzögerungszeiten sein.
Hieraus folgt zumindest für ein diensteintegrierendes Netz die Forderung, daß die Verzögerungszeit des DC-Netzes kleiner als die vom Menschen als störend empfundene Reaktionszeit sein muß. Wie in den „Annahmen und Notation bezüglich der Verkehrslast“ im [Pfit_89 §3.2.2.4] erwähnt, liegt ein sinnvoller Wert der Verzögerungszeit meiner Meinung nach unter 0,2 s, während CCITT maximal 0,4 s erlaubt.

Z2 Soll der Mehrfachzugriffskanal DC-Netz mit einem Reservierungsschema vergeben werden, so sollte die *Alphabetgröße* zumindest im Reservierungskanal *groß genug*, insbesondere größer 2 sein, vgl. §5.4.5.4.3.
Entsprechendes gilt für den in §5.4.5.4.2 beschriebenen Kollisionsauflösungsalgorithmus mit Mittelwertvergleich, der für viele Dienste das bestmögliche Mehrfachzugriffsverfahren darstellt.
Eine große Größe des zur Überlagerung verwendeten Alphabetes ist auch für eine möglichst effiziente Abwicklung von Konferenzschaltungen im eingeschränkten Sinn günstig, vgl. [Pfit_89 §3.1.2.6].

Z3 Der Teilnehmergemeinschaft soll für ihre Nutzdatenübertragung eine *hohe Bitrate* zur Verfügung gestellt werden.

Z4 Die Realisierung soll möglichst *geringen Aufwand* verursachen.

Die für die Erreichung dieser Ziele relevanten *Entwurfsentscheidungen*, nämlich Festlegung der Alphabetgröße, Implementierung der modulo-Addierer und Pseudozufallszahlengeneratoren, Wahl einer

geeigneten Topologie für die globale Überlagerung und Synchronisation des Überlagerns von Informationseinheiten und Schlüsseln werden im folgenden in dieser Reihenfolge behandelt.

Festlegung der Alphabetgröße (betr. vor allem: Z1, Z2, Z4): Geringer Aufwand und geringe Verzögerungszeit einerseits sowie große Größe des zur Überlagerung verwendeten Alphabets andererseits sind offensichtlich nur leicht widersprüchliche Ziele, da bei geeigneter, d.h. zu der Codierung der Informationseinheiten, Schlüssel und Übertragung passender Wahl der Größe des zur Überlagerung verwendeten Alphabets Teile eines Alphabetzeichens von der Teilnehmerstation bereits ausgegeben werden können, *bevor* der Rest des Alphabetzeichens, etwa durch die Nutznachricht, bestimmt ist. Entsprechendes gilt für die globale Überlagerung: Sind nur die Anfänge aller zu überlagernden Zeichen eingetroffen, kann der Anfang des Überlagerungsergebnisses bereits ausgegeben werden. Wie solch eine passende Codierung und passende, schnelle und unaufwendige Addierer bzw. Subtrahierer (modulo der Größe des zur Überlagerung verwendeten Alphabets) gewählt bzw. realisiert werden können, sei kurz skizziert, vgl. Bild 5-16:

Wie allgemein üblich, erfolge die Codierung der Informationseinheiten, Schlüssel und die Übertragung binär. Dann ist es sehr zweckmäßig, die Alphabetgröße als 2^l mit einer festen natürlichen Zahl l zu wählen und auch die Alphabetzeichen in der üblichen Weise binär zu codieren: das neutrale Element als 0, usw. Werden nun die Binärstellen der Alphabetzeichen in aufsteigender Wertigkeit übertragen, so genügt ein Volladdierer bzw. Vollsubtrahierer, ein UND-Gatter und beispielsweise ein Schieberegister der Länge l, um die Überlagerung zweier Bitströme binärstellenweise modulo 2^l durchzuführen: Im Schieberegister befindet sich nur an einer Stelle eine 0. Die Stelle, an der sich die 0 zu Beginn befindet, wird mittels des UND-Gatters mit dem Übertrag des Volladdierers bzw. -subtrahierers konjunktiv verknüpft. Der Ausgang des UND-Gatters dient als Übertrag des Volladdierers bzw. -subtrahierers. Dadurch wird erreicht, daß der Übertrag des Volladdierers bzw. -subtrahierers zu Beginn der binärstellenweisen Überlagerung eines Zeichens immer 0 ist.

Da Volladdierer bzw. -subtrahierer, UND-Gatter und Schieberegister genauso schnell wie Addierer modulo 2 (XOR-Gatter) arbeiten, ist die Verzögerungszeit des gerade skizzierten modulo 2^l arbeitenden DC-Netzes exakt genauso groß wie die eines modulo 2 arbeitenden. Lediglich der Aufwand der Addierer bzw. Subtrahierer wächst linear mit l oder anders formuliert logarithmisch mit der Größe des zur Überlagerung verwendeten Alphabets. Da die Schaltungskomplexität zur Überlagerung für realistische Werte von l (z.B. dürfte immer $l \leq 32$ gelten) immer um Größenordnungen kleiner ist als die zur Erzeugung von kryptographisch starken Pseudozufallsbit- bzw. -zahlenfolgen (vgl. §3) und da die Annahme optimal kurzer Verzögerungszeit für das einzige, ein modulo 2 arbeitendes DC-Netz benötigende (in [Pfit_89 §3.1.2.3.6] als letztes geschilderte) Mehrfachzugriffsverfahren für realistische Bitraten unrealistisch ist, gibt es aus meiner Sicht keinen wirklichen Grund, ein modulo 2 arbeitendes DC-Netz zu errichten. Daß die Annahme optimal kurzer Verzögerungszeiten für realistische Bitraten unrealistisch ist, sei durch folgendes Beispiel verdeutlicht: Angenommen das DC-Netz habe nur die Bandbreite 64000 bit/s und das Übertragungsmedium habe die Signalausbreitungsgeschwindigkeit 200000 km/s, so kann ein DC-Netz mit optimal kurzer Verzögerungszeit allein aufgrund der Signalausbreitungsgeschwindigkeit nur einen Durchmesser von maximal 3,125 km haben. Nichtoptimale Leitungsführung etc. erlaubt nur einen wesentlich geringeren Durchmesser.

Auch eine Kombination eines im wesentlichen Teil seiner Bandbreite modulo 2 und einem kleinen, zur Reservierung verwendeten Teil modulo 2^l arbeitenden DC-Netzes erscheint nicht sehr sinnvoll, da hierbei in einer frühen Phase der Errichtung eines DC-Netzes ohne Not eine weitreichende und die Entwurfskomplexität der höheren Schichten vermutlich steigernde Entscheidung über das Verhältnis der Bandbreiten getroffen werden müßte.

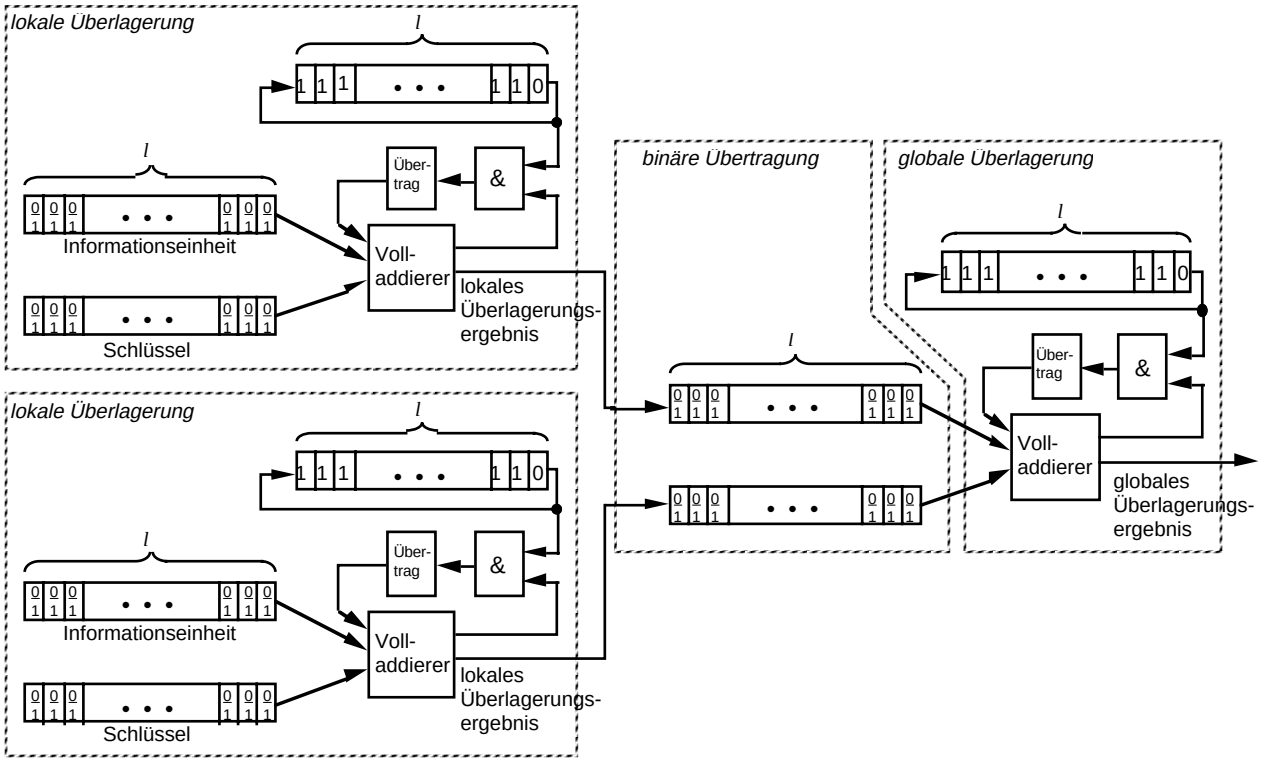


Bild 5-16: Geeignete Codierung von Informationseinheiten, Schlüsseln und lokalen sowie globalen Überlagerungsergebnissen bei der Übertragung: einheitliche Binärcodierung

Implementierung der modulo-Addierer (betr. vor allem: Z1, Z4): Da – wie bei der Festlegung der Alphabetgröße schon gezeigt – modulo-Addierer für alle in Betracht kommenden Alphabetgrößen mit geringem Aufwand so realisiert werden können, daß sie mit der technologieabhängigen minimalen Gatterverzögerungszeit als Verzögerungszeit des modulo-Addierers auskommen, kann jeder modulo-Addierer den gesamten Bitstrom verarbeiten, so daß keine Überlegungen bezüglich Parallelarbeit von modulo-Addierern angestellt zu werden brauchen.

Umgekehrt erscheint es bei Anschluß jeder Teilnehmerstation an mehrere DC-Netze aufwandsmäßig nicht lohnend, vorhandene modulo-Addierer mittels Multiplexern für mehrere der DC-Netze zu verwenden, da Multiplexer nicht wesentlich geringeren Aufwand als die beschriebenen modulo-Addierer verursachen. In §5.4.5.6 wird außerdem noch ausführlich dargestellt, daß ein (möglichst weitgehender) Verzicht auf gemeinsame Teile einen gleichzeitigen Ausfall mehrerer DC-Netze unwahrscheinlicher macht und deshalb wünschenswert ist.

Implementierung der Pseudozufallszahlengeneratoren (betr. vor allem: Z3, Z4): Da die Generierung von kryptographisch starken Pseudozufallsbit- bzw. -zahlenfolgen heutzutage für jedes DC-Netz nennenswerter Übertragungsrate nötig (vgl. §3), aber wesentlich langsamer als ihre Überlagerung und Übertragung ist, müssen ggf. mehrere Pseudozufallszahlengeneratoren parallel betrieben und ihre Ausgaben über einen Multiplexer zu einem um ihre Anzahl schnelleren Bitstrom verschachtelt werden.

Weder der Entwurf noch die Implementierung der Pseudozufallszahlengeneratoren muß innerhalb des DC-Netzes einheitlich sein. Theoretisch können sich je zwei Teilnehmer auf den Entwurf eines Pseudozufallszahlengenerators einigen, sich zwei gleichschnelle (anderenfalls muß die schnellere langsamer arbeiten) Implementierungen beschaffen und brauchen dann nur noch einen gemeinsamen und geheimen Startwert sowie möglicherweise einen öffentlich bekannten genauen Zeitpunkt zum synchronisierten Überlagerungsbeginn ihres Schlüssels. Der Vorteil dieser völlig dezentralen Auswahl von Pseudozufallszahlengeneratoren ist, daß sicherere und leistungsfähigere Pseudozufalls-

zahlengeneratoren nach und nach im DC-Netz eingesetzt werden und vermutlich eine große Vielzahl angeboten wird, so daß es „nicht so schlimm ist“, wenn manche gebrochen werden sollten. Der Nachteil ist offensichtlich: Ein Markt funktioniert nur dann gut, wenn der Konsument die Qualität der Waren (ggf. mit Hilfe von Experten) preiswert beurteilen kann. In [Pfit_89 §2.2.2.2] habe ich meine Skepsis gegenüber den heutzutage auf dem Markt angebotenen, größtenteils nach „geheimgehaltenen“ Algorithmen arbeitenden kryptographischen Systemen, die jeweils nur von (wenn die Geheimhaltung geklappt haben sollte) wenigen, größtenteils namentlich nicht bekannten „Experten“ analysiert wurden, bereits ausgedrückt. Ebenso habe ich in [Pfit_89 §2.2.2.3 (sowie dem Anhang) und §2.2.2.4] einige Vorschläge zur und Begründungen für eine Normung unterbreitet, die im Falle des DC-Netzes den kurzfristigen Austausch von Schlüsseln zwischen beliebigen Teilnehmern und damit eine flexiblere und gezieltere Gestaltung [Cha3_85, Chau_88] des Schlüsselaustauschs erlaubt. Außerdem werden durch die Verwendung standardisierter Implementierungen manche diskutierten Zuverlässigkeitsprobleme leichter lösbar.

Der heutzutage noch hohe Aufwand kryptographisch starker Pseudozufallszahlengeneratoren kann es für DC-Netze hoher Bandbreite erforderlich machen, bezüglich der kryptographischen Stärke der Pseudozufallszahlenerzeugung Kompromisse einzugehen. Diese können in separater oder kombinierter Anwendung der Maßnahmen bestehen, daß

- Teilnehmerstationen nur sehr wenige Schlüssel austauschen, daß
- manche Schlüssel mit effizienteren (und ggf. kryptographisch schwächeren) Pseudozufallszahlengeneratoren erzeugt werden oder daß
- ein Teil der Bandbreite mit schwächer erzeugten Schlüsseln überlagert wird.

Durch letzteres entstünde in jedem Teil der Bandbreite ein separates DC-Netz, wobei die Anonymität der Sender in den verschiedenen DC-Netzen unterschiedlich wäre, und – wie in [Pfit_89 §4.2] diskutiert – für verschieden sensitive Kommunikation verwendet werden könnte. Zu einem späteren Zeitpunkt könnten starke Pseudozufallszahlengeneratoren zusätzlich zu den effizienteren (und ggf. kryptographisch schwächeren) nachgerüstet werden.

Wahl einer geeigneten Topologie für die globale Überlagerung (betr. vor allem: Z1): Die Verzögerungszeit eines DC-Netzes setzt sich aus den für die Übertragung und Überlagerung benötigten Zeiten zusammen. Deshalb sind sowohl die *Übertragungs*- als auch die *Überlagerungstopologie* in aufeinander abgestimmter Weise so zu wählen, daß die Summe aller Verzögerungszeiten und damit die Verzögerungszeit des DC-Netzes möglichst gering ist. Unabhängig von beiden kann die *Schlüsseltopologie* gewählt werden, da die Reihenfolge und Zusammenfassung der Additionen in einer *abelschen* Gruppe irrelevant ist. Die drei zu unterscheidenden Topologien des DC-Netzes zeigt Bild 5-17.

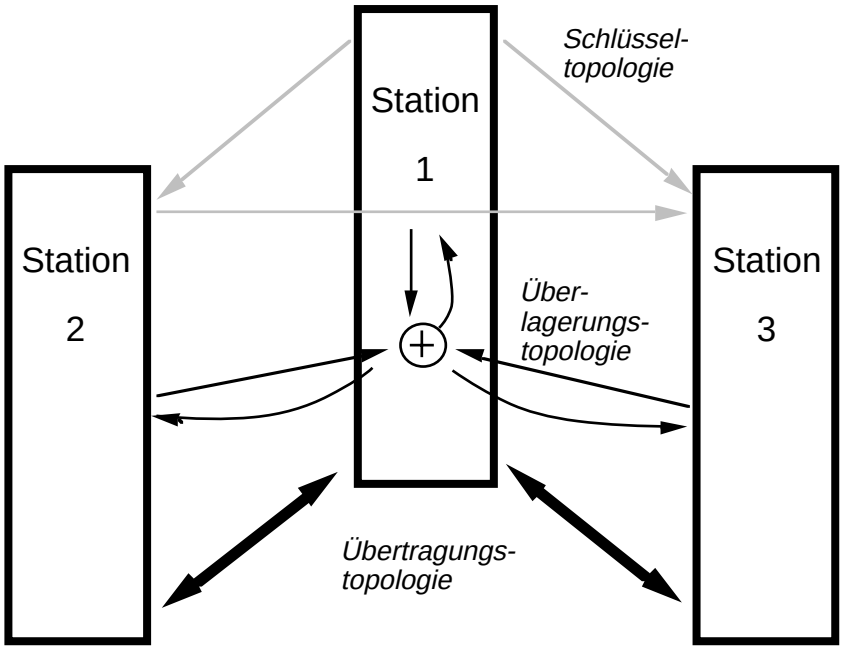


Bild 5-17: Die drei Topologien des DC-Netzes

Bei der Überlagerungstopologie (und auch bei der Übertragungstopologie) gibt es zwei Extremfälle:

1. Die Zeichen werden in einem Ring von Teilnehmerstation zu Teilnehmerstation weitergereicht. Da das Überlagern in jeder Teilnehmerstation mindestens eine Gatterverzögerungszeit dauert, wächst daher die durch Überlagerung verursachte Verzögerungszeit bei m Teilnehmerstationen mit $O(m)$, d.h. mindestens proportional zu m.
2. Die Zeichen werden zu einer zentralen Station (entsprechend einer heutigen Vermittlungszentrale) übertragen und dort überlagert, wobei, wie in Bild 5-18 gezeigt, die durch Überlagerung verursachte Verzögerungszeit bei m Teilnehmerstationen für Gatter mit begrenzt vielen Eingängen und begrenzter Treiberleistung mit $O(\log(m))$, d.h. mindestens proportional zum Logarithmus von m wächst. Dieses geringe Wachstum bleibt erhalten, wenn die Überlagerung dezentral, aber weiterhin baumförmig geschieht, so daß für die globale Überlagerung nicht nur eine stern-, sondern auch eine baumförmige Topologie geeignet ist.
   Bei Überlagerung modulo 2^l ist, wie oben unter „Festlegung der Alphabetgröße“ beschrieben, ein etwa durch ein Schieberegister realisierbarer Zähler modulo l zur Unterdrückung des Überlaufs an den Zeichengrenzen nötig. Dieser Zähler braucht bei zentraler Realisierung der baumförmigen Überlagerung nur einmal vorhanden zu sein, da er alle Volladdierer steuern kann, während er bei dezentraler Realisierung natürlich an jeder Überlagerungsstelle vorhanden sein muß.

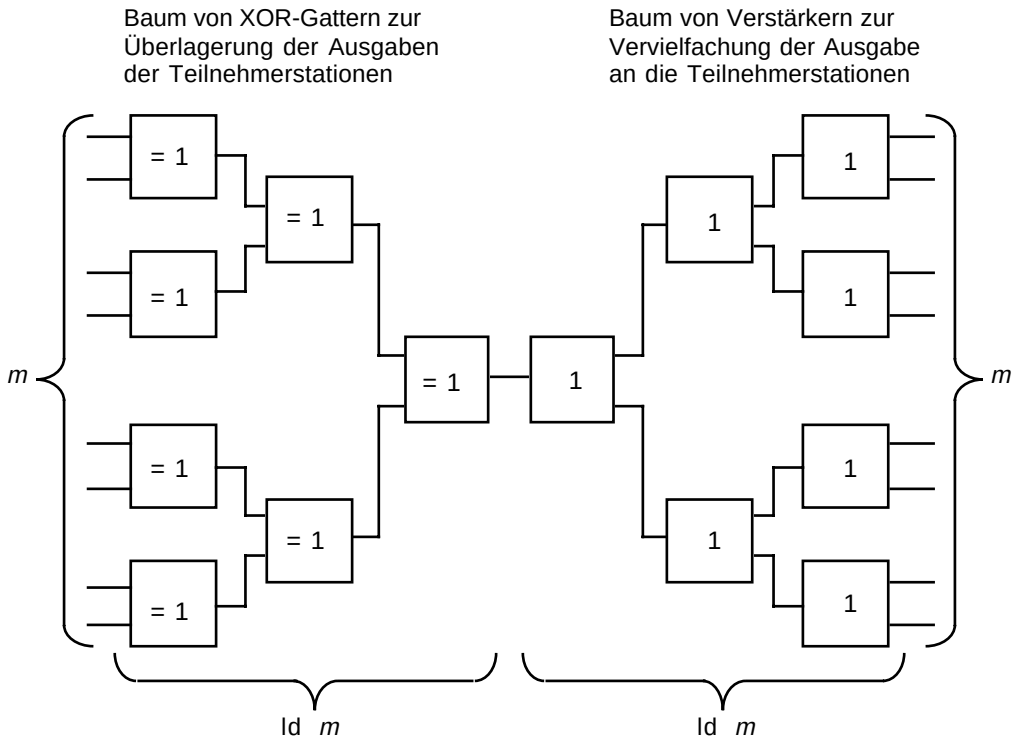


Bild 5-18: Verzögerungszeitminimale Überlagerungstopologie für Gatter mit 2 Eingängen und der Treiberleistung 2 am Beispiel binären überlagernden Sendens

Für diese Überlagerungstopologien geeignete Übertragungstopologien sowie das Wachstum der Summe aller Verzögerungszeiten und damit die Verzögerungszeit des DC-Netzes werden in [Pfit_89 §3.3.3] ausführlich behandelt.

Synchronisation des Überlagerns von Informationseinheiten und Schlüsseln (betr. vor allem: Z3, Z4): Wie schon mehrmals betont, müssen Informationseinheiten und Schlüssel synchronisiert überlagert werden. Geht die Synchronisation auch nur zwischen einem Paar paarweise ausgetauschter Schlüssel verloren, so können auf dem betroffenen DC-Netz keinerlei Informationseinheiten mehr erfolgreich übertragen werden. Die dann zu ergreifenden Maßnahmen werden in §5.4.5.6 behandelt.

Wenn das Übertragungsnetz, wie etwa für das ISDN geplant, in globaler Weise synchron arbeitet, kann diese Synchronität des Übertragungsnetzes auch für eine Synchronität des Überlagerns verwendet werden. Nun sind aber beispielsweise die Bitraten im geplanten ISDN klein verglichen mit der eines DC-Netzes vergleichbarer Nutzleistung. Ebenso ist die Rate akzeptabler Synchronisationsfehler im geplanten ISDN hoch verglichen mit einem DC-Netz vergleichbarer Zuverlässigkeit. Zusätzliche Maßnahmen scheinen also im allgemeinen Fall unumgänglich zu sein, während in speziellen Fällen, beispielsweise bei Implementierung einer dezentralen ringförmigen Überlagerungstopologie auf einem ringförmigen Übertragungsnetz hoher Bitrate und Zuverlässigkeit alle Synchronisationsprobleme der Überlagerung in trivialer Weise lösbar sind.

Eine konzeptionell einfache, in der Realisierung aber sehr aufwendige Methode ist, Zeichenströme mit „Zeitstempeln“ (z.B. Sequenznummern) zu versehen und vor der globalen Überlagerung zu Puffern sowie bezüglich der Puffer die in Rechnernetzen üblichen Flußregelungsmechanismen (flow control mechanisms) einzusetzen.

In der Realisierung weniger aufwendige Methoden können durch Ausnutzung spezieller Überlagerungs- und Übertragungstopologien erreicht werden. Ist die Übertragungstopologie des Verteil-

kanals des DC-Netzes hierarchisch und wird der Takt des Übertragungsnetzes von oben nach unten phasenstabil weitergegeben, so kann jede am Überlagern beteiligte Station ihren Sendetakt aus dem Takt des Verteilkanals herleiten. Tun dies alle Stationen in gleicher Weise, sind ihre Sendetakte phasenstabil. Ist die Überlagerungstopologie ebenfalls hierarchisch und sind die Verzögerungszeiten auf den zugehörigen Übertragungsstrecken konstant, so kann durch stationsindividuelle Wahl der Phasenlage des Sendetaktes zum (Empfangs-)Takt des Verteilkanals des DC-Netzes erreicht werden, daß alle gesendeten Zeichenfolgen das hierarchische Überlagerungsnetz synchron durchlaufen.

5.4.5.6 Fehlertoleranz beim DC-Netz

Beim DC-Netz ist ein beliebiges Bitübertragungsnetz, das die Schicht 0 und die tiefere Teilschicht der Schicht 1 des ISO OSI Referenzmodells umfaßt, mit konventionellen Fehlertoleranz-Maßnahmen ohne Rücksicht auf Anonymität, Unbeobachtbarkeit und Unverkettbarkeit möglich, vgl. §5.4.9. Im folgenden werden deshalb vor allem die obere Teilschicht der Schicht 1 und die Schicht 2 behandelt.

Die Serieneigenschaft des DC-Netzes besteht darin, daß alle Schlüssel fehlerfrei ausgetauscht worden sein müssen, alle Pseudozufallszahlengeneratoren (PZGs) und alle modulo-Addierer fehlerfrei arbeiten müssen und die Synchronisation erhalten bleiben muß.

Anders als beim MIX-Netz, bei dem Fehlertoleranz-Maßnahmen ständig im Hintergrund abgewickelt werden können, lassen sich beim DC-Netz zwei Phasen, später Modi genannt, klar voneinander trennen. Normalerweise übertragen alle Teilnehmerstationen anonym Informationseinheiten. Tritt ein permanenter Fehler im Verfahren zum anonymen Mehrfachzugriff oder im DC-Netz auf, so kann keine Teilnehmerstation Informationseinheiten übertragen, bis dieser Fehler durch Ausgliedern oder Reparatur der defekten Komponente(n) toleriert ist – es sei denn, es gibt mehrere voneinander unabhängige DC-Netze (*statisch erzeugte Parallel-Redundanz*, die wiederum *statisch* oder *dynamisch aktiviert* werden kann).

Die **Realisierung mehrerer unabhängiger DC-Netze** wird nicht vertieft betrachtet, da sie keine besonderen Entwurfs-Schwierigkeiten aufwirft.

Klaus Echtle wies darauf hin, daß es zweckmäßig sein dürfte, zwar jeder Station das Empfangen auf jedem der unabhängigen DC-Netze zu ermöglichen, das Senden jedoch nur auf relativ wenigen (*senderpartitioniertes DC-Netz*). Dadurch wird ein Fehler, der nur wenige Stationen betrifft, bzw. ein verändernder Angreifer, der nur wenige Stationen kontrolliert, daran gehindert, alle anderenfalls eben bezüglich des Fehlers bzw. verändernden Angreifers nicht unabhängigen DC-Netze zu stören. Die Bilder 5-19 und 5-20 zeigen ein für die Fehlervorgabe, beliebiges Fehlverhalten einer beliebigen Station zu tolerieren, konfiguriertes senderpartitioniertes DC-Netz für 10 Stationen. [Nied_87] enthält eine ausführliche und genaue Bewertung der Zuverlässigkeit, d.h. der Wahrscheinlichkeit, daß alle nicht fehlerhaften Stationen noch miteinander kommunizieren können, und Senderanonymität, d.h. unter wieviel Stationen ist ein Sender anonym, solcherart konfigurierter senderpartitionierten DC-Netze.

Wie schon am Beispiel ersichtlich, können auch von für die Fehlervorgabe eines beliebigen Einfachfehlers konfigurierten senderpartitionierten DC-Netzen in diesem Sinne manche Mehrfachfehler toleriert werden, beispielsweise beliebiges Fehlverhalten der Stationen 1, 2 und 5. In diesem Sinne nicht toleriert werden kann beispielsweise beliebiges Fehlverhalten der Stationen 1 und 8, da dann alle nicht an das DC-Netz 5 angeschlossenen Stationen, nämlich die Stationen 2, 3, 5 und 6 nicht mehr ungestört senden können.

Das Konfigurierungsprinzip des Beispiels kann auf die Fehlervorgabe, beliebiges Fehlverhalten von f beliebigen Station zu tolerieren, erweitert werden. Eine notwendige und hinreichende Bedingung hierfür ist, daß die Sendemöglichkeiten keiner Menge von f Stationen die einer einzelnen anderen Station überdecken.

Der Aufwand dieser Fehlertoleranz-Maßnahme ist entgegen dem ersten Eindruck nicht groß, da mehrere, im Grenzfall alle DC-Netze mittels Zeitmultiplex auf einem unterliegenden Bitübertragungsnetz (was zumindest im Grenzfall seine eigenen Fehler tolerieren können muß) realisiert werden können. Nachteilig ist, daß auf jedem der DC-Netze die Senderanonymität deutlich geringer als auf einem alle Stationen umfassenden DC-Netz ist (z.B. kann ein Angreifer, der eine Nachricht auf DC-Netz 1 in Bild 5-19 beobachtet, auch ohne Kooperation der Stationen 5, 6, 7, 8, 9 und 10 wissen, daß keine von ihnen die Nachricht gesendet hat). Bei der Benutzung eines solchermaßen konfigurierten senderpartitionierten DC-Netzes müssen alle Teilnehmerstationen darauf achten, daß verkettbare Informationseinheiten nur auf demselben DC-Netz gesendet werden, was die Lastverteilung auf den DC-Netzen drastisch einschränkt. Anderenfalls ist bei für die Fehlervorgabe, einen beliebigen Fehler einer beliebigen Station zu tolerieren, konfigurierten senderpartitionierten DC-Netzen der Sender identifizierbar, und bei für eine umfassendere Fehlervorgabe konfigurierten senderpartitionierten DC-Netzen die Anonymität des Senders abermals drastisch eingeschränkt. Eine *statische Aktivierung* der Redundanz senderpartitionierter DC-Netze ist also trivialerweise nicht sinnvoll möglich.

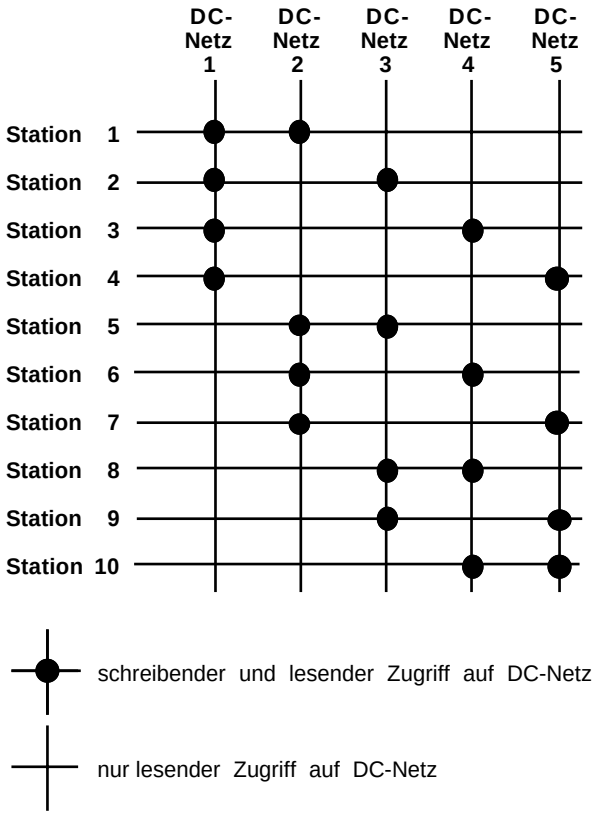


Bild 5-19: Für die Fehlervorgabe, beliebiges Fehlverhalten einer beliebigen Station ohne Fehlerdiagnose zu tolerieren, konfiguriertes senderpartitioniertes DC-Netz von 10 Stationen

Der Vorteil dieser Fehlertoleranz-Maßnahme ist, daß eine Fehlerdiagnose nicht oder zumindest nicht bei wenigen Fehlern nötig ist, und eine vollständige *Fehlerüberdeckung* erreicht wird, d.h. es gibt keine Fehler, die auf diese Weise nicht toleriert werden können.

Wird die Fehlervorgabe überschritten, kann selbstverständlich eine Fehlerdiagnose nötig werden, die mit den im folgenden beschriebenen Methoden durchgeführt werden kann, so daß dann beide Fehlertoleranzverfahren kombiniert werden.

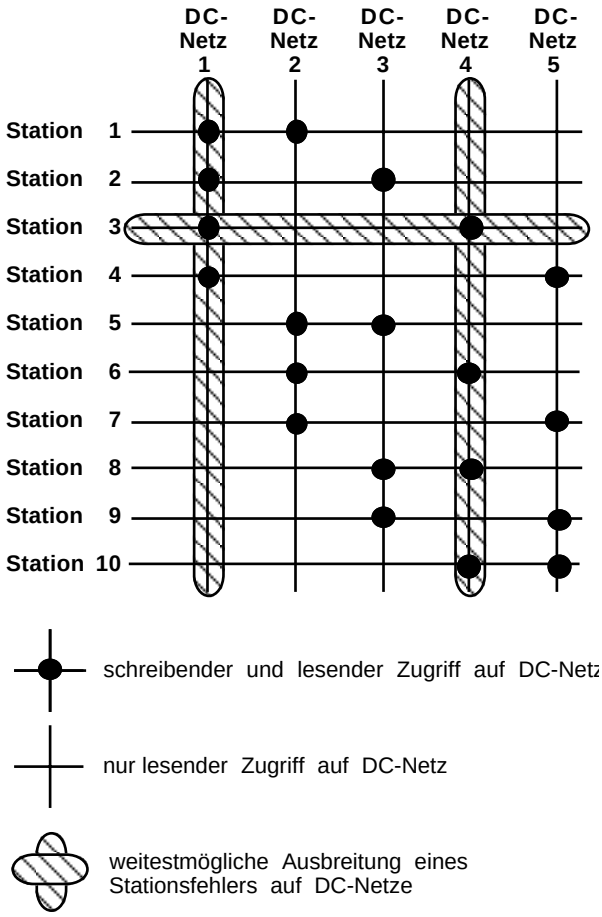


Bild 5-20: Weitestmögliche Ausbreitung eines Fehlers (bzw. verändernden Angriffs) der Station 3

Fehlererkennung, -lokalisierung und -behebung in einem DC-Netz: Transiente wie auch permanente Fehler im Bitübertragungsnetz können – wie erwähnt – durch gesonderte Fehlertoleranz-Maßnahmen in ihm selbst toleriert werden, oder führen andernfalls zu transienten bzw. permanenten Fehlern im DC-Netz. Die transienten Fehler im DC-Netz werden durch Ende-zu-Ende-Protokolle toleriert. Es verbleiben somit die permanenten Fehler im DC-Netz (Ausfall von PZGs, von modulo-Addierern, Verlust der Konsistenz oder Synchronisation der Schlüssel usw.). Diese sind, wenn einmal erkannt und lokalisiert, leicht zu beheben. Die entsprechenden Schlüssel (von defekten PZGs) werden nicht mehr überlagert, Addierer werden ausgetauscht, Schlüssel werden neu verteilt oder sie werden neu synchronisiert. Das Hauptproblem liegt also in der *Fehlererkennung* und *-lokalisierung*.

Fehler der oberen Teilschicht der Schicht 1 lassen sich dadurch erkennen, daß jeder verschlüsselten Informationseinheit von ihrem Sender zusätzliche, allen Stationen zugängliche Redundanz zugefügt wird, z.B. ein linear gebildetes Prüfzeichen, z.B. CRC, am Ende der Informationseinheit. Dies kann immer geschehen, da die Redundanz nur einen Bruchteil der Länge der Informationseinheiten umfaßt, die nutzbare Übertragungsleistung also kaum sinkt. Wenn das Prüfzeichen linear gebildet ist, entsteht auch bei Überlagerungskollisionen ein gültiges Prüfzeichen. Entsteht ein ungültiges Prüfzeichen, so liegt ein Fehler bei der Überlagerung vor.

Die Erkennung von Fehlern des Mehrfachzugriffsverfahrens hängt vom verwendeten Verfahren ab, ist aber für alle empfohlenen Verfahren mit hoher Wahrscheinlichkeit möglich.

Wird ein Fehler des Mehrfachzugriffsverfahrens erkannt, so wird dieses neu initialisiert, um transiente Fehler des Bitübertragungsnetzes oder der oberen Teilschicht der Schicht 1 zu tolerieren. Schlägt dies fehl oder wird auf andere Weise ein andauernder Fehler der oberen Teilschicht der

Schicht 1 erkannt, so wird aus dem die Anonymität garantierenden Anonymitäts-Modus (A-Modus) in den Fehlertoleranz-Modus (F-Modus), in dem Fehler lokalisiert und toleriert werden, geschaltet.

Alle Stationen führen folgendes **Fehlerlokalisierungs- und -behebungs-Protokoll** aus [Pfi1_85 Seite 123]:

Jede Station sichert den momentanen Zustand ihrer paarweise ausgetauschten Schlüssel bzw. PZGs (in der Fachsprache: erstellt einen Rücksetzpunkt, recovery point [AnLe_81]) und führt anschließend folgende Selbstdiagnose durch: sie lädt paarweise zufällige Schlüssel in ihre Schlüsselspeicher bzw. PZGs und überlagert sie lokal mit einer ebenfalls zufälligen Informationseinheit.

Ist das Ergebnis nicht die zufällige Informationseinheit, so ist die Station fehlerhaft und sendet eine dies signalisierende Nachricht über das DC-Netz. Alle anderen Stationen werfen mit dieser Station geteilte Schlüssel weg und wechseln (Einfehlerannahme) in den A-Modus zurück.

Ist das Ergebnis die zufällige Informationseinheit, benutzt die Station den Rücksetzpunkt zur Herstellung des vorherigen Zustands ihrer paarweise ausgetauschten Schlüssel bzw. PZGs. An dieser Stelle des Protokolls gibt es drei wahrscheinliche Fehlertypen:

1. Eine Station ist so fehlerhaft, daß sie sich nicht selbst diagnostizieren und das Ergebnis den anderen mitteilen kann, oder
2. die Synchronisation der Schlüssel(erzeugung und -)überlagerung ging zumindest zwischen zwei Stationen verloren, oder
3. das unterliegende Kommunikationsnetz oder die globale Überlagerung ist fehlerhaft.

Um diese Fehlertypen zu unterscheiden und Fehler zu lokalisieren, wird die Zahl der überlagerten Schlüsselpaare sukzessive halbiert (der entsprechende Schlüsselaustauschgraph hat jeweils nur halb so viele Kanten) und beispielsweise 100 neue und nicht noch einmal verwendete Schlüsselzeichen überlagert.

Ist das globale Überlagerungsergebnis 100 mal das 0 entsprechende Zeichen, so ist der Fehler mit der Wahrscheinlichkeit $1 - g^{-100}$ in der anderen Hälfte (sofern ein 0-Haftfehler (stuck at zero fault) am letzten globalen Überlagerungsgerät ausgeschlossen werden kann. Dies kann dadurch getestet werden, daß alle Stationen vorher 100 zufällige Zeichen senden, was mit derselben Wahrscheinlichkeit ein Ergebnis $\neq 0$ ergibt, sofern dort kein Haftfehler vorliegt).

Ist das globale Überlagerungsergebnis nicht 100 mal das 0 entsprechende Zeichen, so gibt es mindestens einen Fehler bei der Speicherung oder Generierung der Schlüssel oder deren synchronisierter Überlagerung oder aber das unterliegende Kommunikationsnetz oder die globale Überlagerung ist fehlerhaft. Deshalb wird weiterhin halbiert.

Hat der Schlüsselaustauschgraph nur noch eine Kante und ist das globale Überlagerungsergebnis nicht 100 mal das 0 entsprechende Zeichen, tauschen beide Stationen einen neuen Schlüssel oder PZG-Startwert aus, um Schlüsselsynchronisationsfehler zu beheben. Danach wiederholen beide ihren Versuch.

Ist das globale Überlagerungsergebnis abermals nicht 100 mal das 0 entsprechende Zeichen, so werfen beide Stationen den gerade ausgetauschten Schlüssel bzw. PZG-Startwert weg und tauschen mit einer dritten und vierten Station einen Schlüssel oder PZG-Startwert aus. Beide neuen Stationenpaare überlagern nacheinander 100 Schlüsselzeichen.

Entsteht in beiden Fällen nicht 100 mal das 0 entsprechende Zeichen, so ist mit hoher Wahrscheinlichkeit das unterliegende Kommunikationsnetz oder die globale Überlagerung fehlerhaft. Beides weiter zu untersuchen sind übliche Probleme der Fehlertoleranz.

Entsteht nur in einem Fall nicht 100 mal das 0 entsprechende Zeichen, so wird die ursprünglich fehlerverdächtige Station dieses Paares als fehlerhaft angesehen und außer Betrieb genommen. Dies wird allen Stationen mitgeteilt, so daß sie mit dieser Station geteilte Schlüssel wegwerfen und (Einfehlerannahme) in den A-Modus zurückwechseln.

Mit dem Zurückschalten vom F-Modus in den A-Modus wird das Verfahren zum anonymen Mehrfachzugriff neu initialisiert, um Auswirkungen von Fehlern der Schicht 1 (physical) auf die Schicht 2 (data link) rückgängig zu machen. Alle gerade geschilderten Schritte sind in Bild 5-21 zusammengefaßt.

Natürlich gibt es hunderte von Variationen dieses Fehlerlokalisierungs- und -behebungsprotokolls, über deren Zweckmäßigkeit geurteilt werden kann, sobald die Wahrscheinlichkeiten von (Mehrfach-) Fehlern bekannt sind. Ist beispielsweise die Wahrscheinlichkeit von Mehrfachfehlern signifikant, so sollten Hälften des Schlüsselaustauschgraphen, die vom gerade beschriebenen Protokoll nicht getestet werden, auch getestet werden, was den Aufwand des Protokolls höchstens verdoppelt.

Da die Wahrscheinlichkeiten nicht bekannt sind, sind folglich nicht die Details von Fehlerlokalisierung und -behebung interessant, sondern daß sie mit logarithmischem Zeitaufwand (in der Zahl der Stationen) möglich sind.

Es ist beachtenswert, daß bei „Fehlererkennung, -lokalisierung und -behebung in einem DC-Netz“ die Anonymität trotz Fehlertoleranz nicht abgeschwächt wird, wenn entweder echt zufällig generierte Schlüssel oder kryptographisch starke PZGs verwendet werden. Durch das Einführen der zwei Modi und die Verwendung von Schlüsselzeichen entweder im einen oder aber im anderen Modus bleibt im A-Modus die Anonymität jederzeit in ihrem ursprünglichen Umfang garantiert.

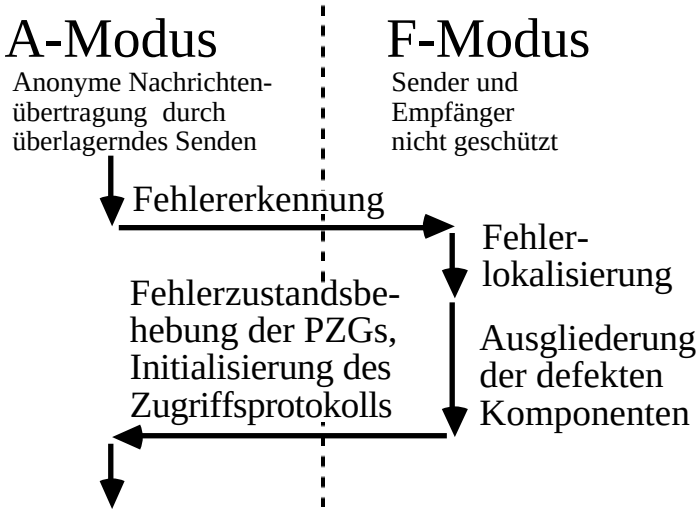


Bild 5-21: Fehlererkennung, -lokalisierung und -behebung beim DC-Netz

5.4.6 MIX-Netz: Schutz der Kommunikationsbeziehung

Statt zusätzlich zum Empfänger auch den Sender verborgen zu halten, kann man zunächst versuchen, nur deren Verbindung geheimzuhalten und so die Anonymität der Kommunikation herzustellen.

Diese Idee wird durch das Verfahren der **umcodierenden MIXe** verwirklicht [Chau_81, Cha1_84].

Bei diesem von David Chaum 1981 für elektronische Post vorgeschlagenen Verfahren werden Nachrichten von ihren Sendern nicht notwendigerweise auf dem kürzesten Weg zu ihren Empfängern geschickt, sondern über mehrere möglichst bezüglich ihres Entwurfes, ihrer Herstellung [Pfit_86 Seite 356] und ihres Betreibers [Chau_81, Cha1_84 Seite 99] *unabhängige* sowie Nachrichten *gleicher Länge puffernde*, *umcodierende* und *umsortierende* Zwischenstationen, sogenannte MIXe, geleitet. Jeder MIX codiert Nachrichten gleicher Länge um, d.h. ent- oder verschlüsselt sie unter

Verwendung eines Konzelationssystems, so daß ihre Wege über ihre Länge und Codierung, zusammen ihr äußeres Erscheinungsbild, nicht verfolgt werden können.

Damit dieses Verfolgen des Weges auch über zeitliche oder räumliche Zusammenhänge nicht möglich ist, muß jeder MIX jeweils mehrere Nachrichten gleicher Länge von genügend vielen unterschiedlichen Absendern[110] abwarten (oder ggf. auch selbst erzeugen oder von Teilnehmerstationen erzeugen lassen – sofern nichts Nützliches zu senden ist, eben bedeutungslose Nachrichten, die von der Teilnehmerstation des Empfängers oder bei geeigneter Verschlüsselungsstruktur und Adressierung bereits von einem der folgenden MIXe ignoriert werden), sie dazu puffern und nach der Umcodierung umsortiert, d.h. in anderer zeitlicher Reihenfolge bzw. auf anderen Leitungen ausgeben. Eine geeignete Reihenfolge ist etwa die alphabetische Ordnung der umcodierten Nachrichten. (Solch eine von vornherein vorgegebene Reihenfolge ist besser als eine zufällige, da so der verborgene Kanal (covert -, hidden channel) der „zufälligen“ Reihenfolge für ein eventuell im jeweiligen MIX vorhandenes Trojanisches Pferd geschlossen wird. Da auch die Anzahl der jeweils gleichzeitig zu mixenden Nachrichten sowie die Zeitverhältnisse exakt vorgegeben werden können und dem MIX das Erzeugen von Nachrichten verboten werden kann, braucht der MIX keinerlei Handlungsspielraum und damit braucht keinerlei Möglichkeit zu bestehen, über die ein eventuell im MIX vorhandenes Trojanisches Pferd einem nicht empfangsberechtigten Empfänger verborgen Information zukommen lassen kann, vgl. §1.2.2 und [PoKl_78, Denn_82 Seite 281].)

Die zum Zwecke des Verbergens zeitlicher oder räumlicher Zusammenhänge zusammen gemixten, d.h. zusammen gepufferten (oder erzeugten), umcodierten und umsortiert (beispielsweise in alphabetischer Reihenfolge) ausgegebenen Nachrichten gleicher Länge werden als ein *Schub* (batch [Chau_81 Seite 85]) bezeichnet.[111]

Zusätzlich zum Puffern, Umcodieren und Umsortieren von Nachrichten gleicher Länge muß jeder MIX darauf achten, daß *jede Nachricht nur einmal gemixt wird* – oder anders herum gesagt: Nachrichtenwiederholungen ignoriert werden. Eine Eingabe-Nachricht stellt genau dann eine Nachrichtenwiederholung dar, wenn sie schon einmal bearbeitet wurde und die jeweils zugehörigen Ausgabe-Nachrichten von Unbeteiligten verkettbar (beispielsweise gleich) wären. (Wann dies genau der Fall ist, hängt vom verwendeten Umcodierungsschema ab und wird deshalb zusammen mit den Umcodierungsschemata weiter unten diskutiert.)

Wird eine Nachricht innerhalb *eines* Schubes mehrfach bearbeitet, so entstehen über die Häufigkeiten der Eingabe- und Ausgabe-Nachrichten dieses Schubes unerwünschte Entsprechungen: einer Eingabe-Nachricht, die *n*-mal auftritt, entspricht eine Ausgabe-Nachricht, die ebenfalls *n*-mal auftritt. Treten alle Eingabe-Nachrichten eines Schubes verschieden häufig auf, so schützt das Umcodieren dieses Schubes also überhaupt nicht.

Wird eine Nachricht in *mehreren* Schüben bearbeitet, so können zwischen diesen Schüben nichtleere Durchschnitte (und Differenzen) gebildet werden, indem jeweils der Durchschnitt (die Differenz)

---

110 Die Forderung „Nachrichten von genügend vielen unterschiedlichen Absendern“ ist eine schwierig zu erfüllende Forderung: 1. sind in manchen Kommunikationsnetzen, z.B. dem Internet, eineindeutige Teilnehmeridentitäten nicht gegeben, d.h. Teilnehmer können sich beliebig viele unterschiedliche Identitäten zulegen. MIX-Varianten, die selbst dann noch sinnvoll eingesetzt werden können, sind in [Kesd_99] beschrieben. 2. ist selbst dann, wenn eineindeutige Teilnehmeridentitäten gegeben sind, deren Prüfung ab dem zweiten durchlaufenen MIX alles andere als trivial, vgl. Aufgabe 5-20.

111 Eine im Internet verbreitete MIX-Implementierung (MIXmaster) arbeitet etwas anders: Bei ihr wird in einem *Pool* eine Mindestanzahl an Nachrichten gehalten und nach dem Mixen einer neu eingetroffenen Nachricht dann zwischen dieser und den im Pool enthaltenen einzelnen Nachrichten eine Zufallsauswahl getroffen, welche Nachricht ausgegeben wird. Wird Poolgröße+1 gleich Schubgröße gewählt, ist die durchschnittliche Verzögerungszeit bei der Poolimplementierung doppelt so groß – dafür ist die Unverkettbarkeit von Eingabe- und Ausgabenachrichten höher. Da bei Kommunikationsdiensten oftmals garantiert werden muß, daß eine maximale Verzögerungszeit nicht überschritten wird und bei der Poolimplementierung keine Garantie gegeben werden kann, erscheint mir die schubweise Implementierung vorteilhaft.

der Eingabe- und Ausgabe-Nachrichten gebildet wird, wobei beim Durchschnitt (bei der Differenz) von letzteren im allgemeinen Fall statt Gleichheit Verkettbarkeit geprüft wird. Natürlich können diese beiden Operationen wiederum auf beliebige Operationsergebnisse angewendet werden. Bei Nachrichten, die in einem Durchschnitt (einer Differenz) der Kardinalität 1 liegen, ist die Entsprechung zwischen Eingabe- und Ausgabe-Nachricht klar – bezüglich ihnen trägt dieser MIX zum Schutz der Kommunikationsbeziehung nichts bei.

Die in diesem Abschnitt hergeleiteten und beschriebenen Grundfunktionen eines MIXes sind in Bild 5-22 dargestellt.

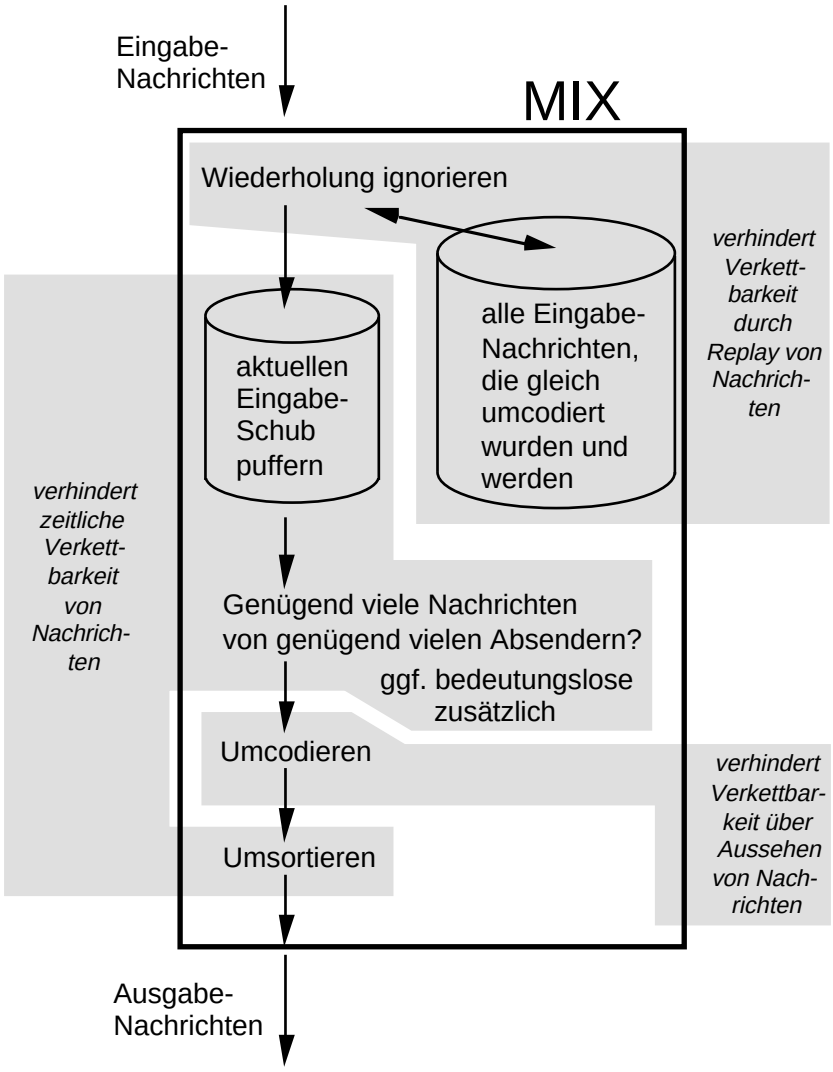


Bild 5-22: Grundfunktionen eines MIXes

5.4.6.1 Grundsätzliche Überlegungen über Möglichkeiten und Grenzen des Umcodierens

Das **Ziel** dieses Verfahrens der umcodierenden MIXe ist, daß

- alle anderen Sender und Empfänger der zusammen in den Schüben der MIXe gemixten Nachrichten [Pfi1_85 Seite 11] oder
- alle MIXe, die von einer Nachricht durchlaufen wurden, [Chau_81 Seite 85]

zusammenarbeiten müssen, um die Kommunikationsbeziehung gegen den Willen von Sender oder Empfänger aufzudecken. Solange beides nicht gegeben ist, sollte aus der Sicht des Angreifers jede

von einem Sender während eines bestimmten Zeitintervalls gesendete Nachricht zu jeder von einem Empfänger empfangenen Nachricht entsprechender Länge gehören können – außer der Angreifer ist Sender *und* Empfänger der Nachricht. Dabei wird die Länge der Zeitintervalle durch die maximal tolerierbaren Verzögerungszeiten der Dienste bestimmt: Jede von einem Sender gesendete Nachricht kann natürlich erst nach ihrem Senden, muß aber vor Ablauf der maximal tolerierbaren Verzögerungszeit des Dienstes beim Empfänger eintreffen.

Dieses Ziel, das in der Diktion von §5.1.2 die Nachrichten und die Teilnehmer bezüglich ihres Sendens und Empfangens lediglich nach Zeitintervallen und Nachrichtenlängen in Klassen einteilt, bezüglich dieser Klassen und des beschriebenen Angreifermodells aber *Unbeobachtbarkeit der Kommunikationsbeziehung* vor Unbeteiligten und – sofern gewünscht – *Anonymität der Kommunikationspartner voreinander* als auch *Unverkettbarkeit der Kommunikationsbeziehungen* garantiert, ist das Maximum des Erreichbaren:

- Arbeiten alle anderen Sender und Empfänger von Nachrichten, die von einem MIX zusammen gepuffert, umcodiert und umsortiert ausgegeben (d.h. in einem Schub gemixt) werden, zusammen, so kann der MIX von ihnen prinzipiell bezüglich der von einem anderen gesendeten Nachrichten überbrückt werden. Praktisch ist dies besonders schlimm, wenn es einem Angreifer möglich ist, von n+1 Nachrichten, die von MIXen zusammen in einem Schub gemixt werden, selbst n zu liefern. Aus diesen für einen einzelnen MIX in [Pfi1_85 Seite 11] gemachten Aussagen folgt, daß, wenn alle anderen Sender und Empfänger von Nachrichten gleicher Länge eines bestimmten Zeitintervalls zusammenarbeiten, sie einen weiteren Sender und Empfänger trotz der Benutzung beliebig vieler, vom Angreifer nicht kontrollierten MIXe, vollständig beobachten können.
- Daß das Verfahren der umcodierenden MIXe, sofern alle von einer Nachricht durchlaufenen MIXe zusammenarbeiten, für diese Nachricht nichts nützt, ist trivial.

Aus dem oben formulierten Ziel folgt, daß *das Umcodieren einer Nachricht (bzw., wie später erläutert wird, eines Nachrichtenteils) relevanten Inhalts nie aus Ver- gefolgt von Entschlüsseln besteht, sofern überflüssiges Umcodieren unterlassen wird und das verwendete Konzelationssystem nur die in §3.1.1.1 genannten Eigenschaften besitzt*: Gibt es eine Ver- gefolgt von einer Entschlüsselung, muß dabei die Entschlüsselung mit dem zur Verschlüsselung passenden Schlüssel geschehen, da anderenfalls die Nachricht nie mehr verstanden werden kann. Dies bedeutet, daß sie in diesen beiden MIXen in gleicher Gestalt vorliegt, so daß diese beiden MIXe jeweils die Kommunikationsbeziehung dieser Nachricht genausogut ohne diese Ver- und Entschlüsselung wie mit ihr aufdecken können – dieses Ver- und Entschlüsseln ist also für den Schutz der Kommunikationsbeziehung im Sinne des Erreichens des gerade formulierten Zieles ohne Belang, zumal sich dadurch die Beteiligung von Sendern und Empfängern nicht ändert.

Aus dem oben formulierten Ziel folgt ebenfalls, daß *alle Nachrichten gleicher Länge des betrachteten Zeitintervalls die MIXe jeweils gleichzeitig (und deshalb auch in gleicher Reihenfolge) durchlaufen müssen*: Durchlaufen nicht alle Nachrichten gleicher Länge des betrachteten Zeitintervalls die MIXe jeweils gleichzeitig, so gibt es einen MIX *m*, den zwei Nachrichten *N1* und *N2* nicht gleichzeitig durchlaufen. Arbeiten nun alle anderen MIXe zusammen, so können sie dies beobachten und damit die zu *N1* bzw. *N2* gehörigen Sender und Empfänger jeweils unterscheiden, vgl. Bild 5-23.

Durchlaufen alle Nachrichten gleicher Länge des betrachteten Zeitintervalls die MIXe jeweils gleichzeitig und sendet und empfängt jede Teilnehmerstation in jedem Zeitintervall mindestens eine Nachricht jeder Länge, wird das *Ziel* des Verfahrens der umcodierenden MIXe sogar *ohne Klasseneinteilung der Teilnehmer* erreicht.

Sendet überdies jede einzelne Teilnehmerstation in jedem Zeitintervall eine konstante Anzahl Nachrichten jeder Länge und empfängt sie eine konstante Anzahl Nachrichten jeder Länge, so wird

perfekte komplexitätstheoretische Unbeobachtbarkeit der Kommunikationsbeziehung vor Unbeteiligten und – sofern gewünscht – *perfekte komplexitätstheoretische Anonymität der Kommunikationspartner voreinander* als auch *perfekte komplexitätstheoretische Unverkettbarkeit der Kommunikationsbeziehungen* erreicht (vgl. §5.1.2).

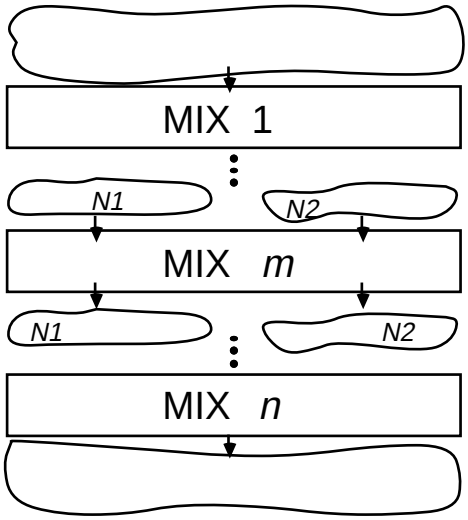


Bild 5-23: Maximaler Schutz: MIXe gleichzeitig und deshalb in gleicher Reihenfolge durchlaufen

Das Umcodieren der MIXe muß natürlich so gestaltet werden, daß der Empfänger die Nachricht trotzdem verstehen kann: Sofern die Nachricht bei ihm verschlüsselt ankommt, muß er alle zur Entschlüsselung notwendigen **Schlüssel** sowie die richtige Reihenfolge ihrer Anwendung bereits **kennen** oder während des Entschlüsselungsprozesses selbst kennenlernen. Außerdem muß natürlich jeder MIX in der Lage sein, die von ihm erwartete Umcodierung durchzuführen, er muß also den Schlüssel zur Ver- oder Entschlüsselung entweder bereits kennen oder während des Prozesses der Umcodierung selbst kennenlernen.

> Eine verschlüsselnde Umcodierung einer Nachricht muß also letztlich vom Empfänger dem durchführenden MIX spezifiziert werden, eine entschlüsselnde Umcodierung einer Nachricht von ihrem Sender.

Außerdem dürfen die vom MIX zu verwendenden Schlüssel diesem nichts Wesentliches über die Identität des Senders oder Empfängers der Nachricht verraten. Es ist wohl akzeptabel, daß der erste MIX einer Nachricht ihren Sender kennt und, sofern keine Verteilung zum Schutz des Empfängers verwendet wird, der letzte MIX den Empfänger. Einem „mittleren“ MIX sollte der zu verwendende Schlüssel nichts über den Sender oder Empfänger verraten, so daß in diesem Fall die Verwendung eines statisch fest ausgetauschten geheimen Schlüssels und damit informationstheoretische Konzelation praktisch nicht möglich ist. (In §5.4.5.4 wurde mit dem paarweisen überlagernden Empfangen zwar eine auf Senderanonymität basierende Methode zum Austausch eines Schlüssels beschrieben. Ist die Senderanonymität informationstheoretisch, so erfolgt der Schlüsselaustausch – ebenfalls in der informationstheoretischen Modellwelt – in Konzelation und Integrität garantierender sowie Anonymität erhaltender Weise. Da informationstheoretische Senderanonymität möglich, aber mit sehr, sehr großem Aufwand verbunden ist, ist das angedeutete Schlüsselaustausch-Verfahren zwar möglich, aber praktisch kaum anwendbar.) Bei „mittleren“ MIXen müssen daher öffentliche Schlüssel und damit asymmetrische Konzelationssysteme verwendet werden.

Also können Nachrichten an und damit auch Nachrichten von „mittleren“ MIXen (praktisch) nur in perfekte komplexitätstheoretische Konzelation garantierender Weise verschlüsselt werden, die durch sie erreichbare Anonymität ist also nur *komplexitätstheoretischer* Natur.

Nach diesen grundsätzlichen Überlegungen über Möglichkeiten und Grenzen des Umcodierens können nun konkrete Umcodierungsschemata systematisch hergeleitet werden. Zunächst soll dies mit Verallgemeinerungen der drei in [Chau_81] einfach angegebenen und dort bezüglich ihrer Grenzen und Notwendigkeit nicht genau diskutierten Umcodierungsschemata geschehen.

5.4.6.2 Senderanonymität

Gesucht sei ein Umcodierungsschema, das es einem Sender erlaubt, einem Empfänger eine Nachricht zukommen zu lassen, dabei aber vor ihm und allen (außer dem ersten) verwendeten MIXen anonym zu bleiben, sowie die Kommunikationsbeziehung zu verbergen, sofern nicht alle MIXe, die von der Nachricht durchlaufen werden, oder alle anderen Sender und Empfänger von Nachrichten im gleichen Zeitintervall zusammenarbeiten. Nach dem vorher Gesagten kann der Sender keinen geheimen Schlüssel mit dem Empfänger und den MIXen, vor denen er anonym bleiben will, verwenden. Also muß bezüglich ihnen ein asymmetrisches Konzelationssystem verwendet werden und der Sender dazu ihre Chiffrierschlüssel kennen.

> Mit den zur Verschlüsselung geeigneten Schlüsseln kann der Sender entweder die Nachricht direkt (*direktes Umcodierungsschema für Senderanonymität*) oder einen eine weitere Nachrichtenumcodierung spezifizierenden geheimen Schlüssel (*indirektes Umcodierungsschema für Senderanonymität*, wird weiter unten und dort gleich in der allgemeineren Form eines *indirekten Umcodierungsschemas* behandelt) verschlüsseln.

Direktes Umcodierungsschema für Senderanonymität: Der Absender einer Nachricht verschlüsselt sie so mit öffentlich bekannten Chiffrierschlüsseln eines asymmetrischen Konzelationssystems (bzw. für den ersten MIX der zu durchlaufenden MIX-Folge ggf. mit einem mit ihm vereinbarten geheimen Schlüssel eines symmetrischen Konzelationssystems), daß sie nacheinander von der von ihm gewählten Folge von MIXen mit deren zugehörigen geheimgehaltenen Dechiffrierschlüsseln (bzw. ggf. mit dem mit dem ersten MIX vereinbarten geheimen Schlüssel eines symmetrischen Konzelationssystems) entschlüsselt werden muß. Dadurch ändert sich das Erscheinungsbild der Nachricht auf jedem Stück ihres Weges, so daß ihr Weg (außer wenn alle MIXe, die sie durchläuft, zusammenarbeiten oder der Sender kooperiert oder alle anderen Sender und Empfänger von Nachrichten im gleichen Zeitintervall (genauer: Schub) zusammenarbeiten) nicht verfolgt werden kann.

Sei $A_1,...,A_n$ die Folge der Adressen und $c_1,...,c_n$ die Folge der öffentlich bekannten Chiffrierschlüssel der vom Sender gewählten MIX-Folge $MIX_1,...,MIX_n$, wobei c_1 auch ein geheimer Schlüssel eines symmetrischen Konzelationssystems sein kann. Sei A_{n+1} die Adresse des Empfängers, der zur Vereinfachung der Notation MIX_{n+1} genannt wird, und c_{n+1} sein Chiffrierschlüssel. Sei $z_1,...,z_n$ eine Folge zufälliger Bitketten (bit strings; die Bildung des deutschen Begriffes erfolgte in Analogie zu „Zeichenkette“). Ist c_1 ein geheimer Schlüssel eines symmetrischen Konzelationssystems, so kann z_1 die leere Bitkette sein. Ist c_i ein Chiffrierschlüssel eines bereits von sich aus indeterministisch verschlüsselnden asymmetrischen Konzelationssystems, so kann z_i ebenfalls die leere Bitkette sein. Der Sender bildet die Nachrichten N_i, die MIX_i erhalten wird, ausgehend von der Nachricht N, die der Empfänger (MIX_{n+1}) erhalten soll:

$$N_{n+1} = c_{n+1}(N)$$
$$N_i = c_i(z_i, A_{i+1}, N_{i+1}) \quad \text{für } i=n,...,1$$

Der Sender sendet N_1 an MIX_1. Jeder MIX erhält nach der Entschlüsselung die Adresse des nächsten und die für diesen bestimmte Nachricht.

> Die Mitverschlüsselung zufälliger Bitketten ist bei Verwendung eines deterministischen asymmetrischen Konzelationssystems nötig, da ein Angreifer ansonsten nicht nur kurze Standardnachrichten erraten und mit dem öffentlich bekannten Chiffrierschlüssel testen kann, sondern in diesem Fall sogar (ganz ohne Raten) die gesamte Ausgabe des MIXes testen könnte.

Bei diesem direkten Umcodierungsschema für Senderanonymität ist das Umcodieren einer Nachricht (zumindest bei allen außer dem ersten MIX) vollständig durch den jeweils öffentlich bekannten Chiffrierschlüssel bestimmt. Um nicht über Nachrichtenhäufigkeiten Entsprechungen zwischen Ein- und Ausgabe dieser MIXe entstehen zu lassen, *bearbeitet jeder MIX, solange er sein Schlüsselpaar beibehält, mit dem zugehörigen geheimgehaltenen Dechiffrierschlüssel nur unterschiedliche Eingabe-Nachrichten*. Erhält ein MIX während dieser Zeitspanne eine Eingabe-Nachricht mehrmals mit dem Auftrag, sie mit seinem geheimgehaltenen Dechiffrierschlüssel zu entschlüsseln, ignoriert er ihr wiederholtes Eintreffen.

c_j ist der Chiffrier-, d_j der Dechiffrierschlüssel von MIX j.
z_i sind zufällige Bitketten, N_i Nachrichten.

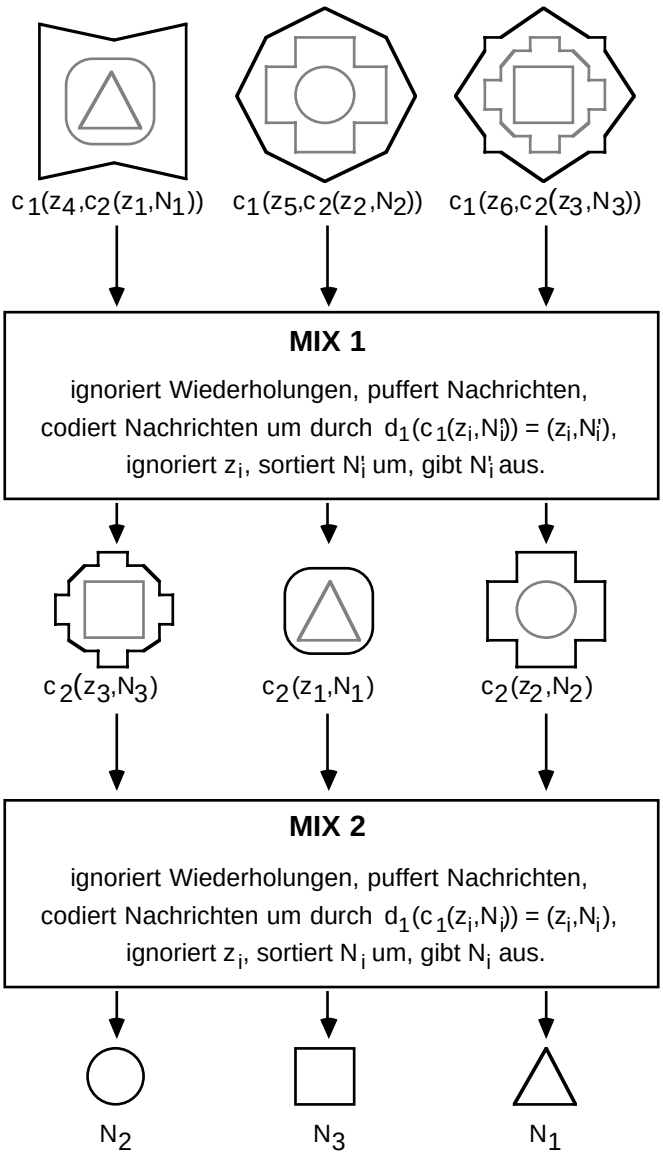


Bild 5-24: MIXe verbergen den Zusammenhang zwischen ein- und auslaufenden Nachrichten

Das bereits verbal beschriebene direkte Umcodierungsschema für Senderanonymität ist in Bild 5-24 anhand zweier MIXe noch einmal graphisch dargestellt. Allerdings wurden dort die Adressen weggelassen und die Indizes von Nachrichten und zufälligen Bitketten nur zu deren Unterscheidung benutzt, da bei Verwendung der Indizierungskonvention des rekursiven Bildungsschemas eine Doppelindizierung nötig geworden wäre.

In Bild 5-25 wird ein etwas größeres Beispiel (n=5) in einer graphischen Darstellung gezeigt, die betont, wer welche Schlüssel kennt und in welcher Reihenfolge Nachrichten verschlüsselt, transferiert und entschlüsselt.

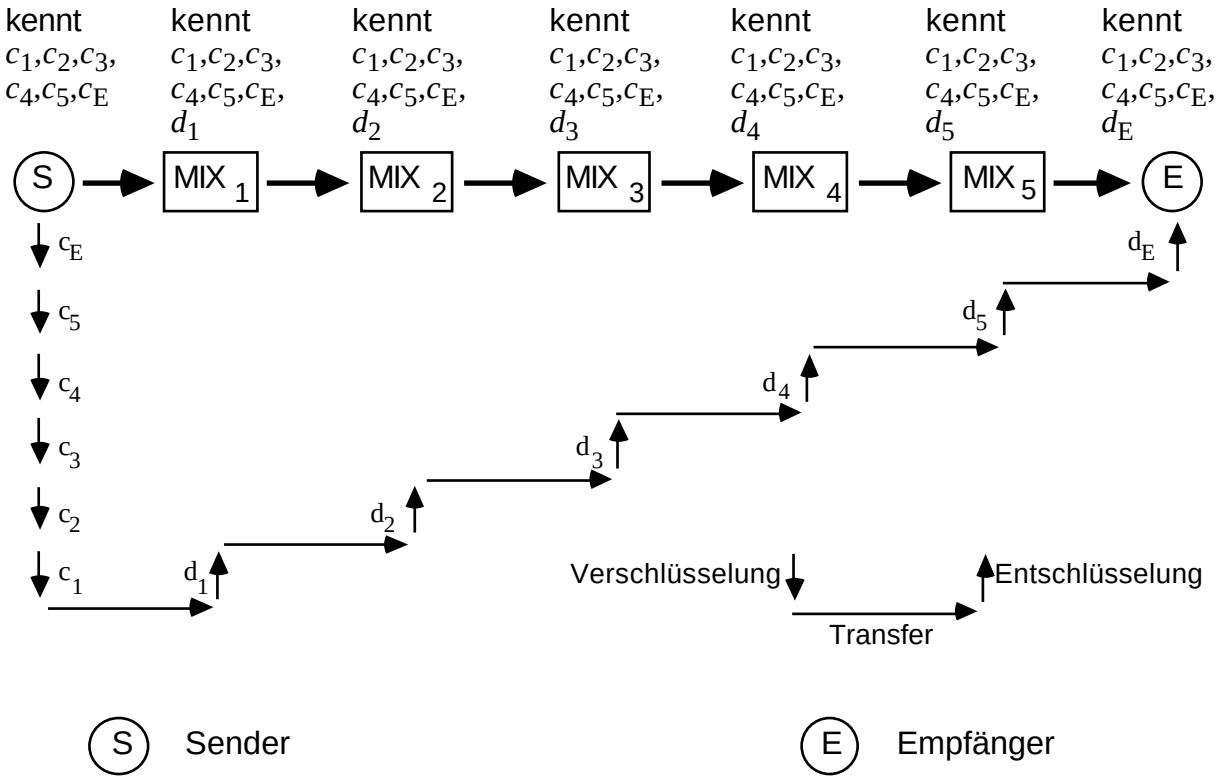


Bild 5-25: Transfer- und Verschlüsselungsstruktur der Nachrichten im MIX-Netz bei Verwendung eines direkten Umcodierungsschemas für Senderanonymität

5.4.6.3 Empfängeranonymität

Neben der Möglichkeit, eine Nachricht bezüglich einer Kommunikationsbeziehung anonym *senden* zu können, ist es zum Schutz der Kommunikationsbeziehung auch nötig, eine Nachricht bezüglich einer Kommunikationsbeziehung anonym *empfangen* zu können. Dies leistet das gerade beschriebene Umcodierungsschema lediglich in der Hinsicht nicht, daß der Sender die Nachricht N_{n+1} natürlich kennt und deshalb den Empfänger in einem normalen Vermittlungsnetz identifizieren kann, sofern

1. der Sender selbst oder jemand, der mit ihm zusammenarbeitet, die entsprechende *Leitung abhören* kann oder die Kooperation des Betreibers oder eines Herstellers des dem MIX-Netz zugrundeliegenden Kommunikationsnetzes hat, oder in vielen Fällen wesentlich einfacher,
2. die Adresse A_{n+1} eine *öffentliche oder explizite Adresse* ist, die einen Personenbezug aufweist oder deren, meist an der Topologie des Kommunikationsnetzes orientiertes Bildungsschema entweder allgemein bekannt ist bzw. auch anderenfalls meist erschlossen werden kann oder dem Sender vom Betreiber oder einem Hersteller des Kommunikationsnetzes der zu dieser Adresse gehörige „Ort“ mitgeteilt wird, oder
3. zusätzlich zu 1. oder 2. der Sender die *Kooperation von MIX_n* hat, was besonders wahrscheinlich ist, wenn er ihn auswählen kann.

Dies Problem kann nun natürlich bezüglich der 1. Bedrohung durch virtuelle Verbindungs-Verschlüsselung zwischen MIX_n und Empfänger [Pfi1_85 Seite 12, 14], bezüglich der 2. Bedrohung durch die Verwendung von impliziten privaten Adressen (beispielsweise könnte A_{n+1} eine zwischen MIX_n und dem Empfänger vereinbarte implizite private Adresse sein und von MIX_n durch eine explizite öffentliche des Kommunikationsnetzes ersetzt werden) und selbst bezüglich aller Bedrohungen mit dem in §5.4.1 beschriebenen Verfahren zum Schutz des Empfängers durch implizite Adressierung und Verteilung gelöst werden. Durch Letzteres wird sogar mehr als nur die Kommunikationsbeziehung geschützt. Je nach Struktur und Leistungsfähigkeit des zugrundeliegenden Kommuni-

kationsnetzes kann die Verteilung „letzter“ (d.h. direkt an den Empfänger gerichteter) Nachrichten die Nutzleistung des Kommunikationsnetzes jedoch in unakzeptabler Weise reduzieren. Also ist eine das Kommunikationsnetz weniger belastende Art des Schutzes des Empfängers bezüglich seiner Kommunikationsbeziehung vor dem Sender gesucht. Diese Art des Schutzes muß nicht unbedingt den Sender vor dem Empfänger schützen, denn sie könnte ja mit einem Umcodierungsschema für Senderanonymität kombiniert werden: Der Sender sendet seine Nachricht unter Verwendung des Umcodierungsschemas für Senderanonymität an irgendeine Instanz *X*, beispielsweise einen MIX, mit der Bitte, sie auf die den Empfänger vor dem Sender schützende Art dem Empfänger zukommen zu lassen. Für den Schutz der Kommunikationsbeziehung zwischen Sender und Empfänger (genauer: ihre *gegenseitige Anonymität*) ist es nun völlig unkritisch, daß der Sender den Empfang seiner Nachricht durch *X* und der Empfänger das Senden seiner Nachricht durch *X* beobachten kann. Nachdem Instanz *X* ihren Zweck – eine einfachstmögliche Erklärung des Prinzips – erfüllt hat, wird sie nun wegrationalisiert: Die Konkatenation der Umcodierungsschemata kann leicht so modifiziert werden, daß der letzte MIX des Senderanonymitätsschemas die bisher von *X* gesendete Nachricht gleich an den ersten MIX des Empfängeranonymitätsschemas sendet.

Ignoriert man (in Analogie zum Umcodierungsschema für Senderanonymität) also den Schutz des Senders bezüglich einer Kommunikationsbeziehung vor dem Empfänger, so ist ein Umcodierungsschema gesucht, das es einem Empfänger erlaubt, von einem Sender einen Nachrichteninhalt zu erhalten, dabei aber vor ihm und allen (außer dem letzten) verwendeten MIXen anonym zu bleiben, sowie die Kommunikationsbeziehung zu verbergen, sofern nicht alle MIXe, die von der Nachricht durchlaufen werden, oder alle anderen Sender und Empfänger von Nachrichten im gleichen Zeitintervall zusammenarbeiten. Um vor dem Sender anonym zu bleiben, muß der Empfänger die Umcodierungen den durchlaufenen MIXen spezifizieren – nicht etwa wie im Umcodierungsschema für Senderanonymität der Sender. Bezüglich des vom Sender generierten Nachrichteninhalts muß es sich also um verschlüsselndes Umcodieren handeln, wozu jedem MIX der von ihm zu verwendende Schlüssel mitgeteilt werden muß, da nach dem vorher Gesagten der Empfänger keinen statisch ausgetauschten geheimen Schlüssel mit dem Sender und den MIXen, vor denen er anonym bleiben will, verwenden kann. Diese Schlüsselmitteilung geschieht am praktischsten dadurch, daß dem vom Sender generierten Nachrichteninhalt noch ein zweiter, **Rückadreßteil** genannter, Nachrichtenteil mitgegeben wird, der Teil einer vom Empfänger gebildeten sogenannten **anonymen Rückadresse** (untraceable return address, [Chau_81]) ist. Der Rückadreßteil muß von jedem MIX mit dem zu seinem öffentlich bekannten Chiffrierschlüssel gehörigen geheimgehaltenen Dechiffrierschlüssel entschlüsselt werden (ggf. kann bezüglich des letzten MIXes stattdessen auch ein zwischen ihm und dem Empfänger ausgetauschter geheimer Schlüssel eines symmetrischen Konzelationssystems verwendet werden), um ihm als Inhalt nicht nur den Rückadreßteil für den nächsten MIX, sondern auch den von ihm für die Verschlüsselung der vom Sender generierten Nachricht zu verwendenden Schlüssel bekannt zu machen. Also gilt:

> Bei einem *Umcodierungsschema für Empfängeranonymität* muß es sich um ein *indirektes* Umcodierungsschema handeln.

Indirektes Umcodierungsschema für Empfängeranonymität: Sei $A_1,...,A_m$ die Folge der Adressen und $c_1,...,c_m$ die Folge der öffentlich bekannten Chiffrierschlüssel der vom Empfänger gewählten MIX-Folge $MIX_1,...,MIX_m$, wobei c_m auch ein geheimer Schlüssel eines symmetrischen Konzelationssystems sein kann. Die von der anonymen Rückadresse begleitete Nachricht wird diese MIXe in aufsteigender Reihenfolge ihrer Indizes durchlaufen. Sei A_{m+1} die Adresse des Empfängers, der zur Vereinfachung der Notation MIX_{m+1} genannt wird. Ebenfalls zur Vereinfachung werde der Sender MIX_0 genannt. Der Empfänger bildet eine **anonyme Rückadresse** (k_0,A_1,R_1), wobei k_0 ein für diesen Zweck generierter Schlüssel eines symmetrischen Konzelationssystems (ein asymmetri-

sches Konzelationssystem wäre natürlich auch möglich, aber aufwendiger) ist, mit dem MIX_0 den Nachrichteninhalt verschlüsseln soll, damit MIX_1 ihn nicht lesen kann, und R_1 der Teil der anonymen Rückadresse ist, der von MIX_0 dem von ihm generierten und mit k_0 verschlüsselten Nachrichteninhalt mitgegeben wird. R_1 wird vom Empfänger ausgehend von einem zufällig gewählten eindeutigen Namen *e* der Rückadresse nach dem folgenden rekursiven Schema gebildet, bei dem R_j jeweils den Rückadreßteil bezeichnet, den MIX_j erhalten wird, und k_j jeweils den Schlüssel eines symmetrischen Konzelationssystems (ein asymmetrisches Konzelationssystem wäre auch möglich, aber aufwendiger), mit dem MIX_j den den Nachrichteninhalt enthaltenden Nachrichtenteil verschlüsseln soll:

$$R_{m+1} = e$$
$$R_j = c_j(k_j, A_{j+1}, R_{j+1}) \quad \text{für } j=m,...,1$$

Diese Rückadreßteile R_j und der (ggf. bereits mehrmals verschlüsselte) vom Sender generierte Nachrichteninhalt *I*, *Nachrichteninhaltsteil* I_j genannt, bilden zusammen die Nachrichten N_j, die jeweils von MIX_{j-1} gebildet und an MIX_j gesendet werden – nach dem folgenden rekursiven Schema also zuerst vom Sender MIX_0 und dann der Reihe nach von den durchlaufenen MIXen $MIX_1,...,MIX_m$:

$$N_1 = R_1, I_1; \qquad I_1 = k_0(I)$$
$$N_j = R_j, I_j; \qquad I_j = k_{j-1}(I_{j-1}) \quad \text{für } j=2,...,m+1$$

Der Empfänger MIX_{m+1} erhält also $e, N_{m+1} = e, k_m(...k_1(k_0(I))...)$ und kann, da er dem eindeutigen Namen *e* der Rückadresse alle geheimen Schlüssel k_j (im Falle der Verwendung eines asymmetrischen Konzelationssystems: alle Dechiffrierschlüssel) in richtiger Reihenfolge zuordnen kann, ohne Probleme entschlüsseln und so den Nachrichteninhalt *I* gewinnen.

> Die Mitverschlüsselung zufälliger Bitketten ist auch bei Verwendung von deterministischen Konzelationssystemen nicht nötig, da die mit öffentlich bekannten Chiffrierschlüsseln der MIXe verschlüsselten Nachrichtenteile R_j mit dem nicht ausgegebenen k_j genügend dem Angreifer nicht bekannte Information enthalten und bezüglich des Umcodierens mit dem Angreifer unbekannten Schlüsseln diesem sowieso kein Testen möglich ist.

Bei diesem indirekten Umcodierungsschema für Empfängeranonymität ist nur das Umcodieren des Rückadreßteils (zumindest bei allen außer dem letzten MIX) vollständig durch den jeweils öffentlich bekannten Chiffrierschlüssel bestimmt. Um nicht über Nachrichtenhäufigkeiten Entsprechungen zwischen Ein- und Ausgabe dieser MIXe entstehen zu lassen, *bearbeitet jeder MIX (solange er sein Schlüsselpaar beibehält) nur unterschiedliche Rückadreßteile*. Hieraus folgt:

> Jede anonyme Rückadresse kann nur einmal verwendet werden.

Eine Wiederholung des Nachrichteninhaltsteils I_j ist unkritisch, sofern zum Umcodieren jeweils ein unterschiedlicher Schlüssel verwendet wird. Dies ist, da jede anonyme Rückadresse nur einmal verwendet werden kann, immer dann der Fall, wenn beim Bilden anonymer Rückadressen jeweils neue Schlüssel k_j verwendet werden. Beachtet ein Empfänger beim Bilden von anonymen Rückadressen diese Regel nicht, schadet er sich nur selbst. Verwendet ein MIX_j einen alten Schlüssel k_j zur Bildung einer Rückadresse, kann er dadurch dem Generierer von k_j auch nicht mehr schaden als durch Verraten des Nachrichtenzusammenhangs, der durch das ursprüngliche Umcodieren mit k_j verborgen werden sollte. Bild 5-29 zeigt das Schema graphisch.

5.4.6.4 Gegenseitige Anonymität

Wird dieses Umcodierungsschema für Empfängeranonymität allein verwendet, so verbirgt es (wie das Umcodierungsschema für Senderanonymität auch) zwar die Kommunikationsbeziehung vor

Unbeteiligten vollständig. Da hier der Empfänger jedoch den Rückadreßteil R_1 kennt, kann er den Sender beobachten, sofern

1. der Empfänger selbst oder jemand, der mit ihm zusammenarbeitet, die entsprechende *Leitung abhören* kann oder die Kooperation des Betreibers oder eines Herstellers des dem MIX-Netz zugrundeliegenden Kommunikationsnetzes hat, oder in vielen Fällen wesentlich einfacher,
2. der Empfänger die *Kooperation von* MIX_1 hat, was besonders wahrscheinlich ist, wenn er ihn auswählen kann, und das dem MIX-Netz zugrundeliegende Kommunikationsnetz gesendeten Nachrichten *öffentliche oder explizite Absenderadressen* zuordnet (und den Netzbenutzern mitteilt – wie im geplanten ISDN vorgesehen), die einen Personenbezug aufweisen oder deren, meist an der Topologie des Kommunikationsnetzes orientiertes Bildungsschema entweder allgemein bekannt ist bzw. auch anderenfalls meist erschlossen werden kann oder dem Sender vom Betreiber oder einem Hersteller des Kommunikationsnetzes der zu dieser Adresse gehörige „Ort“ mitgeteilt wird, oder
3. zusätzlich zu 1. der Empfänger die Kooperation von MIX_1 hat, was – wie schon erwähnt – besonders wahrscheinlich ist, wenn er ihn auswählen kann.

Dies Problem kann nun natürlich auch bezüglich der 1. Bedrohung durch virtuelle Verbindungs-Verschlüsselung [Pfi1_85 Seite 12, 14], bezüglich der 2. Bedrohung durch geeignete Protokollgestaltung im dem MIX-Netz zugrundeliegenden Kommunikationsnetz und bezüglich aller Bedrohungen mit den in §5.4.4 beschriebenen Verfahren zum Schutz des Senders gelöst werden. Durch Letzteres wird sogar mehr als nur die Kommunikationsbeziehung geschützt. Je nach Struktur und Leistungsfähigkeit des zugrundeliegenden Kommunikationsnetzes können diese Verfahren die Nutzleistung des Kommunikationsnetzes in unakzeptabler Weise reduzieren, so daß dann – sofern die Anwendung gegenseitige Anonymität fordert – nur die bereits beschriebene Kombination von einem Umcodierungsschema für Sender- und einem für Empfängeranonymität möglich ist [Pfi1_85 Seite 14f]. (Die Idee ist vermutlich auch in [Chau_81 Seite 85] enthalten, die dortige Formulierung in ihrer Allgemeinheit jedoch falsch.) Diese gegenseitige Anonymität muß natürlich auch bei der ersten Nachricht vorhanden sein, was durch den schon früher beschriebenen Indirektionsschritt über eine (hier tatsächlich nötige) zusätzliche Instanz *X* (bzw. zumindest Funktionalität, die natürlich auch von einer vorhandenen Instanz erbracht werden kann), hier ein **Adreßverzeichnis mit anonymen Rückadressen**, erreicht wird. Da jede anonyme Rückadresse nur einmal verwendet werden kann, ist die Verwendung gedruckter Adreßverzeichnisse sehr umständlich, die Verwendung elektronischer, die jede Adresse nur einmal ausgeben, jedoch kein Problem.

5.4.6.5 Längentreue Umcodierung

Nachdem nun die einfachstmöglichen Umcodierungsschemata für Sender- oder Empfängeranonymität hergeleitet und ihre Einsatzmöglichkeiten beschrieben wurden, soll noch auf eine weitere gemeinsame Eigenschaft hingewiesen werden: In beiden Umcodierungsschemata ändert sich beim Umcodieren die Länge von Nachrichten – sie werden kürzer. Dies ist, wenn alle Nachrichten die gleichen MIXe in gleicher Reihenfolge durchlaufen, was – wie in §5.4.6.1 gezeigt – zum Erreichen des maximal möglichen Anonymitätszieles nötig ist, kein Problem, vgl. Bild 5-26 oben. Nun können – wie in [Pfit_89 §3.2.2.4] gezeigt ist – vor allem Leistungs-, aber auch Kostengesichtspunkte verhindern, daß alle Nachrichten in allen MIXen umcodiert werden. In solchen MIX-Netzen können dann die abnehmenden Nachrichtenlängen einem Angreifer durchaus wertvolle Information liefern. In Bild 5-26 unten ist leicht zu erkennen, daß die Nachrichten innerhalb der MIXe nicht über Kreuz gehen, was im oberen Teil des Bildes nicht zu erkennen ist.

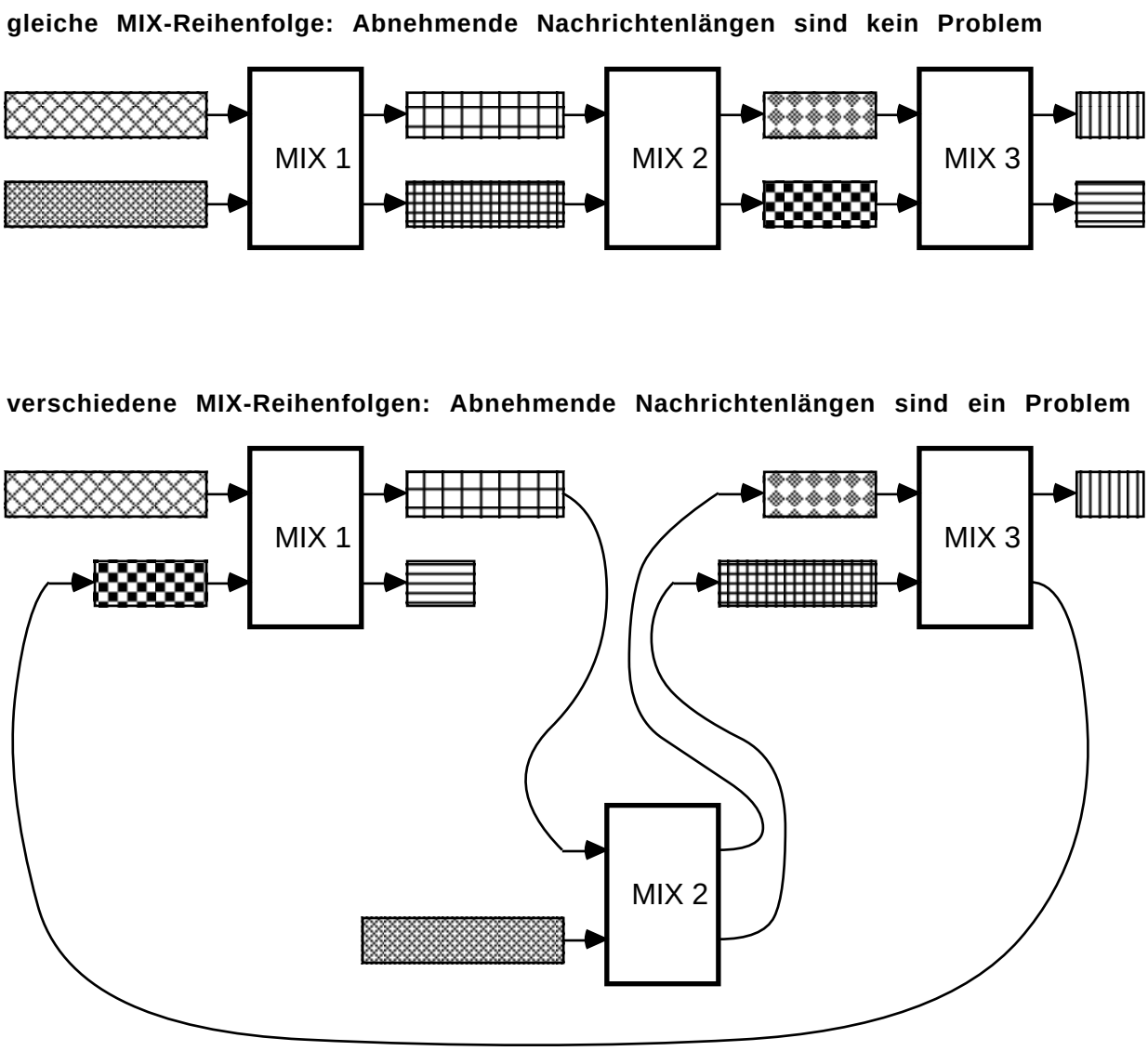


Bild 5-26: Abnehmende Nachrichtenlänge beim Umcodieren

Deshalb ist es wünschenswert, ein Umcodierungsschema zu haben, daß die Länge einer Nachricht beim Umcodieren gleich läßt, so daß ihr (außer ihren ursprünglichen Generierern) niemand ansehen kann, wie oft so bereits bzw. noch umcodiert wurde bzw. werden muß. Außerdem sollten – wie zu Beginn dieses Abschnittes gezeigt – alle Nachrichten gleiche Länge haben und auch durch das verwendete Umcodierungsschema nicht unterschieden werden. Gesucht wird also ein universelles *längentreues Umcodierungsschema*, das nach dem bei der Herleitung eines Umcodierungsschemas für Empfängeranonymität Gesagten zwangsläufig ein *indirektes* Umcodierungsschema sein muß.

Indirektes längentreues Umcodierungsschema: Zusätzlich zu den beim indirekten Umcodierungsschema für Empfängeranonymität bereits eingeführten Bezeichnungen bedeuten eckige Klammern Blockgrenzen. Ausgehend vom Sender (auch MIX_0 genannt) sollen der Reihe nach die MIXe $MIX_1,...,MIX_m$ durchlaufen und schließlich der Empfänger (auch MIX_{m+1} genannt) erreicht werden. Der Empfänger braucht bei Erhalt der Nachricht noch nicht zu wissen, daß er ihr Empfänger ist. Er bearbeitet sie zunächst wie alle vorherigen MIXe.

Jede Nachricht N_j besteht aus b Blöcken gleicher, auf das von den MIXen verwendete asymmetrische Konzelationssystem abgestimmter Länge und wird von MIX_{j-1} gebildet. Die ersten $m+2-j$ Blöcke von N_j enthalten den (Rück-)Adreßteil R_j, die letzten $b-m-1$ Blöcke den Nachrichteninhalt. Jeder MIX_j entschlüsselt den ersten Block der Nachricht N_j mit seinem geheimgehaltenen Dechiffrier-

schlüssel d_j und findet als Ergebnis dieser Entschlüsselung einen Schlüssel k_j eines auf die Blocklänge abgestimmten symmetrischen Konzelationssystems (ein asymmetrisches Konzelationssystem, praktischerweise dasselbe, das von den MIXen verwendet wird, ist auch möglich, aber aufwendiger) zum Umcodieren der übrigen Nachricht und die Adresse A_{j+1} des nächsten MIXes (oder Empfängers). Der Empfänger findet als Adresse des nächsten MIXes (oder Empfängers) seine eigene und erkennt daran, daß er der Empfänger dieser Nachricht ist. Dieser erste Block von N_j wird von den folgenden MIXen (bzw. dem Empfänger) nicht benötigt und deshalb von MIX_j weggeworfen. Mit k_j codiert MIX_j die restlichen b-1 Blöcke um und hängt vor den ersten Block mit Nachrichteninhalt, um die Länge der Nachricht nicht zu ändern, einen Block zufälligen Inhalts Z_j (in [Chau_81 Seite 87] wird auch dieser Block mit k_j umcodiert, was, da er zufälligen Inhalt hat, überflüssig ist).

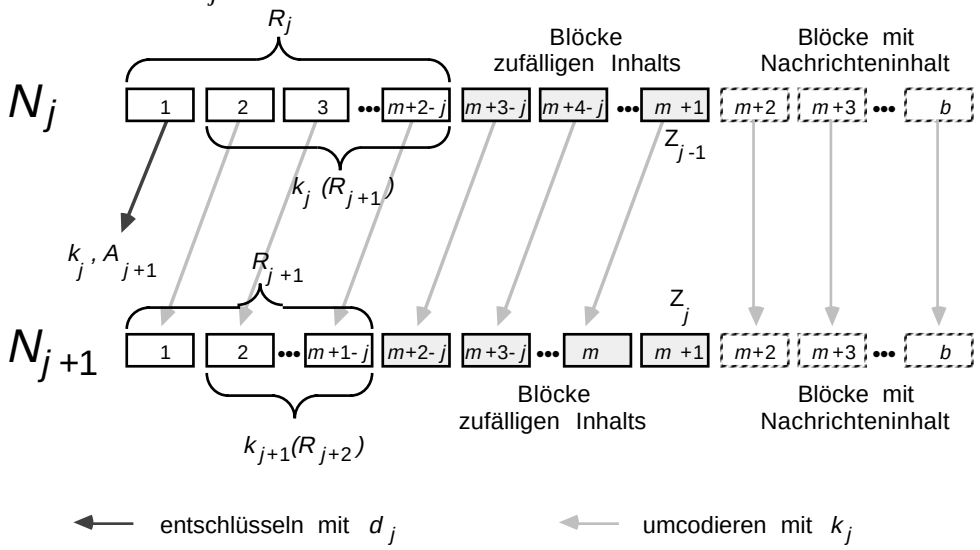


Bild 5-27: Indirektes längentreues Umcodierungsschema

Damit dies klappt, wird der vom Sender verwendete (Rück-)Adreßteil R_1 (ausgehend von einem zufällig gewählten eindeutigen Namen e) nach dem folgenden rekursiven Schema gebildet:

$$R_{m+1} = [e]$$
$$R_j = [c_j(k_j,A_{j+1})],k_j(R_{j+1}) \qquad \text{für } j=m,\ldots,1$$

Das Bildungsschema der Nachrichten N_j und insbesondere die Längentreue dieses rekursiven Umcodierungsschemas ist in Bild 5-27 veranschaulicht. Da MIX_j seinen Index j nicht zu kennen braucht, braucht er zwischen Blöcken des (Rück-)Adreßteils und Blöcken zufälligen Inhalts nicht unterscheiden zu können. Bezogen auf Bild 5-27 bedeutet dies, daß MIX_j lediglich von Block m+1 der Nachricht N_{j+1} weiß, daß er ein Block zufälligen Inhalts (im Bild: grau hinterlegt) ist.

> Da alle MIXe m erfahren müssen, um zu wissen, wo sie ihren Block zufälligen Inhalts einfügen sollen, kennen sie damit sowohl m als obere Schranke für die Anzahl der zu durchlaufenden MIXe als auch b-m-1 als obere Schranke für die Länge (in Blöcken) des Nachrichteninhalts.

Dem Bildungsschema des (Rück-)Adreßteils ist zu entnehmen, daß alle Blöcke des (Rück-)Adreßteils außer dem ersten von MIX_j mit k_j zu *entschlüsseln* sind. Ob die letzten, den Nachrichteninhalt enthaltenden b-m-1 Blöcke mit k_j zu *ent-* oder zu *verschlüsseln* sind, hängt davon ab, ob die Umcodierung durch MIX_j der Sender- oder Empfängeranonymität dient. Soll die Umcodierung mit k_j der Senderanonymität dienen, muß MIX_j entschlüsseln, da der Empfänger ansonsten den Schlüssel k_j auch erfahren müßte – dann wäre die Umcodierung als Schutz vor ihm aber nutzlos – oder den Nachrichteninhalt nicht verstehen könnte. Entsprechend muß verschlüsselt werden, soll die Umcodierung der

Empfängeranonymität dienen. Statt in obiges rekursives Bildungsschema für die (Rück-)Adreßteile noch zusätzliche Bits aufzunehmen, die den MIXen anzeigen, ob sie die Blöcke mit Nachrichteninhalt jeweils ver- oder entschlüsseln sollen, wird diese Information als Teil der Schlüssel k_j betrachtet.

Damit MIX_j seinen Index j nicht zu kennen braucht, sollten die von ihm erhaltenen Blöcke zufälligen Inhalts genauso behandelt werden wie alle Blöcke (außer dem ersten) des (Rück-)Adreßteils R_j. Sie sollten also entschlüsselt werden. Dies wird formal dadurch ausgedrückt, daß die (Rück-)Adresse R_j und die Blöcke zufälligen Inhalts zum Nachrichtenkopf H_j (header) zusammengefaßt werden. Im folgenden bedeute $[H_j]_{x,y}$ die Blöcke x bis y (jeweils einschließlich) von H_j. Ist $x>y$, so bedeutet $[H_j]_{x,y}$ keinen Block.

$$\begin{aligned} H_1 &= R_1 \\ &= [c_1(k_1,A_2)],k_1(R_2) \end{aligned}$$

$$\begin{aligned} H_j &= k_{j-1}^{-1}([H_{j-1}]_{2,m+1}),[Z_j] \\ &= [c_j(k_j,A_{j+1})],k_j(R_{j+1}),k_{j-1}^{-1}([H_{j-1}]_{m+3-j,m+1}),[Z_j] \qquad \text{für } j=m,...,2 \end{aligned}$$

Soll das indirekte längentreue Umcodierungsschema für **Senderanonymität** eingesetzt werden, so muß der Sender die Schlüssel k_1, k_2, ..., k_m generieren und den Chiffrierschlüssel c_{m+1} und die Adresse A_{m+1} des Empfängers kennen. Damit der Nachrichteninhalt vom Empfänger verstanden werden kann, muß ihn der Sender vor dem Absenden der Nachricht mit den von ihm generierten Schlüsseln verschlüsseln. Also wird die Nachricht N_1 vom Sender nach dem folgenden Schema ausgehend von dem in $i=b-m-1$ Blöcke zerlegten Nachrichteninhalt $I=[I1],...,[Ii]$ (ist der Nachrichteninhalt zu kurz, muß er auf die passende Länge aufgefüllt werden) und dem Nachrichtenkopf H_1 gebildet:

$$\begin{aligned} N_1 &= H_1,I_1; \qquad I_1 = k_1(k_2(...k_m(c_{m+1}(I_1))...)) \\ &\qquad\qquad = k_1(k_2(...k_m(c_{m+1}([I1]))...)),...,k_1(k_2(...k_m(c_{m+1}([Ii]))...)) \end{aligned}$$

$$N_j = H_j,I_j; \qquad I_j = k_{j-1}^{-1}(I_{j-1}); \qquad \text{für } j=2,...,m+1$$

Der aus jeweils $m+2-j$ Blöcken bestehende (Rück-)Adreßteil R_j, $j-1$ Blöcke zufälligen, ggf. bereits mehrmals „entschlüsselten“ Inhalts sowie der ggf. bereits mehrmals entschlüsselte, vom Sender generierte und von ihm mit den Schlüsseln c_{m+1}, k_m, ... k_1 verschlüsselte Nachrichteninhaltsteil I_j bilden zusammen die Nachricht N_j. Aus ihr bildet MIX_j nach obigem rekursiven Schema die Nachricht N_{j+1}.

Bild 5-28 veranschaulicht in der von Bild 5-25 bekannten graphischen Darstellung den Einsatz des längentreuen Umcodierungsschemas für Senderanonymität am Beispiel $m=5$.

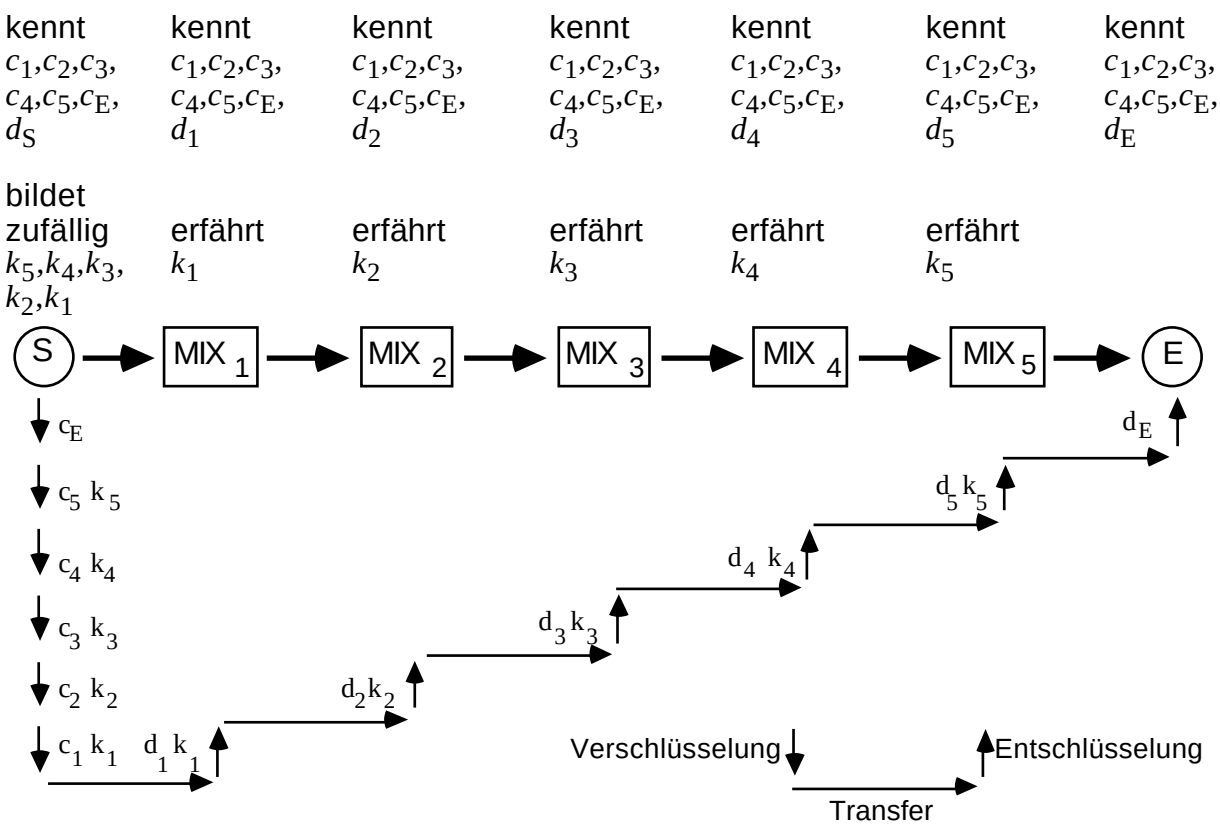


Bild 5-28: Indirektes Umcodierungsschema für Senderanonymität

Soll das indirekte längentreue Umcodierungsschema für **Empfängeranonymität** eingesetzt werden, so generiert der Empfänger die Schlüssel k_0, k_1, ..., k_m und die anonyme Rückadresse (k_0,A_1,R_1), die er dem Sender mitteilt. Der Sender bildet unter Verwendung von (k_0,A_1,R_1) die Nachricht N_1 sowie jeder MIX_j unter Verwendung von N_j die Nachricht N_{j+1} nach dem folgenden rekursiven Schema:

$$N_1 = H_1,I_1; \qquad I_1 = k_0(I) = k_0([I1]),...,k_0([Ii])$$

$$N_j = H_j,I_j; \qquad I_j = k_{j-1}(I_{j-1}) \qquad \text{für } j=2,...,m+1$$

Nach Erhalt von N_{m+1} erkennt der Empfänger an dem eindeutigen Namen e die Rückadresse und kann mit den zugehörigen, ihm bekannten Schlüsseln k_m, k_{m-1}, ..., k_1, k_0 entschlüsseln und so den Klartext erhalten.

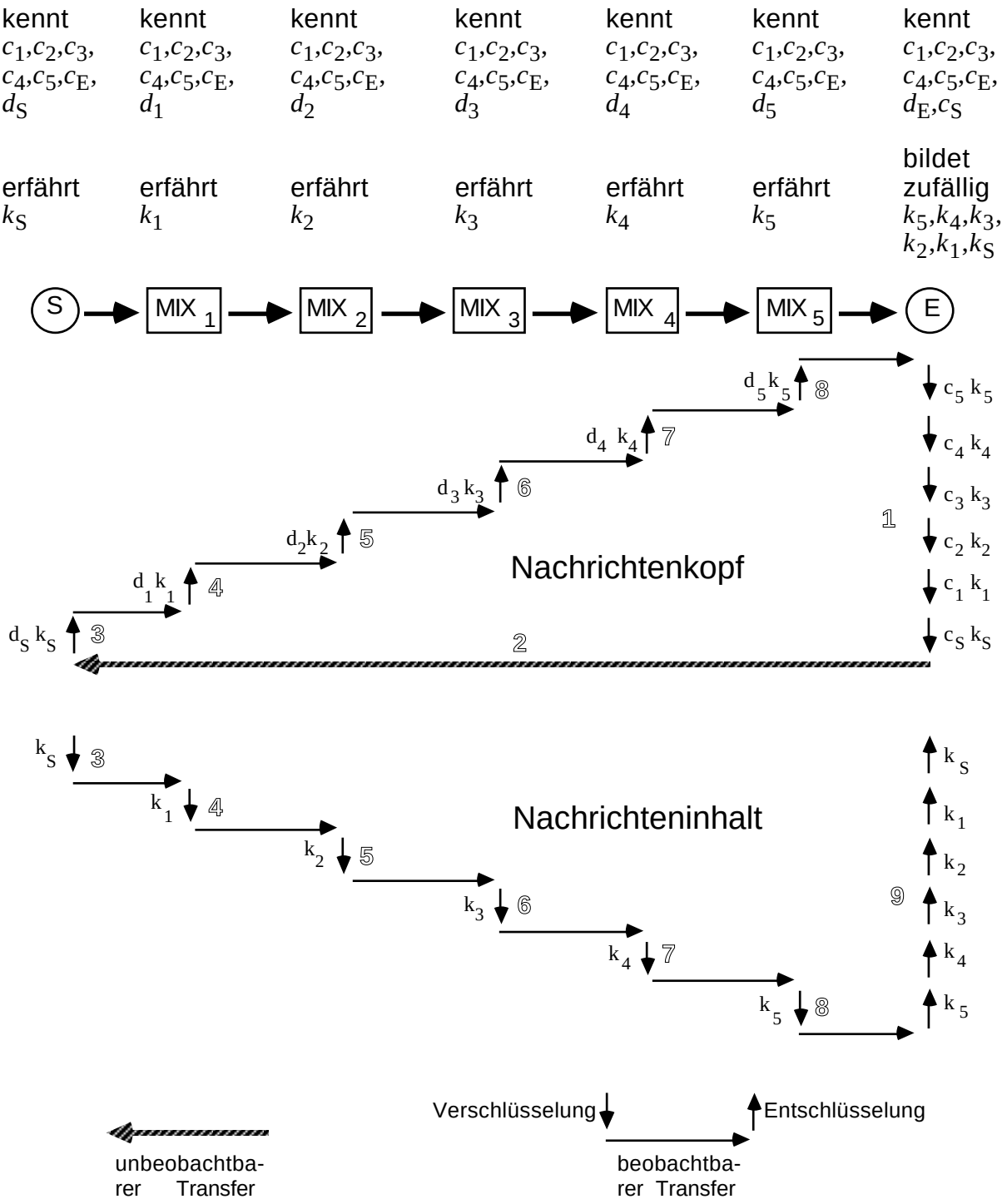


Bild 5-29: Indirektes Umcodierungsschema für Empfängeranonymität

Dies ist in Bild 5-29 für das Beispiel m=5 in der graphischen Darstellung von Bild 5-25 gezeigt. Nicht dargestellt ist, wie genau E dem Sender S die anonyme Rückadresse zukommen läßt. Die Zahlen in Konturschrift geben die Reihenfolge der Schritte an.

Sollen mit diesem indirekten längentreuen Umcodierungsschema **Sender und Empfänger** voreinander in überprüfbarer Weise **anonym** bleiben, müssen die Blöcke der (Rück-)Adresse von m bis zu einem gewissen Index g ($1<g\leq m$) vom Sender (sei es als (Rück-)Adresse in einem Adreßverzeichnis oder als Absender in der vorherigen Nachricht) und ab g-1 vom Empfänger gebildet werden. Der Sender erhält also R_g und bildet aus diesen m+2-g Blöcken die aus m+1 Blöcken bestehende (Rück-)Adresse R_1. Er verschlüsselt den Nachrichteninhalt mit dem ihm als Teil der Rückadresse mit-

geteilten Schlüssel k_S und den g-1 von ihm generierten Schlüsseln. Der Empfänger muß nach Erhalt des mehrfach verschlüsselten Nachrichteninhaltes mit den Schlüsseln k_g, ..., k_m, k_S entschlüsseln.

Bild 5-30 verdeutlicht dies am Beispiel m=5 und g=4 in der graphischen Notation von Bild 5-25. Wie in Bild 5-29 geben die Zahlen in Konturschrift die Reihenfolge der Schritte an.

Um Übertragungsaufwand und -zeit zu sparen, kann die Funktion des letzten MIXes des Senderanonymitätsteils und des ersten MIXes des Empfängeranonymitätsteils von einem MIX erbracht werden (im Beispiel von Bild 5-30 die Funktion der MIXe MIX_3 und MIX_4). Dies ist immer dann sinnvoll, wenn Sender und Empfänger keine speziellen Vertrauensvorlieben für einzelne MIXe haben.

Sollen Nachrichten zwischen gegenseitig anonymen Partnern mit Rückadressen versehen sein, so fällt an dieser Stelle auf, daß von den i Nachrichteninhaltsblöcken mitnichten alle für den eigentlichen Nachrichteninhalt zur Verfügung stehen: Es werden m+2-g für die Rückadresse benötigt – und dies in jeder Nachricht (wenn auch g für jede Nachricht unterschiedlich gewählt werden kann), denn genau wie beim indirekten Umcodierungsschema für Empfängeranonymität *darf jeder MIX (solange er sein Schlüsselpaar beibehält) auch bei dem indirekten längentreuen Umcodierungsschema nur unterschiedliche (Rück-)Adreßteile bearbeiten.*

Entsprechendes gilt für die umzucodierenden Blöcke, die zur *selben* Nachricht gehören und folglich mit demselben Schlüssel umcodiert werden: ist das verwendete Konzelationssystem eine Blockchiffre, so dürfen keine zwei umzucodierenden Blöcke gleich sein. Entsprechendes gilt für eine selbstsynchronisierende Stromchiffre, vgl. §3. Lediglich bei Verwendung einer synchronen Stromchiffre brauchen MIXe keine diesbezüglichen Maßnahmen zu ergreifen. Das Gesagte gilt sinngemäß bei jedem indirekten Umcodierungsschema, ist hier aber, da explizit von „Blöcken“ geredet wird, besonders hervorzuheben. Dieser **aktive Angriff durch Wiederholung indirekt umzucodierender Nachrichtenteile** wurde in [Chau_81, Pfi1_85 Seite 14, 27] übersehen.

Außerdem sei noch einmal hervorgehoben, daß die durch die Schlüssel c_2 bis c_{m-1} spezifizierten Umcodierungen des (Rück-)Adreßteils natürlich nur komplexitätstheoretisch sicher sein können und die von den MIXen MIX_2 bis MIX_{m-1} erreichbare Anonymität der Kommunikationsbeziehung also auch nur komplexitätstheoretischer Natur sein kann. Die durch c_1 bzw. c_m spezifizierten Umcodierungen können jedoch durchaus mit informationstheoretischer Sicherheit erfolgen, indem Sender bzw. Empfänger mit MIX_1 bzw. MIX_m einen geheimen Schlüssel ausgetauscht haben. Sofern Sender oder Empfänger die Reihenfolge der MIXe frei wählen können, sollten sie diesen geheimen Schlüssel jeweils mit dem MIX austauschen, dem sie am meisten vertrauen. Wie bereits begründet schließt eine solche freie Wahl der MIX-Reihenfolge das Erreichen des maximalen Zieles des Verfahrens der umcodierenden MIXe jedoch aus.

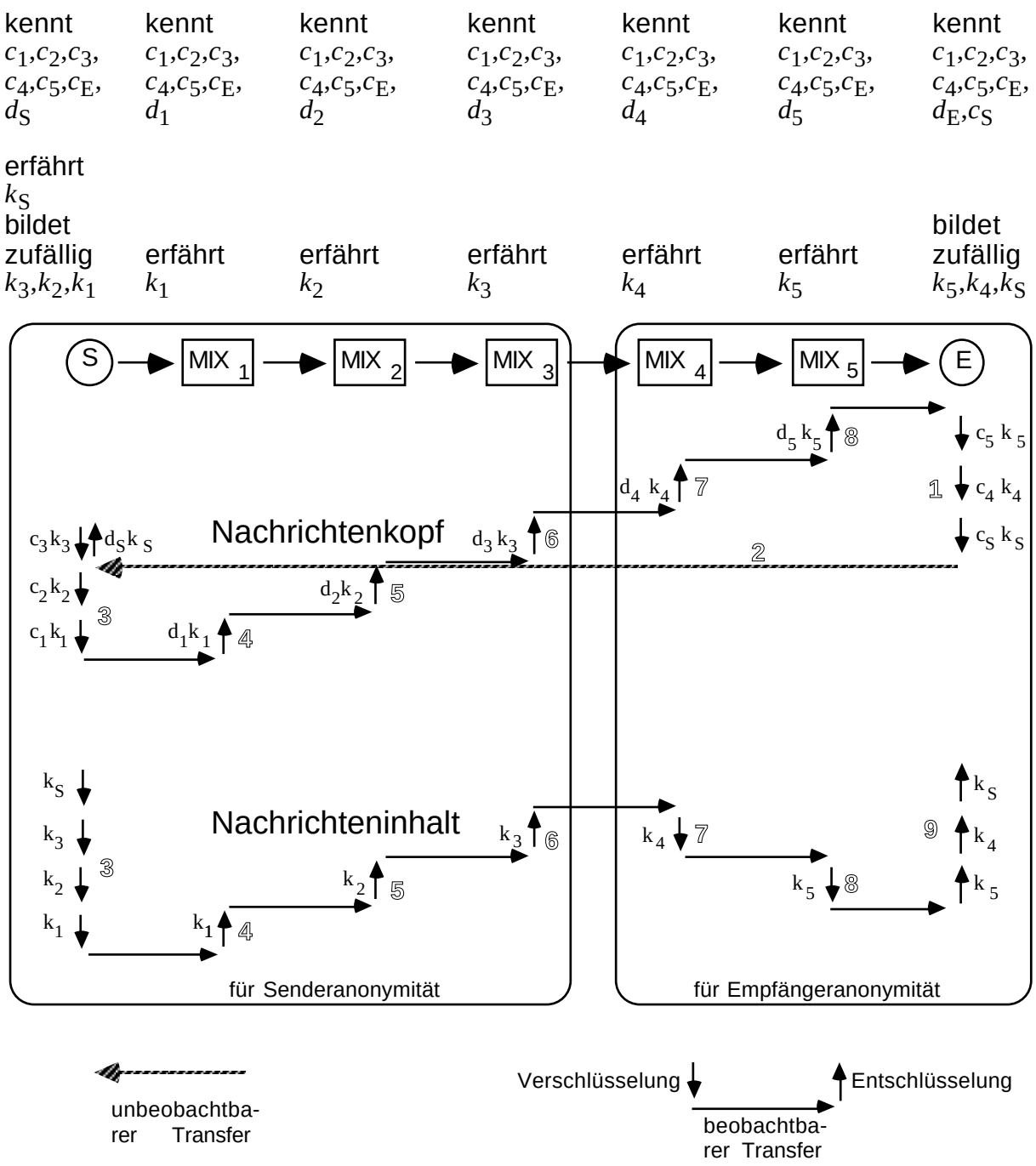


Bild 5-30: Indirektes Umcodierungsschema für Sender- und Empfängeranonymität

Hat das symmetrische Konzelationssystem zusätzlich zu der in §3 für alle Schlüssel *k* und Klartexte *x* geforderten Eigenschaft $k^{-1}(k(x)) = x$ **auch die Eigenschaft** $\boldsymbol{k(k^{-1}(x)) = x}$, d.h. sind nicht nur Ver- und Entschlüsselung, sondern auch Ent- und Verschlüsselung zueinander invers, so braucht beim obigen indirekten längentreuen Umcodierungsschema nicht zwischen dem Gebrauch für Sender- oder Empfängeranonymität unterschieden zu werden, wie im folgenden erläutert wird. Zusätzlich kann damit dem einzelnen MIX auch die Kenntnis von *m* als obere Schranke für die Anzahl der zu durchlaufenden MIXe als auch *b*-*m*-1 als obere Schranke für die Länge (in Blöcken) des Nachrichteninhalts vorenthalten werden: Statt als *m*+1-ten Block fügt jeder MIX seinen Block zufälligen Inhalts als letzten Block an. Der Empfänger erhält den Nachrichteninhalt dann in den Blöcken ab Block zwei statt in den letzten Blöcken, vgl. Bild 5-31.

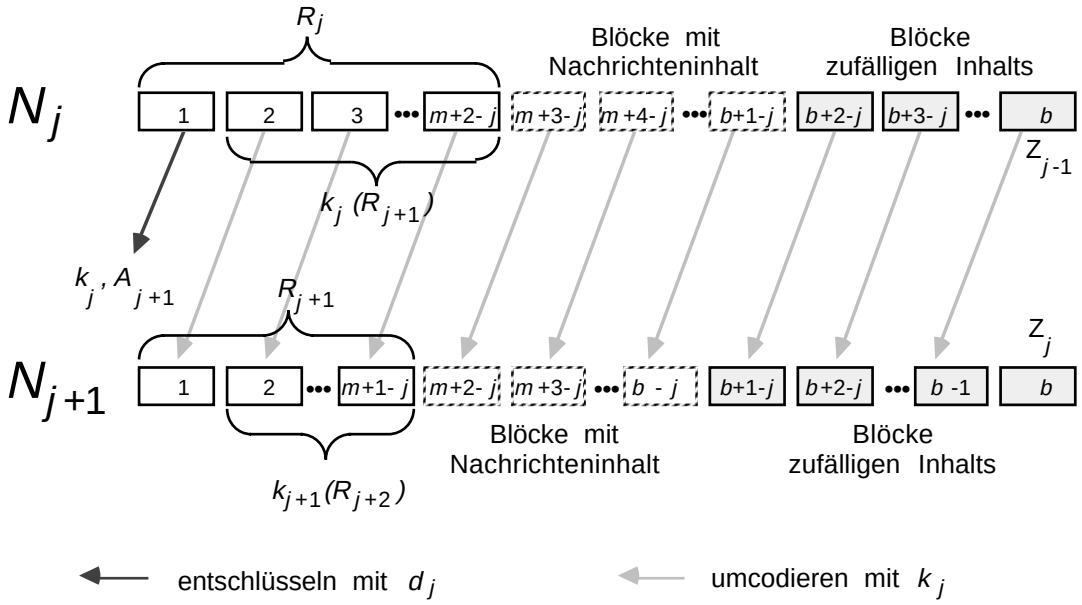


Bild 5-31: Indirektes längentreues Umcodierungsschema für spezielle symmetrische Konzelationssysteme

Wird willkürlich festgelegt, daß jeder MIX alle Blöcke der Nachricht verschlüsselt, so erhält man das einzige längentreue Umcodierungsschema, das in [Chau_81] angegeben ist. Wie bereits erwähnt, wird eine überflüssige Verschlüsselung des angefügten Blocks zufälligen Inhalts vermieden:

$$R_{m+1} = [e]$$
$$R_j = [c_j(k_j,A_{j+1})],k_j^{-1}(R_{j+1}) \quad \text{für } j=m,...,1$$

$$N_1 = R_1,I_1; \quad I_1 = k_0(I) = k_0([I1]),...,k_0([Ii])$$
$$N_j = R_j,I_j; \quad I_j = k_{j-1}(I_{j-1}),[Z_{j-1}] \quad \text{für } j=2,...,m+1$$

Handelt es sich beim Nachrichteninhalt I um Klartext, kann der Empfänger ihn nur verstehen, wenn er die Schlüssel k_0, k_1, ..., k_{m+1} kennt, es sich also mit anderen Worten bei (k_0,A_1,R_1) um eine anonyme Rückadresse handelt und damit das Umcodierungsschema für *Empfängeranonymität* eingesetzt wird.

Soll das Umcodierungsschema für *Senderanonymität* eingesetzt werden, so generiert der Sender die Schlüssel k_1, k_2, ..., k_m und muß den Chiffrierschlüssel c_{m+1} und die Adresse A_{m+1} des Empfängers kennen. Damit der Nachrichteninhalt vom Empfänger verstanden werden kann, muß ihn der Sender vor dem Absenden der Nachricht mit den von ihm generierten Schlüsseln entschlüsseln. Von den obigen Formeln ist lediglich die für I_1 zu modifizieren:

$$I_1 = k_1^{-1}(k_2^{-1}(...k_m^{-1}(c_{m+1}(I_1))...))$$
$$= k_1^{-1}(k_2^{-1}(...k_m^{-1}(c_{m+1}([I1]))...)),...,k_1^{-1}(k_2^{-1}(...k_m^{-1}(c_{m+1}([Ii]))...))$$

5.4.6.6 Effizientes Vermeiden wiederholten Umcodierens

Wie nach den bisherigen drei Beispielen zu vermuten, müssen MIXe bezüglich aller Umcodierungsschemata darauf achten, daß **keine spezielle Umcodierung** (d.h. gleicher umzucodierender Nachrichtenteil, gleiches Konzelationssystem und gleicher, die Umcodierung dann genau spezifizierender Schlüssel) **mehrmals** ausgeführt wird. Anderenfalls könnte ein Angreifer ihnen mehrmals unterschiedliche Nachrichten mit demselben Nachrichtenteil schicken und beobachten, welcher Ausgabe-Nachrichtenteil sich in den entsprechenden MIX-Ausgaben in der entsprechenden Häufig-

keit wiederholt und damit diesen MIX auch bezüglich der den Nachrichtenteil ursprünglich enthaltenden Nachricht überbrücken.

Eine, leider falsche, Idee, die man bezüglich der Lösung dieses ggf. in jedem MIX viel Speicher- und auch Nachrichtenvergleichsaufwand verursachenden Problems haben kann, besteht darin, dem Angreifer den Zugriff auf Nachrichten (und damit auch auf Nachrichtenteile) zwischen MIXen bzw. zwischen Sender und erstem sowie letztem MIX und Empfänger jeweils durch virtuelle Verbindungs-Verschlüsselung zu verwehren. Der Fehler dieser Idee besteht darin, daß so natürlich nur den am Umcodieren dieser Nachricht Unbeteiligten der beschriebene Angriff unmöglich wird, nicht jedoch den Beteiligten, insbesondere den beteiligten MIXen. Der erste und letzte MIX zusammen könnten dann nämlich immer noch die Kommunikationsbeziehung aufdecken, indem sie einen Nachrichtenteil der vom ersten MIX auszusendenden Nachricht mehrmals auf die Reise durch alle „mittleren" (und in diesem Fall völlig unnützen) MIXe schicken.

Somit bleiben nur *vier Möglichkeiten, den Speicher- und Nachrichtenvergleichsaufwand zu verkleinern*: Wie in [Chau_81 Seite 85] beschrieben, können

1. die *öffentlich bekannten Dechiffrierschlüssel der MIXe öfter gegen neue ausgetauscht werden* oder
2. die Teile, deren Umcodierung durch den öffentlichen Dechiffrierschlüssel des betreffenden MIXes spezifiziert ist, einen „*Zeitstempel*" enthalten, der den eindeutigen[112] Zeitpunkt ihrer einzigzulässigen Umcodierung bestimmt.

Letzteres senkt den Aufwand fast auf Null, jedoch ist beides für die Anwendung bezüglich bzw. in anonymen Rückadressen denkbar schlecht geeignet, soll der Sender frei wählen können, wann er antwortet.

Deshalb ist die in [Pfi1_85 Seite 74f] genannte

3. Möglichkeit wichtig, statt der mit den geheimgehaltenen Dechiffrierschlüsseln der MIXe umzucodierenden Nachrichtenteile, die nach der Definition eines asymmetrischen Konzelationssystems mindestens hundert, aus Gründen der Sicherheit der heute bekannten asymmetrischen Konzelationssysteme viele hundert Bit lang sein müssen, das Bild dieser Nachrichtenteile unter einer ihre variable Länge auf eine konstante Länge, z.B. 50 Bit, verkürzende *Hashfunktion* zu speichern.

Diese Hashfunktion muß lediglich ihren Urbildraum in etwa gleichmäßig auf ihren Bildraum abbilden – es darf durchaus mit vertretbarem Aufwand möglich sein, zwei Urbilder zu finden, die auf dasselbe Bild abgebildet werden. Bevor ein MIX seinen geheimgehaltenen Dechiffrierschlüssel auf einen Nachrichtenteil anwendet, bildet er sein Bild unter der Hashfunktion und prüft, ob es schon in seiner Bereits-gemixt-Liste verzeichnet ist. Wenn ja, ignoriert er die Nachricht, wenn nein, wird das Bild in seine Bereits-gemixt-Liste eingetragen und die Entschlüsselung durchgeführt. Der Nachteil dieses Verfahrens ist, daß mit sehr kleiner Wahrscheinlichkeit die Situation auftritt, daß zufällig zwei verschiedene Nachrichtenteile auf dasselbe Bild abgebildet werden und deshalb der später ankommende fälschlicherweise nicht bearbeitet wird. In diesem Fall muß der Sender es mit einer anderen Nachricht, etwa indem er Schlüssel oder zufällige Bitketten ändert, noch mal probieren. Daß er dies kann, erfordert Fehlertoleranz-Maßnahmen, die aber aus anderen Gründen viel dringender und häufiger (kurz: sowieso) gebraucht und in [Pfit_89 §5.3] ausführlich behandelt sind.

Eine 3. sehr ähnliche Möglichkeit wird in einer im Internet verbreiteten MIX-Implementierung (MIXmaster) gewählt:

4. es wird ein bestimmter *Teil* der Nachricht gespeichert (und jeweils verglichen). Dies kann ein Teil der umzucodierenden Nachricht sein oder auch der umcodierten. Bei der MIXmaster-

[112] Es kann auch sinnvoll sein, mit dem Zeitstempel festzulegen, wann die Nachricht *spätestens* umcodiert wird. Dies ermöglicht ein Abwägen zwischen dem Kleinhalten der Bereits-gemixt-Liste und der flexiblen Verwendung von beispielsweise Rückadressen.

Implementierung werden beispielsweise die zufälligen Bitketten (vgl. §5.4.6.2) als dieser Teil gewählt.

Die 4. Möglichkeit spart gegenüber der 3. den Aufwand der Anwendung der Hashfunktion, setzt aber für die Speichereffizienz voraus, daß ein Nachrichtenteil hoher Zufälligkeit (in der Fachsprache: Entropie) gewählt wird. Zufällige Bitketten haben natürlich diese Eigenschaft. Die MIXmaster-Implementierung ist also geschickt gemacht.

Mittels des im folgenden beschriebenen Verfahrens des **anonymen Abrufs** kann die 1. und 2. Möglichkeit zur Verkleinerung des Speicher- und Nachrichtenvergleichsaufwands der MIXe auch bei Schutz des Empfängers angewandt werden: Statt einer anonymen Rückadresse (mit langer Gültigkeit) wird einem Partner nur eine *Kennzahl* mitgeteilt. Der Partner sendet seine mit der Kennzahl adressierte und Ende-zu-Ende-verschlüsselte Antwort unter Verwendung eines Senderanonymitätsschemas an die in diesem Abschnitt schon mehrmals erwähnte, beim anonymen Abruf eine Zwischenspeicherung durchführende, nichtanonyme Instanz *X*. Hin und wieder sendet die Teilnehmerstation des Empfängers unter Verwendung eines Senderanonymitätsschemas eine anonyme Rückadresse an *X*, mit der sie in einer festgelegten Schubfolge die Antwort erhält, sofern diese schon eingetroffen ist. Obwohl beim anonymen Abruf offensichtlich mehr Nachrichten zu senden sind, dürfte der Vorteil, daß jeweils vorher genau bekannt ist, wer welche Umcodierung in welchem Schub durchführen wird, und deshalb sowohl die 1., als auch die 2. Möglichkeit anwendbar sind, den Speicher- und Nachrichtenvergleichsaufwand der MIXe so sehr verkleinern, daß trotzdem insgesamt der Aufwand wesentlich gesenkt wird.

Das Verfahren des anonymen Abrufs ist insbesondere dort relevant, wo wegen schmalbandiger Teilnehmeranschlußleitungen Verteilung zum Schutz des Empfängers nicht anwendbar ist, vgl. §5.5.

5.4.6.7 Kurze Vorausschau

Wird all das bisher Gesagte beachtet, so ist die schlimmste Folge eines **verändernden Angriffs** (oder Fehlers) durch einen MIX der Verlust einer Nachricht, nicht jedoch der Verlust der Anonymität der Kommunikationsbeziehung. Durch öffentlichen Zugriff auf die von MIXen gesendeten Nachrichten oder – auch im Falle von virtueller Verbindungs-Verschlüsselung wirksam – ihr Signieren mit einem MIX-spezifischen Signierschlüssel ist ein Verlust jedoch feststellbar und beweisbar, vgl. §5.4.7. Durch geeignete Fehlertoleranz-Maßnahmen ist zudem die Wahrscheinlichkeit des Verlustes von Nachrichten durch Ausfall eines MIXes (ein ausgefallener MIX innerhalb der MIX-Folge einer Nachricht reicht, um sie zu verlieren!) zu vermindern, wie in dem bereits angekündigten §5.4.6.10 gezeigt wird.

Wegen der Zeitverzögerung durch Warten auf Nachrichten, durch Prüfen auf Wiederholungen und Umsortieren der Nachrichten sowie wegen des Zeitaufwandes für die Entschlüsselungen in einem asymmetrischen Konzelationssystem ist das soweit beschriebene Verfahren der umcodierenden MIXe nicht für Anwendungen geeignet, die kurze Übertragungszeiten fordern.

Wandelt man dieses Verfahren aber wie erstmals in [Pfi1_85 Seite 25-29] beschrieben ab, so kann man **anonyme Kanäle** schalten, die z.B. den Realzeitanforderungen des Telefonverkehrs genügen und in §5.4.6.9 genauer erklärt werden. Hierzu wird zum Kanalaufbau eine spezielle Nachricht nach dem oben beschriebenen Verfahren übertragen, die jedem gewählten MIX einen Schlüssel eines schnelleren symmetrischen Konzelationssystems übergibt, den dieser MIX von da an für die Entschlüsselung des Verkehrs auf diesem Kanal verwendet. Das Umcodieren bei Kanälen nützt natürlich nur dann etwas, wenn mindestens zwei Kanäle von gleicher Kapazität durch denselben MIX *gleichzeitig auf- und abgebaut* werden.

Will man durch umcodierende MIXe nicht nur Kommunikationsbeziehungen, sondern auch das **Senden** und **Empfangen** von Teilnehmerstationen schützen, muß jede Teilnehmerstation ein MIX sein oder sehr viele bedeutungslose Nachrichten senden, wie dies in §5.4.3 beschrieben wurde sowie für die Randbedingung schmalbandiger Teilnehmeranschlußleitungen in §5.5.1 beschrieben wird.

5.4.6.8 Notwendige Eigenschaften des asymmetrischen Konzelationssystems und Brechen der direkten RSA-Implementierung

Als Schluß dieses Abschnitts sei ausdrücklich darauf hingewiesen, daß bei (mittleren) MIXen das verwendete asymmetrische Konzelationssystem nicht nur – wie jedes asymmetrische Konzelationssystem – einem adaptiven aktiven Angriff mit gewähltem Klartext, sondern im wesentlichen (d.h. bis auf die Erschwernis, daß der MIX die zufälligen Bitketten nicht ausgibt) auch einem **aktiven Angriff mit gewähltem Schlüsseltext** (chosen-ciphertext attack) widerstehen muß. Zumindest wenn anonyme Rückadressen verwendet werden, können die Schlüsselpaare der MIXe nicht nach jedem Schub ausgetauscht werden, so daß das verwendete asymmetrische Konzelationssystem dann sogar einem **adaptiven aktiven Angriff mit gewähltem Schlüsseltext** (adaptive chosen ciphertext attack) widerstehen muß.

Es sei an §3 erinnert, wo bereits gesagt wurde, daß es bisher kein *einschrittiges* asymmetrisches Konzelationssystem gibt, das solch einem Angriff allein unter der Annahme der Schwierigkeit eines reinrassigen Standardproblems in beweisbarer Form standhält und damit ein kanonischer Kandidat zur Implementierung (mittlerer) MIXe wäre. Da es *mehrschrittige* asymmetrische Konzelationssysteme gibt, die dies in beweisbarer Form tun, kann man aus ihnen beweisbar sichere MIX-Implementierungen gewinnen. Sie sind wegen der Mehrschrittigkeit zwischen Sender (bzw. dem die Umcodierung Spezifizierenden) und *jedem* durchlaufenen MIX allerdings sehr aufwendig: Sollen die (mittleren) MIXe MIX_1 bis MIX_n verwendet werden, so muß der Schlüsselaustauschdialog zwischen Sender und MIX_i, $i>1$, jeweils durch alle MIX_j mit $1 \leq j < i$ geschehen [Bött_89 §3.3.1]. Dies macht den Hauptvorteil von MIXen gegenüber den anderen Verfahren zum Schutz der Verkehrs- und Interessensdaten zunichte, nämlich die vergleichsweise geringe benötigte Bandbreite zwischen Teilnehmerstation und Kommunikationsnetz. Im Rest dieser Arbeit wird deshalb jeweils nur der Fall explizit behandelt, daß zur Implementierung von (mittleren) MIXen ein einschrittiges asymmetrisches Konzelationssystem verwendet wird.

In §3.6.3 wurde gezeigt, daß RSA ohne geeignetes, vom Entschlüsseler geprüftes Redundanzprädikat bereits einem nicht-adaptiven aktiven Angriff mit gewähltem Klartext nicht widerstehen kann. Im folgenden wird gezeigt, daß die Verwendung von **RSA ohne ein zusätzliches Redundanzprädikat und mit zusammenhängenden zufälligen Bitketten**, d.h. so wie von David Chaum in [Chau_81 Seite 84] beschrieben, **vollkommen unsicher ist**. Dies wird, um die Notation einfach zu halten, anhand der in Bild 5-24 verwendeten gezeigt.

Der Angreifer beobachte eine Nachricht $c(z,N)$ am Eingang des MIXes. Der Angreifer wählt sich einen geeigneten Faktor f und reicht dem MIX die Nachricht $c(z,N) \bullet c(f)$ zum Mixen im selben bzw. einem späteren Schub ein. Nachdem in §3.6 die perfekte Unverkettbarkeit von $c(z,N)$ und $c(z,N) \bullet c(f)$ bei Wahl beliebiger Faktoren f gezeigt wurde, gilt „praktisch“ perfekte Unverkettbarkeit, wenn nur „viele“ Faktoren möglich sind, d.h., der MIX hat dann keine Chance, einen aktiven Angriff zu erkennen. Es bleibt also nur zu zeigen, daß der Angreifer durch seinen Angriff etwas erhält, das es ihm erlaubt, N aus den Ausgabe-Nachrichten des entsprechenden Schubes herauszufinden. Intuitiv und näherungsweise richtig kann dies in der bisherigen allgemeinen Notation folgendermaßen dargestellt werden: Der MIX bildet intern $d(c(z,N) \bullet c(f)) = (z,N) \bullet f$. Kann der Angreifer f so wählen, daß an der Stelle der auszugebenden Nachricht $N \bullet f$ entsteht, was in der bisherigen Notation $(z,N) \bullet f = (?, N \bullet f)$ lautet, ist sein Angriff gelungen: Er muß nur noch prüfen, welche beiden Nachrichten des

bzw. der beiden Schübe die Gleichung $X = Y \cdot f$ erfüllen. Was an der Stelle der zufälligen Bitkette entsteht ist irrelevant, da dies vom MIX weder geprüft noch ausgegeben wird.

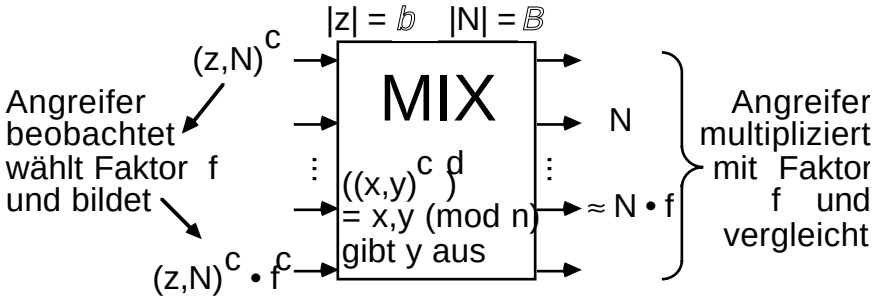


Bild 5-32: Brechen der direkten RSA-Implementierung von MIXen

Warum dieser Angriff gelingt, ist ohne die Verwendung weiterer Details von RSA nicht zu erklären. Die für das Verständnis notwendigen lauten, daß bei RSA Ver- und Entschlüsselung durch Exponentiation innerhalb eines Restklassenrings erfolgen. Der den Restklassenring charakterisierende Modulus $n = p \cdot q$, wobei p und q zufällig gewählte Primzahlen von – aus Sicherheitsgründen – je mindestens 350 Bit Länge sind, bildet mit dem zur Verschlüsselung verwendeten Exponenten den öffentlich bekannten Chiffrierschlüssel. In jedem Block dieser Blockchiffre können $\mathbb{b}$ Bit lange zufällige Bitketten und $\mathbb{B}$ Bit lange Nachrichten untergebracht werden, sofern nur $2^{\mathbb{b}} \cdot 2^{\mathbb{B}} < n$ ist. Bei der Implementierungsbeschreibung von David Chaum, wie bei jeder effizienten Implementierung, ist $\mathbb{b}$ sehr viel kleiner als $\mathbb{B}$. Werden, wie bei David Chaum und in der obigen Notation angedeutet die zufälligen Bitketten an den Bitstellen höherer Wertigkeit untergebracht, so gilt $(z,N) = z \cdot 2^{\mathbb{B}} + N$ und $(z,N) \cdot f \equiv (z \cdot 2^{\mathbb{B}} + N) \cdot f \equiv z \cdot 2^{\mathbb{B}} \cdot f + N \cdot f$.

Aus der die Werte der Bezeichner z' und N' definierenden[113] Kongruenz $(z,N) \cdot f \equiv z' \cdot 2^{\mathbb{B}} + N'$ folgt $z \cdot 2^{\mathbb{B}} \cdot f + N \cdot f \equiv z' \cdot 2^{\mathbb{B}} + N'$, daraus folgt $2^{\mathbb{B}} \cdot (z \cdot f - z') \equiv N' - N \cdot f$ und daraus folgt $z \cdot f - z' \equiv (N' - N \cdot f) \cdot (2^{\mathbb{B}})^{-1}$. Wählt der Angreifer $f \le 2^{\mathbb{b}}$, so gilt $-2^{\mathbb{b}} < z \cdot f - z' < 2^{2\mathbb{b}}$. Der Angreifer setzt nun in die Formel $z \cdot f - z' \equiv (N' - N \cdot f) \cdot (2^{\mathbb{B}})^{-1}$ für N und N' jeweils alle Ausgabe-Nachrichten des bzw. der betroffenen Schübe ein und prüft, ob $z \cdot f - z'$ im entsprechenden Intervall liegt. Dies ist, da $\mathbb{b}$ sehr viel kleiner als $\mathbb{B}$ ist, mit sehr hoher Wahrscheinlichkeit nur für genau ein Paar der Ausgabe-Nachrichten der Fall. Die erste Nachricht N dieses Paares ist die Nachricht, bezüglich der der MIX mittels des aktiven Angriffs überbrückt werden soll. Die zweite Nachricht N' ist die der vom Angreifer gebildeten Eingabe-Nachricht entsprechende Ausgabe-Nachricht.

Erfüllen zwei oder mehr Paare diese Bedingung, so wiederholt der Angreifer seinen Angriff mit einem anderen Faktor und führt bei seinem zweiten sowie ggf. allen weiteren Versuchen das paarweise Einsetzen nur noch mit den durch die bisherigen Tests nicht ausgeschlossenen Nachrichten des zuerst erwähnten Schubes durch. Dieser Angriff auf die Kommunikationsbeziehung einer Nachricht N ist mit in der bei N verwendeten Schubgröße quadratischem Aufwand durchzuführen: Der Angreifer muß bei jedem Versuch nur einmal mit RSA verschlüsseln (sowie das Ergebnis mit N multiplizieren und modulo n reduzieren) sowie für jede Nachricht des Schubes eine Addition, zwei Multiplikationen, zwei Reduktionen modulo n und zwei Vergleiche durchführen. In [PfPf_90 Seite 376] ist beschrieben, wie der Aufwand dieses Angriffs auf O(Schubgröße • ld(Schubgröße)) reduziert werden kann.

---

113 Es wird angenommen, daß auf beiden Seiten der definierenden Kongruenz jeweils die kleinsten nichtnegativen Repräsentanten der Restklasse stehen, so daß die Kongruenz in Wirklichkeit eine Gleichung ist, die zusammen mit den sich aus der Länge von z' und N' ergebenden Ungleichungen die Werte von z' und N' eindeutig bestimmt.

Die bisher unterstellte Annahme, daß $\mathbb{b}$ sehr viel kleiner als $\mathbb{B}$ ist, ist nicht nötig. Ein entsprechender Angriff ist bereits immer dann möglich, wenn $\mathbb{B}$ so groß ist, daß eine Wahl von $f \leq 2^{\mathbb{B}-1}$ für den zu überbrückenden MIX „praktische“ Unverkettbarkeit von (z,N) und $(z,N)\bullet f$ ergibt. Die resultierende Ungleichung $-2^{\mathbb{b}} < z\bullet f - z' < 2^{\mathbb{b}+\mathbb{B}-1}$ genügt bereits, damit der Angreifer in jeder Iteration seines Angriffs einen von der Iterationszahl unabhängigen Anteil der noch möglichen Nachrichten des ursprünglichen Schubes ausscheiden kann. Der Aufwand des Angriffs steigt damit lediglich um einen logarithmischen Faktor.

Ist eine adaptive Wiederholung des Angriffs nicht möglich, etwa weil der MIX nach jedem Schub sein Schlüsselpaar wechselt, kann der Angreifer dieselbe Information wie durch seinen iterativen Angriff dadurch erhalten, daß er gleich mehrere Faktoren wählt, entsprechend mehrere Nachrichten bildet und alle diese zusammen mit der Nachricht, bezüglich der er den MIX überbrücken will, zum Mixen im selben Schub einreicht.

Es wurde bisher nicht geprüft, ob mit den in jüngerer Vergangenheit veröffentlichten Angriffen auf RSA eine Senkung des Angriffsaufwands möglich ist.

Alles bisher am Beispiel eines direkten Umcodierungsschemas für Senderanonymität Gezeigte gilt natürlich auch für jedes indirekte Umcodierungsschema, indem für den Angriff statt ganzer Nachrichten nur (Rück-)Adreßteile verwendet werden. Dies ist bei allen in [Chau_81] angegebenen Umcodierungsschemata möglich [PfPf_90 §3.2]. Dies kann jedoch durch Verwendung geeigneter Umcodierungsschemata, d.h. solcher, die keine direkt sondern nur indirekt umcodierte Teile in beobachtbarer Weise verwenden, vermieden werden. Hierbei sollte das zur indirekten Umcodierung verwendete (symmetrische) Konzelationssystem völlig anders als RSA arbeiten.

Werden die zufälligen Bitketten an den Bitstellen niedrigerer Wertigkeit untergebracht, so existiert ein analoger und ebenfalls erfolgreicher Angriff [PfPf_90 §3.2].

Ein analoger Angriff versagt, wenn statt einer zusammenhängenden zufälligen Bitkette in nicht zu großen Abständen (möglichst also äquidistantem Abstand) genügend viele einzelne zufällige Bits in die Nachricht „eingestreut“ und vom MIX entsprechend „herausgesiebt“ werden. Allerdings sind schwächere Angriffe möglich, wenn entweder die zufälligen Bitketten kürzer als die mitverschlüsselten Nachrichten sind oder auch nur ein größerer Block von Nachrichtenbits nicht von zufälligen Bits unterbrochen ist [PfPf_90 §3.2]. Hierbei können 10 Bits bereits als größerer Block gelten.

Allerdings kann selbst der schwächere Angriff leicht dadurch verhindert werden, daß die eingestreute zufällige Bitkette und die mitverschlüsselte Nachricht durch **Verschlüsselung in einem völlig anderen (symmetrischen) Konzelationssystem** (z.B. DES mit einem öffentlich bekannten Schlüssel) vollständig „gemischt“ werden, bevor mit RSA verschlüsselt wird.

Ein alternativer Ansatz ist, daß die Klartexte ein geeignetes **Redundanzprädikat** erfüllen müssen, vgl. §3.6.4.1. Erfüllt der vom MIX entschlüsselte Klartext z,N das Redundanzprädikat nicht, gibt der MIX keinen Teil dieser Nachricht aus. Um ausgiebiges Probieren zu verhindern, falls man sich der kryptographischen Stärke des Redundanzprädikates nicht sicher ist, können die Sender falsch gebildeter Nachrichten durch Zusammenarbeit aller MIXe identifiziert werden: Jeder MIX_i gibt zu der rückzuverfolgenden Nachricht die im Falle der Verwendung eines deterministischen asymmetrischen Konzelationssystems explizit mitverschlüsselte zufällige Bitkette z_i an (bzw. bei Verwendung eines indeterministisch verschlüsselnden asymmetrischen Konzelationssystems die hierbei verwendete. Kann MIX_i dies nicht, ist das indeterministisch verschlüsselnde asymmetrische Konzelationssystem für diesen Zweck ungeeignet). Die Sender können zur Rechenschaft gezogen werden, sofern alle Nachrichtenübertragungen öffentlich sind oder alle Nachrichten vom jeweiligen Sender (Teilnehmerstation oder MIX) authentifiziert werden. Bei der Rückverfolgung von Nachrichten ist zu beachten, daß bei Verwendung von anonymen Rückadressen nicht der Verwender, sondern nur der Generierer zur Rechenschaft gezogen wird. Der Verwender muß dabei anonym

bleiben können, da ansonsten Kommunikationspartner einander durch Zusenden falsch gebildeter Rückadressen identifizieren könnten.

5.4.6.9 Schnellere Übermittlung durch MIX-Kanäle

Das in den bisherigen Abschnitten erläuterte MIX-Netz ist im wesentlichen zur unbeobachtbaren Übermittlung einzelner Nachrichten gedacht. Das MIX-Netz ist prinzipiell aber auch für Kanäle geeignet.

Im folgenden wird gezeigt, wie für Kanäle die Verzögerung durch die MIX-Operation verringert und die Verteilung aller Nachrichten (d.h. Kanäle) vermieden werden kann. Die sich so ergebenden MIX-Kanäle sind zwar unter den Randbedingungen des schmalbandigen ISDN nicht direkt einsetzbar, bilden jedoch das Grundverfahren der durchaus einsetzbaren Zeitscheibenkanäle (vgl. §5.5.1).

In gewisser Weise realisiert bereits die hybride Verschlüsselung Simplexkanäle:

Der Hauptteil jeder Nachricht N_i ist lediglich mit einer symmetrischen *Stromchiffre* verschlüsselt. Da geeignete Stromchiffren Ver- und Entschlüsselungsgeschwindigkeiten im Mbit/s-Bereich haben und jedes Bit sofort zu ver- bzw. entschlüsseln erlauben, wird die Übermittlung dieses Hauptteiles praktisch nicht verzögert.

Der Anfang jeder Nachricht N_i, $i > 1$, wird jedoch merklich verzögert: Der erste Block der Nachricht ist asymmetrisch verschlüsselt. Zur Entschlüsselung muß meist der Empfang des gesamten ersten Blocks abgewartet werden, und die eigentliche Entschlüsselung ist bei Verwendung heutiger asymmetrischer Konzelationssysteme vergleichsweise langsam. Hinzu kommt der Aufwand zur Umsortierung und für den Test auf Wiederholung von Nachrichtenanfängen.

Daher wird der langsame Kanalaufbau von der eigentlichen Nachrichtenübermittlung getrennt [Pfi1_85 §2.1.2] und durch eigenständige Signalisierungsnachrichten realisiert.

Hierdurch wird eine Aufteilung der verfügbaren Bandbreite des Teilnehmeranschlusses in beiden Richtungen in jeweils einen Signalisierungsanteil für den Kanalaufbau und einen Datenanteil für die eigentliche Nachrichtenübermittlung vorgenommen. Die Art der Aufteilung ist für das folgende Verfahren irrelevant, doch wird in diesem Abschnitt der Einfachheit halber die im ISDN verwendete statische Aufteilung in einen Signalisierungskanal und mehrere Datenkanäle übernommen.

Zum Kanalaufbau wird über den Signalisierungskanal eine MIX-Eingabenachricht N_1, genannt **Kanalaufbaunachricht**, gesendet, die den MIXen lediglich die für den symmetrischen Teil der hybriden Verschlüsselung benötigten Schlüssel k_i, $i > 1$, mitteilt. Folglich wird sie von M_m nicht verteilt. Der Einheitlichkeit halber sei mit k_1 im folgenden der Schlüssel k_{1A} abgekürzt, den der Sender A mit M_1 fest vereinbart hat.

Jeder MIX merkt sich die Zuordnung der Kanalaufbaunachricht vom Eingabe- zum Ausgabeschub. Jeder Kanalaufbaunachricht ist nun entsprechend ihrer Position im Eingabeschub von M_i ein Eingabekanal von M_i zugeordnet. Die Zuordnung der Kanalaufbaunachrichten vom Eingabe- zum Ausgabeschub von M_i definiert zugleich auch eine Vermittlung der Eingabekanäle von M_i auf die Ausgabekanäle von M_i, z.B. bei Frequenzmultiplex eine Zuordnung von Eingabe- zu Ausgabefrequenzen. Damit ist durch das Mixen der Kanalaufbaunachricht ein Kanal durch alle MIXe definiert, auf dem nun die eigentliche Nachricht übermittelt werden könnte.

Die Verteilung aller Kanäle an alle Netzabschlüsse erscheint wenig sinnvoll, da davon ausgegangen wird, daß jeder Netzabschluß nur einige wenige, im ISDN meist zwei Kanäle empfangen kann. Das Problem der Unbeobachtbarkeit des Empfängers vor dem Sender wird daher auf eine etwas andere Art gelöst, ähnlich den anonymen Rückadressen in [Chau_81].

Das Problem wird zunächst umgangen, indem der Empfänger dem Sender gleichgestellt wird: ein Kanal kann als **Sendekanal** oder als **Empfangskanal** genutzt werden.

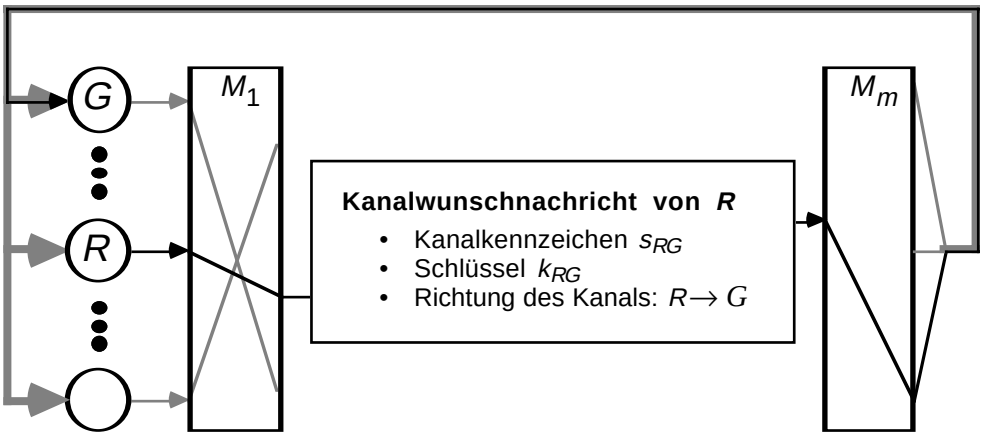
Ein **Sendekanal** ist das genaue (d.h. oben suggerierte) Analogon zur hybriden Verschlüsselung: der Sender baut den Kanal auf, verschlüsselt den Informationsstrom N fortlaufend als $k_1(k_2(\ldots k_m(N)\ldots))$ und überträgt ihn an M_1. M_1 entschlüsselt fortlaufend mit k_1 und gibt den Nachrichtenstrom auf demselben Wege wie die zugehörige Kanalaufbaunachricht an M_2 weiter. Alle MIXe verfahren ebenso, d.h. M_m bildet am Ende den Klartextstrom N.

Ein **Empfangskanal** ist ein „rückwärts" betriebener Sendekanal: Der *Empfänger* baut den Kanal auf (mit derselben Art von Kanalaufbaunachricht). M_m erhält vom *Sender* den nicht speziell für die MIXe codierten (aber natürlich Ende-zu-Ende-verschlüsselten) Informationsstrom N, verschlüsselt ihn mit k_m und leitet $k_m(N)$ „zurück" an M_{m-1}. Alle MIXe verfahren ebenso, d.h. M_1 gibt am Ende den verschlüsselten Strom $k_1(\ldots k_m(N)\ldots)$ aus. Da der Empfänger alle Schlüssel k_i kennt, kann er hieraus N bestimmen.

Sende- und Empfangskanäle sind hinsichtlich ihrer Unbeobachtbarkeit völlig *symmetrisch*: Ebenso wie der Sender gefahrlos direkt seinen Sendekanal nutzen kann, ohne dadurch vom Empfänger identifiziert werden zu können, ist der Empfänger eines Empfangskanals vor Identifizierung durch den Sender geschützt. Hingegen ist der Empfänger eines Sendekanals durch den Sender und der Sender eines Empfangskanals durch den Empfänger beobachtbar.

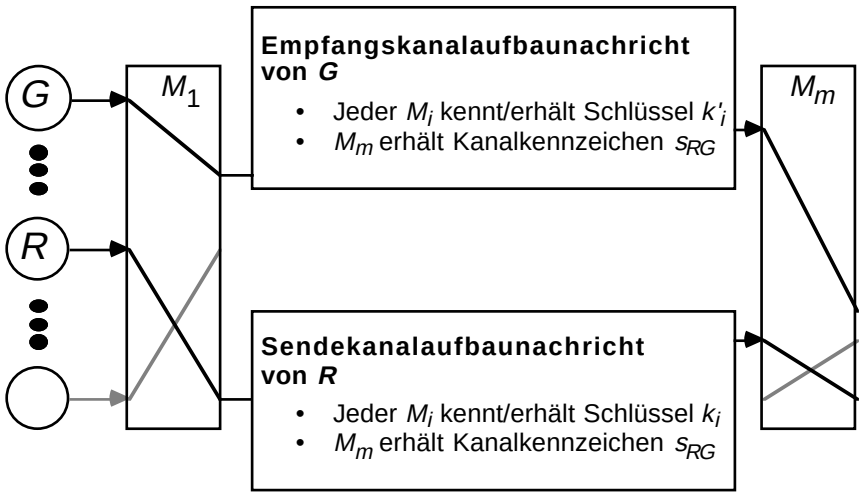
Um nun beide Vorteile zu vereinen und beide Nachteile auszuschließen, werden die eigentlichen **MIX-Kanäle** als *Verknüpfungen* von Sende- und Empfangskanälen gebildet (Bilder 5-33, 5-34, 5-35): Der Sender baut einen Sendekanal auf, der bei M_m endet, und der Empfänger einen Empfangskanal, der bei M_m anfängt. M_m leitet den Informationsstrom, der auf dem Sendekanal eintrifft, auf den Empfangskanal um.

Die zu verknüpfenden Kanäle werden durch ein gemeinsames **Kanalkennzeichen** bezeichnet, das M_m übereinstimmend in beiden Kanalaufbaunachrichten erhält. Es muß natürlich beiden Partnern bekannt sein, und da man kaum von statischen und außerhalb des Netzes geknüpften Beziehungen ausgehen kann, muß es vom rufenden Partner R gewählt und mit dem üblichen Schema dem gerufenen Partner G in einer **Kanalwunschnachricht** mitgeteilt werden (sinnvollerweise zusammen mit einem Schlüssel zur Ende-zu-Ende-Verschlüsselung und der Angabe, ob R zu senden oder zu empfangen gedenkt). In Bild 5-33 wurde willkürlich R als Sender und G als Empfänger des MIX-Kanals angenommen. Es ist natürlich möglich, die Kanalwunschnachricht und die Kanalaufbaunachricht des Rufenden zu einer einzigen Signalisierungsnachricht zusammenzufassen.



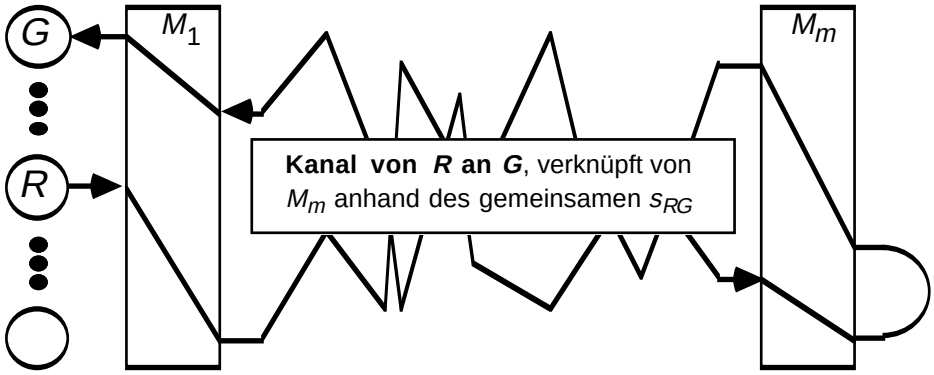
Der Netzabschluß *R* des rufenden Teilnehmers sendet über den Signalisierungskanal an den Netzabschluß *G* des gerufenen Teilnehmers eine Kanalwunschnachricht. Die Nachricht wird wie in Bild 5-24 dargestellt gemixt, der Ausgabeschub von M_m wird an alle Netzabschlüsse verteilt. *R* beabsichtigt zu senden.

Bild 5-33: Verarbeitung der von *R* gesendeten Kanalwunschnachricht



Beide Netzabschlüsse bauen nun einen Kanal auf. Die Kanalaufbaunachrichten werden über den Signalisierungskanal übertragen und wie in Bild 5-24 dargestellt gemixt, aber nicht verteilt. Die MIXe merken sich die Zuordnung der Kanalaufbaunachrichten von Eingabe- zu Ausgabeschüben, M_m verknüpft anhand des gemeinsamen Kanalkennzeichens s_{RG} den Sendekanal von *R* mit dem Empfangskanal von *G*.

Bild 5-34: Verarbeiten der Kanalaufbaunachrichten



R sendet auf seinem Datenkanal Daten *N* verschlüsselt als $k_1(\ldots k_m(N)\ldots)$ an M_1. Der Kanal wird von M_1 über die weiteren MIXe bis zu M_m wie die Sendekanalaufbaunachricht in Bild 5-34 weitergeleitet und dabei werden die Daten entschlüsselt. M_m verschlüsselt *N* mit k'_m. Der Kanal wird zurückgeleitet und dabei werden die Daten verschlüsselt, *G* erhält über seinen Datenkanal $k'_1(\ldots k'_m(N)\ldots)$.

Bild 5-35: Verknüpfen der Kanäle

Kanalwunschnachrichten, Sende- und Empfangskanalaufbaunachrichten stellen die drei Typen von Signalisierungsnachrichten des Verfahrens dar. Die Übermittlung aller Signalisierungsnachrichten erfolgt nach dem Grundschema aus §5.4.6.2, außer daß die Kanalwunschnachrichten verteilt werden.

Da die Größe der Eingabeschübe den Aufwand der einzelnen MIXe entscheidend beeinflußt, ist es sinnvoll, die Nachrichten der drei verschiedenen Typen getrennt zu mixen. Die Unbeobachtbarkeit wird dadurch nicht eingeschränkt.

Zur Verkleinerung des Suchraums bzgl. Wiederholungen müssen entweder die Zeitstempel der MIX-Eingabenachrichten sowohl den Schub als auch den Nachrichtentyp eindeutig charakterisieren, oder aber die MIXe müssen für die verschiedenen Nachrichtentypen verschiedene Schlüssel zur Umcodierung verwenden. Die Angabe des Nachrichtentyps verursacht für die Netzabschlüsse zwar weniger Synchronisationsaufwand, andererseits wird aber ohnehin von einem streng synchronen Betrieb ausgegangen, und der Signalisierungskanal stellt den Hauptengpaß des Verfahrens dar. Daher wird im folgenden stets von der zweiten Möglichkeit ausgegangen.

Wie im Grundschema wird das Problem der Beobachtbarkeit des Sendens auf einem Sendekanal gelöst, indem jeder Netzabschluß jederzeit gleichviele Sendekanäle (notfalls Scheinsendekanäle) unterhält.

Ohne Verteilung ist nun natürlich das tatsächliche Empfangen von einem Empfangskanal beobachtbar. Daher muß jeder Netzabschluß jederzeit auch gleichviele Empfangskanäle (notfalls Scheinempfangskanäle) unterhalten.

Die gleiche Länge aller Kanäle wird durch die MIXe garantiert: wird auf einem Sende- oder Empfangskanal nichts übertragen, so überbrückt jeder MIX M_i dieses Fehlen durch bedeutungslose Daten.

Hierdurch wird insbesondere verhindert, daß der Sender den Empfänger identifiziert, indem er auf dem Sendekanal keine Information überträgt und der Empfänger nichts erhält. Dies ist der Grund, weshalb hier, anders als in [Pfi1_85], der Empfänger selbst seinen Empfangskanal aufbaut.

Zugleich wird verhindert, daß Scheinempfangskanäle durch Fehlen von Daten auffallen.

Ein **Duplexkanal** wird durch zwei gegenlaufende Simplexkanäle eingerichtet. Beide Kanäle könnten vom Rufenden mit derselben Kanalaufbaunachricht aufgebaut werden. Dies spart zwar Übertragungskapazität, wird sich aber im folgenden als hinderlich herausstellen, so daß schon jetzt von zwei völlig

getrennten MIX-Kanälen ausgegangen werden soll. Allerdings genügt eine gemeinsame Kanalwunschnachricht.

Im §5.4.1.3 wurde allgemein beschrieben, wie ein verändernder Angriff eines Senders auf den zugehörigen Empfänger verhindert werden kann, in [PfPW1_89 §2.2.3] geschieht dies speziell für die hier behandelte Situation. Da aber der in [PfPW1_89 §2.2.4.2] beschriebene verändernde Angriff eines Empfängers auf den zugehörigen Sender nicht verhindert werden kann und auch für MIX-Kanäle möglich ist, könnte man argumentieren, daß diese Maßnahme für Duplexkanäle völlig sinnlos ist, da der Angreifer seinen Partner ja wahlweise als Sender oder als Empfänger angreifen kann.

Aus praktischen Gründen ist die Verhinderung des Angriffs auf den Empfänger aber dennoch sinnvoll, da er sehr viel einfacher durchzuführen wäre als der Angriff auf den Sender: Statt, wie in [PfPW1_89 §2.2.4.2] beschrieben, Senden aktiv zu stören, muß der Angreifer lediglich alle Empfangskanäle beobachten können. Zudem gelingt der Angriff auf den Empfänger sofort, während der Angriff auf den Sender lediglich einen Test darstellt, der dann logarithmisch in der Senderanzahl oft wiederholt werden muß, um den Sender wirklich zu finden.

Da für die eigentlichen MIX-Kanäle die üblicherweise aufwendigsten MIX-Aufgaben, nämlich die Sortierung von Nachrichten und die Kontrolle auf Wiederholungen, entfallen, d.h. lediglich eine einfache Umcodierung und Vermittlung von Eingangs- auf Ausgangskanäle erfolgt, kann der Aufwand der MIXe zur Verwaltung der MIX-Kanäle vernachlässigt werden. Die Umcodierung der Daten mit der symmetrischen Stromchiffre fällt, da sie mit einer sehr viel höheren Rate erfolgen kann als die Übertragung (vgl. §5.5.2), kaum ins Gewicht.

In dieser Form könnte das Verfahren jedoch unter den Randbedingungen des schmalbandigen ISDN noch nicht direkt eingesetzt werden. Neben der üblichen Überlegung, daß das Verfahren nicht homogen für alle Netzabschlüsse auf einmal, sondern nur innerhalb gewisser Anonymitätsgruppen[114] durchgeführt werden könnte, gibt es nämlich noch ein schwerwiegendes Problem. Wie für die Nachrichten in §5.4.6.1 gilt auch für MIX-Kanäle:

- Jeder Netzabschluß muß für jeden Kanalwunscheingabeschub von M_1 gleich viele Kanalwunschnachrichten und Sende- und Empfangskanalaufbaunachrichten beisteuern, und entsprechend gleich viele Sende- und Empfangskanäle unterhalten.
- Nachrichten eines Schubes müssen gleich lang sein, was hier insbesondere heißt, daß gleichzeitig aufgebaute Kanäle auch *gleichzeitig* wieder abgebaut werden müssen.

Dies würde vor allem bedeuten, daß jeder Netzabschluß mit dem Abbau eines Kanals, sogar eines Scheinkanals, warten müßte, bis der längste gleichzeitig aufgebaute Kanal beendet wäre. Da dem Teilnehmer mit Basisanschluß im schmalbandigen ISDN nur zwei 64-kbit/s-Kanäle zur Verfügung stehen, wäre dies unzumutbar.

Diese Einschränkung wird durch das Verfahren in §5.5.1 beseitigt.

5.4.6.10 Fehlertoleranz beim MIX-Netz

Im MIX-Netz ohne Verteilung „letzter“ (vgl. §5.4.6.3) Informationseinheiten sind andere Fehlerbehebungs-Verfahren als Ende-zu-Ende-Zeitschranken und nochmalige Übermittlung bei Überschreitung der Zeitschranken ganz besonders nötig, da dieses Fehlerbehebungs-Verfahren bei manchen Diensten nicht zufriedenstellend funktioniert: Bei elektronischer Post (electronic mail) etwa gibt es keine sinnvollen Zeitschranken. Außerdem muß bei anonymen Rückadressen der ursprüngliche Generierer

---

114 „Gruppe“ ist hierbei umgangssprachlich gemeint, d.h. im Sinne von eine „Menge von Instanzen, die eine gemeinsame Eigenschaft verbindet“. Insbesondere hat hier „Gruppe“ nichts mit der mathematischen Struktur „Gruppe“ (abgeschlossene transitive Verknüpfung, Existenz eines neutralen und jeweils eines inversen Elementes) zu tun. Im mathematischen Sinne müßte man also von „Anonymitätsmengen“ statt „Anonymitätsgruppen“ sprechen.

eine neue anonyme Rückadresse als Ersatz einer durch Ausfall eines MIXes unbrauchbar gewordenen bilden – der Verwender der Rückadresse kann dies nicht (vgl. §5.4.6.3). Dieses Problem kann allerdings durch das Verfahren des anonymen Abrufs statt „normaler“ anonymer Rückadressen vermieden werden, vgl. §5.4.6.6.

Für die hiermit motivierten und in den folgenden Unterabschnitten hergeleiteten Fehlertoleranzverfahren wird als einheitliche und angemessene graphische Notation die von Bild 5-25 verwendet. Dargestellt wird jeweils die Übermittlung einer Nachricht (als Beispiel einer von anderen unabhängigen Informationseinheit) vom Sender *S* über 5 MIXe zum Empfänger *E*, die dabei verwendeten Schlüssel und ihre Kenntnis, sowie die Reihenfolge von Verschlüsselungs-, Transfer- und Entschlüsselungsoperationen. Wie in den Bildern 5-24 und 5-25 wird das einfachste Verschlüsselungsschema (direktes Umcodierungsschema für Senderanonymität) abgebildet.

Da keine Informationseinheiten eingezeichnet sind, kann man Bild 5-25 auch als Darstellung des (abstrakten) Verschlüsselungs-, Transfer- und Entschlüsselungsschemas verstehen. Ebenso kann, da keine genauen Verschlüsselungsstrukturen gezeigt sind, Bild 5-25 auch als Abbildung eines indirekten Umcodierungsschemas – sei es für Sender-, sei es für Empfängeranonymität oder beides – verstanden werden, das nur die wesentlichen, nämlich nachrichtenunabhängigen Schlüssel sowie Ver- und Entschlüsselungen zeigt.

Man erkennt an Bild 5-25 noch deutlicher die Serieneigenschaft als an Bild 5-24. Fällt nur ein MIX auf dem Weg zwischen *S* und *E* aus, so kann die Nachricht nicht weiter entschlüsselt und übertragen werden.

Wie in §5.4.9 begründet wird und aus Bild 5-40 ersichtlich ist, basiert das MIX-Netz auf einem in den Schichten 0 (medium), 1 (physical) und 2 (data link) sowie der unteren Teilschicht der Schicht 3 (network) ohne Rücksicht auf Anonymitäts- und Unverkettbarkeitsforderungen realisierbaren Kommunikationsnetz. Wie schon begründet, impliziert dies, daß diese Schichten des Kommunikationsnetzes konventionelle Fehlertoleranz-Maßnahmen verwenden können, ohne dadurch die Anonymität bzw. Unbeobachtbarkeit der Netzteilnehmer zu gefährden. Transiente sowie permanente Fehler in diesen Schichten können also von diesen selbst toleriert werden.

Die MIXe selbst müssen und können so gebaut oder mittels organisatorischer Maßnahmen betrieben werden, daß sich Fehler (oder verändernde physische Angriffe) mit an Sicherheit grenzender Wahrscheinlichkeit nicht so auswirken, daß der MIX seinen geheimen Dechiffrierschlüssel ausgibt, Informationseinheiten mehrfach mixt oder in falscher Reihenfolge ausgibt. Hingegen ist es akzeptabel, daß er bei schwerwiegenden Fehlern (oder verändernden physischen Angriffen) seine Schlüssel „vergißt“ und seinen Dienst vollständig einstellt (fail-stop Betrieb [ScSc_83]). Um möglichst selten Fehler extern tolerieren zu müssen, ist es natürlich möglich, den MIX intern fehlertolerant aufzubauen, solange bei diesem Aufbau obige Bedingungen auch bei Fehlern oder verändernden Angriffen eingehalten werden.

Extern wahrnehmbare transiente Fehler in den MIXen werden von den Ende-zu-Ende-Protokollen zwischen Sender *S* und Empfänger *E* erkannt und durch wiederholtes Senden toleriert. (Zur Leistungssteigerung ist auch eine weitere Zwischenstufe mit MIX-zu-MIX-Protokollen möglich.) Es bleiben somit nur noch die extern wahrnehmbaren permanenten Fehler in MIXen. Diese Fehler sind mit den üblichen Techniken zu erkennen bzw. zu lokalisieren, z.B. der Ausfall eines MIXes durch Zeitschranken bzw. die Diagnose, welcher MIX ausfiel, durch das Ausbleiben eines zyklisch zu gebenden Lebenssignals (Nachricht mit Datum und Zeit sowie Signatur, I’m alive message). Die permanenten Ausfälle von MIXen erfordern darüber hinaus geeignete Methoden zur Fehlerbehebung. Prinzipiell gibt es drei Möglichkeiten:

Die erste, in [Pfi1_85 §4.4.6.10.1] beschriebene, erfordert keine Koordination von MIXen. Sie stellt aber ein Ende-zu-Ende-Protokoll dar, was – wie erwähnt – für die Unverkettbarkeit und damit auch die Anonymität bzw. Unbeobachtbarkeit ungünstig ist.

Im Gegensatz hierzu ist Koordination zwischen MIXen bei den beiden anderen nötig und eine nicht Ende-zu-Ende arbeitende Fehlerbehebung deshalb möglich, da bei ihnen MIXe in die Lage versetzt werden, andere zu ersetzen. Wie dies geschehen kann und was dabei zu beachten ist, wird in [Pfi1_85 §4.4.6.10.2] behandelt.

Besonderheiten beim Schalten von Kanälen sind für alle drei Möglichkeiten gemeinsam in [Pfit_89 §5.3.3] behandelt. In [Pfit_89 §5.3.4] wird eine quantitative Bewertung aller drei Möglichkeiten durchgeführt.

5.4.6.10.1 Verschiedene MIX-Folgen

Bei der ersten Möglichkeit versucht der Sender eine Wiederholung der Übermittlung auf einem anderen Weg (Bild 5-36), d.h. über eine disjunkte MIX-Folge (in der Begriffswelt der Fehlertoleranz [EcGM_83, gekürzt auch in BeEG_86]: *dynamisch aktivierte Redundanz*). Diese Lösung ist einfach, aber (wie dynamisch aktivierte Redundanz generell) langsam, da das Ende-zu-Ende-Protokoll zuerst den Fehler erkennen (i. allg. durch Ende-zu-Ende-Zeitschranken) und dann eine zweite Übermittlung einleiten muß, möglicherweise ohne zu wissen, welcher MIX defekt ist.

Im MIX-Netz ohne Verteilung „letzter“ Informationseinheiten und ohne anonymen Abruf sind gemäß §5.4.6.2 anonyme Rückadressen mit langer Gültigkeit zum Schutz des Empfängers nötig. Bei anonymen Rückadressen mit langer Gültigkeit ist obiges Fehlertoleranzverfahren dann sehr aufwendig, wenn keine zeitliche Beziehung zwischen Nachrichten und Antworten besteht, wie dies z.B. bei elektronischer Post (electronic mail) der Fall ist. Der Empfänger *E* kann, falls die erhaltene Rückadresse ausfällt (wenigstens ein MIX in dieser Folge ist defekt), keine Übermittlung auf einem anderen Weg versuchen. Dies gilt auch, wenn immer zwei oder mehr anonyme Rückadressen mit langer Gültigkeit (*statisch erzeugte Redundanz*) ausgetauscht werden und in jeder Folge ein MIX defekt ist. Der ursprüngliche Sender *S* (genauer: seine Teilnehmerstation) müßte also, solange er von *E* keine Antwort erhielt, *E* (genauer: seiner Teilnehmerstation) immer wieder neue anonyme Rückadressen mitteilen, was in den meisten Fällen völlig überflüssig ist, da *E* lediglich bisher noch keine Zeit fand, eine Antwort zu formulieren. Alternativ kann sich *S* auch immer wieder informieren, welche MIXe inzwischen ausgefallen sind[115], und *E* nur dann neue anonyme Rückadressen mit langer Gültigkeit mitteilen, wenn alle *E* bisher mitgeteilten anonymen Rückadressen mit langer Gültigkeit ausgefallen sind oder waren, denn auch im letzteren Fall können sie – nach einem Fehlversuch von *E* – für diesen inzwischen permanent unbrauchbar sein.

Diese erste Möglichkeit der Fehlertoleranz läuft auf ein alle Stationen involvierendes **Zwei-Phasen-Konzept** hinaus. In der einen Phase werden Informationseinheiten anonym übertragen, in der anderen werden Fehler toleriert, z.B. dadurch daß nach jedem Ausfall eines MIXes alle Sender allen Empfängern neue anonyme Rückadressen zukommen lassen, in denen der ausgefallene MIX nicht benötigt wird (in der Begriffswelt der Fehlertoleranz: *dynamisch erzeugte*, dynamisch aktivierte *Redundanz*). Da im MIX-Netz ohne Verteilung „letzter“ Informationseinheiten an alle Stationen und ohne anonymen Abruf aus Gründen der gegenseitigen Anonymität von Sender und Empfänger alle Adressen anonyme Rückadressen sein müssen (§5.4.6.4), erfordert diese Neuverteilung in ihm zwingend eine Indirektionsstufe in Form von nichtanonymen Stellen zur Neuverteilung der Adressen (etwa die bereits in §5.4.6.4 erwähnten Adreßverzeichnisse mit anonymen Rückadressen): Nach Ausfall eines MIXes wird dies allen Stationen mitgeteilt und jede sendet den nichtanonymen Stellen zur Adreßverteilung mit einem Senderanonymitätsschema über die noch intakten MIXe eine Liste der

---

115 Auch hier ist wieder gleichartiges Verhalten aller Teilnehmerstationen und konsistente Verteilung der Information, welche MIXe zur Zeit ausgefallen sind, wichtig. Andernfalls sind Angriffe auf die Anonymität des Rückadreßinhabers möglich. Sie können analog zu dem in §5.4.1.1 2) beschriebenen Angriff auf die Empfängeranonymität erfolgen.

unbrauchbar gewordenen anonymen Adressen, jeweils eine anonyme Ersatzadresse und eventuell noch weitere anonyme Adressen, da diese ja je nur einmal verwendet werden können und also von Zeit zu Zeit sowieso nachgeliefert werden müssen.

Im MIX-Netz mit Verteilung „letzter“ Informationseinheiten an alle Stationen sind anonyme Rückadressen überflüssig, da durch Verteilung und normale implizite Adressen der Empfänger vollständig geschützt ist. Also müssen in solch einem MIX-Netz auch nach Ausfall von MIXen keine neuen Adressen ausgetauscht werden. Eine Liste der ausgefallenen MIXe sollte natürlich an alle Stationen verteilt werden, damit sie diese MIXe für das Senderanonymitätsschema nicht verwenden.

Entsprechendes gilt beim Verfahren des anonymen Abrufs.

Eine im Fehlerfall Zeit sparende Variation dieser Möglichkeit der Ende-zu-Ende-Protokolle mit statisch oder dynamisch aktivierter Redundanz besteht darin, jeden anonymen Informationstransfer parallel über mehrere disjunkte MIX-Folgen auszuführen (statisch oder dynamisch erzeugte, *statisch aktivierte Redundanz*).

Nachteil jeder Ende-zu-Ende-Fehlerbehebung ist, daß statistische Angriffe über Senderaten von Informationseinheiten – wie zu Beginn dieses Kapitels bereits erwähnt – möglich sind, die ggf. eine Verkettung von Verkehrsereignissen bewirken können. Besonders ausgeprägt ist dies bei MIX-Netzen ohne Verteilung „letzter“ Informationseinheiten und bei individueller Wahl der Ende-zu-Ende-Zeitschranken (genauer: der Zeitschrankenintervalle, innerhalb derer zu einem zufälligen Zeitpunkt reagiert wird) bzw. bei individueller Wahl der Anzahl der parallel auszuführenden Informationstransfers.

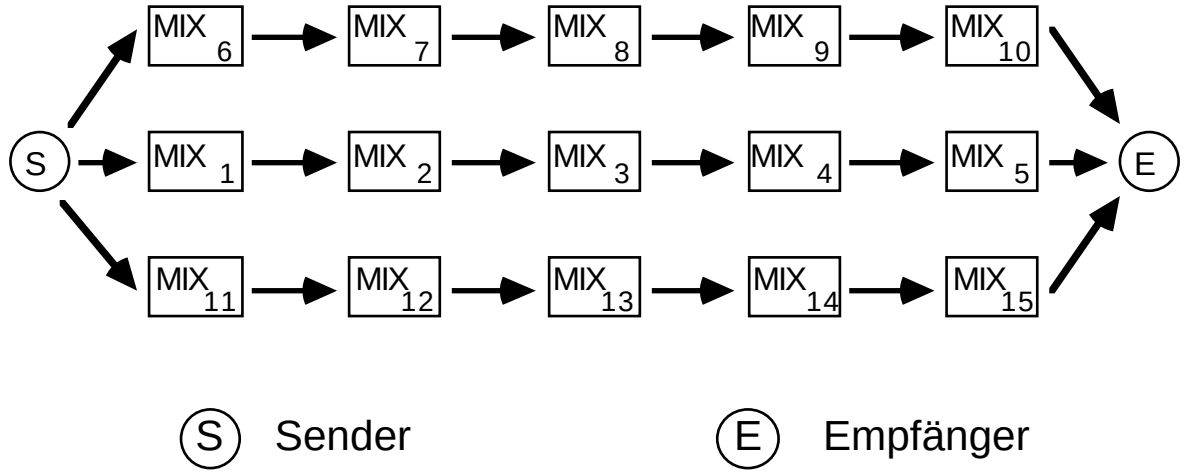


Bild 5-36: Zwei zusätzliche alternative Wege über disjunkte MIX-Folgen

Können statistische Angriffe über Senderaten von Informationseinheiten ignoriert werden, so gilt das folgende Schutzkriterium.

> **Schutzkriterium**:
> Verschiedene MIX-Folgen sind genauso sicher wie die ursprünglichen Schemata (vgl. §5.4.6), sofern bei jeder verwendeten MIX-Folge mindestens ein MIX nicht vom Angreifer kontrolliert wird.
> Hierbei ist es unerheblich, ob der nicht kontrollierte MIX korrekt mixt oder (etwa, sofern er ausgefallen ist) gar nicht.

5.4.6.10.2 Ersetzen von MIXen

Um eine nicht Ende-zu-Ende arbeitende Fehlerbehebung zu ermöglichen und damit alle Nachteile der ersten Möglichkeit zu vermeiden, werden bei der zweiten und dritten Möglichkeit MIXe in die Lage versetzt, andere zu ersetzen. Dies ist insbesondere für die Verfügbarkeit sehr vorteilhaft, verursacht aber das im folgenden Unterabschnitt behandelte *Koordinations-Problem*.

Die Möglichkeit des Ersetzens von MIXen wird statisch vorgesehen (*statisch erzeugte Redundanz*) – bei dynamisch erzeugter Redundanz bräuchten keine MIXe ersetzt zu werden. Es könnte, wie in §5.4.6.10.1, eine ganz andere MIX-Folge gewählt werden.

Statisch erzeugte Redundanz kann generell entweder statisch oder dynamisch aktiviert werden.

Dabei ist eine statische Aktivierung bezüglich des Ersetzens eines MIXes nichts anderes als eine verteilte, intern fehlertolerante Implementierung eines MIXes (vgl. §5.4.6.10): An den vorangehenden MIX (bzw. das unterliegende Kommunikationsnetz) wird die Funktion des Sendens der Eingabe an alle verteilten Teile des MIXes delegiert. An den nachfolgenden MIX wird die Funktion des Sammelns der Ausgaben aller verteilten Teile und die Bildung des Gesamtergebnisses delegiert. Nicht delegierbar ist eine Koordinierung aller verteilten Teile des MIXes (vgl. auch §5.4.6.10.2.1). Die Koordinierung muß sicherstellen, daß alle nicht ausgefallenen Teile in jedem Schub (vgl. §5.4.6) jeweils die gleichen Eingabe-Informationseinheiten bearbeiten. Anderenfalls kann ein verändernder Angreifer Differenzen zwischen den Eingabe- und Ausgabe-Informationseinheiten einzelner Teile des MIXes herbeiführen und zur Überbrückung des verteilt implementierten MIXes bezüglich des Schutzes der Kommunikationsbeziehung nutzen. Dies gelingt genau dann, wenn Durchschnitte oder Differenzen der Schübe die Kardinalität 1 haben, vgl. §5.4.6.

Eine statische Aktivierung bezüglich des Ersetzens mehrerer MIXe in Folge ist bezüglich des Schutzes der Kommunikationsbeziehung problematisch, da

- das bereits mehrfach erwähnte Koordinations-Problem dann jeweils für die MIXe in Folge bezüglich jeder Stufe besteht und
- eine Informationseinheit nur unter solchen Informationseinheiten bezüglich der Kommunikationsbeziehung geschützt werden kann, die jeweils gleichzeitig dieselben MIXe in Folge durchlaufen, d.h. sowohl jeweils dieselben MIXe hintereinander als auch gleichviele Folgen parallel.

Eine statische Aktivierung bezüglich des Ersetzens mehrerer MIXe in Folge ist also nur bei fest vorgegebenen MIX-Kaskaden (vgl. [Pfit_89 §4.2.3.1]) möglich.

Da beide Möglichkeiten statisch erzeugter, statisch aktivierter Redundanz einerseits sehr aufwendig und andererseits bezüglich des Entwurfsspielraums uninteressant zu sein scheinen, wird in §5.4.6.10.2.1 bis §5.4.6.10.2.3 ausschließlich statisch erzeugte, *dynamisch aktivierte* Redundanz behandelt.

5.4.6.10.2.1 Das Koordinations-Problem

Im MIX-Netz, wie es in §5.4.6 beschrieben wurde, war jeder MIX dafür verantwortlich, solange er sein Schlüsselpaar beibehält, Informationseinheiten mit ihm höchstens einmal zu mixen. Anderenfalls könnte ein Angreifer einen MIX bezüglich einer Informationseinheit, die er zweimal mixt, überbrücken, da sie mit hoher Wahrscheinlichkeit die einzige in den entsprechenden Ausgaben des MIXes zweimal enthaltene Informationseinheit ist.

Können MIXe in die Lage versetzt werden, andere zu ersetzen, so wird diese Verantwortung jedes *einzelnen* MIXes auf ein von ihm und denjenigen MIXen, die ihn möglicherweise ersetzen können, gebildetes *Team* übertragen. Dabei muß das Team dieser Verantwortung auch dann gerecht werden, wenn eine beliebige Teilmenge des Teams ausgefallen und die Kommunikation zwischen nicht

ausgefallenen Teammitgliedern unterbrochen ist. Insbesondere dürfen Informationseinheiten, die von ausgefallenen oder gerade unerreichbaren Teammitgliedern bereits gemixt wurden, nicht nochmal von einem Teammitglied gemixt werden – obwohl ein Angreifer versuchen wird, genau das zu erreichen.

Diese Aufgabe kann von einem einfachen Protokoll zwischen den Teammitgliedern gelöst werden. Allerdings ist der Aufwand dieses Protokolls, nämlich die durch seine Ausführung verursachte zusätzliche Verzögerungszeit der eigentlichen Nachrichten sowie die zusätzliche Kommunikation zwischen sowie der zusätzliche Speicherplatz in MIXen, erheblich.

> **Ineffizientes Koordinations-Protokoll** [Pfi1_85 Seite 74]:
> Bevor ein MIX Informationseinheiten mixt, schickt er alle seine Eingabe-Informationseinheiten an alle Teammitglieder, die nicht ausgefallen sind.
> Der MIX mixt eine Informationseinheit nur, nachdem ihm von allen nicht ausgefallenen Teammitgliedern bestätigt wurde, daß sie diese Eingabe-Informationseinheit noch nicht gemixt haben und auch nicht mixen werden.
> Ein ausgefallener MIX bekommt von den anderen Teammitgliedern alle Informationseinheiten, die gemixt wurden oder gemixt werden sollen, bevor er selbst wieder zu mixen beginnt.

Um dieses Koordinations-Protokoll ausführen zu können, muß jeder MIX des Teams alle Informationseinheiten, die von anderen Teammitgliedern oder ihm selbst gemixt wurden (bzw. gemixt werden wollten), speichern. Um den dazu benötigten Speicherplatz erträglich zu halten, können die in §5.4.6.6 beschriebenen vier Möglichkeiten zur Verkleinerung des Speicher- und Suchaufwands (öffentlich bekannte Dechiffrierschlüssel der MIXe öfter tauschen; Zeitstempel; verkürzende Hashfunktion; nur bestimmten Teil der Nachricht speichern und vergleichen) verwendet werden.

Wird die Methode der verkürzenden Hashfunktion oder die Methode von Speicherung und Vergleich nur eines bestimmten Teils der Nachricht bereits vom Anfrager und nicht erst von den Antwortern (im obigen Koordinations-Protokoll) angewandt, so reduziert diese Möglichkeit nicht nur den Speicher- und Suchaufwand, sondern auch den Kommunikationsaufwand.

Andere Methoden zur Verringerung des Kommunikationsaufwandes und der Verzögerungszeit der eigentlichen Nachrichten sind spezifisch für die zweite und dritte Möglichkeit und werden deshalb in §5.4.6.10.2.2 und §5.4.6.10.2.3 behandelt.

Das obige ineffiziente Koordinations-Protokoll setzt voraus, daß MIXe sicher wissen, welche MIXe des Teams ausgefallen bzw. nicht ausgefallen sind. Anderenfalls könnte ein Angreifer zwei Gruppen voneinander zu isolieren und zu überzeugen versuchen, daß die MIXe der anderen Gruppe ausgefallen sind. Ist ihm das gelungen, so ist ein erfolgreicher Angriff leicht. Um dies auch dann zu verhindern, wenn MIXe nicht sicher wissen, welche MIXe des Teams ausgefallen bzw. nicht ausgefallen sind, kann obiges Koordinations-Protokoll um die Regel erweitert werden, daß

> nur dann gemixt wird, wenn eine absolute Mehrheit funktioniert (und miteinander kommunizieren kann).

Dies kann mit den üblichen Authentifikationstechniken garantiert werden, vgl. §3. Hier – wie auch bei den folgenden effizienten Koordinations-Protokollen – wird der anfragende MIX bei der Bestimmung der absoluten Mehrheit immer als aktiv zustimmend reagierend, d.h. positiv mitgezählt.

5.4.6.10.2.2 MIXe mit Reserve-MIXen

Die zweite Möglichkeit (der drei am Ende von §5.4.6.10 angekündigten) besteht darin, daß der geheime Schlüssel des ausgefallenen MIXes einem oder mehreren anderen MIXen, die z.B. von derselben Organisation betrieben werden, bekannt ist (Bild 5-37) oder von vielen MIXen, die z.B.

von sich gegenseitig mißtrauenden Organisationen betrieben werden, durch Verwendung eines Schwellwertschemas (vgl. §3.9.6) rekonstruiert werden kann [Pfi1_85].

Beides ermöglicht ohne Änderungen der Adreß- oder Verschlüsselungsschemata ein MIX-zu-MIX-Protokoll, indem der jeweilige Sender oder MIX bei Ausfall eines (oder mehrerer) MIXe die Informationseinheit zu einem nicht ausgefallenen Mitglied des entsprechenden Teams übermittelt. Dazu benötigt er Information über Ausfälle von MIXen sowie die Teamstruktur. Beides kann entweder global bekannt sein oder auf Anfrage mitgeteilt werden – der Ausfall eines MIXes etwa durch Ausbleiben einer Antwort (Empfangsquittung). Auf die ebenfalls nötige Information über die Erreichbarkeit von MIXen, die durch Ausfälle von Teilen des unterliegenden Kommunikationsnetzes eingeschränkt sein kann, wird aus den zu Beginn dieses Kapitels dargelegten Gründen nicht weiter eingegangen.

In der Begriffswelt der Fehlertoleranz ausgedrückt handelt es sich also um *statisch erzeugte, dynamisch aktivierte Parallel-Redundanz.*

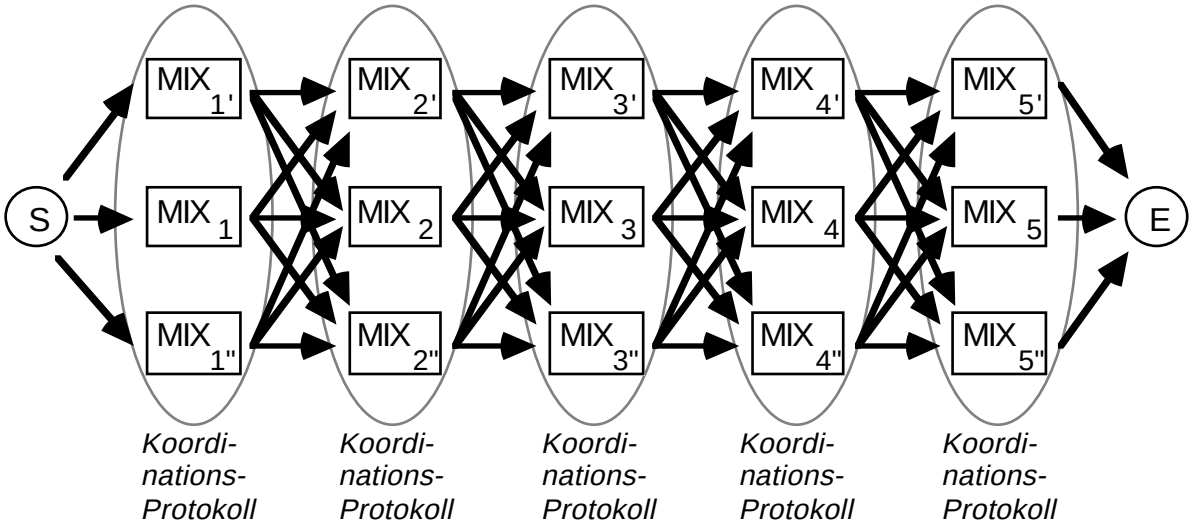


Bild 5-37: MIX_i kann alternativ von MIX_i' oder MIX_i'' ersetzt werden (i = 1, 2, 3, 4, 5)

Wie angekündigt, kann der Aufwand des Koordinations-Protokolls zur Verhinderung mehrfachen Mixens derselben Informationseinheit drastisch reduziert werden:

> **Effizientes Koordinations-Protokoll für MIXe mit Reserve-MIXen** [Pfi1_85 Seite 75f]:
> Jegliche zusätzliche Verzögerung des Mixens kann vermieden werden, wenn jeder MIX mit seinen Reserve-MIXen vereinbart, daß sie nur dann mit seinem Schlüssel mixen, wenn er ausgefallen ist.
> Es genügt dann, daß er seine Eingabe-Informationseinheiten an eine deutlich größere als absolute Mehrheit aller Teammitglieder spätestens dann abschickt, wenn er die zugehörigen Ausgabe-Informationseinheiten ausgibt. Hört er auf zu mixen, wenn er innerhalb eines vorgegebenen Zeitraums keine authentifizierten Empfangsbestätigungen einer absoluten Mehrheit aller anderen Teammitglieder erhält, so kann ein Angreifer ihn mittels Isolierung höchstens bezüglich der Informationseinheiten überbrücken, für die er noch keine Empfangsbestätigung erhielt.
> Reserve-MIXe, deren Ausfall ein MIX überprüfen kann (etwa mittels authentifizierter Kommunikation mit dessen Kommunikationsprozessor), brauchen bei der Ermittlung einer (absoluten) Mehrheit nicht gezählt zu werden.

Ist ein MIX ausgefallen, so können die Reserve-MIXe einen von ihnen als seinen *Vertreter* bestimmen (und etwa zur Rekonstruktion des MIX-Schlüssels befähigen, falls ein Schwellwertschema, vgl. §3.9.6, verwendet wurde). Der Vertreter verhält sich genauso wie der MIX, sofern er nicht ausgefallen wäre, bis er entweder selbst auch ausfällt (und die übrigen Reserve-MIXe einen neuen Vertreter bestimmen) oder der vertretene MIX repariert ist und synchronisiert mit dem Ende des stellvertretenden Mixens seines Vertreters mit seinem Mixen beginnt.

Meiner Meinung nach wird das Risiko, daß ein Angreifer einen MIX bezüglich der Informationseinheiten, für die er noch keine Empfangsbestätigungen erhielt, mittels Isolierung überbrücken kann, bei weitem durch die Vermeidung jeder zusätzlichen Verzögerung des Mixens aufgewogen. Denn verändernde Angriffe, die diese Schwäche des Koordinations-Protokolls auszunutzen versuchen, können leicht entdeckt werden (zumindest wenn sie oft erfolgen) und müssen an sehr vielen Stellen in aufeinander abgestimmter Weise eingreifen, wenn Informationseinheiten jeweils mehrere MIXe durchlaufen: Um den Weg einer bestimmten Informationseinheit von ihrem Sender zu ihrem Empfänger zu verfolgen, muß ein Angreifer in jedem Team, von dem jeweils ein Mitglied die Informationseinheit mixen muß, den gerade aktiven MIX (kontrollieren oder) zum Ausfall bringen oder isolieren, nachdem er diese Informationseinheit mixte.

Ein MIX sollte solche MIXe als seine Reserve-MIXe wählen, die physisch zumindest einige zehn Kilometer auseinanderliegen, damit sich so wenig Fehler (oder verändernde physische Angriffe) wie möglich auf mehrere auswirken [Pfi1_85 Seite 76].

Betreibt eine Organisation mehrere MIXe an räumlich weit entfernten Stellen, so scheint es sehr vernünftig zu sein, daß diese MIXe füreinander als Reserve-MIXe fungieren.

Schutzkriterium:
Wenn die MIXe geeignet koordiniert sind, so sind MIXe mit Reserve-MIXen genauso sicher wie die ursprünglichen Schemata (vgl. §5.4.6), sofern bei einem der in der Verschlüsselungsstruktur verwendeten MIXe sein Team nicht vom Angreifer kontrolliert wird.
Ein Team wird zumindest dann nicht vom Angreifer kontrolliert, wenn er kein Teammitglied kontrolliert oder aber bei Verwendung eines Schwellwertschemas sowohl nicht den eigentlichen MIX, als auch keinen Vertreter, als auch weniger als zur Rekonstruktion des Schlüssels nötige MIXe des Teams kontrolliert (und damit nie den Schlüssel erhält) als auch keine absolute Mehrheit der MIXe des Teams kontrolliert (sonst könnte diese Mehrheit auch ohne Ausfall des gerade mixenden MIXes einen Stellvertreter etablieren lassen und beide parallel und unkoordiniert mixen lassen).

5.4.6.10.2.3 Auslassen von MIXen

Die dritte Möglichkeit besteht darin, daß jeder MIX in jeder Nachricht genügend Information erhält, um einen MIX (oder im allgemeinen Fall: bis zu *ü* MIXe mit einer beliebigen natürlichen Zahl *ü*) überbrücken und auslassen zu können [Pfi1_85 Seite 77ff].

Im Gegensatz zur zweiten benötigt die dritte Möglichkeit Änderungen der Adreß- oder Verschlüsselungsschemata, um ebenfalls ein MIX-zu-MIX-Protokoll zu ermöglichen. Diese geänderten Adreß- und Verschlüsselungsschemata werden hier teilweise hergeleitet.

Bezüglich weiterer Varianten sei auf [Pfit_89] verwiesen. In [Pfit_89 §5.3.2.3.2] werden Schutzkriterien zur Charakterisierung der mit diesen Adreß- und Verschlüsselungsschemata bei geeigneter Koordinierung erreichbaren Anonymität der Kommunikationsbeziehung aufgestellt. In [Pfit_89 §5.3.2.3.3 und §5.3.2.3.4] werden passende und möglichst effiziente Koordinations-Protokolle für die zwei möglichen Betriebsarten entwickelt: Entweder werden nur ausgefallene, d.h. möglichst

wenig, MIXe ausgelassen, oder aber möglichst viele. Letzteres verkompliziert das Koordinations-Problem, erlaubt aber in manchen Situationen einen Effizienzgewinn.

Um das Verständnis zu erleichtern, wird zuerst der Fall beschrieben, daß jeder MIX den nächsten MIX in einer Folge gewählter MIXe auslassen kann. In diesem Fall kann dies Fehlertoleranzverfahren den Ausfall eines oder sogar mehrerer nicht aneinandergrenzender MIXe tolerieren.

Um einen MIX auslassen zu können, muß sein Vorgänger nicht nur die für ihn bestimmte Nachricht bilden können, sondern auch die für seinen Nachfolger bestimmte (Bild 5-38).

Wenn jeder MIX die Nachrichten sowohl für seinen Nachfolger als auch für dessen Nachfolger separat erhält, wächst die Länge der Nachrichten exponentiell mit der Zahl der benutzten MIXe. Um dies zu vermeiden, wählt der Sender einer Nachricht (wie bei allen indirekten Umcodierungsschemata, vgl. §5.4.6) für jeden MIX einen unterschiedlichen Schlüssel (beispielsweise eines effizienten symmetrischen Konzelationssystems), mit dem dieser MIX die für seinen Nachfolger bestimmte Nachricht entschlüsselt. Jeder MIX erhält diesen und den für seinen Nachfolger bestimmten Schlüssel mit seinem öffentlich bekannten Dechiffrierschlüssel verschlüsselt. Separat und mit den öffentlich nicht bekannten Schlüsseln entsprechend verschlüsselt erhält er den Rest der Nachricht. Die Adressen der nächsten beiden MIXe können jeder Nachricht unverschlüsselt vorangestellt oder mit den beiden erwähnten Schlüsseln zusammen mit dem öffentlich bekannten Dechiffrierschlüssel des MIXes verschlüsselt werden. Im folgenden wird dies formaler beschrieben.

kennt $c_1,c_2,c_3,$ c_4,c_5,c_E	kennt $c_1,c_2,c_3,$ $c_4,c_5,c_E,$ d_1	kennt $c_1,c_2,c_3,$ $c_4,c_5,c_E,$ d_2	kennt $c_1,c_2,c_3,$ $c_4,c_5,c_E,$ d_3	kennt $c_1,c_2,c_3,$ $c_4,c_5,c_E,$ d_4	kennt $c_1,c_2,c_3,$ $c_4,c_5,c_E,$ d_5	kennt $c_1,c_2,c_3,$ $c_4,c_5,c_E,$ d_E
bildet zufällig $k_5,k_4,k_3,$ k_2,k_1	erfährt k_1,k_2	erfährt k_2,k_3	erfährt k_3,k_4	erfährt k_4,k_5	erfährt k_5	

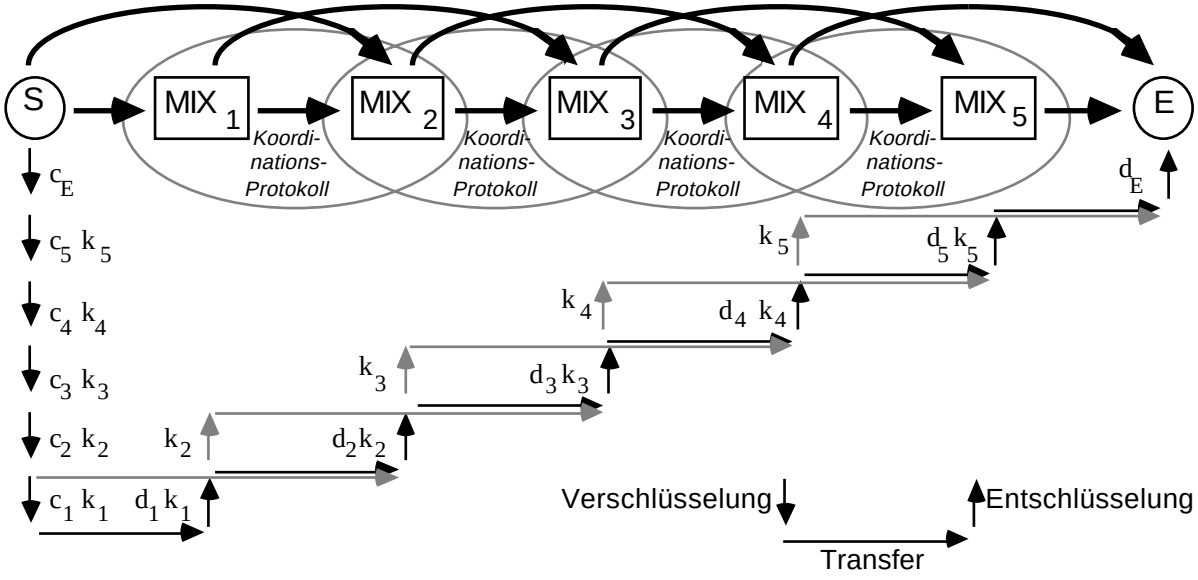


Bild 5-38: Jeweils ein MIX kann ausgelassen werden; Die Koordinations-Protokolle werden im Bild zwischen Gruppen minimalen Umfangs abgewickelt.

Sei wie in §5.4.6 $A_1,...,A_n$ die Folge der Adressen und $c_1,...,c_n$ die Folge der öffentlich bekannten Chiffrierschlüssel der gewählten MIX-Folge $MIX_1,...,MIX_n$, wobei c_1 auch ein geheimer Schlüssel eines symmetrischen Konzelationssystems sein kann. Sei A_{n+1} die Adresse des Empfängers, der zur Vereinfachung der Notation MIX_{n+1} genannt wird, und c_{n+1} sein Chiffrierschlüssel. Sei $k_1,...,k_n$ die gewählte Folge nichtöffentlicher Schlüssel (beispielsweise eines symmetrischen Konzelationssystems). Der Sender bildet die Nachrichten N_i, die MIX_i erhalten wird, gemäß dem folgenden rekursivem Bildungsschema ausgehend von der Nachricht N, die der Empfänger (MIX_{n+1}) erhalten soll:

$$\begin{aligned} N_{n+1} &= c_{n+1}(N) && \text{(Schema 1)} \\ N_n &= c_n(k_n),k_n(A_{n+1},N_{n+1}) \\ N_i &= c_i(k_i,A_{i+1},k_{i+1}),k_i(A_{i+1},N_{i+1}) \qquad \text{für } i = 1,...,n\text{-}1 \end{aligned}$$

Dies Schema ist natürlich nicht das einziggeeignete. Beispielsweise erfüllt das folgende Schema denselben Zweck:

$$\begin{aligned} N_{n+1} &= c_{n+1}(N) && \text{(Schema 2)} \\ N_n &= c_n(k_n,A_{n+1}),k_n(N_{n+1}) \\ N_i &= c_i(k_i,A_{i+1},k_{i+1},A_{i+2}),k_i(N_{i+1}) \qquad \text{für } i = 1,...,n\text{-}1 \end{aligned}$$

Der Sender sendet jeweils N_1 an MIX_1.

In beiden Schemata kann MIX_i die Nachrichten N_{i+1} und N_{i+2} sowie die Adressen A_{i+1} und A_{i+2} aus N_i berechnen.

Bei einigen im folgenden diskutierten Koordinations-Protokollen ist es auch wichtig, daß MIX_i überprüfen kann, daß sein Vorgänger MIX_{i-1} kein Bit der Nachricht N_i geändert hat (und entsprechend auch sonst niemand, da MIX_{i-1} am meisten über die Nachricht erfuhr und deswegen am zielgerichtetsten handeln konnte).

Solange die Konzelationssysteme nicht gebrochen sind[116], kann ein Angreifer nur Nachrichtenteile gezielt verändern, zu denen er die verwendeten Schlüssel kennt. Anderenfalls geht die Nachricht einfach verloren oder wird von einer Station, bei der die Nachricht nach ihrer Entschlüsselung mittels fehlererkennender Codes als fehlerhaft erkannt wird, einfach ignoriert. In beiden Fällen gewinnt der Angreifer dadurch keine für ihn nützliche Information. Also hat MIX_{i-1} die Möglichkeiten, den ganzen mit c_i verschlüsselten Teil zu ersetzen (ohne den Inhalt des ursprünglichen lesen zu können, da er zwar c_i, nicht aber d_i kennt) oder nur an dem mit k_i verschlüsselten Teil zu manipulieren.

Die Gefahr hierbei ist, daß er versuchen kann, die Adressen zu ändern. Dies ist für den Schutz der Kommunikationsbeziehung nicht kritisch, solange Adressen nur für die Vermittlung von Nachrichten verwendet werden, da sowieso unterstellt ist, daß der Angreifer alle Leitungen kontrolliert. Es kann aber dann kritisch werden, wenn die Nachrichten festlegen, welche MIXe sich miteinander koordinieren müssen, damit keine Nachricht zweimal gemixt wird.

In beiden Schemata müßte der Angreifer den mit c_i verschlüsselten Teil ändern, um eine Adresse zu ändern. Um dies zu tun, muß er ein neues k_i erfinden. Dann wird dies aber nach der Entschlüsselung des jeweils mit k_i verschlüsselten Teiles, nämlich $k_{i+1}(A_{i+2},N_{i+2})$ in Schema 1 oder $k_{i+1}(N_{i+2})$ in Schema 2 erkannt. Dazu müssen diese Teile jeweils genug Redundanz enthalten, damit MIX_i dies bereits feststellen kann.

---

116 Hier wird, um die Umcodierungsschemata bei ihrer Erklärung möglichst einfach zu halten, den verwendeten Konzelationssystemen mehr unterstellt, als sie nach ihrer Definition leisten müssen, vgl. §3 und Aufgabe 3-7 d). Leisten die Konzelationssysteme dies nicht bereits von sich aus, so müssen sie (oder alternativ die Umcodierungsschemata) geeignet erweitert werden. Dies kann etwa durch Redundanzbildung mittels einer kollisionsresistenten Hashfunktion geschehen, vgl. §3.6.4.1.

Wird ein Koordinations-Protokoll verwendet, bei dem diese Überprüfung nicht nötig ist, kann in Schema 1 das erste A_{i+1} weggelassen werden.

Die Wahl zwischen Schema 1 und Schema 2, d.h. ob A_{i+2} im ersten, mit einem aufwendigen asymmetrischen Konzelationssystem verschlüsselten Teil oder im zweiten, mit einem weniger aufwendigen symmetrischen Konzelationssystem verschlüsselten Teil (unter dem Namen A_{i+1} für $i := i+1$) enthalten ist, hängt vom verwendeten asymmetrischen Konzelationssystem ab: Muß etwa aus Gründen der Sicherheit des asymmetrischen Konzelationssystems die Länge der mit ihm verschlüsselten Informationseinheiten so groß sein (wie dies etwa für RSA, vgl. §3.6 der Fall ist), daß A_{i+2} auf jeden Fall mit hineinpaßt, ist es natürlich sinnvoller, Schema 2 zu verwenden, anstatt in Schema 1 mit zufälligen Bitketten aufzufüllen. Denn es ist trivialerweise weniger aufwendig, A_{i+2} mit dem aufwendigeren asymmetrischen Konzelationssystem ohne zusätzlichen Aufwand mitverschlüsseln zu lassen als A_{i+2} (unter dem Namen A_{i+1} für $i := i+1$) mit dem weniger aufwendigen symmetrischen Konzelationssystem mit zusätzlichem Aufwand zu ver- und entschlüsseln. Wenn A_{i+2} irgendwelche Redundanz enthält, so ist (im Prinzip) Vorsicht geboten, daß die Entropie (d.h. der zufällige und deshalb dem Angreifer unbekannte Informationsgehalt) von $k_i, A_{i+1}, k_{i+1}, A_{i+2}$ nicht zu gering wird. Aber da das symmetrische Konzelationssystem eine vollständige Suche über seinen Schlüsselraum vereiteln muß (vgl. §3), die Schlüssellänge dafür also zu groß sein muß (was ab etwa 100 Bit Länge sicher der Fall ist), enthalten sowohl k_i als auch k_{i+1} genug Entropie.

Der interessierte Leser findet in [Pfit_89] viele weitere Verschlüsselungsschemata und eine quantitative Bewertung.

5.4.7 Tolerierung verändernder Angriffe bei RING-, DC- und MIX-Netz

In einem Kommunikationsnetz kann jeder Fehler als verändernder Angriff angesehen werden. Fehler „aus Versehen“ bilden eine Untermenge aller möglichen (physischen, logischen) verändernden Angriffe. Trotzdem wurden Fehler bisher so behandelt, daß unter allen Umständen die Anonymität gewahrt blieb. Ein Angreifer kann jetzt soweit gehen, daß er ständig Fehler erzeugt, d.h. Nachrichten falsch umschlüsselt, fälschlicherweise nicht weiterüberträgt, eigene sendet, wenn er es gemäß des Protokolls zum anonymen Mehrfachzugriff nicht darf, oder Schlüssel falsch überlagert, um jede Diensterbringung zu unterbinden (*denial of service*). Dieses Problem verändernder Angriffe auf die Verfügbarkeit kann auf zwei Arten gelöst werden [MaPf_87 Seite 403f].

Zum einen wird nach jedem Fehler untersucht, ob eventuell ein verändernder Angriff und, wenn ja, durch wen vorliegt. Da die erforderlichen Maßnahmen sehr aufwendig sind und darüber hinaus u.U. den Sender bzw. Empfänger einer Nachricht identifizieren, ist es besser, erst auf Verdacht (anormale Fehlerrate u.ä.) die entsprechenden Maßnahmen einzuleiten. Um verändernde Angriffe erkennen zu können, muß bei allen Grundverfahren *jede Station ihre Aktivitäten aufdecken*:

- Beim MIX-Netz muß jeder MIX alle ein- und auslaufenden Informationseinheiten ab einem festgelegten Zeitpunkt offenlegen – sofern dies durch das zugrundeliegende Kommunikationsnetz nicht schon sowieso geschehen ist. Dann kann jeder prüfen, ob eine von ihm gesendete Informationseinheit von diesem MIX falsch ent- oder verschlüsselt oder gar ganz unterschlagen wurde. Um dies einer dritten Partei zu beweisen, müssen die Zusammenhänge der betroffenen Informationseinheiten aufgedeckt werden[117]. Es ist jedoch nicht erforderlich, die Zu-

[117] Wer die Entschlüsselung einer Informationseinheit spezifiziert, kurzum: sie verschlüsselt hat, kann dies folgendermaßen tun: Er gibt die bei der Verschlüsselung der Informationseinheit verwendete Zufallszahl an (die er sich zu diesem Zwecke merken muß). Dann kann bei einem *deterministischen* asymmetrischen Konzelationssystem jeder prüfen, ob die Informationseinheit zusammen mit der angegebenen Zufallszahl verschlüsselt eine der Eingaben des MIXes ergibt. Ist dies der Fall und erscheint die Informationseinheit nicht als eine der Ausgaben des MIXes, so hat der MIX einen verändernden Angriff durchgeführt – es sei denn, er kann nachweisen, daß er die Eingabe-Informa-

sammenhänge aller Informationseinheiten zu offenbaren, so daß diese Maßnahme zwar hohen Aufwand verursacht, nicht aber die Anonymität integer übertragener Informationseinheiten gefährdet [Chau_81 Seite 85].

- Beim DC-Netz muß jede Station alle Schlüsselzeichen (nicht aber den Startwert des PZG, sofern einer verwendet wurde) und ihr gesendetes Zeichen für die Zeichen aufdecken, bei denen ein verändernder Angriff vermutet wird [Cha3_85, Chau_88].
- Beim RING- und BAUM-Netz muß jede Station ihre gesendeten Zeichen für die Zeichen aufdecken, bei denen ein verändernder Angriff vermutet wird.

Sowohl beim DC- als auch beim RING- und BAUM-Netz ist es Aufgabe eines global vereinbarten **Aufdeckverfahrens**, dafür zu sorgen, daß

A) einerseits *alle Zeichen potentiell aufgedeckt werden können*, denn anderenfalls könnte zumindest bei manchen Zeichen ohne Aufdeckrisiko gestört werden, und daß

B) andererseits *keine Zeichen aufgedeckt werden, die die Senderanonymität* von entweder selbst sensitiven oder mit solchen auch nur indirekt verkettbaren Nachrichten *untergraben*.

Letzteres erfordert anscheinend die Verwendung eines Reservierungsverfahrens als Mehrfachzugriffsverfahren.

In [Cha3_85, Chau_88] ist (nur) das folgende kombinierte Reservierungs- und **Aufdeckverfahren für das DC-Netz** enthalten:

- Alle Teilnehmerstationen reservieren in jeder Runde genau einen Übertragungsrahmen. Werden bei der Reservierung nicht genausoviel Übertragungsrahmen reserviert wie Teilnehmerstationen am DC-Netz teilnehmen, werden alle Reservierungsbits aufgedeckt und geprüft, ob jede Station genau ein Reservierungsbit gesendet hat. Da alle Teilnehmerstationen in jeder Runde genau einen Übertragungsrahmen reservieren, ergibt dies Aufdecken keinerlei Information über die Stationen, die sich an das Reservierungsschema gehalten haben.
- Bereits vor der Reservierung sendet jede Teilnehmerstation eine ihr zuordenbare, mögliche *Aufdeckforderung*, nämlich die Bitposition ihrer Reservierung und die zu sendende Nachricht – beides zusammen mit einer Blockchiffre verschlüsselt. Wird die Teilnehmerstation beim Senden gestört und möchte, daß die Störung aufgedeckt wird, so veröffentlicht sie (in ihr zuordenbarer Weise) den Schlüssel, mit dem ihre mögliche Aufdeckforderung verschlüsselt ist, wodurch aus der möglichen eine tatsächliche Aufdeckforderung wird. Da dies die Senderanonymität ihrer Nachricht untergräbt, wird eine Teilnehmerstation dies nur dann tun, wenn sie nichts zu senden hatte und ihre Nachricht also eine bedeutungslose Nachricht, in diesem Zusammenhang eine bedeutungsvolle Falle (trap) für Störer ist.

David Chaums Aufdeckverfahren leistet B) überhaupt nicht und – je nach vom Leser unterstelltem Teilnehmerstationenverhalten[118] – A) höchstens im *komplexitätstheoretischen* Sinne:

- Der Angreifer kann durch das Veröffentlichen möglicher Aufdeckforderungen, die sich gar nicht auf seine Nachricht beziehen, ein Aufdecken für beliebige, von ihm *vorher* gewählte

---

tionseinheit schon in einem früheren Schub bearbeitet, er also lediglich eine Wiederholung ignoriert hat. Bei einem *indeterministischen* asymmetrischen Konzelationssystem muß der Verschlüsseler, sofern neben der bisher diskutierten Zufallszahl bei der Verschlüsselung weitere Zufallsinformation verwendet wurde, auch diese bekannt geben. Auch hier ist die *Entschlüsselung deterministisch*, so daß ohne Bekanntgabe der Entschlüsselungsfunktion – was alle Zusammenhänge und nicht nur die strittigen aufdecken würde – auf die richtige Ausgabe-Informationseinheit geschlossen werden kann.

118 [Cha3_85, Chau_88 Seite 72] ist nicht zu entnehmen, ob Teilnehmerstationen auch für sinnvolle Nachrichten mögliche Aufdeckforderungen senden und ob diese möglichen, aber nie tatsächlichen Aufdeckforderungen auch die richtige Bitpositionen und Nachrichten enthalten. Wie gleich ersichtlich wird, ist zumindest ersteres sinnvoll. Außerdem vereinfacht beides die Beschreibung.

(und zumindest hin und wieder von anderen benutzte) Zeichen erreichen. Werden die Reservierungsbits mit aufgedeckt, so entlarvt dies den verändernden Angreifer allerdings.

- Kann ein Angreifer die für diese Aufdeckverfahren verwendete Blockchiffre brechen, so kann er je nach Verhalten der reservierenden Teilnehmerstationen und der verwendeten Blockchiffre möglicherweise
  a) ohne Aufdeckrisiko stören und
  b) sogar *nach* dem Senden gewählte Zeichen aufdecken lassen.

a) gilt, falls die Teilnehmerstationen nicht immer eine mögliche Aufdeckforderung senden oder bei sinnvollen Nachrichten falsche Bitpositionen verwenden und die verwendete Blockchiffre, wie z.B. DES, manche Abbildungen von Klar- auf Schlüsseltext ausschließt. Letzteres ist für $(2^{\text{Blocklänge}})! > 2^{\text{Schlüssellänge}}$ immer der Fall, vgl. §3.

b) gilt selbst *nach* dem Senden der Aufdeckforderung des Angreifers, falls die Blockchiffre alle möglichen Abbildungen von Klar- auf Schlüsseltext zuläßt. Anderenfalls kann der Angreifer nur vorher das bzw. die Zeichen beliebig wählen, das bzw. die er aufgedeckt haben will.

Bei ungünstiger Wahl von Teilnehmerverhalten und Blockchiffre gilt also sogar a) sowie b) mit nachträglicher Wahl des Angreifers.

Nach dem Gesagten ist klar, daß a) durch Ausführung des kombinierten Reservierungs- und Aufdeckverfahrens, wie es oben von mir beschrieben wurde, vermieden wird, sofern der Angreifer sinnvolle und bedeutungslose Nachrichten nicht unterscheiden kann. In [WaPf_89, WaPf1_89] werden Aufdeckverfahren beschrieben, die auch die Schwäche b) nicht haben. Die Forderungen A) und B) sind sogar in der informationstheoretischen Modellwelt in beweisbarer Form erfüllbar, wobei versucht werden sollte, diese Resultate auf weitere Klassen von Mehrfachzugriffsverfahren auszudehnen oder einen Beweis zu erbringen, daß dies nicht möglich ist.

Aufdeckverfahren für das DC-Netz können in kanonischer Weise auf RING- und BAUM-Netz übertragen werden.

Es sei darauf hingewiesen, daß, um ein Aufdecken zu ermöglichen, beim DC- und insbesondere beim RING- und BAUM-Netz – auch falls keine „Fehler“ auftreten – erheblicher zusätzlicher Aufwand zur Speicherung von Informationseinheiten nötig ist.

Im Rest dieses Abschnitts bleibt nun noch zu beschreiben, wie auf Dispute beim Aufdecken so reagiert werden kann, daß zumindest langfristig der „Schuldige“ bestraft wird.

Sind aus verläßlicher Quelle die *Übertragungen auf allen Leitungen bekannt* (z.B. bei ausschließlicher Verwendung eines physischen Broadcast-Mediums – im Gegensatz zu mit Hilfe von Protokollen realisierten Verteilnetzen, beispielsweise Ringen mit Punkt-zu-Punkt-Leitungen), so kann der verändernde Angreifer beim MIX-Netz sicher entdeckt werden. (Beim RING- und BAUM-Netz natürlich auch, aber hier besteht der Witz der Verfahren gerade darin, daß niemand die Übertragungen auf allen Leitungen kennt.) Anders ist dies beim DC-Netz, da hier auch die von einer Station verwendeten Schlüssel bekannt sein müssen, um zu entscheiden, ob sie gesendet hat. Sind nicht alle Übertragungen auf Leitungen öffentlich bekannt, so kann derselbe Effekt dadurch erreicht werden, daß jede Station jede von ihr auf der Leitung gesendete Informationseinheit signiert. Das Signieren und Speichern von Signaturen verursacht zwar einen erheblichen zusätzlichen Aufwand, erlaubt es aber, hinterher sicher festzustellen, was genau auf der Leitung übertragen wurde. (Ein Sonderfall liegt vor, wenn beide Stationen an einer Leitung Angreifer sind und gemeinsam lügen. Dann kann natürlich nicht festgestellt werden, was auf der Leitung übertragen wurde.)

Sind *nicht alle Übertragungen auf Leitungen bekannt und wird nicht jede Informationseinheit signiert*, so können im MIX-, RING- und BAUM-Netz sowie im DC-Netz (dort über strittige gesendete Nachrichten oder strittige Schlüssel) drei Gruppen von Stationen identifiziert werden, wovon die erste Gruppe weder beschuldigt wird noch beschuldigt, während zumindest eins von beiden bei der

zweiten und dritten Gruppe der Fall ist. Eine von ihnen ist der verändernde Angreifer. Die andere ist unschuldig, wird aber vom Angreifer beschuldigt. Dabei wird davon ausgegangen, daß sich nur ein Angreifer im Kommunikationsnetz befindet, der aber mehrere Stationen besitzen oder unterwandert haben kann, und Widersprüche nur in der Umgebung des Angreifers vorkommen. Der Angreifer hat nur dann eine Chance, nicht sofort ermittelt zu werden, wenn er wenige andere Stationen beschuldigt. Ansonsten spräche eine Mehrheit von zu Unrecht beschuldigten „ehrlichen“ Stationen gegen ihn. Im nicht entscheidbaren Fall jeweils weniger sich gegenseitig beschuldigender Stationen wird man normal weiterarbeiten, aber vorher so rekonfigurieren, daß sich im Wiederholungsfall diese wenigen Stationen nicht mehr gegenseitig beschuldigen können. Dies wird beispielsweise dadurch erreicht, daß

- beim MIX-Netz andere Wege (Adressen) verwendet werden, d.h. jeder Sender bildet Adressen nur noch so, daß keine Nachricht diese beiden MIXe direkt hintereinander passiert.
- beim DC-Netz keine Schlüssel zwischen den beiden betroffenen Stationen mehr ausgetauscht werden (die vorhandenen Schlüssel werden entfernt). Eine Alternative wäre, daß jeweils beide Partner ihren gemeinsamen Schlüssel vor seiner Verwendung digital signieren, so daß jeder Partner die Signatur des anderen hat. Dann kann bei gegenseitiger Beschuldigung von einem Dritten nach Vorlage der Signaturen entschieden werden, welche Station im Unrecht ist [Cha3_85].
  An dem direkten Signieren des gemeinsamen Schlüssels ist nachteilhaft, daß jeder der beiden einzeln – und damit möglicherweise ohne Wissen seines Partners – dann einem Dritten den Wert des gemeinsamen Schlüssels nicht nur mitteilen, sondern auch *nachweisen* kann, indem er die Signatur des anderen vorweist (seine eigene nützt nichts, da er selbst natürlich auch einen anderen, falschen Schlüssel signieren könnte). Deshalb wird in [Chau_88 Seite 73f] vorgeschlagen, daß nicht der gemeinsame Schlüssel, sondern zwei Teile, deren Summe den Schlüssel ergibt und deren einer Teil zufällig gewählt ist, jeweils von einem der Partner signiert werden. Um den Wert des gemeinsamen Schlüssels einem Dritten nachzuweisen, bedarf es nun – wie im Fall ohne Signaturen – der Mitwirkung beider, da die Signatur jeweils von dem Partner angefordert werden muß, der sie nicht geleistet hat.[119]
- beim RING- bzw. BAUM-Netz wird der Ring bzw. Baum rekonfiguriert. Dies ist bei einer Realisierung als virtueller Ring bzw. Baum, d.h. auf einem beliebigen Kommunikationsnetz wird ein logischer Ring bzw. Baum durch Verbindungs-Verschlüsselung zur Simulation der Unbeobachtbarkeit der Ring- bzw. Baum-Leitungen realisiert (was auf eine eigene Schutz-Schicht führt), sehr leicht möglich. Bei einer physischen Realisierung ist die Rekonfigurierung sehr aufwendig und daher kostenintensiv.

Ist ein Netzteilnehmer mehrfach in einer Gruppe, die eine andere beschuldigt oder von ihr beschuldigt wird, so wird er als der verändernde Angreifer angesehen und aus dem Kommunikationsnetz ausgeschlossen. Dieses Verfahren ist nur dann praktikabel, wenn relativ wenige Netzteilnehmer verändernde Angreifer sind. Ansonsten könnten die verändernden Angreifer zusammenarbeiten und die korrekt kommunizierenden Stationen aus dem Kommunikationsnetz ausschließen, wobei sie dann natürlich immer weniger Stationen zum Beobachten hätten.

---

119 Seit Erfindung der nicht herumzeigbaren Signaturen ist die gerade beschriebene Aufspaltung des gemeinsamen Schlüssels in zwei Teile nicht mehr nötig. Beide Partner signieren den gemeinsamen Schlüssel jeweils mit einer nicht herumzeigbaren Signatur, wodurch das Gleiche erreicht wird, da der Signierer jeweils an der Prüfung der Signatur aktiv mitwirken muß, vgl. §3.1.1.2.

5.4.8 Aktiver Verkettungsangriff über Betriebsmittelknappheit

In diesem Abschnitt wird ein universeller aktiver Verkettungsangriff über Betriebsmittelknappheit und Möglichkeiten zu seiner Erkennung und Abwehr beschrieben.

Ziel des Angriffs ist herauszufinden, ob zwei unterschiedliche Pseudonyme (z.B. implizite Adressen) zur selben Station gehören oder nicht. Der Angriff besteht darin, von diesen beiden Pseudonymen Dienste in solchem Umfang anzufordern, daß aus der Geschwindigkeit der Diensterbringung darauf geschlossen werden kann, ob beide Pseudonyme zur selben Station gehören – dann sind die Betriebsmittel knapper und die Geschwindigkeit der Diensterbringung ist folglich geringer, als wenn die Dienste von unterschiedlichen Stationen erbracht werden. Als knappe Betriebsmittel werden Rechenleistung, Sende- und Empfangsbandbreite betrachtet.

Ist die

- lokale *Rechenleistung* von Stationen begrenzt oder die
- anonym und unverkettbar durch *Senden* oder *Empfangen* nutzbare Bandbreite des Kommunikationsnetzes pro Station auf einen echten Bruchteil der Gesamtbandbreite beschränkt,

so kann eine Station Dienstanforderungen, die diese Grenzen überschreiten, natürlich nicht in Realzeit erfüllen oder gar gar nicht gleichzeitig empfangen. Hieraus resultieren drei Angriffsarten entsprechend den drei knappen Betriebsmitteln. Sie haben gemeinsam, daß von zwei unterschiedlichen Pseudonymen (z.B. impliziten Adressen) jeweils Dienste angefordert werden, die nur erbracht werden können, wenn die beiden Pseudonyme unterschiedlichen Stationen gehören. Hierzu muß der Angreifer entweder

- die Rechenleistung der angegriffenen Station schätzen können, oder
- ihre maximale Sendebandbreite kennen, was leicht ist, wenn sie durch das Kommunikationsnetz, etwa beim MIX-Netz, fest vorgegeben ist. Anderenfalls muß er auch dies schätzen. Um seine Schätzung zu erleichtern, kann er selbst natürlich auch gegen ein faires Zugriffsprotokoll verstoßen (vgl. z.B. §5.4.5.4.2 Kollisionsauflösungsalgorithmus mit Mittelwertbildung und überlagerndem Empfangen) und dadurch der bzw. den angegriffenen Stationen die maximale Sendebandbreite verknappen.
- Kennt der Angreifer die maximale Empfangsbandbreite, was leicht ist, wenn sie durch das Kommunikationsnetz, etwa beim MIX-Netz ohne Verteilung „letzter“[120] Nachrichten, fest vorgegeben ist, oder kann er sie schätzen, so müssen sich also so viele Angreiferstationen zusammentun, daß sie überschritten wird. Ggf. können natürlich auch wenige Angreiferstationen gegen ein faires Zugriffsprotokoll verstoßen.

Wird der Dienst rechtzeitig erbracht, so ist das Ergebnis des Angriffs, daß (ggf. unter der Annahme richtiger Schätzung) zwei Pseudonyme nicht zur selben Station gehören können. Wird der Dienst nicht rechtzeitig erbracht, so kann dies natürlich auch an sonstigen Dienstanforderungen an eine der beiden Stationen liegen. Kennt der Angreifer von jeder Station ein Pseudonym, z.B. eine ihr öffentlich zugeordnete Adresse, und erbringen Stationen unter diesen Adressen auch Dienste, so wählt der Angreifer als eines der beiden Pseudonyme jeweils ein nichtanonymes. Damit kann er jedes Pseudonym mit in der Stationenanzahl linearem Aufwand einer Station zuordnen.

Dieser *aktive* Verkettungsangriff über Betriebsmittelknappheit kann einerseits von der angegriffenen Station erkannt werden: Solange eine Station nicht aufgrund externer Dienstanforderungen an ihre Kapazitätsgrenzen stößt, wurde sie nicht angegriffen. Dies ist eine für die Überprüfbarkeit von Schutz weit bessere Situation als die *passiver* Angriffe.

Andererseits kann der aktive Angriff leicht erschwert werden:

---

120 d.h. direkt an den Empfänger gerichteter Nachrichten, vgl. §5.4.6.3

- Jeder Teilnehmer ist sowohl frei, wieviel Rechenleistung er „besitzt“ als auch, wieviel davon er für externe Dienstanforderungen bereitstellt. Letzteres kann er frei wählen und dies sogar von seiner Station (pseudo)zufällig über die Zeit verteilt tun lassen.
- Beim DC-, RING- und BAUM-Netz werden alle Stationen so ausgelegt, daß jede einzelne die gesamte Bandbreite zum Senden nutzen kann. Dies verhindert allerdings einige in [Pfit_89] beschriebene Optimierungen zur Einsparung von Betriebsmitteln.
- Bei Verteilung werden alle Stationen so ausgelegt, daß sie die gesamte Bandbreite zum Empfangen nutzen können. Dies verhindert allerdings einige in [Pfit_89] beschriebene Optimierungen zur Einsparung von Betriebsmitteln.

Mit großem Aufwand und einschneidenden Einschränkungen der nutzbaren Leistung kann der aktive Angriff auch durch *statische Aufteilung der Betriebsmittel* vollständig verhindert werden: Vor der eigentlichen Dienstanforderung werden alle Pseudonyme anonym und unverkettbar bekanntgegeben, unter denen in der anstehenden Dienstphase ein Dienst angefordert werden wird. Sei *n* die Anzahl dieser Pseudonyme. Jede Station stellt dann jeder Dienstanforderung genau den *n*-ten Teil der minimalen im Kommunikationsnetz verfügbaren Rechenleistung sowie der minimalen im Kommunikationsnetz verfügbaren Sende- und Empfangsbandbreite zur Verfügung.

Für den praktischen Einsatz ist ein geeigneter Kompromiß zwischen der statischen und dynamischen Betriebsmittelaufteilung zu finden.

Es sei angemerkt, daß Angriffe über Betriebsmittelknappheit und ihre Verhinderung durch statische Aufteilung der Betriebsmittel in einem anderen Kontext altbekannt sind: Benutzen zwei Prozesse gemeinsame Betriebsmittel, z.B. zwei Rechenprozesse denselben Prozessor, so kann bei fairer dynamischer Betriebsmittelaufteilung jeder der beiden Prozesse den anderen dadurch verzögern, daß er Betriebsmittel länger benutzt. Hat einer Zugriff auf die reale Zeit, so kann er diese Verzögerung feststellen. Diese modulierbare Verzögerung stellt einen verborgenen Kanal vom anderen Prozeß zu ihm dar, vgl. §1.2.2.

5.4.9 Einordnung in ein Schichtenmodell

Um den Entwurf, das Verständnis, die Implementierung und die Verbindung von Kommunikationsnetzen zu erleichtern, werden sie als **geschichtete Systeme** entworfen. Jede Schicht benutzt die **Dienste** der nächst tieferen Schicht sowie durch ihr sogenanntes **Protokoll** reglementierte Kommunikation zwischen ihren Instanzen, um der nächst höheren Schicht einen komfortableren Dienst anzubieten. Die Internationale Normungsbehörde (International Standards Organization, abgekürzt ISO) normte ein 7-Schichten-Modell, das „Grundlegende Referenzmodell für die Verbindung offener Systeme“ (Basic reference model for Open Systems Interconnection, abgekürzt OSI), dessen Schichten – wenn nötig – zu Teilschichten verfeinert werden, etwa um es Lokalen Netzen anzupassen. Da alle internationale Normungsarbeit sich auf dieses ISO OSI Referenzmodell bezieht, geben die folgenden beiden Bilder die passenden Schichten für Ende-zu-Ende- und Verbindungs-Verschlüsselung bzw. die Grundverfahren zum Schutz der Verkehrs- und Interessensdaten an.

Um beispielsweise Vermittlungszentralen keine unnötige Protokollinformation zugänglich zu machen, muß Ende-zu-Ende-Verschlüsselung in der tiefsten Schicht erfolgen, deren Protokoll Ende-zu-Ende, d.h. direkt zwischen den Teilnehmerstationen der Kommunikationsnetzbenutzer arbeitet. Deshalb ist Schicht 4 (Transportschicht, transport layer) die für Ende-zu-Ende-Verschlüsselung geeignete Schicht.[121]

---

[121] Dies gilt nicht unbedingt für *nichtkanonische* Kommunikationsnetze, etwa solche, die aus mehreren, auf einer höheren Schicht als der Transportschicht gekoppelten Netzen bestehen. Bei solchen nichtkanonischen Netzen erfolgt eine Protokollumsetzung mittels eines oder mehrerer Protokollumsetzer (Gateways) auf einer so hohen Schicht, daß das Transportprotokoll nicht mehr Ende-zu-Ende, sondern nur Ende-zu-Gateway und Gateway-zu-Ende arbeiten kann.

Da Angreifer, die Leitungen (oder allgemeiner: Kommunikationskanäle) abhören, auch keine unnötige Protokollinformation erhalten sollten, muß Verbindungs-Verschlüsselung in der tiefsten Schicht erfolgen, die mit digitalen Daten (im Gegensatz zu analogen Signalen) arbeitet. Deshalb ist Schicht 1 (Bitübertragungsschicht, physical layer) die für Verbindungs-Verschlüsselung geeignete Schicht. Auf Kommunikationskanälen, die jeweils zwei Instanzen der Schicht 1 exklusiv zugeordnet sind, kann – wie in §3 beschrieben – durch Einsatz einer Stromchiffre vor dem Angreifer sogar ohne zusätzliche Kosten verborgen werden, ob und wieviel Information auf dem Kommunikationskanal fließt.

Diese Schichten (aber leider noch viele andere) werden in [ISO7498-2_89] als mögliche Implementierungsorte für Ende-zu-Ende- und Verbindungs-Verschlüsselung genannt. Hoffentlich führen die obigen Argumente dazu, daß sie (und nicht nur jeweils höher liegende oder gar keine) als Implementierungsort gewählt werden, auch wenn sich manche Implementierer davon eine leichtere oder schnellere Implementierung und alle Geheimdienste umfassendere Arbeitsmöglichkeiten versprechen mögen.

Mißtraut man der Implementierung einzelner (Teil)Schichten bezüglich Vertraulichkeit oder Integrität (vgl. §5.1), sollten die Nutzdaten dieser Schicht von der nächst höheren zum Zwecke der Konzelation oder Integrität verschlüsselt werden (vgl. §3). Hierdurch können zusätzliche Ende-zu-Ende- oder Verbindungs-Verschlüsselungen entstehen. Letzteres erscheint weniger notwendig, da die Entwurfskomplexität tieferer (Teil)Schichten geringer als die höherer (Teil)Schichten und unter anderem deshalb Fernwartung zur Korrektur von Fehlern in tieferen (Teil)Schichten überflüssig und nicht vorgesehen ist.

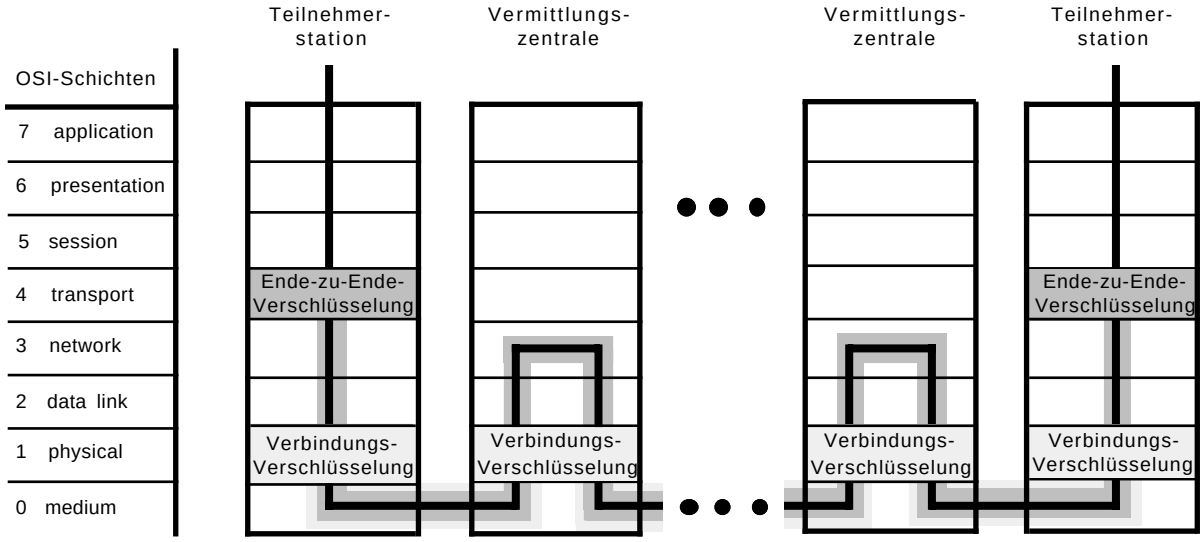


Bild 5-39: Einordnung der Ende-zu-Ende- und Verbindungs-Verschlüsselung in das ISO OSI Referenzmodell

Das Grundverfahren zum Schutz des Empfängers verwendet Verteilung (broadcast).

Verteilung kann in einem beliebigen Kommunikationsnetz durch geeignete Wegeermittlung (routing) in Schicht 3 (Vermittlungsschicht, network layer) realisiert werden, so daß Schicht 4 nur noch implizite Adressen generieren und auswerten muß. Eine mögliche Realisierung von Verteilung

---

Da zur Protokollumsetzung auf einer höheren Schicht als der Transportschicht die Daten der Transportschicht interpretiert werden müssen, müssen sie vom Gateway interpretiert werden können und dürfen folglich nicht Ende-zu-Ende-verschlüsselt sein. Bei nichtkanonischen Kommunikationsnetzen kann die Ende-zu-Ende-Verschlüsselung folglich bestenfalls auf einer höheren Teilschicht der höchsten Schicht erfolgen, auf der die Netze gekoppelt sind.

ist Überflutung (flooding [Tane_81]), die mit einer sehr einfachen Wegeermittlungs-Strategie erreicht werden kann: Jede Station überträgt jede Nachricht an alle Nachbarstationen, von denen sie diese Nachricht (noch) nicht empfangen hat.

In speziell für Verteilung entworfenen Kommunikationsnetzen wird Verteilung effizienter durch Schicht 1 realisiert. Zumindest für manche Dienste kann Schicht 1 auch gleich die Information auswählen, die für die Teilnehmerstation bestimmt ist. Dies wird in Bild 5-40 Kanalselektion genannt.

MIX- und DC-Netz verwenden spezielle kryptographische Mechanismen, um innerhalb des Kommunikationsnetzes Anonymität und Unverkettbarkeit zu schaffen (bei einer globalen Sicht wäre der Terminus „Anonymität und Unverkettbarkeit schaffen“ ein Widerspruch in sich, da einem Angreifer natürlich durch die Gestaltung des Kommunikationsnetzes keine Information weggenommen werden kann, so daß bei globaler Sicht alle Maßnahmen nur Anonymität und Unverkettbarkeit erhalten können, vgl. §5.1.2). Diese Mechanismen ordne ich in die höchsten Teilschichten (sublayers) von Schicht 3 bzw. 1 ein, da

- das MIX-Netz die Wegeermittlung (routing) von (den tieferen Teilschichten von) Schicht 3 nutzt, aber keine Ende-zu-Ende-Maßnahme ist, die in die Schicht 4 einzuordnen wäre;
- das DC-Netz die Übertragung von Zeichen (heutzutage vor allem: Bits) von (den tieferen Teilschichten der) Schicht 1 nutzt, aber nicht mit der Entdeckung von Übertragungsfehlern, Mehrfachzugriff oder anderen Diensten der Schicht 2 (Sicherungsschicht, data link layer) befaßt ist.[122]

RING- und BAUM-Netz – wie alle nach der Idee „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ funktionierenden Kommunikationsnetze – schaffen innerhalb des Kommunikationsnetzes Anonymität, indem sie ein ring- bzw. baumförmiges – allgemeiner: passend förmiges – Medium in Schicht 0 (medium) und digitale Signalregenerierung in Schicht 1 verwenden.

Bild 5-40: Einordnung der Grundverfahren zum Schutz der Verkehrs- und Interessensdaten in das ISO OSI Referenzmodell

Hieraus folgt, daß Verteilung als Grundverfahren zum Schutz des Empfängers

- beliebig implementierte Schichten 0, 1, und 2 ohne Effizienzeinbuße nutzen kann, sofern eine Implementierung in einem nicht speziell für Verteilung entworfenen Kommunikationsnetz erfolgt, oder

[122] Wird das DC-Netz auf einem Medium mit Mehrfachzugriff implementiert, z.B. LAN, so muß es notgedrungen in Schicht 2 oder höher implementiert werden.

- nur eine beliebig implementierte Schicht 0 nutzen kann, sofern eine Implementierung in einem speziell für Verteilung entworfenen Kommunikationsnetz erfolgt.

Das MIX-Netz kann beliebig implementierte Schichten 0, 1, 2, und tiefere Teilschichten von Schicht 3 ohne Effizienzeinbuße nutzen, das DC-Netz eine beliebig implementierte Schicht 0 und tiefere Teilschichten von Schicht 1.

Alle diese (Teil)Schichten können ohne Einschränkungen durch Anonymitätsanforderungen implementiert werden, so daß in ihnen alle Verfahren zur Steigerung von Leistung und Zuverlässigkeit eingesetzt werden können.

Alle Schichten oberhalb der (innerhalb des Kommunikationsnetzes) Anonymität und Unverkettbarkeit schaffenden (Teil)Schichten müssen die **Anonymität** und ggf. auch die **Unverkettbarkeit erhalten**. Dies bedeutet, daß diese oberen Schichten keine (oder zumindest nicht zu viele) Alternativen ausschließen oder verketten, die die Anonymität und Unverkettbarkeit schaffenden Schichten ermöglichen – jeweils gesehen aus der Perspektive des Angreifers.[123] Anderenfalls könnte ein Angreifer das allgemein verfügbare Wissen über die höheren Schichten verwenden, um Sender oder Empfänger zu identifizieren oder sie (oder Sendeereignisse) miteinander in Beziehung zu setzen. Beispielsweise müssen die Protokolle zum anonymen Mehrfachzugriff nicht nur die Bandbreite des gemeinsamen Kanals des DC-, RING-, oder BAUM-Netzes effizient nutzen, sondern auch die Anonymität der Sender (sowie möglichst die Unverkettbarkeit von Sendeereignissen) erhalten.

Für die Normung der Protokolle dieser Schichten hat dies Konsequenzen, die von den zuständigen Normungsgremien entweder noch nicht als solche erkannt oder aber ignoriert werden. Wenn eine Norm nicht eine Teilmenge hinreichender Funktionalität enthält, die Anonymität (und ggf. auch Unverkettbarkeit) erhält, ist sie für Kommunikationsnetze mit überprüfbarem Datenschutz nutzlos. Folglich sind solche internationalen Normen in Ländern, deren Verfassung oder Gesetze Datenschutz vorschreiben, nicht anwendbar, und nationale Normen möglicherweise illegal. Insbesondere die erste Möglichkeit und die Tatsache, daß das Ändern von Normen sehr lange dauert, kann in der Zukunft internationale Kommunikation wesentlich behindern. Außerdem ist es eine offene Frage, ob Menschen in anderen Ländern gewillt sind, ein Kommunikationsnetz zu benutzen, das sie leicht beobachten kann.

Während es aus den gerade geschilderten Gründen nötig ist, daß die Protokolle der Schichten oberhalb der (innerhalb des Kommunikationsnetzes) Anonymität und Unverkettbarkeit schaffenden (Teil)Schichten die Anonymität der Beteiligten und möglichst auch die Unverkettbarkeit der Sendeereignisse erhalten, so sind insbesondere für letzteres bezüglich der Beteiligten sinnvolle Grenzen durch die Natur der zu erbringenden Dienste vorgegeben. Beispielsweise kann vor den an einem Telefongespräch Beteiligten nicht verborgen werden, daß sich Sätze und damit auch die sie transportierenden Informationseinheiten, seien es nun durch Kanalvermittlung geschaltete kontinuierliche Bitströme oder nur Sprachpakete, wenn tatsächlich gerade etwas zu hören ist (TASI = time assignment speech interpolation [Amst_83, LiFl_83, Rei1_86]), in beiden Kommunikationsrichtungen aufeinander beziehen. Allgemeiner ausgedrückt: Es ist nicht möglich, vor Beteiligten mehr Anonymität oder Unverkettbarkeit von Kommunikationsereignissen zu schaffen, als der gerade abgewickelte Dienst zuläßt. Die Forderung, daß einem Ereignis der höheren Schicht möglichst alle Ereignisse der tieferen Schicht entsprechen und daß Ereignisse möglichst nicht verkettet werden können, bezieht sich also ausschließlich auf die Unbeobachtbarkeit durch Unbeteiligte. Wenn die zu berücksichtigenden Angreifer entweder selbst Beteiligte sind oder die Kooperation von Beteiligten haben, so kann ggf. er-

[123] Anders formuliert bedeutet dies, daß keine (oder zumindest nicht allzu viele) der auf der Anonymität und Unverkettbarkeit schaffenden Schicht möglichen und vom Angreifer ununterscheidbaren Alternativen von den höheren Schichten ausgeschlossen (oder verkettet) werden.

heblicher Aufwand gespart werden, wenn gar nicht erst versucht wird, einem Angreifer die Verkettung von über den Dienst bereits verketteten Verkehrsereignissen unmöglich zu machen.

Es sollte erwähnt werden, daß die speziellen Verfahren zum Schutz des Empfängers, des MIX-, DC-, RING- und BAUM-Netzes natürlich auch in *höheren* (Teil)Schichten implementiert werden können, beispielsweise auf einer beliebig gestalteten Schicht 3 (Vermittlungsschicht). Verteilung und MIXe würden dann in der Schicht 4 (Transportschicht) implementiert, die dann einige Funktionen der Schicht 3, z.B. Wegeermittlung, nochmals enthalten müßte. Wenn überlagerndes Senden in der Schicht 4 (Transportschicht) implementiert würde, müßte sogar anonymer Mehrfachzugriff nochmals in den höheren (Teil)Schichten implementiert werden. Dasselbe gilt für das RING- und BAUM-Netz, bei denen zusätzlich noch Verschlüsselung zwischen den Einheiten der Schicht 4 (virtuelle Verbindungs-Verschlüsselung) nötig wäre, um die physische Unbeobachtbarkeit der Leitungen zu simulieren.

Verteilung, MIX-, DC- und sogar RING- und BAUM-Netz können daher als *virtuelle*, d.h. in fast beliebigen Schichten implementierbare Konzepte betrachtet werden. Aber eine effiziente Implementierung muß unnötig komplexe Schichtung und nochmalige Implementierung von Funktionen vermeiden. Deshalb gibt Bild 5-40 den kanonischen und vernünftigen Entwurf an.

Aus Bild 5-40 ist ebenfalls zu ersehen, wie die im §5.4.4 und seinen Unterabschnitten beschriebene Erweiterung des Konzeptes der „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ um „überlagerndes Senden“ kanonischer- und vernünftigerweise implementiert wird: Auf einem Medium passender Topologie (Schicht 0) und einer (Teil)Schicht 1 mit digitaler Signalregenerierung setzt eine weitere, höhere (Teil)Schicht 1 mit Überlagerung von Schlüsseln und Nachrichten auf. Darüber arbeitet dann das anonyme Mehrfachzugriffsverfahren.

Wie in [DiHe_79 Seite 420] erklärt, darf die Implementierung der (Teil)Schichten, die (innerhalb des Kommunikationsnetzes) Anonymität und Unverkettbarkeit schaffen oder erhalten müssen, keine Eigenschaften einführen, die ein Angreifer verwenden kann, Alternativen zu unterscheiden, die im abstrakten Entwurf für ihn ununterscheidbar sind. Beispiele solch unbrauchbarer Implementierungen wären

- Modulation der Ausgabe eines MIXes gemäß den in den Nachrichten enthaltenen zufälligen Bitketten,
- direkte Ausgabe des Ergebnisses eines modulo-2-Addierers, für dessen Ausgabe $1+1 \neq 0+0$ und $1+0 \neq 0+1$ gilt, wenn die analogen Signalcharakteristika betrachtet werden, oder
- zeichenfolgensensitive Taktverschiebungen der digitalen Signalregenerierung (pattern sensitive timing jitter) beim RING-Netz.

Diese und andere Fehler, vor allem aber geeignete Implementierungen werden in [Pfit_89 §3.1 und §3.2] noch in größerer Tiefe betrachtet.

5.4.10 Vergleich von Stärke und Aufwand von RING-, DC- und MIX-Netz

Um die Stärke der beschriebenen Verfahren und ihren Aufwand zu vergleichen, ist die folgende Tabelle hilfreich. Man sieht, daß das DC-Netz zwar stärkeren Angreifern als das MIX-Netz widersteht. Dafür ist der Aufwand des MIX-Netzes signifikant geringer.

In der Tabelle entsprechen die jeweils 3 unterschiedenen Bereiche beim DC-Netz sich in Angreifermodell und Aufwand. Beim Zeichen ≥ trifft = genau bei Duplex-Kommunikation zu. Beim MIX-Netz entsteht der Wachstumswert, der praktisch relevant ist, indem das sehr langsame Längenwachstum der Nachrichten durch die notwendigen Zufallszahlen vernachlässigt wird.

	Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung, z.B. RING-Netz	DC-Netz	MIX-Netz
Angreifermodell	physisch beschränkt	informationstheoretisch --- bzgl. Vertraulichkeit informationstheoretisch, bzgl. Diensterbringung komplexitätstheoretisch beschränkt --- komplexitätstheoretisch beschränkt • kryptographisch stark • wohluntersucht	komplexitätstheoretisch beschränkt; Es sind nicht einmal wohluntersuchte, gegen adaptive aktive Angriffe sichere asymmetrische Konzelationssysteme bekannt, die praktikabel sind.
Aufwand pro Station und pro Bit	Übertragung: $O(n)$ (genauer: $\geq \frac{n}{2}$)	polynomial in n, aber unpraktikabel --- Übertragung: $O(n)$ (genauer: $\geq \frac{n}{2}$) Schlüsselaustausch: $O(k \bullet n)$ --- Übertragung: $O(n)$ (genauer: $\geq \frac{n}{2}$) Schlüsselgenerierung: $O(k \bullet n)$	Übertragung im Teilnehmeranschlußbereich: $O(k)$, praktisch: ≈ 1; Übertragung im Innern des Netzes, insgesamt: $O(k^2)$, praktisch: $\approx k$

n = Teilnehmerzahl k = Zusammenhang Schlüsselgraph DC-Netz, bzw. Anzahl MIXe

Tabelle: Aufwand der Grundverfahren

5.5 Ausbau zu einem breitbandigen diensteintegrierenden Digitalnetz

Der Ausbau dieses schmalbandigen zu einem **breitbandigen Vermittlungs-/Verteilnetz** sollte je nach Erschließungszustand und Besiedelungsstruktur des zu versorgenden Gebietes erfolgen (Bild 5-41).

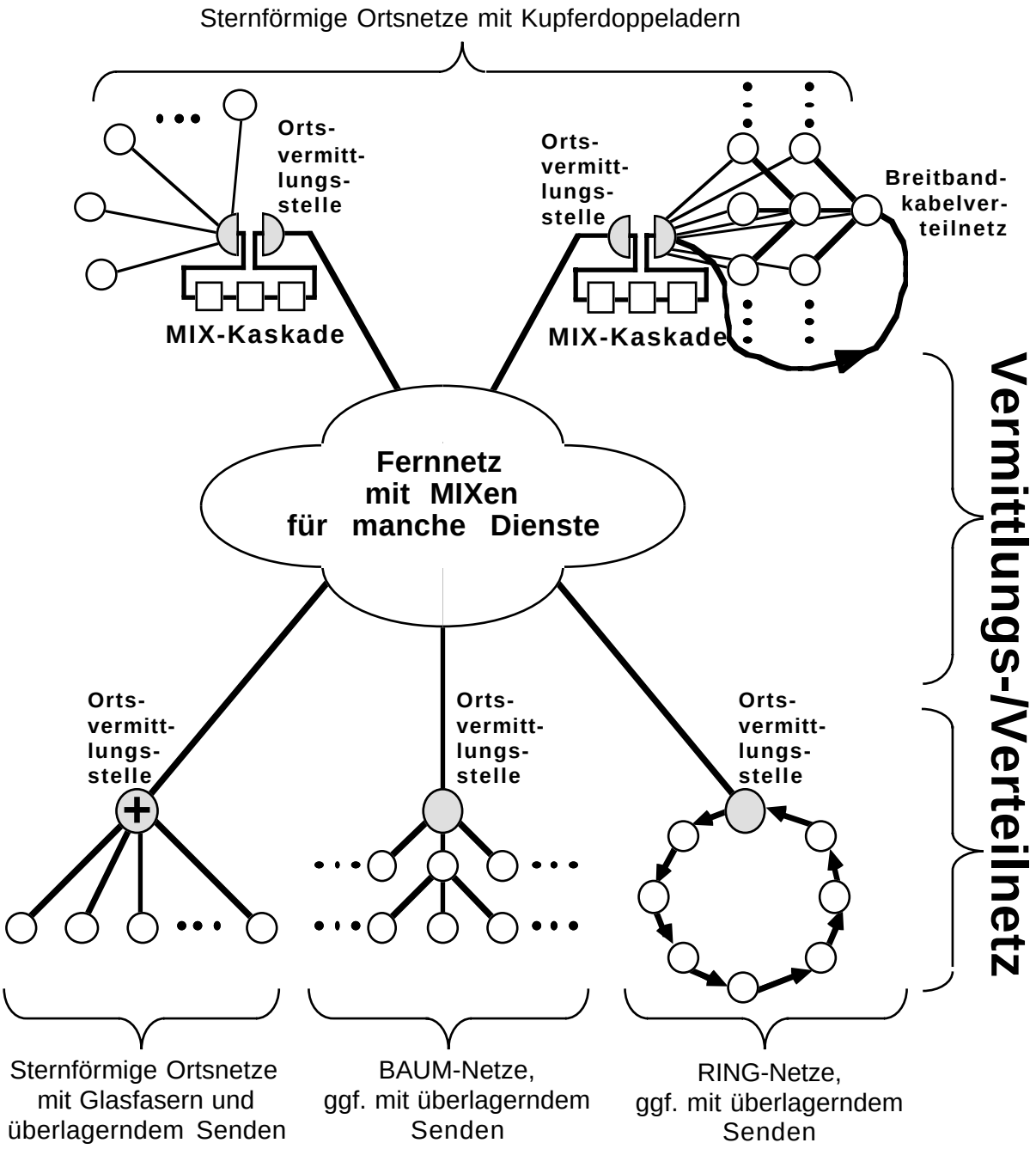


Bild 5-41: Integration verschiedener Schutzmaßnahmen in einem Netz

Sind im zu versorgenden Gebiet schon Kabelkanäle vorhanden oder handelt es sich um ein ländliches Gebiet und sind inzwischen genügend schnelle Pseudozufallszahlengeneratoren mit guten Sicherheitseigenschaften verfügbar, so dürfte es billiger sein, die Verkabelungsstruktur beizubehalten und die Verteilnetze mit überlagerndem Senden zu realisieren, selbst wenn die Pseudozufallszahlengeneratoren noch teuer sind, als die Verkabelungsstruktur zu ändern und RING-Netze als Verteilnetze einzusetzen. Sind im zu versorgenden Gebiet noch keine Kabelkanäle vorhanden und sind noch keine genügend schnellen Pseudozufallszahlengeneratoren mit guten Sicherheitseigenschaften verfügbar, sollten RING-Netze als Verteilnetze realisiert werden. Trifft keine der obigen Bedingungen zu und besteht unmittelbarer Handlungsbedarf, so sollte trotz erhöhter Kosten ringförmig verkabelt und bei ländlichen Gebieten überlagerndes Senden später nachgerüstet werden.

Im Laufe der Zeit kann der Umfang der durch überlagerndes Senden realisierten Verteilnetze vergrößert werden. Bis die Bundesrepublik vielleicht in sehr ferner Zukunft über ein Verteilnetz mit allen

Kommunikationsdiensten versorgt wird, sollten für besonders sensitive und schmalbandige Dienste zur Erhöhung des Schutzes über das durch die Verteilnetze bzw. MIX-Kaskaden bei den Ortsvermittlungsstellen gegebene Maß hinaus MIX-Kaskaden bei den Fernvermittlungsstellen benutzt werden.

Dies und die beschriebenen Etappen sprechen dafür, MIX-Kaskaden zunächst bei den Ortsvermittlungsstellen und später entweder zusätzliche bei den Fernvermittlungsstellen zu errichten oder nach Umstellung eines Ortsnetzes von Kupferdoppeladern auf Glasfasern die der Ortsvermittlungsstelle zugeordnete MIX-Kaskade dann einer Fernvermittlungsstelle zuzuordnen. Insbesondere in Großstädten, deren Ortsnetz wegen der hohen Dichte an Anschlüssen, insbesondere Geschäftsanschlüssen, einerseits früh mit digitalen Ortsvermittlungsstellen und andererseits auch früh mit Glasfasern ausgebaut werden soll, kann dies ohne großen Aufwand geschehen, da Großstädte sowohl Orts- als auch Fernvermittlungsstellen besitzen.

Parallel zum beschriebenen Netzausbau bezüglich Nutzleistung muß er auch bezüglich Zuverlässigkeit erfolgen. Dabei werden Fehlertoleranz-Maßnahmen stetig an Gewicht gewinnen.

Zum Schluß sei erwähnt, daß weitgehend unklar ist, wie die *Kosten* bzw. *Effizienz* des beschriebenen Netzausbaus sich zu den des geplanten verhalten.

5.5.1 Telefon-MIXe

Das folgende Verfahren, **Telefon-MIXe** genannt, erlaubt den Teilnehmern das Unterhalten **unbeobachtbarer Verbindungen**, d.h. unbeobachtbarer Ende-zu-Ende-Kanäle. Die unbeobachtbaren Verbindungen können im Gegensatz zu den MIX-Kanälen beliebig lange andauern und von jedem der Partner nahezu jederzeit abgebaut werden.

Es wird im wesentlichen nur die Verbindung zwischen den Netzabschlüssen der Teilnehmer betrachtet. Die darauf aufbauende Verbindung zwischen den Endgeräten bzw. den i. allg. menschlichen Teilnehmern selbst wird nur dort betrachtet, wo sie sich von der Situation im ISDN unterscheidet. Über einen ISDN-Basisanschluß mit – wie vorgesehen – 144 kbit/s vollduplex kann ein Teilnehmer gleichzeitig zwei 64-kbit/s-Vollduplexverbindungen unterhalten.

Die Telefon-MIXe erfordern keine Veränderung des Fernnetzes und erlauben einen schrittweisen Ausbau, d.h. das Nebeneinander von analogen bzw. Standard-ISDN-Ortsvermittlungsstellen und den neuen, hier vorgeschlagenen.

Schmalstbandige Dienste, die im ISDN zusammen mit der Signalisierung über den 16-kbit/s-Signalisierungskanal abgewickelt werden könnten, werden im folgenden nur am Rande betrachtet.

Im Gegensatz zu bisher soll hier gleich ein hierarchisches Netz mit Ortsnetzen und Fernnetz betrachtet werden.

In §5.5.1.1 und §5.5.1.2 wird das Grundprinzip der Telefon-MIXe, die Verwendung von Zeitscheibenkanälen, dargestellt. In §5.5.1.3 und §5.5.1.4 wird der Auf- und Abbau von Verbindungen behandelt. In §5.5.1.5 werden die Abrechnung der Netzbenutzung, in §5.5.1.6 Verbindungen zwischen Ortsvermittlungsstellen (OVSt'n) mit MIX-Kaskade und herkömmlichen OVSt'n und in §5.5.1.7 die Zuverlässigkeit betrachtet.

5.5.1.1 Grundprinzip der Zeitscheibenkanäle

Beide am Ende von §5.4.6.9 für MIX-Kanäle genannten Probleme werden gelöst, indem beliebig lange andauernde Verbindungen auf mehrere kurze (≈ 1 s) und unmittelbar aufeinanderfolgende MIX-Kanäle, genannt **Zeitscheibenkanäle**, aufgeteilt werden. Mit jeder neuen Zeitscheibe hat jeder Teilnehmer die Möglichkeit, eine neue Verbindung einzurichten oder eine bestehende abzubrechen.

Jeder Zeitscheibenkanal habe die Zeitdauer *z*. Die Einteilung in Zeitscheiben erfolgt synchron für alle Teilnehmer des gesamten Netzes. Wie in §5.4.6.9 besteht ein Zeitscheibenduplexkanal aus einem Sende- und einem Empfangskanal, die mit **ZS-Kanal** und **ZE-Kanal** abgekürzt werden. Für jede Zeitscheibe baut ein Netzabschluß eine konstante Anzahl von ZS- und ZE-Kanälen auf (d.h. hier jeweils ein Paar je 64-kbit/s-Kanal).

Eine reale Verbindung wird durch eine **Verbindungswunschnachricht** eingeleitet, die, wie oben die Kanalwunschnachricht, der Netzabschluß *R* des rufenden Teilnehmers an den Netzabschluß *G* des gerufenen Teilnehmers sendet. Im Idealfall, d.h. wenn *G* den Wunsch sofort akzeptiert und beide sich an ihr Protokoll halten, verknüpfen *R* und *G* ab einer vorherbestimmten Zeitscheibe ihre ZS- und ZE-Kanäle, bis einer von beiden den Abbruch signalisiert und sie wieder synchron ihre ZS- und ZE-Kanäle trennen. (Weitere Kanalwunschnachrichten für die einzelnen Zeitscheibenkanäle werden bei geeigneter Verbindungswunschnachricht nicht benötigt, vgl. §5.5.1.3.)

Solange ein Netzabschluß keine reale Verbindung unterhält, sendet er zufällige Daten an sich selbst, d.h. sein ZS-Kanal ist direkt mit seinem ZE-Kanal verbunden. Da bedeutungsvolle Daten Ende-zu-Ende-verschlüsselt sind, also Außenstehenden ebenfalls zufällig erscheinen, sind zufällige Daten von einem wirklichen Ende-zu-Ende-Kanal nicht zu unterscheiden. (Alternativ könnte man Scheinkanäle auch wie in §5.4.6.9 „offen“ lassen, d.h. mit keinem passenden Gegenstück verknüpfen. Dann könnte M_m sie aber von wirklichen Kanälen unterscheiden. Dies ist auch der Grund, die beiden Teile eines Duplexkanals einzeln aufzubauen.)

Die notwendige Signalisierung erfolgt im wesentlichen wie in §5.4.6.9 über einen eigenen Signalisierungskanal, der in dem nicht für Datenkanäle genutzten Anteil der verfügbaren Bandbreite untergebracht werden muß, beim ISDN-Basisanschluß also in den in beiden Richtungen verbliebenen 16 kbit/s. Dieser Signalisierungskanal wird der Hauptengpaß des Verfahrens sein, da über ihn die Verbindungswunsch- und die ZE- und ZS-Kanalaufbaunachrichten übertragen und erstere sogar verteilt werden müssen.

Im Gegensatz zum ISDN findet daher der Teil der Signalisierung, der nach Aufbau der Verbindung zwischen den Anschlüssen noch notwendig ist, innerhalb des Datenkanals statt, allerdings nur soviel, daß der Datenkanal transparent bleibt. Die Signalisierung zum Zwecke des Verbindungsaufbaus findet jedoch ausschließlich im Signalisierungskanal statt.

Da für die Telefon-MIXe die schmalbandigen Teilnehmeranschlußleitungen und das Fernnetz unverändert gelassen werden sollen, kann das Verfahren nur im *lokalen Bereich* eingesetzt werden, d.h. jeder Ortsvermittlungsstelle (OVSt) wird jeweils eine MIX-Kaskade von *m* MIXen zugeordnet.

Die Aufgabenverteilung zwischen OVSt und MIXen ist recht einfach: Die MIXe sind ausschließlich für die MIX-Funktionen zuständig, alle übrigen Aufgaben werden von der OVSt erfüllt. Funktional kann man sich eine solche OVSt zweigeteilt vorstellen: Die eine Hälfte übernimmt die Kommunikation zwischen dem ersten MIX der Kaskade und den Teilnehmern, insbesondere auch die Aufgaben der Verteilzentrale in §5.4.1 (für diese Anwendung genau beschrieben in [PfPW1_89 §2.2.3]). Die andere Hälfte übernimmt die Kommunikation zwischen dem letzten MIX der Kaskade und dem Fernnetz (sowie einige Umsetzungsaufgaben, vgl. §5.5.1.6).

In großen Ortsnetzen (mit mehr als 5000 Teilnehmern) wird es wegen der beschränkten Bandbreite des Teilnehmeranschlusses notwendig sein, die Teilnehmer ein für allemal logisch (d.h. ohne neue OVSt oder MIXe) in mehrere Anonymitätsgruppen einzuteilen, so daß Mixen und Verteilung immer nur innerhalb einer Anonymitätsgruppe stattfinden.

Der durch die Scheinsende- und Scheinempfangskanäle entstehende zusätzliche Kommunikationsaufwand stört hier nicht, da er nur die Teilnehmeranschlußleitung des Teilnehmers selbst betrifft, die diesem exklusiv zugeordnet ist und in derselben Zeit andernfalls ungenutzt wäre.

Im folgenden soll gleich der allgemeine Fall betrachtet werden, daß *R* und *G* möglicherweise zu verschiedenen OVSt'n gehören. Sei hierzu MR_1, …, MR_m die MIX-Kaskade von *R*, MG_1, …, MG_m die von *G* (Bild 5-42).

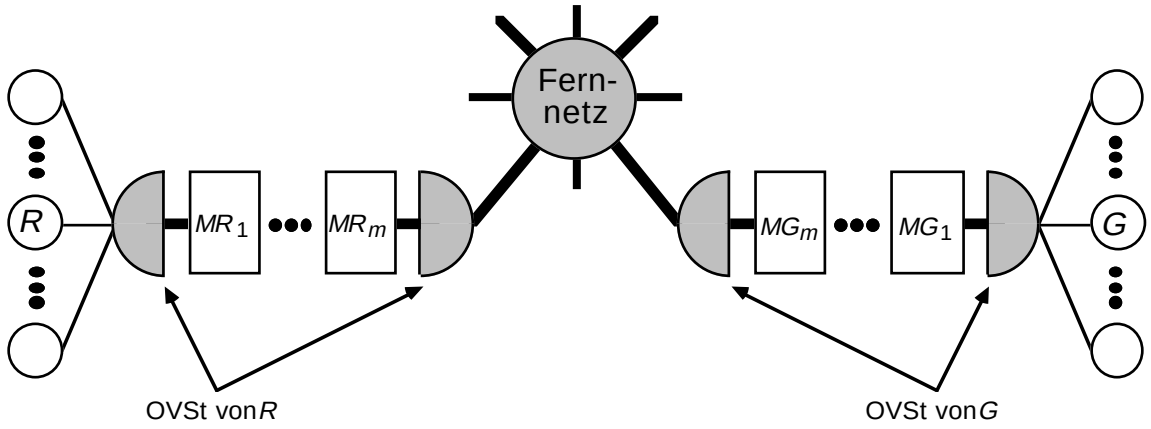


Bild 5-42: Netzabschlüsse des rufenden und des gerufenen Teilnehmers mit jeweiliger OVSt (aufgespalten in die zwei Funktionshälften) und zugeordneter MIX-Kaskade

Der Sonderfall, daß einer der beiden Netzabschlüsse an eine OVSt *ohne* MIX-Kaskade angeschlossen ist, wird in §5.5.1.6 behandelt.

5.5.1.2 Aufbau der Zeitscheibenkanäle

Zeitscheibenkanäle zeichnen sich gegenüber den MIX-Kanälen in §5.4.6.9 lediglich durch ihre vorgegebene Dauer sowie vorgegebene Anfangszeitpunkte aus, so daß hier nichts prinzipiell Neues zu sagen ist. Zu beachten ist lediglich der Einfluß des Fernnetzes auf das Nachrichtenformat und den Beginn der Zeitscheiben.

Sind *R* und *G* nicht an dieselbe OVSt angeschlossen, so müssen die MIXe MR_m und MG_m die Funktion des letzten MIXes M_m in §5.4.6.9 gemeinsam erbringen:

Ein ZS-Kanal z.B. von *R* an *G* wird von MR_m über die OVSt von *R*, das Fernnetz und die OVSt von *G* an MG_m übermittelt, der ihn in den passenden ZE-Kanal von *G* einleiten muß. Um dies zu ermöglichen, muß *R* in seiner ZS-Kanalaufbaunachricht MR_m zusätzlich die OVSt von *G* mitteilen.

Da die Verbindung zwischen *R* und *G* aus sehr vielen Zeitscheibenkanälen besteht, ist es sinnvoll, den zur Übermittlung notwendigen Kanal im Fernnetz zwischen den OVSt'n von *R* und *G* für die gesamte Verbindung zu nutzen. Der Zusammenhang der verschiedenen Zeitscheibenkanäle der Verbindung muß dabei für die MIXe MR_m und MG_m bzw. die OVSt'n nicht explizit hergestellt werden; es genügt, daß der Kanal im Fernnetz erst dann abgebaut wird, wenn erstmalig *kein* Zeitscheibenkanal zwischen den OVSt'n von *R* und *G* zu übermitteln ist.

Wie üblich muß eine MIX-Kaskade mit dem Beginn der ZE-Kanäle der aktuellen Zeitscheibe warten, bis die Anfänge aller entsprechenden ZS-Kanäle eingetroffen sind.

MG_m bzw. MR_m muß daher die schnell eintreffenden, aus demselben Ortsnetz stammenden ZS-Kanäle solange zwischenpuffern, bis ZS-Kanäle von überall aus dem Fernnetz Zeit hatten einzutreffen. (Bei einer maximalen Laufzeit von 0,2 s im Fernnetz wären dies 12,8 kbit je lokalem 64-kbit/s-Simplexkanal.)

In den Bildern 5-43 und 5-44 sind die von den MIXen benötigten Daten für den Kanalaufbau zusammengefaßt. Die Schlüssel des symmetrischen Konzelationssystems werden durch die hybride Ver-

schlüsselung übergeben, die restlichen Daten in den D_i-Feldern der Kanalaufbaunachricht. Das Feld „Entgeltmarke“ wird in §5.5.1.5 erklärt.

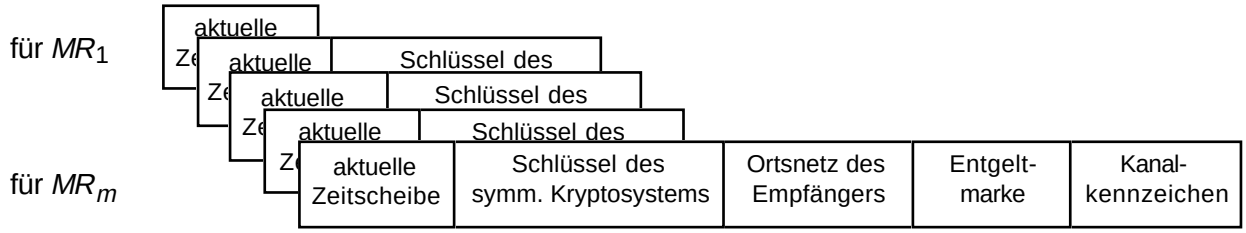


Bild 5-43: Von den MIXen MR_i benötigte Daten für den ZS-Kanalaufbau

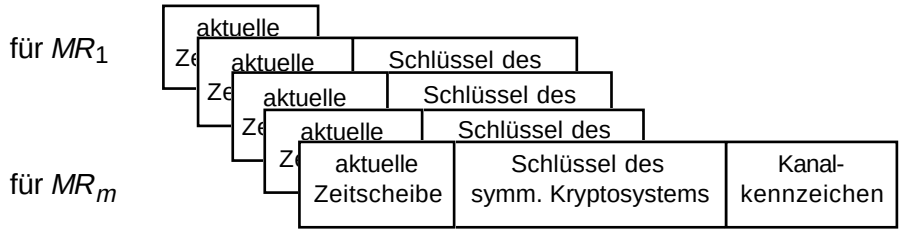


Bild 5-44: Von den MIXen MR_i benötigte Daten für den ZE-Kanalaufbau

5.5.1.3 Aufbau einer unbeobachtbaren Verbindung

Zum Aufbau einer unbeobachtbaren Verbindung sendet der Netzabschluß R des rufenden Teilnehmers über seinen Signalisierungskanal eine **Verbindungswunschnachricht**. Die Nachricht wird mit dem Schema aus §5.4.6.2 übermittelt, d.h. sie wird von der Kaskade MR_1, …, MR_m gemixt, ggf. über das Fernnetz zur OVSt des Empfängers übermittelt und dort an alle Teilnehmeranschlüsse verteilt.

Da mit jeder neuen Zeitscheibe jeder Teilnehmer die Möglichkeit haben soll, eine neue Verbindung einzurichten, müssen auch Verbindungswunschnachrichten einmal pro Zeitscheibe gemixt werden. Im folgenden wird angenommen, daß jeder Netzabschluß pro Zeitscheibe und Datenkanal genau eine, ggf. bedeutungslose, solche Nachricht beisteuert.

Der Netzabschluß G des Gerufenen muß die Nummer t_0 der ersten Zeitscheibe erfahren, ab der die Verbindung bestehen soll.

Die Kanalwunschnachrichten der einzelnen Zeitscheibenkanäle (hier zwei je Zeitscheibe) werden alle zusammengefaßt und in die Verbindungswunschnachricht integriert:

- G erhält das feste Ortsnetzkennzeichen von R.
- Die Kanalkennzeichen der Zeitscheibenkanäle werden systematisch mit einem Pseudozufallsbitfolgengenerator (PZBFG) erzeugt, wozu R an G einen Startwert s_{RG} des PZBFG schickt.

  Bezeichnet $s_{RG}(x)$ das x-te aus s_{RG} abgeleitete Kennzeichen, so wird für die Zeitscheibe $(t_0 + t)$, $t = 0, 1, \ldots$, in Richtung $R \rightarrow G$ das Kennzeichen $s_{RG}(2 \bullet t+1)$ und in Richtung $G \rightarrow R$ das Kennzeichen $s_{RG}(2 \bullet t+2)$ verwendet.
- R und G verwenden für alle Zeitscheibenkanäle einen einzigen Schlüssel k_{RG} zur Ende-zu-Ende-Verschlüsselung (was außer R und G aber niemand feststellen kann, also auch nichts schadet). Auch k_{RG} kann aus s_{RG} abgeleitet werden.

Alternativ könnte man auch einen von s_{RG} unabhängigen Schlüssel k_{RG} mitschicken, oder für die beiden Richtungen $R \rightarrow G$ und $G \rightarrow R$ zwei getrennte Startwerte des PZBFG verwenden. Dies würde aber die von der OVSt von *G* zu *verteilende* Verbindungswunschnachricht nur unnötig verlängern.

Im korrekten Fall baut *R* ab Zeitscheibe t_0 nun wie angekündigt einen ZS- und ZE-Kanal mit den oben angegebenen aus s_{RG} abgeleiteten Kennzeichen auf.

Im Idealfall wird der Kanal zwischen den Anschlüssen *R* und *G* geschaltet, indem *G* ebenfalls ab t_0 seinen ZS- und ZE-Kanal passend aufbaut (Bild 5-45).

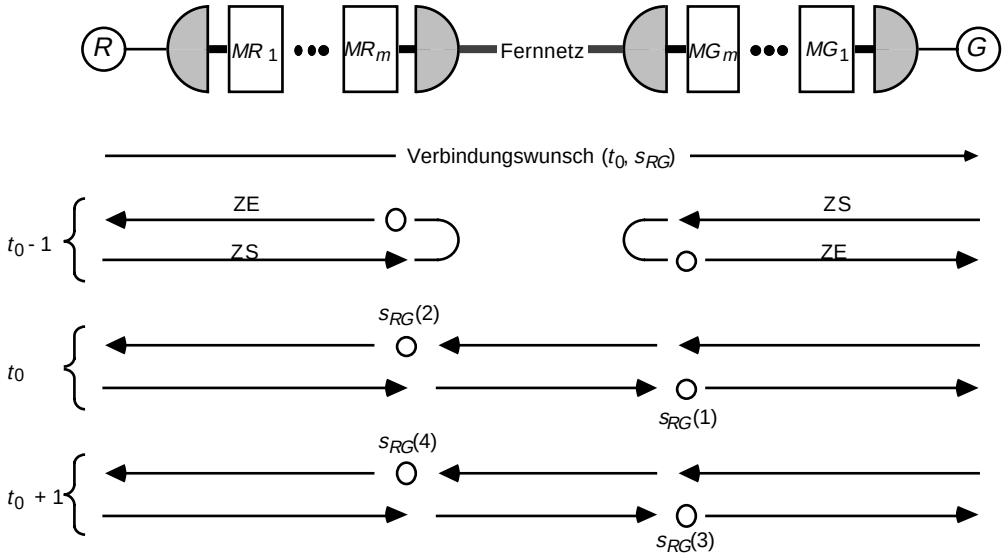


Bild 5-45: Verbindungsaufbau im einfachsten Fall. Das Senden der Startnachrichten, d.h. der eigentliche Beginn der Verbindung, wurde nicht dargestellt

Im Gegensatz zum ISDN, wo die gesamte Signalisierung außerhalb des 64-kbit/s-Datenkanals erfolgt, erscheint es hier zur Entlastung des Signalisierungskanals sinnvoller, wie im analogen Fernsprechnetz die weitere Signalisierung für den Verbindungsaufbau im noch ungenutzten Datenkanal vorzunehmen:

Über diesen Datenkanal wird zunächst das Zustandekommen der Verbindung von *G* durch eine **Startnachricht** an *R* signalisiert, der sein noch bestehendes Interesse an der Verbindung ggf. ebenfalls mit einer Startnachricht an *G* signalisiert. (Damit nicht durch das Auffüllen eines partnerlosen ZE-Kanals durch einen MIX zufällig eine Startnachricht entsteht, müssen diese hinreichend lang gewählt sein.)

Daran anschließend kann die Verbindung zwischen den Endgeräten hergestellt werden, d.h. für den Fernsprechdienst könnte nun das Telefon des Gerufenen anfangen zu klingeln, das Telefon des Rufenden könnte anfangen, Freizeichen von sich zu geben. Die notwendige Signalisierung für den Übergang von „Warten auf Partner“ zu „Partner am Apparat“ u.ä. erfolgt wiederum über den noch ungenutzten Datenkanal.

Bauen *R* und *G* ihre Zeitscheibenkanäle nicht synchron auf, z.B. weil *G* keinen ungenutzten Datenkanal zur Verfügung hat, so schadet dies der Unbeobachtbarkeit nichts, da ja ein fehlender ZS-Kanal durch wenigstens einen der MIXe aufgefüllt wird. ZS-Kanäle ohne passenden ZE-Kanal werden vom letzten MIX des beabsichtigten Empfängers ignoriert.

Damit kann der Verbindungsaufbau völlig asynchron erfolgen. Hierzu muß *G* alle eingegangenen Verbindungswünsche zumindest für eine bestimmte Zeit speichern.

Ist *G* besetzt (d.h. ist er bereits vollständig mit echten Verbindungen ausgelastet, die der Teilnehmer auch nicht abbrechen will) oder nicht empfangsbereit, so kann der rufende Teilnehmer seinen

Netzabschluß *R* im Prinzip beliebig lange auf *G* warten lassen. Um aber zu vermeiden, daß *G*, nachdem *R* seinen Wunsch längst aufgegeben hat, sinnlos einen Kanal durchs Fernnetz zur OVSt von *R* aufbauen läßt, wird die maximale Wartezeit von *R* durch eine Zeitschranke beschränkt. Kann der rufende Teilnehmer diese Schranke beliebig wählen und sie in der Verbindungswunschnachricht *G* mitteilen, so entfällt damit die im Verfahren der Telefon-MIXe den Signalisierungskanal arg belastende Wahlwiederholung durch dauerndes Senden von Verbindungswunschnachrichten. Bei längeren Zeitschranken muß hierzu nicht unbedingt der Teilnehmer persönlich am Endgerät warten, sondern es ist denkbar, ihm das Zustandekommen der Verbindung zwischen den Anschlüssen wiederum durch ein Klingelzeichen anzuzeigen.

Wartet *R* trotzdem nicht, d.h. bricht den Verbindungsaufbau ab, und baut *G* später seine Kanalhälfte doch noch auf, so bleibt die Startnachricht von *R* natürlich aus und *G* bricht den Verbindungsaufbau ebenfalls ab.

Wartet *G* sehr lange, bevor er den Verbindungswunsch von *R* befriedigt, so muß er, um die aktuellen Kennzeichen erzeugen zu können, zuerst alle vorangegangenen erzeugen. Statt der linearen Anordnung der Kennzeichen kann man diese aber auch baumförmig so anordnen (die genaue Konstruktion ist in [GoGM_86] beschrieben), daß sich jedes Kennzeichen $s_{RG}(x)$ statt in $|x|$ Schritten in $\log(|x|)$ Schritten erzeugen läßt [GoGM_86]. Steht ausreichend Speicherplatz zur Verfügung, um jeweils einen Zweig des Baumes vollständig abzuspeichern, so sind zur sequentiellen Erzeugung im Mittel weniger als zwei Schritte erforderlich, d.h. im Normalfall verdoppelt sich der Aufwand.

Wartet einer der beiden Anschlüsse auf den anderen und sind sie nicht beide an dieselbe OVSt angeschlossen, so würde im obigen Schema das Fernnetz durch noch ungenutzte Datenkanäle unnötig belastet. Dies kann mit einer geringen Einbuße an Unbeobachtbarkeit vermieden werden, wenn man den jeweils letzten MIXen bzw. den OVSt'n ermöglicht, die Zusammengehörigkeit aller Zeitscheibenkanäle einer Verbindung innerhalb des Fernnetzes zu erkennen.

Hierzu verwenden *R* und *G* für alle ZE- und ZS-Kanäle ihrer Verbindung im Falle eines Ferngesprächs ein *gemeinsames* Kennzeichen.

Der (nicht sehr große) Nachteil ist, daß dadurch die genaue Dauer der einzelnen Ferngespräche erkennbar wird. Bestehen zwischen zwei bestimmten OVSt'n innerhalb eines längeren Zeitraums nur wenige Verbindungen, so ist dies aber auch im bisher beschriebenen Verfahren der Fall: Ein Angreifer kann mit hoher Wahrscheinlichkeit annehmen, daß alle unmittelbar aufeinanderfolgenden Zeitscheibenkanäle zwischen diesen beiden OVSt'n genau einer Verbindung entsprechen.

Der Vorteil ist dafür, daß in diesem Fall der letzte MIX in der Kaskade des Senders bzw. die OVSt bedeutungslose Daten im Fernnetz unterdrücken können. Dazu werden die Startnachrichten von *R* und *G* jeweils für MR_m bzw. MG_m erkennbar übermittelt. Erst bei Eintreffen der Startnachricht von *G* läßt MG_m den Kanal im Fernnetz aufbauen. Die ZE-Kanäle werden bis zum Eintreffen des richtigen ZS-Kanals wie üblich von MR_m bzw. MG_m mit bedeutungslosen Daten aufgefüllt.

Wird die Verbindung nicht sofort zur Übermittlung von Daten genutzt, etwa weil es im Fernsprechdienst nach dem Verbindungsaufbau zwischen den Endgeräten erst einige Zeit beim Gerufenen klingeln muß, bevor er den Hörer abnimmt, so können auch hier bedeutungslose Daten unterdrückt werden. Hierzu muß das Signal für „Verbindung zum *Teilnehmer* bereit“ den letzten MIXen MG_m bzw. MR_m wie schon die Startnachrichten über den Datenkanal im Klartext übermittelt werden.

Verbindungswunschnachrichten müssen im Ortsnetz des Empfängers *verteilt* werden. Damit ist es aber möglich, daß durch einen koordinierten Angriff vieler Teilnehmer ein bestimmtes Ortsnetz durch Verbindungswünsche überflutet wird, so daß die Verbindungswünsche anderer Teilnehmer nicht mehr verteilt werden können (Angriff auf die Verfügbarkeit, *denial of service attack*). Dieser Angriff ist analog in jedem mehrstufigen Netz, also auch im ISDN, möglich. Da der Angreifer nichts Unzu-

lässiges tut, kann der Angriff auch nicht verhindert, sondern höchstens erkannt oder in seiner Wirkung begrenzt werden.

Letzteres kann erreicht werden, indem man die Verbindungswunschnachrichten mit Prioritäten versieht, und jedem Teilnehmer lediglich eine begrenzte Anzahl von Verbindungswünschen höherer Prioritäten erlaubt. Die Verteilung der Verbindungswünsche erfolgt sortiert nach ihrer Priorität. Die Zuordnung von Prioritäten zu Verbindungswünschen erfolgt adaptiv durch den Netzabschluß: Jeder Verbindungswunsch erhält zunächst die geringste Priorität. Wird der Verbindungswunsch nicht beantwortet und nach einer gewissen Zeit wiederholt, so erhält er die nächsthöhere noch verfügbare Priorität.

Die Einhaltung der zulässigen Anzahl von Verbindungswünschen höherer Priorität je Teilnehmer wird von den OVSt'n anhand nur einmal gültiger „Berechtigungsmarken“ kontrolliert. Eine solche Berechtigungsmarke ist eine von der OVSt digital signierte Nachricht, die die Priorität angibt. Jeder Teilnehmer erhält für jede höhere Priorität nur eine begrenzte Anzahl von Berechtigungsmarken je Zeiteinheit (z.B. Woche). Sie können in Zeiten geringer Netzauslastung durch die OVSt verschickt werden. Enthält ein Verbindungswunsch keine Berechtigungsmarke, so wird ihr die geringste Priorität zugeordnet.

Nun könnte allerdings die OVSt durch Vergleich der erhaltenen mit den ausgegebenen Berechtigungsmarken die Sender der Verbindungswünsche identifizieren. Um dies zu vermeiden, werden zur Unterzeichnung der Berechtigungsmarken **blind geleistete Signaturen** [Cha8_85, Chau_89] verwendet. Diese erlauben es den Teilnehmern, jede von der OVSt signierte Berechtigungsmarke genau einmal so umzuformen, daß sie nach wie vor von der OVSt signiert ist, aber von dieser nicht mehr wiedererkannt werden kann, vgl. §3.9.5 und §6.4.1.

Damit enthält eine Verbindungswunschnachricht für MR_i im Feld D_i

- einen aktuellen Zeitstempel (d.h. die Nummer der aktuellen Zeitscheibe)

und für MR_m zusätzlich

- den Hinweis, ob die Verbindungswunschnachricht bedeutungslos ist und ignoriert werden kann,
- das Ortsnetzkennzeichen des Empfängers,
- eine Berechtigungsmarke.

Die Bestandteile einer Verbindungswunschnachricht sind in Bild 5-46 zusammengefaßt.

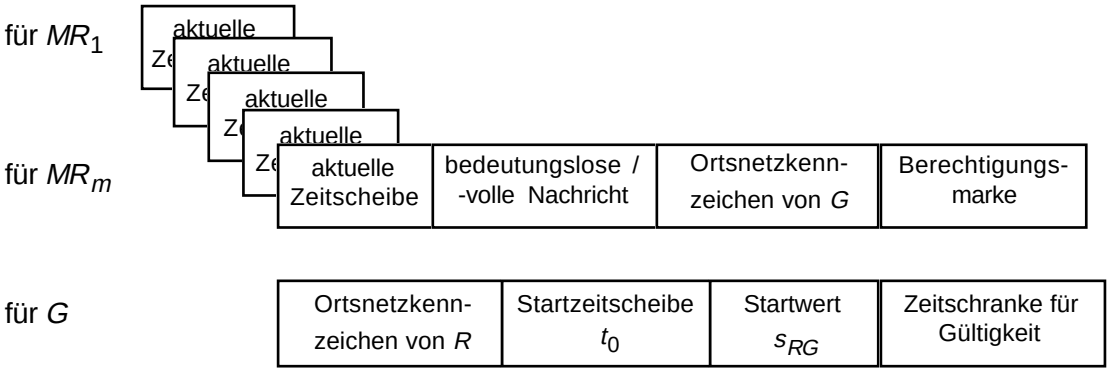


Bild 5-46: Verbindungswunschnachricht von R an G

5.5.1.4 Abbau einer unbeobachtbaren Verbindung

Die Verbindung kann von jedem der beiden Partner geordnet abgebaut werden. Der Abbau erfolgt, indem die Partner ab einer bestimmten Zeitscheibe ihre ZS- und ZE-Kanäle nicht mehr miteinander verbinden, sondern wieder „kurzschließen“ oder für eine neue Verbindung nutzen.

Bei strikter Beibehaltung der Transparenz gibt es im wesentlichen zwei Möglichkeiten, wie einem Partner ermöglicht werden kann, dem anderen den Verbindungsabbauwunsch zu signalisieren.

Zum einen kann der abbauende Netzabschluß eine eigenständige Verbindungsabbaunachricht über den Signalisierungskanal an seinen Partner senden. Da der Signalisierungskanal aber ohnehin schon sehr belastet ist, scheint diese „natürliche“ Lösung nicht sehr sinnvoll.

Zum anderen kann man aber auch einen Teil des Signalisierungskanals, genauer 1 bit je Zeitscheibe, fest dem Datenkanal zuordnen, d.h. dieser 1-bit/z-Kanal würde zusammen mit dem Datenkanal über ZS- und ZE-Kanäle übermittelt. Der Verbindungsabbau könnte z.B. durch Übermittlung einer '1' auf diesem Kanal signalisiert werden.

5.5.1.5 Abrechnung der Netzbenutzung

Die bereits in [PfPW_88] genannten Möglichkeiten zur Abrechnung der Netz- und Dienstbenutzung ohne Identifikation der Teilnehmer sind natürlich auch hier anwendbar.

Für Gespräche innerhalb eines Ortsnetzes erscheint eine Abrechnung von Einzelgesprächen technisch sinnlos, da sie im Netz keinen zusätzlichen Aufwand verursachen. Zudem sind solche Gespräche völlig unbeobachtbar.

Soll dennoch eine benutzungsabhängige Abrechnung erfolgen, so kann diese unter Beibehaltung der völligen Unbeobachtbarkeit von Ortsgesprächen durch lokale Entgeltzähler bei den Teilnehmeranschlüssen selbst vorgenommen werden (die natürlich nur in fest vorgegebenen Intervallen, z.B. monatlich, vom Betreiber ausgelesen werden dürfen und können).

Wird die Unbeobachtbarkeit von Ortsgesprächen eingeschränkt, so können aber auch dieselben Abrechnungstechniken wie für das Fernnetz verwendet werden.

Zur Abrechnung von Verbindungen, die das Fernnetz benutzen (und erst damit echten Mehraufwand verursachen), können den entsprechenden Kanalaufbaunachrichten in den Daten für den jeweils letzten MIX Entgeltmarken (von derselben Art wie die Berechtigungsmarken in §5.5.1.3) beigelegt werden. Kann eine OVSt den Zusammenhang aller Zeitscheibenkanäle eines Ferngesprächs erkennen (vgl. §5.5.1.3), so ist es wie bisher möglich, daß erst bei Abnahme beider Hörer bezahlt werden muß, und es genügt, wenn z.B. in den ZS-Kanalaufbaunachrichten bezahlt wird. Auch muß in diesem Fall nicht jede Zeitscheibe einzeln bezahlt werden.

In Bild 5-43 wurden die zur Abrechnung notwendigen Daten willkürlich den ZS-Kanalaufbaunachrichten zugeordnet. Für ZS-Kanäle, die nicht über das Fernnetz übermittelt werden müssen, ist das Feld für die Entgeltmarke leer, aber natürlich vorhanden.

Die Entgeltmarken können mit Hilfe eines die Unbeobachtbarkeit wahrenden digitalen Zahlungssystems (§6 sowie [PWP_90, BüPf_87, BüPf_89]) jederzeit von den Teilnehmern bei deren OVSt eingekauft werden. Der notwendige Nachrichtenaustausch geschieht dabei über eine normale unbeobachtbare Verbindung.

Geht man davon aus, daß für eine Verbindung über das Fernnetz für alle Zeitscheibenkanäle ein gemeinsames Kennzeichen verwendet wird, so kann wahlweise der Rufende oder der Gerufene bezahlen: Wer die Verbindung bezahlen wird, kann in der Verbindungswunschnachricht vom Rufenden für den Gerufenen und MR_m angegeben werden. MR_m teilt MG_m mit, wer bezahlen soll. Nach Austausch der Startnachrichten, die das Zustandekommen der Verbindung zwischen den Endgeräten sig-

nalisiert, akzeptiert der letzte MIX des Zahlenden nur noch ZS-Kanalaufbaunachrichten mit gültiger Entgeltmarke und baut bei Fehlen der Marken die Verbindung ab.

Statt der jeweils letzten MIXe können natürlich auch die OVSt'n von *R* und *G* diese Abrechnung übernehmen, wenn sie von den MIXen die jeweils notwendigen Daten (Verbindungskennzeichen, Angabe wer bezahlt, Entgeltmarke) erhalten.

5.5.1.6 Verbindungen zwischen OVSt mit MIX-Kaskade und herkömmlicher OVSt

Für Verbindungen zwischen OVSt'n mit MIX-Kaskade und herkömmlichen OVSt'n muß der letzte MIX in der MIX-Kaskade der erweiterten OVSt die Rolle eines Umsetzers spielen.

Verfügt nur die OVSt von *R* über eine MIX-Kaskade, so teilt *R* dem MIX MR_m in einer speziellen Verbindungswunschnachricht die Rufnummer von *G* und all die Daten mit, die im normalen Schema *G* erhielte, d.h. den für die Verbindung verwendeten Startwert s_{RG} und die Nummer der ersten Zeitscheibe. MR_m baut über seine OVSt und das Fernnetz eine herkömmliche Verbindung zu *G* auf und verknüpft diese mit den Zeitscheibenkanälen von *R*.

Der rufende Teilnehmer ist nun ebenso unbeobachtbar wie im Fall, daß auch bei *G* eine MIX-Kaskade ist. Der gerufene Teilnehmer ist so beobachtbar wie heute und im ISDN.

Verfügt nur die OVSt von *G* über eine MIX-Kaskade, so stellt sich in erster Linie die Frage der geeigneten Adressierung.

Verwendet der rufende Teilnehmer eine Rufnummer im heutigen Sinne, so ist die Unbeobachtbarkeit des gerufenen Teilnehmers damit zwangsläufig verloren. Daher sollte man auch für solche Verbindungen implizite Adressen vorsehen.

Im folgenden wird sich herausstellen, daß eine Anonymitätsgruppe aus jeweils ungefähr 5000 Teilnehmern bestehen kann. Nimmt man an, daß jeder Teilnehmer ungefähr 50 verschiedene offene implizite Adressen verwendet (vgl. §5.4.1.2), und daß, um Kollisionen zu verhindern, die Menge der möglichen Adressen ungefähr quadratisch in der Anzahl der benötigten sein soll, so kommt man auf einen Adreßraum von $6{,}25 \cdot 10^{10}$, d.h. eine offene implizite Adresse käme mit 11 Dezimalziffern aus. Hinzu kommt allerdings noch die Auswahl der Anonymitätsgruppe von *G*. Nimmt man an, daß das größte Ortsnetz sicher weniger als 50 000 000 Anschlüsse hat und jede Anonymitätsgruppe 5000 Teilnehmer, so kommt man mit 4 Dezimalziffern aus.

Damit liegt für solche Verbindungen folgendes Schema nahe: *R* wählt zunächst wie heute die Vorwahl des Ortsnetzes von *G* und stellt damit eine herkömmliche Verbindung zu MG_m her. Durch 15 weitere Dezimalziffern teilt er MG_m die implizite Adresse von *G* mit. MG_m spielt Stellvertreter und sendet *G* eine entsprechende Verbindungswunschnachricht. Nimmt *G* die Verbindung an, so verknüpft MG_m wie oben die herkömmliche und die datenschützende Verbindungshälfte. Nimmt *G* die Verbindung nicht an, so kann MG_m zur Vermeidung von Wahlwiederholungen an *R* das Angebot zurücksenden, den Verbindungswunsch aufrecht zu erhalten und ihn bei Annahme des Wunsches durch *G* zurückzurufen.

Eine verdeckte implizite Adresse wird hingegen aus mindestens 200 Dezimalziffern bestehen und setzt zudem beim Rufenden gewisse kryptographische Fähigkeiten voraus: Der Rufende benötigt einen Rechner zur Speicherung von Schlüsseln und Berechnung von Adressen, und muß in der Lage sein, Daten von diesem Rechner an MG_m übermitteln zu können, sei es über das ISDN oder über das herkömmliche Fernsprechnetz mittels Modem oder Akustikkoppler. Die Übermittlung erfolgt innerhalb der Verbindung zwischen *R* und MG_m, d.h. nicht im Signalisierungskanal.

Es sei angemerkt, daß an einer OVSt, die eine MIX-Kaskade hat, auch herkömmliche Teilnehmeranschlüsse beibehalten werden können.

5.5.1.7 Zuverlässigkeit

Offensichtlich stellen die MIXe einer Kaskade hinsichtlich der Zuverlässigkeit ein *Seriensystem* dar. Hier würde es aber wohl genügen, jeden MIX intern fehlertolerant auszulegen. Falls trotzdem ein MIX ausfallen sollte, können die an diese OVSt angeschlossenen Netzabschlüsse in allen Nachrichten diesen MIX auslassen.

Es gibt auch sich über mehrere MIXe erstreckende Fehlertoleranzverfahren, vgl. §5.4.6.10 und [Pfi1_85 §3.1, MaPf_87, Pfit_89 §5.3]. Diese erhöhen aber den Aufwand, und das Problem der anonymen Rückadressen, das diese Verfahren vor allem nötig macht, tritt im hier beschriebenen Verfahren nicht auf.

Hinsichtlich der Sicherung vor Übertragungsfehlern ändert sich gegenüber dem ISDN nichts. Auch hier wird nur eine streng synchrone, aber ansonsten ungesicherte Übertragung angenommen.

Durch Verwendung einer geeigneten synchronen Stromchiffre (Pseudo-one-time-pad; Betriebsart OFB) wird eine *zusätzliche* Ausbreitung von Bitverfälschungen auf den 64-kbit/s-Kanälen verhindert. Auf dem Signalisierungskanal wird die Verwendung entsprechender Protokolle, z.B. HDLC, zur Sicherung der Übertragung gegen Fehler angenommen.

5.5.2 Aufwand

Im folgenden soll anhand einer sehr einfachen Leistungsabschätzung die Praktikabilität der in den vorhergehenden Abschnitten beschriebenen Telefon-MIXe demonstriert werden.

Es wird sich zeigen, daß bei Beschränkung auf 2 • 64 + 16 kbit/s Duplexkanäle für $m = 10$ MIXe ungefähr 5000 Netzabschlüsse an eine MIX-Kaskade angeschlossen werden können.

Als Hauptengpässe werden sich herausstellen

- die Übertragung der ZS- und ZE-Kanalaufbaunachrichten, Verbindungswunschnachrichten und Rückmeldungen von den Netzabschlüssen an die OVSt sowie
- die Verteilung der Verbindungswunschnachrichten an die Netzabschlüsse.

Beides muß über den 16-kbit/s-Signalisierungskanal erfolgen. Da ein Teil seiner Bandbreite für die Sicherung gegen Übertragungsfehler verwendet werden muß und möglicherweise auch einige schmalbandigere Dienste über den Signalisierungskanal abgewickelt werden, soll im folgenden lediglich mit einer verfügbaren Bandbreite zur Signalisierung von $b = 12$ kbit/s in beiden Richtungen gerechnet werden.

Die Vorbereitung und Verarbeitung der Nachrichten durch die Netzabschlüsse, das Mixen und die eigentlichen Datenkanäle stellen demgegenüber ein geringeres Problem dar.

In §5.5.2.1 werden zunächst einige Parameter zur Beschreibung der benötigten kryptographischen Systeme eingeführt und typische Werte angegeben. In §5.5.2.2 wird die Dauer z einer Zeitscheibe abgeschätzt, und in §5.5.2.3 die Anzahl n der Netzabschlüsse, die an eine MIX-Kaskade angeschlossen werden können. In §5.5.2.4 wird die Komplexität der von den Netzabschlüssen und MIXen je Zeitscheibe durchzuführenden Aufgaben betrachtet, und in §5.5.2.5 schließlich die Verbindungsaufbauzeit abgeschätzt.

Zur Erinnerung und Anpassung der Notation sei das Bildungsschema der Nachrichten gegeben:

Angenommen, ein Teilnehmer möchte über seinen Netzabschluß A die (bereits Ende-zu-Ende-verschlüsselte) Nachricht N an den Netzabschluß B eines anderen Teilnehmers senden, und zusätzlich jedem MIX M_i gewisse Daten D_i zukommen lassen. Dann bildet er nacheinander die m+1 Nachrichten

$$\begin{aligned} N_{m+1} &:= N \\ N_i &:= c_i^*(D_i, N_{i+1}) \text{ für } i = m, m\text{-}1, \ldots, 2, \end{aligned} \qquad (1)$$

$$N_1 \quad := k_{1A}(D_1, N_2)$$

und sendet N_1 an M_1. N_1 wird als **MIX-Eingabenachricht** bezeichnet.

Jeder M_i erhält von M_{i-1} bzw. von A die Nachricht N_i; die Umcodierung von N_i besteht in der Entschlüsselung mit k_{1A} bzw. d_i und dem Beseitigen von D_i.

Für M_1 könnte natürlich statt des symmetrischen ebenfalls das hybride Kryptosystem verwendet werden, was jedoch den Aufwand zur Bildung von N_1 nur unnötig erhöhen und die Sicherheit in gewissen Fällen verschlechtern würde.

5.5.2.1 Parameter zur Beschreibung der benötigten kryptographischen Systeme

Die im folgenden verwendeten Parameter orientieren sich an den jeweils bekanntesten Vertretern der symmetrischen bzw. asymmetrischen kryptographischen Systeme, nämlich DES und RSA, vgl. §3.7 und §3.6.

Wiederholt (vgl. §5.4.6.8) und bekräftigt sei an dieser Stelle, daß die Sicherheit dieser kryptographischen Systeme und damit auch die jeder darauf basierenden MIX-Implementierung bislang noch nicht einmal relativ zu einem als schwer erachteten Problem, z.B. der Faktorisierung, *bewiesen* wurde.

Dieser Nachteil wird hier nur deshalb ignoriert, weil einerseits dasselbe auf alle praktikablen Konzelationssysteme zutrifft, andererseits DES und RSA zumindest als praktisch gut untersucht und im öffentlichen Bereich als ungebrochen gelten.

Die Parameter und ihre angenommenen Werte sind in Bild 5-47 zusammengefaßt.

Parameter	symmetrische Stromchiffre	asymmetrische Blockchiffre
Schlüssellänge	s_{sym} = 128 bit	irrelevant
Blocklänge	$b_{sym} = \infty$	b_{asym} = 660 bit
Länge der Verschlüsselung von N	$\|N\|$	b_{asym} (falls $\|N\| \le b_{asym}$)

Bild 5-47: Parameter der verwendeten kryptographischen Systeme

5.5.2.1.1 Symmetrisches Konzelationssystem

Im folgenden wird von einer sicheren Blockchiffre mit 128 bit Schlüssellänge ausgegangen, z.B. einer Verallgemeinerung von DES mit anderer Teilschlüsselerzeugung, vgl. §3.7.7.

Für das hier vorgestellte Verfahren benötigt man eine synchrone Stromchiffre, d.h. die symmetrische Blockchiffre wird im OFB-Modus betrieben, vgl. §3.8.2.4. Es entsteht keine Expansion durch Auffüllen von Blöcken oder ähnlichem.

5.5.2.1.2 Asymmetrisches Konzelationssystem

Für das asymmetrische kryptographische System RSA erscheint zu Zeit eine Moduluslänge von 660 bit als ausreichend, vgl. §3.6.

5.5.2.1.3 Hybride Verschlüsselung

Um Übertragungskapazität zu sparen, wird bei der hybriden Verschlüsselung stets ein möglichst großer Teil N' des Klartextes N in den ersten, asymmetrisch verschlüsselten, Block hineingezogen, d.h. es ist, wenn N'' der Rest von N ist,

$$c_E^*(N) := \begin{cases} c_E(N) & \text{falls } |N| \le b_{asym} \\ c_E(k_{SE},N'),\ k_{SE}(N'') & \text{sonst.} \end{cases} \qquad (2)$$

Zur Bestimmung der Länge von $c_E^*(N)$ ist zu beachten, daß zur Gewährleistung der Unbeobachtbarkeit die in §5.4.6.2 verwendete Verschlüsselung *indeterministisch* sein muß.

Bei Verwendung von RSA muß daher dessen Determinismus beseitigt werden, indem die Nachricht vor der Verschlüsselung indeterministisch codiert, d.h. im wesentlichen um eine Anzahl a zufällig gewählter Bits erweitert wird.

Die dadurch verursachte *Längenexpansion* kann sinnvoll genutzt werden, indem eben jene a zusätzlichen Bits als symmetrischer Schlüssel k_{SE} verwendet werden. Damit ist $a = s_{sym}$.

Die genaue Realisierung der „Erweiterung“ ist leider nicht trivial. So wurde gezeigt, daß speziell bei Verwendung von RSA die in [Chau_81] nahegelegte Codierung durch Voranstellen zufällig gewählter Bits ein unsicheres MIX-Netz ergibt, vgl. §5.4.6.8.

Ungebrochen ist jedoch folgende Erweiterung, von der im folgenden auch ausgegangen werden soll: Vor der Verschlüsselung einer Nachricht wählt sich der Verschlüßler einen zufälligen, s_{sym} bit langen String und konkateniert ihn mit der Nachricht. Die so erweiterte Nachricht wird mit einer festen und möglichst wenig strukturerhaltenden Permutation π des Klartextraumes „verwürfelt“. Die Permutation π und ihre Inverse π^{-1} sind allgemein bekannt. (Sie könnten z.B. durch einen festen Schlüssel eines symmetrischen Konzelationssystems beschrieben werden.) Beim Entschlüsseln wird umgekehrt verfahren, d.h. zunächst die inverse Permutation π^{-1} angewandt und dann der s_{sym} bit lange String entfernt.

Geht man von dieser Erweiterung aus, so ist

$$|c_E^*(N)| = \max\ \{\ b_{asym},\ |N| + s_{sym}\ \}. \qquad (3)$$

5.5.2.2 Minimale Dauer einer Zeitscheibe

Jeder Netzabschluß muß für jede Zeitscheibe und jeden ihm zur Verfügung stehenden 64-kbit/s-Datenkanal drei MIX-Eingabenachrichten senden: eine ZS- und eine ZE-Kanalaufbaunachricht sowie eine Verbindungswunschnachricht. Deren Längen seien mit L_{ZS}, L_{ZE} bzw. L_V bezeichnet.

Die Länge einer Rückmeldung (vgl. [PfPW1_89 §2.2.3]) wird mit L_R bezeichnet. Wird RSA als Signatursystem verwendet, so gilt im wesentlichen $L_R = b_{asym}$, was im folgenden auch angenommen werden soll. Da die Rückmeldungen ob ihrer kurzen Länge ohnehin kaum ins Gewicht fallen, wird angenommen, daß jeder Netzabschluß für jede Zeitscheibe eine Rückmeldung sendet.

Ein Netzabschluß muß also in der Lage sein, über seinen Signalisierungskanal in jeder Zeitscheibe $2 \cdot (L_{ZS} + L_{ZE} + L_V) + L_R$ bit zu übertragen, d.h. es muß gelten

$$z \ge \frac{2 \cdot (L_{ZS} + L_{ZE} + L_V) + L_R}{b}\ . \qquad (4)$$

Die Längen L_{ZS}, L_{ZE} und L_V sollen im folgenden abgeschätzt werden. In §5.5.2.2.1 wird hierzu die Länge einer MIX-Eingabenachricht allgemein abgeschätzt, in §5.5.2.2.2 werden die speziellen Werte in Gleichung (4) eingesetzt.

5.5.2.2.1 Länge einer MIX-Eingabenachricht

Gemäß §5.5.2 (1) gilt

$$|N_1| \le |D_1| + |N_2|$$

bzw. für $i = 2, \ldots, m$ gemäß §5.5.2.1.3 (3)

$$|N_i| = \max \{ b_{asym}, |D_i| + |N_{i+1}| + s_{sym} \}.$$

Da für $i < m$ die Nachricht N_{i+1} wenigstens aus einem asymmetrisch verschlüsselten Block besteht, gilt für $i < m$ stets $|N_{i+1}| \ge b_{asym}$. Damit folgt nach Auflösung der obigen rekursiven Gleichungen

$$|N_1| = \sum_{j=1}^{m-1} |D_j| + (m\text{-}2) \bullet s_{sym} + \max \{ b_{asym}, |D_m| + |N_{m+1}| + s_{sym} \}. \qquad (5)$$

Im folgenden soll stets von Gleichung (5) ausgegangen werden.

5.5.2.2.2 Abschätzung

Die Strukturen der drei interessierenden Nachrichtentypen sind in den Bildern 5-43, 5-44 und 5-46 dargestellt. Den Feldern sind folgende Längen zugeordnet worden:

Bild 5-43, 5-44, 5-46:

- Zeitscheibennummer: 30 bit
- Ortsnetzkennzeichen: 22 bit

Bild 5-43, 5-44:

- Entgeltmarke zur Bezahlung der Fernnetzbenutzung: 670 bit
- Kanalkennzeichen: 28 bit

Bild 5-46:

- Unterscheidung bedeutungslose Nachricht / bedeutungsvolle Nachricht: 1 bit
- Startwert s_{RG} des PZBFG: 128 bit
- Zeitschranke für Gültigkeit (d.h. das Warten von R auf G): 20 bit
- Berechtigungsmarke zur Angabe der Priorität eines Verbindungswunsches: 670 bit

Für die beiden Marken wurde jeweils angenommen, daß die Priorität der Berechtigungsmarke bzw. der Wert der Entgeltmarke nicht nur implizit in der Signatur (bei RSA: 660 bit) durch unterschiedliche Signaturen, sondern auch explizit (in weiteren 10 bit) angegeben wird. Dies erspart der kontrollierenden OVSt das Durchprobieren aller möglichen Schlüssel zum Prüfen von Signaturen.

Für eine ZS- oder ZE-Kanalaufbaunachricht gilt für $i = 1, \ldots, m$-1 jeweils $|D_i| = 30$ bit, da der Schlüssel des symmetrischen Konzelationssystems bereits für die hybride Verschlüsselung gezählt wurde, und $|N_{m+1}| = 0$.

Für eine ZS-Kanalaufbaunachricht ist $|D_m| = 750$ bit, d.h. nach (5) $|N_m| = 750$ bit + s_{sym}. Für eine ZE-Kanalaufbaunachricht ist $|D_m| = 58$ bit, d.h. nach (5) $|N_m| = b_{asym}$.

Damit folgt aus (5)

$$L_{ZS} = (m\text{-}1) \bullet (30 \text{ bit} + s_{sym}) + 750 \text{ bit}$$
$$L_{ZE} = (m\text{-}1) \bullet (30 \text{ bit} + s_{sym}) - s_{sym} + b_{asym}. \qquad (6)$$

Für eine Verbindungswunschnachricht gilt für $i = 1, \ldots, m$-1 jeweils $|D_i| = 30$ bit, und $|D_m| = 723$ bit. Nach (5) ist $|N_m| = 723$ bit + $|N_{m+1}| + s_{sym}$. Umgekehrt reicht $|N_{m+1}| = b_{asym}$ aus, da der Empfänger lediglich 200 bit erhalten muß.

Damit ist

$$L_V = (m\text{-}1) \bullet (30 \text{ bit} + s_{sym}) + b_{asym} + 723 \text{ bit}. \qquad (7)$$

Gemäß (6) und (7) ist

$$L_{ZS} + L_{ZE} + L_V = (m\text{-}1) \bullet 90 \text{ bit} + (3 \bullet m\text{-}4) \bullet s_{sym} + 2 \bullet b_{asym} + 1473 \text{ bit}. \quad (8)$$

Setzt man nun die angenommenen Werte für s_{sym}, b_{asym} (vgl. Bild 5-47) und b = 12 kbit/s in (4) ein, so erhält man

$$z \geq \frac{474 \text{ bit} \bullet m + 2191 \text{ bit}}{6000 \text{ bit/s}} + \frac{660 \text{ bit}}{12000 \text{ bit/s}} = 0{,}079 \text{ s} \bullet m + 0{,}42016\overline{6} \text{ s}, \quad (9)$$

wozu für m = 10 ein Wert $z \geq 1{,}22$ s hinreichend ist.

5.5.2.3 Maximale Anzahl der von einer MIX-Kaskade bedienbaren Netzabschlüsse

Die Anzahl n der von einer MIX-Kaskade bedienbaren Netzabschlüsse wird im wesentlichen durch die Verteilung der Verbindungswunschnachrichten begrenzt: jedem Netzabschluß steht anteilig lediglich ein Empfangskanal von b/n zur Verfügung.

Wie stark diese Begrenzung ist, hängt von der Verkehrsstatistik ab. Vereinfachend wird angenommen, daß das Verteilnetz ein M/D/1-System sei, wobei λ die maximale mittlere Rate sei, mit der jeder der n Teilnehmer Verbindungswünsche erhält. Als Wert wird λ = 1/300 1/s angenommen, d.h. in Stoßzeiten im Mittel zwölf Verbindungswünsche je Teilnehmer und Stunde.

(Dieser Wert für λ ist sicher ausreichend, eher sogar zu hoch: Laut Fernsprechstatistik [SIEM_87] wurden 1985 in der Bundesrepublik *im Mittel* weniger als 3,1 Gespräche je Hauptanschluß und *Tag* geführt. Vertraut man den Erwartungen von Siemens, so dürfte der Wert λ = 1/300 1/s auch für Spitzenzeiten eher zu hoch sein: Laut [SIEM_88] verarbeitet der leistungsfähigste Vermittlungsrechner von Siemens im Endausbau 4,8 Vermittlungsversuche je Teilnehmer und Stunde. Außerdem interessieren beim Verfahren der Telefon-MIXe für λ sogar nur die ankommenden Gespräche.)

Jede verteilte Verbindungswunschnachricht besteht aus genau einem asymmetrisch verschlüsselten Block (vgl. §5.5.2.2.2) und hat daher die Länge b_{asym}, so daß maximal $\mu := b / b_{asym}$ Verbindungswunschnachrichten pro Sekunde verteilt werden können.

Ist T_v die mittlere Systemzeit, d.h. die Zeit, die ein Verbindungswunsch im Mittel benötigt, um bei den Netzabschlüssen einzutreffen, so gilt [Klei_75 §5.5, insbesondere (5.74)] in Stoßzeiten im Mittel

$$T_v = \frac{2 \bullet \mu - n \bullet \lambda}{2 \bullet \mu \bullet (\mu - n \bullet \lambda)}. \quad (10)$$

Hieraus ergibt sich

$$n = \frac{\mu}{\lambda} \bullet \frac{\mu \bullet T_v - 1}{\mu \bullet T_v - 0{,}5}. \quad (11)$$

Für λ = 1/300 1/s, $T_v \leq 0{,}5$ s, b = 12 kbit/s und b_{asym} = 660 bit ergibt dies $n \leq 5137$.

Speziell für Verbindungswunschnachrichten würde sich die Verwendung der in §5.4.1.2 erklärten offenen impliziten Adressierung lohnen: Da der gerufene Netzabschluß G den rufenden Netzabschluß R bereits anhand der offenen Adresse a_{RG} wiedererkennt, muß der Verbindungswunsch lediglich die Nummer der gewünschten Startzeitscheibe angeben. Alle anderen Angaben kann G „wiederverwenden“. Verwendet der rufende Teilnehmer nicht seinen eigenen Netzabschluß R, sondern ruft von einem anderen Ortsnetz aus an, so kann die Verbindung über R zu dem aktuell verwendeten Netzabschluß umgeleitet werden. Ist eine Adresse 100 bit lang, so ist ein Verbindungswunsch damit maximal 130 bit lang. Werden nur solche Adressen verwendet, so ergibt sich mit $\mu = b$ / 130 bit gemäß (11) und obigen Werten eine Schranke von $n \leq 27389$.

Die M/D/1-Annahme ist, wie erwähnt, natürlich etwas vereinfachend. Allerdings schadet dies hier fast nichts: Die Ankünfte sind näherungsweise „gedächtnislos“, da eine Wahlwiederholung durch Wiederholen der Verbindungswunschnachricht nicht notwendig ist. Die Bedienzeit ist tatsächlich deterministisch, bis auf seltene, aufgrund von Fehlern notwendige Wiederholungen. Die verfügbare Bandbreite braucht natürlich nicht statisch nach oben beschränkt zu werden. Zudem hat bei der hier gemachten Annahme von $T_v \gg 1/\mu$ die Wartezeit ohnehin nur einen recht geringen Einfluß auf n, d.h. (11) ist im wesentlichen die triviale Forderung $n \cdot \lambda \leq \mu$.

5.5.2.4 Berechnungsaufwand der Netzabschlüsse und MIXe

Der Berechnungsaufwand der Netzabschlüsse und MIXe besteht zu einem großen Teil aus dem Anwenden kryptographischer Operationen: Vorbereiten und Erkennen von impliziten Adressen, Vorbereiten und Verarbeiten von MIX-Eingabenachrichten, Signieren und Testen von Rückmeldungen, Ver- und Entschlüsseln von Datenkanälen.

Da aber die Verschlüsselungsraten beider Konzelationssysteme erheblich höher sind als die Übertragungsraten, stellen diese Operationen kein Problem dar.

Der sonstige Aufwand der Netzabschlüsse, z.B. zur Verwaltung von Schlüsseln, impliziten Adressen usw., ist zu vernachlässigen.

Der sonstige Aufwand der MIXe besteht im wesentlichen aus dem Puffern und Umsortieren von Nachrichten sowie dem Testen von Zeitstempeln. Dies betrifft je Zeitscheibe 6 Eingabeschübe mit je n Nachrichten, also mit den Schranken aus §5.5.2.3 weniger als 27389 Nachrichten. Dies sollte, zumindest mit spezieller Hardware, in 0,01 s pro MIX möglich sein. Man kann, um hier noch etwas Zeit zu sparen, auch die Verbindungswunschnachrichten so gestalten, daß sie zunächst einen Sendekanal zu MR_m bzw. der OVSt schalten, über den dann erst die zusätzlichen Daten für MR_m und G übermittelt werden. Dadurch kann der Teilnehmer die Entscheidung, ob und mit wem er eine echte Verbindung wünscht, etwas später treffen.

Generell ist zu beachten, daß die Nachrichten der verschiedenen Zeitscheiben von den MIXen pipeline-artig verarbeitet werden können, also insbesondere die Zeitscheiben bei den verschiedenen MIXen zu verschiedenen Zeitpunkten beginnen können.

5.5.2.5 Verbindungsaufbauzeit

Die Verbindungsaufbauzeit setzt sich im wesentlichen zusammen aus

- dem Warten auf die Übermittlung des Verbindungswunsches durch die MIX-Kaskade des rufenden Netzabschlusses ($\leq z$),
- der Verzögerungszeit der MIX-Kaskade des rufenden Netzabschlusses ($m \cdot 0{,}01$ s),
- der Laufzeit im Fernnetz ($\leq 0{,}2$ s),
- dem Warten auf die Verteilung an den Empfänger (in Spitzenzeiten im Mittel T_v),
- dem Warten auf das Aufbauen des ZS-Kanals durch den gerufenen Netzabschluß ($\leq z$, sofern der gerufene Netzabschluß sofort antwortet), und schließlich
- der Verzögerungszeit der MIX-Kaskade des gerufenen Netzabschlusses ($m \cdot 0{,}01$ s).

Dies ergibt zusammen eine Zeit von $2 \cdot (z + m \cdot 0{,}01 \text{ s}) + T_v + 0{,}2$ s, bei den obigen Werten also ungefähr 3,34 s. Im Mittel wird die Zeit etwa halb so groß sein.

5.6 Netzmanagement

In diesem Kapitel wird zunächst in §5.6.1 diskutiert, wer welche Teile des Kommunikationsnetzes (errichten sowie) betreiben und entsprechend die Verantwortung für die Dienstqualität übernehmen sollte.

Danach wird in §5.6.2 skizziert, wie eine Abrechnung der Kosten für die Benutzung des Kommunikationsnetzes mit dem Betreiber ohne Untergraben von Anonymität, Unbeobachtbarkeit und Unverkettbarkeit sicher und komfortabel erfolgen kann.

Die Bezahlung von über das Kommunikationsnetz erbrachten (höheren) Diensten kann mittels eines beliebigen digitalen Zahlungssystems erfolgen, was in §6 dargestellt wird.

5.6.1 Netzbetreiberschaft: Verantwortung für die Dienstqualität vs. Bedrohung durch Trojanische Pferde

Bezüglich Netzbetreiberschaft und Verantwortung für die Dienstqualität besteht folgendes Dilemma [Pfi1_85 Seite 129]:

Verwendet jeder eine beliebige **Teilnehmerstation** (was man sich bis hierher, da nichts anderes explizit gesagt wurde, vermutlich vorgestellt hat), so wird keine Organisation willens und in der Lage sein, die Verantwortung für die Dienstqualität, d.h. die von den Teilnehmern wahrnehmbare Nutzleistung und Zuverlässigkeit, zu übernehmen. Dies gilt zumindest dann, wenn fehlerhaftes Verhalten oder ungenügende Leistung einer Teilnehmerstation die Dienstqualität für andere Teilnehmer auch dann beeinträchtigen können, wenn diese anderen Teilnehmer nicht gerade mit dieser fehlerhaften oder ungenügend leistungsfähigen Teilnehmerstation kommunizieren wollen (weswegen ab §5.4 nicht mehr von Netzbenutzern, sondern von Netzteilnehmern gesprochen wurde). Ein solches „anarchisch-liberales“ Netzmanagement ist zwar für die (Teilnehmer-)Überprüfbarkeit von Anonymität, Unbeobachtbarkeit und Unverkettbarkeit optimal, wegen seiner unkalkulierbaren Dienstqualität aber nur für spezielle Anwendungen, keinesfalls aber für diensteintegrierende Kommunikationsnetze geeignet. Es kann natürlich auf einem beliebigen Kommunikationsnetz um den Preis geringer Kosteneffizienz, geringer Nutzleistung, geringer Zuverlässigkeit und, vielleicht, großen Mißtrauens durch den Rest der Gesellschaft, was seine Teilnehmer wohl zu verbergen haben, errichtet und betrieben werden.

Entwirft, produziert, errichtet und betreibt eine Organisation andererseits sämtliche Netzkomponenten, insbesondere auch die Teilnehmerstationen, so wird diese Organisation zwar willens und in der Lage sein, die Verantwortung für die Dienstqualität zu übernehmen. Aber niemand, insbesondere kein Teilnehmer, kann ausschließen, daß Teile solch einer Organisation irgendwo Trojanische Pferde installieren, vgl. §1.2.2. Solch eine Organisation wäre also bezüglich Sicherheit, insbesondere Vertraulichkeit, nicht kontrollierbar.

Da jede Schutzmaßnahme, die innerhalb des Kommunikationsnetzes angesiedelt ist, durch die Teilnehmerstationen realisiert (oder mit realisiert, z.B. Abfragen und Überlagern oder MIXe) werden muß, ist es zur Verkleinerung dieses Dilemmas sinnvoll, die Teilnehmerstation bezüglich Entwurf, Produktion, Installation und Betreiberschaft geeignet in zwei Teile zu zerlegen: Das (bzw. die) Teilnehmerendgerät(e) und den Netzabschluß.

Zum **Teilnehmerendgerät** zählen dabei die Teile der Teilnehmerstation, die mit der Kommunikation mit anderen und der anderen Teilnehmerstationen nichts zu tun haben. Folglich kann das Teilnehmerendgerät ausschließlich unter der Kontrolle des einzelnen Teilnehmers stehen.

Zum **Netzabschluß** zählen all die Teile der Teilnehmerstation, die mit der Kommunikation mit anderen oder der anderen Teilnehmerstationen zu tun haben. Folglich ist der Betreiber des Kommunikationsnetzes für den Netzabschluß verantwortlich.

Beispielsweise wird von CCITT bei X.25 das Teilnehmerendgerät DTE (Data Terminal Equipment) und der Netzabschluß DCE (Data Circuit-terminating Equipment) genannt [Tane_81, Tane_88], bei ISDN heißt das Teilnehmerendgerät TE (Terminal Equipment) und der Netzabschluß NT (Network Termination) [ITG_87, Tane_88].

Sofern es einen Netzbetreiber geben und dieser für die Dienstqualität verantwortlich sein soll, so sollte aus Gründen der Überprüfbarkeit der Sicherheit (insbesondere der Vertraulichkeit) der Funktionsumfang der Netzabschlüsse möglichst klein sein, damit ihr Entwurf, ihre Produktion, ihre Installation und ihr Betrieb möglichst rigoros öffentlich überprüft werden können. Damit sollte es auch bezüglich der Überprüfbarkeit von Anonymität, Unbeobachtbarkeit und Unverkettbarkeit akzeptabel sein, daß ein oder wenige Lieferanten die Netzabschlüsse liefern, und die Qualität ihrer Produkte vom Netzbetreiber anerkannt wird. Die Lieferanten haben ihre Entwürfe inkl. aller Entwurfshilfsmittel und Entwurfskriterien zu veröffentlichen. Konsumenten- oder Bürgervereinigungen wie beispielsweise die „Stiftung Warentest“ sollten in den Netzabschlüssen (wie auch in den Teilnehmerendgeräten) kontinuierlich nach Trojanischen Pferden suchen. Dies ist hauptsächlich zu Beginn aufwendig, wenn die Entwurfshilfsmittel und vor allem die Entwürfe überprüft werden müssen. Die Kontrolle der unveränderten Produktion dürfte wesentlich einfacher sein.

Netzabschlüsse dürften nur wenige integrierte digitale Schaltkreise und analoge Sender- und Empfänger-Bausteine umfassen. Entsprechend sollte Wartung bzw. Reparatur (und anschließende Überprüfung) selten sein. Insbesondere besteht keine Notwendigkeit für *Fernwartung*, d.h. die Fähigkeit des Herstellers oder Netzbetreibers, die Software des Netzabschlusses per Zugriff über das Kommunikationsnetz modifizieren zu können, wie dies bei den heutigen Vermittlungszentralen vorgesehen und (wegen bei der Auslieferung noch nicht gefundenen, geschweige denn korrigierten Entwurfsfehlern oder einer großen Zahl möglicher Hardwareausfälle) wohl auch nötig ist. Wie in §5 erläutert wurde, können Hersteller mittels Fernwartung Trojanische Pferde beliebig installieren und die von ihnen eingebauten beliebig beseitigen. Dies gilt dann nicht, wenn Fernwartung nur in der sehr eingeschränkten Form der *Fernabfrage* möglich ist: Hierbei können per Zugriff über das Kommunikationsnetz keine Programme modifiziert, sondern nur vor Ort vorhandene Diagnoseprogramme gestartet und ihre Ergebnisse entgegengenommen werden. Der Zweck der Fernabfrage ist, daß ein Wartungstechniker gezielt Ersatzteile mitnehmen kann. Sicherzustellen, daß Fernwartung auf Fernabfrage beschränkt ist, ist allerdings das bereits in §1.2 und §5 dargestellte Problem des Ausschließens Trojanischer Pferde.

In den folgenden Unterabschnitten wird beschrieben, welchen Funktionsumfang die Netzabschlüsse bei den verschiedenen Grundverfahren zum Schutz der Verkehrs- und Interessensdaten aufweisen müssen. Bild 5-48 gibt einen Überblick.

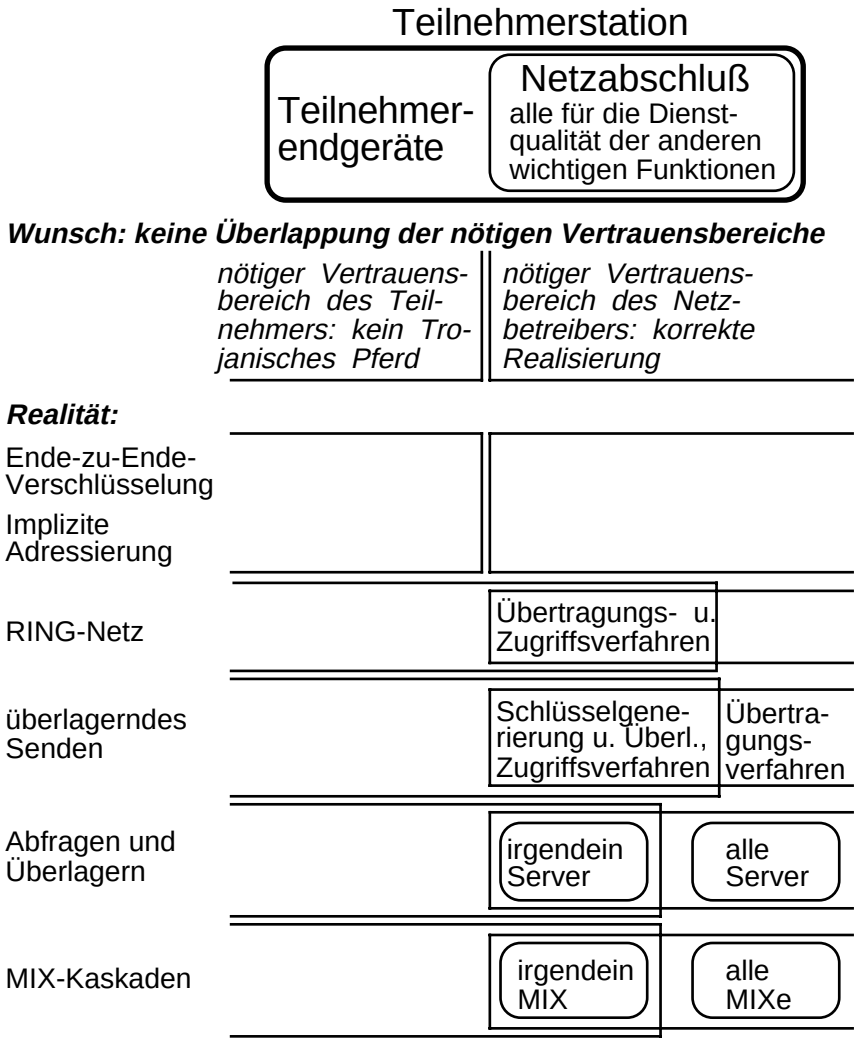


Bild 5-48: Funktionsumfang der Netzabschlüsse

5.6.1.1 Ende-zu-Ende-Verschlüsselung und Verteilung

Da eine Teilnehmerstation bei fehlerhafter Ende-zu-Ende-Ver- bzw. -Entschlüsselung, fehlerhafter Generierung oder Erkennung impliziter Adressen oder fehlerhaftem Empfang verteilter Informationseinheiten nur die Dienstqualität ihres Teilnehmers und die seiner direkten Kommunikationspartner beeinträchtigt, kann all dies im Teilnehmerendgerät abgewickelt werden.

5.6.1.2 RING- und BAUM-Netz

Der Netzabschluß des RING- bzw. BAUM-Netzes muß neben dem physischen Netzanschluß mit digitaler Signalregenerierung, das Zugriffsverfahren und auch alle vorgesehenen Fehlertoleranz-Maßnahmen umfassen. Denn sofern diese Funktionen bei einem Teilnehmer Leistungs- oder Zuverlässigkeitsmängel aufweisen, sind auch alle anderen Teilnehmer des RING- bzw. BAUM-Netzes betroffen.

Ist im Netzabschluß vom Netzbetreiber oder dem Hersteller ein Trojanisches Pferd untergebracht, so kann dieses also das Senden der betroffenen Station registrieren, wodurch die Schutzwirkung von Übertragungstopologie und digitaler Signalregenerierung für die Sender vollständig verloren geht. Somit wurde das Problem Trojanischer Pferde von den Vermittlungszentralen auf die Netzabschlüsse verlagert. Dennoch ist, wie in §5.6.1 bereits begründet, die Situation hier erheblich besser, da Netzabschlüsse um einige Größenordnungen einfacher als Vermittlungsrechner sind und damit eine Prü-

fung auf Trojanische Pferde viel leichter möglich und wegen seltenerer Wartung bzw. Reparatur auch weniger häufig durchzuführen ist.

5.6.1.3 DC-Netz

Der Netzabschluß des DC-Netzes muß neben dem physischen Netzanschluß die Pseudozufallszahlengenerierung, das Zugriffsverfahren und auch alle vorgesehenen Fehlertoleranz-Maßnahmen umfassen. Denn sofern diese Funktionen bei einem Teilnehmer Leistungs- oder Zuverlässigkeitsmängel aufweisen, sind auch alle anderen Teilnehmer des DC-Netzes betroffen.

Ist im Netzabschluß vom Netzbetreiber oder dem Hersteller ein Trojanisches Pferd untergebracht, so kann dieses also das Senden der betroffenen Station registrieren, wodurch die Schutzwirkung des überlagernden Sendens vollständig verloren geht. Das Problem Trojanischer Pferde wurde somit wie beim RING- und BAUM-Netz von den Vermittlungszentralen auf die Netzabschlüsse verlagert. Wie in §5.6.1.2 ist aber die Situation dadurch erheblich besser geworden.

Bei modularem Aufbau des Netzabschlusses gemäß der Schichtung von Bild 5-40 sind zusätzlich die Module, die den physischen Netzanschluß darstellen (Medium und untere Teilschicht der Schicht 1), gemäß dem Angreifermodells des DC-Netzes bezüglich Anonymität, Unbeobachtbarkeit und Unverkettbarkeit unkritisch. Hier darf, wer immer will, beobachtend angreifen. Selbst verändernde Angriffe in diesen Modulen schwächen bei geeigneter Implementierung (vgl. die Anmerkungen in §5.4.1 und §5.4.5.2) nicht die Anonymität ab, sondern verhindern nur die Diensterbringung. Geeignete Protokolle zur Fehler- bzw. Angriffsdiagnose sind in §5.4.5.6 und §5.4.7 sowie [WaPf_89, WaPf1_89] beschrieben.

Für den Fall eines umfangreichen öffentlichen DC-Netzes mit einem Netzbetreiber ist es noch wichtiger als bei lokalen oder Spezialnetzen, daß die kryptographische Stärke der Pseudozufallszahlengenerierung öffentlich bewiesen oder zumindest validiert ist (vgl. §3). Anderenfalls kann die Pseudozufallszahlengenerierung eine verborgene Falltür (trapdoor) enthalten, die es dem Netzbetreiber oder -entwerfer ermöglicht, das Senden der Teilnehmerstationen zu beobachten. Oder die Pseudozufallszahlengenerierung wird einige Monate oder Jahre, nachdem sie für dieses umfangreiche DC-Netz ein De-facto-Standard geworden und nur noch unter sehr hohen Kosten zu ändern ist, gebrochen.

5.6.1.4 Abfragen und Überlagern sowie MIX-Netz

Werden Server oder MIXe von normalen Netzteilnehmern betrieben, so umfaßt der Netzabschluß eine große Datenbank mit Addierwerken oder (kleine) Vermittlungszentrale und eine größere Zahl von sehr leistungsfähigen Ver- und Entschlüsselungsgeräten sowie alle Maßnahmen zur Fehlertoleranz. Wie in §5.4.6.7 erwähnt, kann dann versucht werden, mittels MIXen nicht nur die Kommunikationsbeziehung, sondern auch das Senden und Empfangen von Teilnehmerstationen zu schützen. Neben den aus [Pfit_89 §3.2.2.4] ersichtlichen Schwierigkeiten, daß dann entweder sehr viele bedeutungslose Informationseinheiten erzeugt oder lange Wartezeiten hingenommen werden müssen, sei darauf hingewiesen, daß dieser sehr umfangreiche Netzabschluß das Senden und Empfangen (nicht aber die Kommunikationsbeziehungen) einer Teilnehmerstation kanonischerweise beobachten kann. Der geschilderte Versuch ist also nicht nur sehr aufwendig und unkomfortabel, sondern sein Erfolg auch höchst ungewiß.

Die Vermittlungsfunktion der MIXe kann zwar weitgehend eliminiert werden, indem die MIXe jeweils in fester Reihenfolge durchlaufen werden müssen (vgl. §5.4.6.1). Aber auch für den Rest kann Wartung oder Reparatur öfter nötig sein als dies für in privaten Räumen untergebrachte Geräte akzeptabel, geschweige denn wünschenswert ist.

Die Wartungs- und Reparatursituation ist völlig unproblematisch, sofern, wie beispielsweise in §5.5.1 beschrieben, normale Teilnehmerstationen nicht als Server oder MIX fungieren.

Da bei Abfragen und Überlagern die Dienstqualität Unbeteiligter durch „Fehler“ anderer Teilnehmer, die keine Server-Funktion realisieren oder deren Server-Funktion vom Fehler nicht betroffen ist, nicht beeinträchtigt wird, kann die den einzelnen Teilnehmer schützende Maßnahme, nämlich das Bilden der Abfragevektoren, das Verschlüsseln und Senden an die Server, das Empfangen und Entschlüsseln der Antworten der Server sowie deren Überlagerung, ausschließlich im Teilnehmerendgerät stattfinden.

Da beim MIX-Netz die Dienstqualität Unbeteiligter durch „Fehler“ anderer Teilnehmer, die keine MIX-Funktion realisieren oder deren MIX-Funktion vom Fehler nicht betroffen ist, nicht beeinträchtigt wird, kann die den einzelnen Teilnehmer schützende Maßnahme, nämlich das mehrfache Ver- und Entschlüsseln, ausschließlich im Teilnehmerendgerät stattfinden. Mit dem in §5.5.1 beschriebenen Verfahren der MIXe mit bedeutungslosen Zeitscheibenkanälen und Verteilung der Gesprächswünsche kann also Senden und Empfangen mit erheblich besserer Aussicht auf Erfolg geschützt werden als mit dem in §5.4.6.7 beschriebenen Verfahren, bei dem jede Teilnehmerstation als MIX fungiert.

Für den Fall eines umfangreichen öffentlichen MIX-Netzes ist es noch wichtiger als bei lokalen oder Spezialnetzen, daß die kryptographische Stärke der verwendeten Konzelationssysteme auch für diesen Anwendungszweck öffentlich bewiesen oder zumindest validiert ist (vgl. §3 und §5.4.6.8).

5.6.1.5 Kombinationen sowie heterogene Netze

Werden – wie etwa in §5.4.4 beschrieben – mehrere Grundverfahren zum Schutz der Verkehrs- und Interessensdaten kombiniert, so ist es bezüglich der Überprüfbarkeit des Schutzes günstig, nicht einen Netzabschluß mit der Vereinigungsmenge aller notwendigen Funktionen zu realisieren, sondern eine Kaskadierung der für die einzelnen Verfahren nötigen Netzabschlüsse entsprechend der Schichtung gemäß §5.4.9 vorzunehmen. Dann kann ein in einem vom Teilnehmerendgerät weit entfernter, einer niedrigen Schicht entsprechender Netzabschluß mit Trojanischem Pferd das Kommunikationsverhalten einer Teilnehmerstation nur wenig beobachten.

5.6.1.6 Verbundene, insbesondere hierarchische Netze

Werden die Übergänge zwischen den Netzen bzw. Netzebenen und ggf. die von allen Teilnetzen hin und wieder benötigten oberen Netzebenen so realisiert, daß eine Gruppe von Betreibern die Verantwortung für die Dienstqualität dieser Netzteile übernehmen kann, so können die restlichen Netzteile beliebig realisiert werden. Netzbetreiberschaft und Verantwortung für die Dienstqualität können also in unterschiedlichen Netzteilen so gut wie unabhängig voneinander geregelt werden.

5.6.2 Abrechnung

Bei einem offenen Kommunikationsnetz muß ggf. eine komfortable und sichere Abrechnung der Kosten für die Netzbenutzung mit dem Netzbetreiber möglich sein. Bei der Organisation der Abrechnung muß darauf geachtet werden, daß durch Abrechnungsdaten die Anonymität, Unbeobachtbarkeit und Unverkettbarkeit im Kommunikationsnetz nicht verloren geht.

Prinzipiell hat man dabei zwei Möglichkeiten: **Individuelle** Abrechnung nach Einzelnutzung (oder auch für Abonnements u.ä.) mit Verfahren, bei denen der bezahlende Teilnehmer anonym ist oder **generelle**, d.h. von allen Netzteilnehmern zu leistende, pauschale Bezahlung, die nicht anonym erfolgen muß, da dabei keine interessanten Abrechnungsdaten entstehen.

Für individuelle Abrechnung können entweder

- nicht manipulierbare Zähler [Pfi1_83 Seite 36f] oder
- anonyme digitale Zahlungssysteme (siehe §6),

verwendet werden.

Die **nicht manipulierbaren Zähler** werden zweckmäßigerweise im Netzabschluß untergebracht. Sie entsprechen in ihrer Funktion den heutigen Elektrizitätszählern, nur daß sie über das Kommunikationsnetz ausgelesen werden können. Sind die Zähler technisch so gestaltet, daß dieses Auslesen nur in großen Zeitintervallen, z.B. alle Monate einmal, geschehen kann, so geben diese Zähler nur sehr wenig personenbezogene Information ab, so daß zwischen Netzbetreiber und Teilnehmern über deren Zählerstände nichtanonym abgerechnet werden kann.

Hauptvorteil dieser Lösung ist, daß im Kommunikationsnetz so gut wie kein Aufwand für Abrechnungszwecke getrieben werden muß. Hauptnachteile sind, daß ein Umgehen des Zählers durch Umgehen des Netzabschlusses genauso verhindert oder zumindest entdeckt werden muß wie eine Manipulation am Zählerstand. Beide Nachteile sind allerdings nicht sehr schwerwiegend, da (im Gegensatz zu allgemein verwendeten Zahlungsmitteln, vgl. §6) mit diesen Zählern nur die Bezahlung einer einzigen Dienstleistung von – zumindest bei Privatleuten – üblicherweise eher geringfügigem Wert möglich ist. Auch heute sind Briefmarken wesentlich leichter zu fälschen als Geldscheine.

Bei Verwendung von anonymen digitalen Zahlungssystemen müssen die genauen Abrechnungsprotokolle so entworfen werden, daß von vornherein niemand betrügen kann, da bei ihnen die Anonymität, Unbeobachtbarkeit und Unverkettbarkeit eine nachträgliche Strafverfolgung generell be- oder gar verhindert [WaPf_85, PWP_90, BüPf_90]. Dies ist beim Abrechnungsproblem der Netznutzung sehr einfach zu erreichen: Der ersten aller verkettbaren Informationseinheiten wird jeweils ein digitales, d.h. durch eine binäre Nachricht repräsentiertes Zahlungsmittel vorangestellt. Den Wert dieser „**digitalen Briefmarke**“ läßt sich der Netzbetreiber gutschreiben, bevor er die verkettbaren Informationseinheiten weiterbefördert.

Selbst bei lokaler Kommunikation in DC-, bzw. BAUM-Netzen kann der Netzbetreiber unfrankierte Nachrichten unterdrücken, wenn er die globale Überlagerung durchführt bzw. die Wurzel des Baumes kontrolliert. Beim RING-Netz kann er dies nicht immer. Allerdings kann bei zwei Ringteilnehmern *A* und *B* entweder nur *A* unentgeltlich an *B* senden oder *B* an *A*. Kosten in einem Übertragungsrahmen realisierte Duplex-Kanäle genauso viel wie ein ebenfalls in einem Übertragungsrahmen realisierter Simplex-Kanal, so kann zumindest bei Diensten, die einen Duplex-Kanal erfordern, eine Dienstleistung nicht ohne Bezahlung erlangt werden.

Hauptvorteil dieses Verfahrens der „digitalen Briefmarken“ ist, daß es ohne nicht umgehbare und nicht manipulierbare Zähler auskommt und bei einem „guten“ anonymen Zahlungssystem keinerlei personenbezogene Information anfällt. Hauptnachteil ist der nötige Kommunikationsaufwand. Um ihn erträglich zu halten und insbesondere die Verzögerungszeit kurz, sollte der Netzbetreiber die Funktion der Bank im digitalen Zahlungssystem übernehmen, vgl. §6.

Wünscht der Teilnehmer einen Einzelentgeltnachweis, so kann ihm (und nur ihm!) dies je nach Funktionsverteilung entweder das Teilnehmerendgerät oder der Netzabschluß erstellen.

Durch Verwendung von generellen Pauschalen vermeidet man alle Probleme bezüglich Betrugssicherheit und fast den gesamten Aufwand des Abrechnungsverfahrens.

Sobald genügend Bandbreite zur Verfügung steht, ist z.B. ein pauschales Entgelt an den Netzbetreiber für schmalbandiges Senden und Fernsehempfang möglich.

5.7 Öffentlicher mobiler Funk

Als Ergänzung des bisher behandelten Ausbaus der offenen Kommunikationsnetze zwischen ortsfesten, durch Leitungen verbundenen Teilnehmerstationen erfolgt ein Ausbau der offenen Funknetze

zwischen mobilen Teilnehmerstationen [Alke_88]. Deshalb soll hier kurz skizziert werden, wie die hierbei auftretenden Datenschutzprobleme gelöst oder zumindest erträglich klein gehalten werden können – ausführlicher ist dies in [Pfit_93, ThFe_95, FJKP_95, Fede_99] beschrieben.

Die Unterschiede zu den bisherigen Kapiteln sind, daß

- Übertragungsbandbreite bei Funknetzen sehr knapp ist und *bleiben wird*, da das elektromagnetische Spektrum im freien Raum „nur einmal“ vorhanden ist.
- nicht nur (technisch gesehen) die Nutzdaten und Vermittlungsdaten bzw. (inhaltlich gesehen) die Inhaltsdaten, Interessensdaten und Verkehrsdaten einen Personenbezug aufweisen und deshalb ggf. geschützt werden müssen, sondern auch der *momentane Ort* der mobilen Teilnehmerstation bzw. des sie benutzenden Teilnehmers.

Wie in §5.4.4 erläutert wurde, ist es zumindest bezüglich technisch versierter Angreifer unrealistisch zu fordern, daß Signale, die von verschiedenen Stationen gesendet werden, nicht unterschieden werden können: Wegen der analogen Charakteristika des Senders und ihrer bei jedem Produktionsprozeß unvermeidbaren Streuung wäre dies praktisch und wegen der Änderung des Signals bei seiner Ausbreitung (Dispersion) bei kontinuierlichem Senden auch theoretisch nicht erfüllbar.

Ich gehe im folgenden deshalb davon aus, daß eine mobile Teilnehmerstation immer identifizierbar und peilbar ist, wenn sie sendet[124] (auch wenn Hochfrequenztechniker das Funksystem so auslegen sollten, daß Identifikation und Peilung der mobilen Teilnehmerstationen möglichst schwierig ist, und obwohl bei manchen Anwendungen dem Angreifer unbekannte, die Signalausbreitung beeinflussende Umgebungen eine Identifikation praktisch sehr erschweren).

Im Gegensatz dazu gehe ich im folgenden davon aus, daß Teilnehmerstationen so ausgelegt werden können, daß sie nicht identifizierbar und peilbar sind, wenn sie nur (passiv) empfangen.

Wegen dem auch durch Außenstehende sehr leicht abhörbaren Funkverkehr ist neben der immer notwendigen Ende-zu-Ende-Verschlüsselung auch Verbindungs-Verschlüsselung zwischen der mobilen Teilnehmerstation und der (aus ihrer Sicht) ersten ortsfesten Station angebracht, sofern die Protokollinformation der Schichten 1 bis 3 des ISO OSI Referenzmodells (vgl. §5.4.9) irgendeinen Personenbezug aufweist.

Wegen der Knappheit der Übertragungsbandbreite und einer ansonsten jederzeit möglichen Identifikation und Peilung der mobilen Teilnehmerstation sind die in §5.4.3 bis §5.4.5 beschriebenen Möglichkeiten zum Schutz des Senders („bedeutungslose Nachrichten“, „Unbeobachtbarkeit angrenzender Leitungen und Station sowie digitale Signalregenerierung“, „überlagerndes Senden“) sowohl nicht anwendbar als auch nicht empfehlenswert.

Da damit alle Maßnahmen zum Schutz der Verkehrs- und Interessensdaten im ortsfesten Teil des Kommunikationsnetzes abgewickelt werden müssen, bietet sich folgendes Vorgehen an (Bild 5-49):

Sofern die Codierung der Nutzdaten in mobilen genauso wie in ortsfesten Teilnehmerstationen erfolgt, kann das Verfahren der umcodierenden MIXe (§5.4.6) direkt angewendet werden, sofern die mobilen Teilnehmerstationen über genügend Verschlüsselungskapazität verfügen.

Ist die Codierung der Nutzdaten in mobilen Teilnehmerstationen anders als in ortsfesten, um beispielsweise Übertragungsbandbreite zu sparen (z.B. 13 kbit/s Sprachkanal statt 64 kbit/s), so könnte der Empfänger und das Kommunikationsnetz dies zur Verkettung verwenden. In diesem Fall sollte, zumindest solange mobile Teilnehmerstationen nur einen sehr kleinen Teil aller Teilnehmerstationen darstellen, von der mobilen Teilnehmerstation ein Verbindungs-verschlüsselter Kanal zu einer ortsfesten Teilnehmerstation (möglichst des gleichen Teilnehmers) hergestellt, das Signal dort an die übliche Signalcodierung angepaßt und von dort mit den üblichen Verfahren zum Schutz der

---

124 Dies ist nicht willkürlich aus der Luft gegriffen, sondern eine Aufwandsfrage. In [BMI_89 Seite 297] ist zu lesen: „Computergesteuerte Empfangs- und Auswerteeinrichtungen erlauben es der Fernmeldeaufklärung, aus einer Vielzahl von Signalen die Signale einzelner Geräte (jedes Einzelgerät weist besondere Charakteristiken auf) zu selektieren.“

Verkehrs- und Interessensdaten weiterübertragen werden. Entsprechendes gilt, wenn zwar die Codierung der Nutzdaten in mobilen genauso wie in ortsfesten Teilnehmerstationen erfolgt, die mobilen Teilnehmerstationen aber nicht über genügend Verschlüsselungskapazität verfügen.

Eine gerade nur (passiv) empfangende mobile Teilnehmerstation sollte (auch bei Zellularfunksystemen) vom Kommunikationsnetz nicht lokalisiert werden können. Liegt für sie ein Verbindungswunsch oder eine lange Nachricht vor, so sollte eine entsprechende implizite Adresse im ganzen Funknetz verteilt werden, worauf sich die mobile Teilnehmerstation (aktiv) meldet und dadurch lokalisierbar ist. Da Adressen nur wenige Bytes umfassen müssen, ist der Aufwand für diese Schutzmaßnahme gering – falls dies nicht sogar zu einer Aufwandssenkung führt, da die Verwaltung eines Zellularfunksystems erheblich vereinfacht wird.

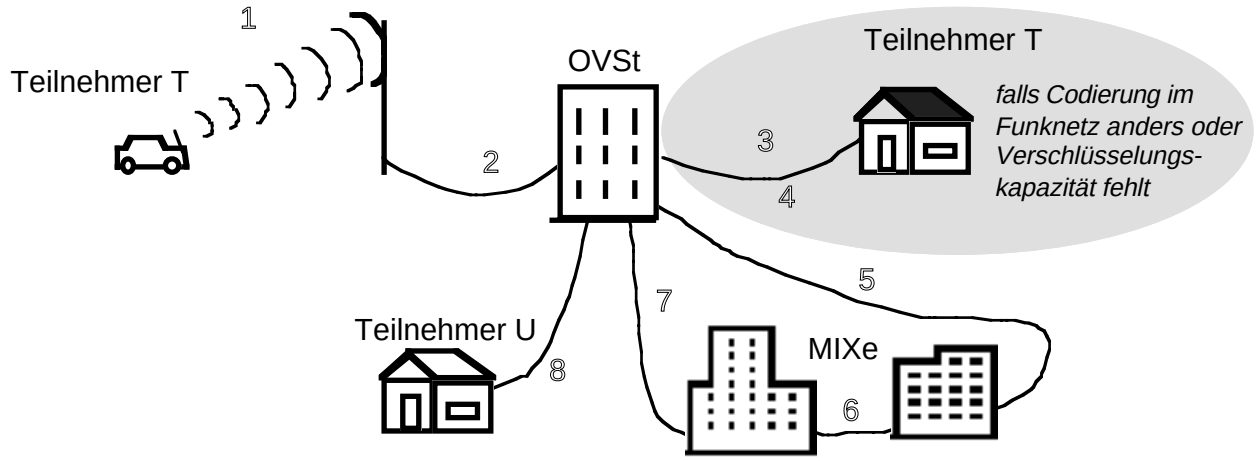


Bild 5-49: Anschluß von Funk-Teilnehmern an ein MIX-Netz

Eine von den bisher beschriebenen Maßnahmen unabhängige ist, den Benutzer gegenüber seiner Sendestation anonym zu halten. Dies macht immer dann Sinn, wenn Sendestationen Benutzern nicht fest zugeordnet sind, wie dies etwa im D-Netz (GSM Zellularfunknetz) vorgesehen ist. In ihm erhält jeder Teilnehmer eine Chipkarte (SIM = Subscriber Identity Module), mit der er jede Sendestation benutzen kann. Ein Beispiel einer Anwendung wäre ein Mietwagen, wo weder Funknetz noch Sendestation des Mietwagens erfahren sollten, wer die Sendestation gerade benutzt. Diese Anonymisierungsmaßnahme entspricht der in §5.3.1 diskutierten Benutzung öffentlicher Anschlüsse.

Zum Schluß sei noch daran erinnert, daß die mobilen Teilnehmerstationen aus den zu Beginn von [Pfit_89 §5 und §6.1] genannten Gründen so konzipiert werden sollten, daß sie in Katastrophensituationen weitgehend ohne den ortsfesten Teil des Kommunikationsnetzes in der näheren Umgebung auskommen und zusätzliche, normalerweise für Unterhaltung (Rundfunk) verwendete Frequenzen nach Erhalt einer „Freigabenachricht“ für Notrufe verwenden können.

Das über öffentlichen mobilen Funk Gesagte ist bei der Gestaltung von **Verkehrsleitsystemen** zu beachten: Es ist bezüglich Anonymität, Unbeobachtbarkeit und Unverkettbarkeit unkritisch, Informationen an Fahrzeuge zu verteilen; es ist sehr kritisch, wenn Fahrzeuge Informationen dauernd oder sehr oft senden müssen, wie dies im Projekt PROMETHEUS [FO_87, Walk_87] vorgesehen ist.

Aus den in §5.1 dargelegten Gründen sollten Verkehrsleitsysteme aus Schutzgründen zusätzlich so entworfen werden, daß Sensoren zwar Fahrzeuge erkennen, aber weder Fahrzeugtypen noch gar Fahrzeugexemplare unterscheiden können – anderenfalls entstehen Bewegungsbilder, die genauso wenig geschützt werden können, wie die im Rest von §5 behandelten Vermittlungsdaten.

Leseempfehlungen

Verschlüsselung: SFJK_97, PSWW_98

Schutz der Verkehrs- und Interessensdaten: Pfit_89 (nur zur Vertiefung)

MIXe: FrJP_97, FJMP_97, FGJP_98, JMPP_98

Einordnung in ein Schichtenmodell: SFJK_97

Abrechnung: Pede_96, FrJW_98

Öffentlicher mobiler Funk: FeJP_96, FJKPS_96, Fede_99

Aktuelle Forschungsliteratur:

Computers & Security, Elsevier

Tagungsserien ACM Conference on Computer and Communications Security, IEEE Symposium on Security and Privacy, ESORICS, IFIP/Sec, VIS; Tagungsbände erscheinen bei ACM, IEEE, in der Serie LNCS des Springer-Verlags Heidelberg, bei Chapman & Hall London, in der Serie Datenschutz-Fachberichte des Vieweg-Verlags Wiesbaden

Hinweise auf aktuelle Forschungsliteratur und Kommentare zu aktuellen Entwicklungen:

http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/ oder
abonnieren unter cipher-request@itd.nrl.navy.mil

Datenschutz und Datensicherheit DuD, Vieweg-Verlag

6 Werteaustausch und Zahlungssysteme

Nach einer kurzen Diskussion, welche Grade an Anonymität es vor Beteiligten gibt, wird dargestellt, wie Rechtssicherheit von Geschäftsabläufen unter Wahrung der Anonymität erreicht werden kann. Danach werden zwei Beispiele ausführlich behandelt: Zunächst, wie Werteaustausch, d.h. der Tausch von Ware gegen Geld, betrugssicher abgewickelt werden kann. Danach, wie digitale Zahlungssysteme realisiert werden können.

Dieses Kapitel ist leicht überarbeitet aus [PWP_90] entnommen.

6.1 Anonymität vor Beteiligten

Unbeobachtbarkeit eines Geschäftes auch vor einem Beteiligten zu fordern ist offenbar sinnlos. Um die Erfassungsmöglichkeit unnötiger personenbezogener Daten zu verhindern, ist hier vielmehr das Ziel, die Identität eines Benutzers so weit wie möglich vor seinen Partnern zu verbergen, ihn also **anonym** zu halten.

Für die Stärke von Anonymität gibt es drei Kriterien.

Das erste ist das verwendete **Angreifermodell** (vgl. §1.2.5), das beschreibt, vor welchen Partnern Anonymität herrschen soll und ob diese bestehen bleibt, wenn mehrere Partner oder auch Unbeteiligte ihre Informationen zusammentragen. Hierzu gehört insbesondere die Frage, ob es speziell vorgesehene Instanzen gibt, die im Streitfall auf Wunsch eines Geschäftspartners die Anonymität des anderen aufheben können. Wenn immer möglich, sollte letzteres vermieden werden, da, wenn bereits wenige Instanzen zusammen die Anonymität aufheben können, diese eine zu starke Machtposition haben, wogegen die Aufhebung zu unzuverlässig wird, wenn es sehr viele sein müssen (im Grenzfall die Gesamtheit aller Benutzer).

Zum zweiten kann man bezogen auf einen bestimmten Angreifer fragen, unter **wieviel möglicherweise Handelnden** sich der wirklich Handelnde verbirgt. Bei offenen Systemen sollten, wenn möglich, immer alle Benutzer bei allen Handlungen als Handelnde in Frage kommen. Praktische Grenzen kann es allerdings durch das Leistungsvermögen des Kommunikationsnetzes geben, vgl. §5.4, so daß etwa der eine Erklärung Abgebende bei Zusammenarbeit seines Geschäftspartners und des Netzbetreibers sich zwar unter vielen, nicht aber unter allen Benutzern verbergen kann. Dabei ist zu beachten, daß die Anzahl der möglichen Handelnden so groß sein muß, daß der Schaden, der nach einer Deanonymisierung einem einzelnen entstünde, nicht einfach allen möglichen Handelnden zugefügt werden kann, ohne daß der Schädiger dadurch einen größeren Nach- als Vorteil gewinnt [Mt 2,16]. Zum Beispiel könnte, wenn sich unter nur 10 Personen bekanntlich ein Kunde eines Verlages für verfassungsfeindliche Schriften befindet, die Verfassungstreue aller 10 in Frage gestellt und ihnen damit eine Anstellung in manchen Bereichen erschwert werden.

Drittens kann man die Anonymität nicht nur bezüglich isolierter Handlungen betrachten, sondern muß auch berücksichtigen, inwieweit der gerade betrachtete Angreifer mehrere Geschäfte oder Geschäftsteile miteinander in Beziehung setzen, sie **verketten** kann, vgl. §5.1.2. Konkret besteht solch eine Verkettung meist darin, daß der Partner weiß, daß zwei Handlungen von derselben Person ausgeführt wurden. Die Verkettbarkeit muß aus Gründen der Vertraulichkeit möglichst gering gehalten werden, andererseits kann sie aus Gründen der Rechtssicherheit manchmal gewollt sein, insbesondere zwischen Teilen desselben Geschäfts oder etwa bei mehrfacher Kommunikation mit einer Bank zu Authentikationszwecken.

Die Stärke der Anonymität eines Benutzers wird dabei nicht nur durch die Kennzeichen bestimmt, die der Partner automatisch über den Benutzer erfährt, etwa Art und Uhrzeit des abgewickelten

Geschäfts, sondern vor allem durch eigens gewählte Kennzeichen wie Kennummern oder Testschlüssel für digitale Signaturen (§3.1.1.2), sogenannte **Pseudonyme**.

Eine aus praktischer Sicht zweckmäßige grobe Einteilung von Pseudonymen nach der Stärke der durch sie realisierten Anonymität ist in Bild 6-1 dargestellt.

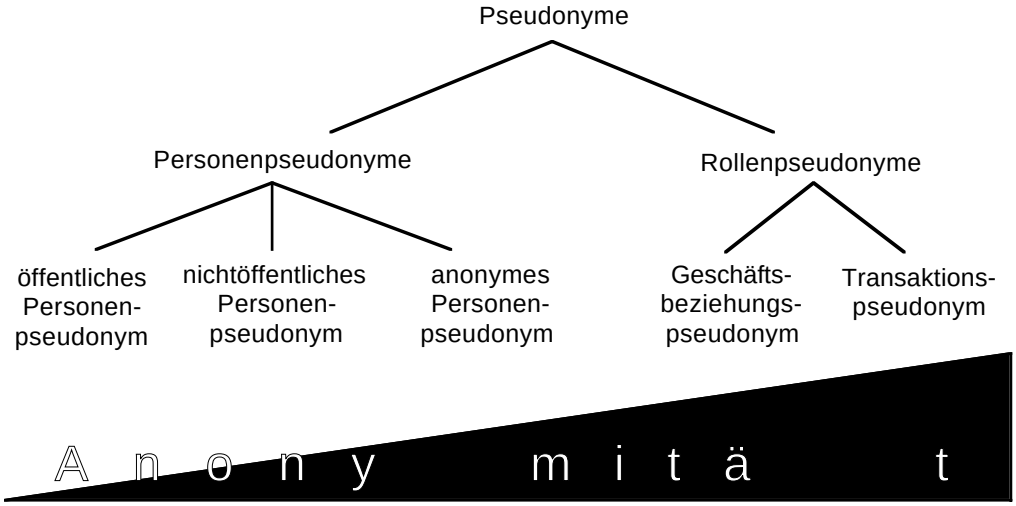


Bild 6-1: Einteilung der Pseudonyme nach ihrem Personenbezug

Ein Pseudonym wird als **Personenpseudonym** bezeichnet, wenn sein Inhaber es für viele verschiedene Geschäftsbeziehungen über lange Zeit hinweg verwendet, es somit einen Namensersatz darstellt. Hinsichtlich der Verkettungsmöglichkeiten mit der Person seines Inhabers können grob drei Arten von Personenpseudonymen unterschieden werden.

Betrachtet man den Zeitpunkt, zu dem ein Personenpseudonym erstmals verwendet wird, so ist bei **öffentlichen Personenpseudonymen** die Zuordnung zu einer Person zumindest im Prinzip allgemein bekannt (z.B. Telefonnummern), bei **nichtöffentlichen Personenpseudonymen** ist diese Zuordnung nur wenigen Stellen bekannt (z.B. Kontonummern ohne Nennung des Namens oder nicht im Teilnehmerverzeichnis aufgeführte Telefonnummern) und bei **anonymen Personenpseudonymen** ist diese Zuordnung nur dem Inhaber bekannt (z.B. biometrische Merkmale des Inhabers oder gar seine DNA). Bei Verwendung von Personenpseudonymen sammelt sich bei einem Beobachter laufend personenbezogene Information an, so daß nach einer gewissen Zeit der Inhaber eines nichtöffentlichen oder anonymen Personenpseudonyms deanonymisiert werden kann. Jedes Personenpseudonym ist also ein potentielles Personenkennzeichen.

Diesen Nachteil vermeiden **Rollenpseudonyme**, die im Gegensatz zu Personenpseudonymen nicht der Person, sondern nur ihrer momentan ausgeübten Rolle zugeordnet sind. **Geschäftsbeziehungspseudonyme** sind solche Rollenpseudonyme, die für viele Transaktionen verwendet werden, z.B. eine Kontonummer bei den vielen Buchungen eines Kontos oder ein Künstlername. **Transaktionspseudonyme** hingegen werden nur für eine Transaktion verwendet, z.B. Kennwörter bei anonym aufgegebenen Chiffreanzeigen. Bei Verwendung von Rollenpseudonymen können verschiedene Parteien über den Pseudonymträger gesammelte Information zumindest nicht einfach über die Gleichheit von Pseudonymen, sondern allenfalls über Korrelation von Zeiten, Geldbeträgen etc. verketten. Aber trotzdem besteht bei Geschäftsbeziehungspseudonymen die Gefahr, daß bei intensiv genutzten Beziehungen der Partner genügend pseudonymbezogene Information zur Deanonymisierung erhält. Aus der Sicht der Vertraulichkeit sollten daher, wenn immer möglich, Transaktionspseudonyme verwendet werden.

Wird im folgenden ein Pseudonym genauer angesprochen, etwa als das Pseudonym, das eine Person X bei einer Transaktion t, in der sie in der Rolle R (z.B. Kunde) auftritt, gegenüber einer anderen Person, die in einer Rolle S (z.B. als Dienstanbieter) auftritt, verwendet, so wird dies mit $p_R^S(X,t)$ bezeichnet. Im Einzelfall überflüssige Teile einer solchen Bezeichnung, z.B. die Transaktion bei Geschäftsbeziehungspseudonymen, werden weggelassen.

Kommunikation zwischen solchermaßen voreinander anonymen Partnern ist über ein Verkehrsdaten schützendes Kommunikationsnetz ohne weiteres möglich, denn es gestattet, Nachrichten ohne Absenderangabe zu senden und unter beliebigen Pseudonymen zu empfangen, die nicht (wie sonst Adressen) den physischen Ort des Benutzers oder gar ihn selbst bezeichnen. Ebensowenig wird die Verwendung eines kryptographischen Systems erschwert, da die Schlüssel eines asymmetrischen kryptographischen Systems, das entweder selbst zum Verschlüsseln oder zum Schlüsselaustausch für ein symmetrisches System verwendet wird, statt identifizierbaren Benutzern auch Pseudonymen zugeordnet sein können.

Nachdem in diesem Abschnitt daran erinnert wurde, daß Anonymität für überprüfbaren Datenschutz in offenen digitalen Systemen notwendig und technisch realisierbar ist, soll in den folgenden Abschnitten behandelt werden, wie die gewünschte Rechtssicherheit erreicht werden kann, ohne die Anonymität etwa bei der Authentikation wieder aufzugeben.

6.2 Rechtssicherheit von Geschäftsabläufen unter Wahrung der Anonymität

In diesem Abschnitt wird allgemein untersucht, was bei der Gestaltung von Geschäftsabläufen zu beachten ist, um Rechtssicherheit trotz Anonymität zu garantieren, ohne hierbei auf die derzeitige juristische Situation näher einzugehen (siehe hierzu [Rede_84, Clem_85, Köhl_86, Rede_86]).

Dazu wird in der Reihenfolge vorgegangen, in der auch die Abwicklung des Rechtsgeschäfts mitsamt eventueller Schadensregulierung erfolgt. Vieles im folgenden Gesagte gilt auch für nicht anonyme Geschäftsabläufe in offenen digitalen Systemen, da auch dort davon auszugehen ist, daß sich Geschäftspartner am Anfang weder persönlich kennen noch in üblicher Weise voreinander ausweisen können, so daß anfänglich Anonymität (wenn auch nicht Unbeobachtbarkeit) herrscht.

6.2.1 Willenserklärungen

6.2.1.1 Anonymes Abgeben und Empfangen

Die Möglichkeit, Erklärungen anonym abzugeben und zu empfangen, ist bereits durch das Verkehrsdaten schützende Kommunikationsnetz gegeben.

Wenn korrespondierende Erklärungen abgegeben werden sollen, bei denen man wünscht, daß entweder alle Beteiligten diese signieren oder keiner (was bei den meisten für offene Systeme vorgesehenen Geschäften nicht der Fall ist), kann man dies wie einen kompletten Geschäftsablauf zum Austausch signierter Erklärungen auffassen und etwa so abwickeln wie den nachher ausführlicher behandelten Austausch von Ware gegen Geld. Wo gewünscht, können spezielle Vertragsabschlußprotokolle [BGMR_85, EvGL_85] verwendet werden, die einen fast gleichzeitigen Austausch der Erklärungen erlauben, ohne dabei die Dienste eines Dritten in Anspruch nehmen zu müssen. Allerdings erhöhen solche Vertragsabschlußprotokolle den Kommunikationsaufwand stark.

6.2.1.2 Authentikation von Erklärungen

Häufig muß der eine Erklärung Abgebende eine Berechtigung zur Abgabe vorweisen. Ein wesentliches Hilfsmittel, um dies über ein Kommunikationsnetz tun zu können, sind **digitale Signaturen**.

6.2.1.2.1 Digitale Signaturen

Anstelle der eigenhändigen Unterschrift soll in Rechtsgeschäften über offene digitale Systeme eine sogenannte digitale Signatur (vgl. §3.1.1.2) dazu dienen, sicherzustellen, daß eine bestimmte Erklärung von einer bestimmten Person (oder auch Personengemeinschaft etc.), gekennzeichnet durch ein Pseudonym, abgegeben wurde.

Die Grundforderungen an eine digitale Signatur sind daher:

1. Niemand außer dem Inhaber eines Pseudonyms sollte fähig sein, ein Dokument mit der zu diesem Pseudonym gehörenden Signatur zu versehen.
2. Jeder kann nachprüfen, ob eine bestimmte Erklärung mit einer zu einem bestimmten Pseudonym gehörenden Signatur versehen ist.

Die erste Forderung bezieht sich dabei nur auf den *Willen* des Benutzers: Natürlich kann in einem rein digitalen System niemand einen Benutzer daran hindern, einem anderen Benutzer zu gestatten, unter einem von ihm verwendeten Pseudonym zu handeln. Dies entspricht der auch heute gegebenen Möglichkeit, einem anderen beliebig viele Blankounterschriften zur Verfügung zu stellen, und kann höchstens dem Benutzer selbst schaden, da man auch hier davon ausgehen muß, daß die so zustandegekommenen Erklärungen dem Inhaber des Pseudonyms als seine eigenen zugerechnet werden.

Der einfache Ansatz, auf die Einzigartigkeit der Pseudonyme zu vertrauen und diese wie herkömmliche Unterschriften der Erklärung beizufügen, genügt obigen Anforderungen leider nicht, denn jeder, der einmal eine unterschriebene Erklärung von einem Benutzer erhalten hätte, könnte das beigefügte digitale Pseudonym auf beliebig viele weitere Erklärungen kopieren. Insbesondere zählt zu diesem ungenügenden Ansatz die bisweilen diskutierte Möglichkeit, eine digitalisierte Kopie der handgeschriebenen Unterschrift zur Authentikation von Erklärungen zu verwenden, denn auch diese läßt sich, selbst wenn sie quer über eine Papierversion der Erklärung geschrieben war, in der digitalen Version mühelos von der Erklärung trennen und weiterkopieren. (Die Gedanken aus [Köhl_86] sind hier nicht anwendbar, da dort die Faksimileunterschriften wie bei Banknoten mit speziellen Papierformularen kombiniert werden, die nicht im digitalen System verschickt werden können. Außerdem ist schon beim heutigen Stand der Kopiertechnik der Aufdruck solcher Formulare nicht sicherer als die eigenhändige Unterschrift, höchstens noch die Papierstruktur o.ä.).

Deswegen benötigt man auch im nichtanonymen Fall ein vom Namen verschiedenes digitales Pseudonym (öffentliches Personenpseudonym), was einer der Gründe ist, ihn als Spezialfall des anonymen aufzufassen.

Abhilfe gegen obiges Problem schafft ein **digitales Signatursystem**.

In Bild 6-2 ist die Übergabe einer signierten Nachricht vom Benutzer X an den Benutzer Y dargestellt, wobei X das Pseudonym p_A (für Abgebender) verwendet; links in einer funktionalen Schreibweise, rechts in der für den Rest des Kapitels gewählten graphischen Notation, in der Nachrichten als Dokumente und Signaturen als Siegel dargestellt werden.

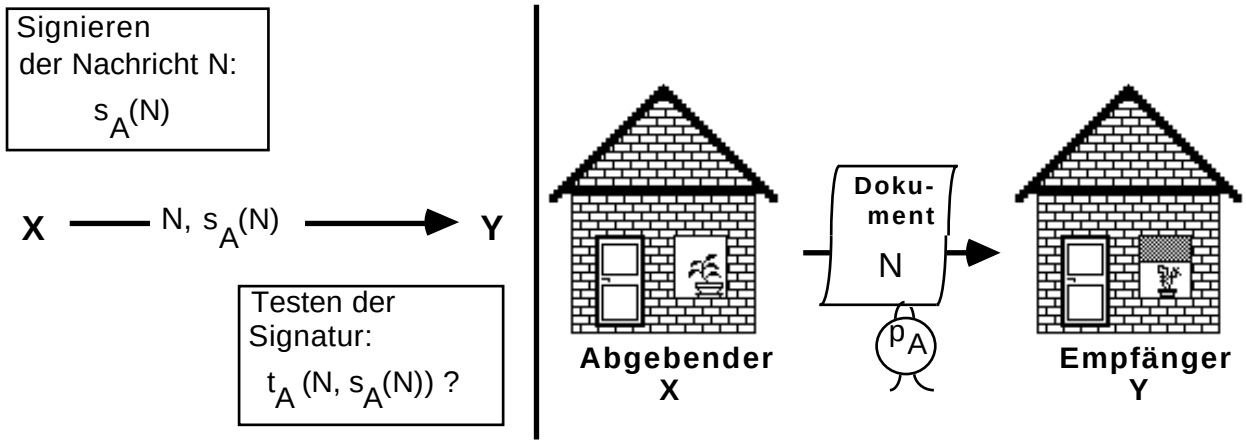


Bild 6-2: Übergabe einer signierten Nachricht von X an Y, in funktionaler Schreibweise links, in graphischer rechts

Für manche Anwendungen ist es sinnvoll, für digitale Signatursysteme zusätzliche Eigenschaften zu fordern:

Der Empfänger z.B. einer mit GMR signierten Nachricht kann diese jedem anderen Teilnehmer zeigen und ihn damit von der Authentizität der Nachricht überzeugen. Der Unterzeichner hat über das Weitergeben seiner Signatur keine Kontrolle. Dieser Mangel wird durch **nicht herumzeigbare Signaturen** (undeniable signatures, §3.1.1.2 und §3.9.4) behoben: Damit ein anderer Teilnehmer die Echtheit der Signatur glaubt, muß der angebliche Unterzeichner befragt werden; dieser kann von ihm tatsächlich geleistete Signaturen nicht ableugnen.

Jedes Signatursystem (wie auch jedes asymmetrische Konzelationssystem) kann mit hinreichend großem Aufwand gebrochen werden. In üblichen Signatursystemen trägt das Risiko hierfür der Unterzeichner. Durch ein **Fail-Stop-Signatursystem** (§3.9.2) kann dieses Risiko auf den Empfänger verlagert werden: Wird eine Signatur gefälscht, so kann der angebliche Unterzeichner diese Fälschung jedem Dritten nachweisen.

Eine weitere und im folgenden auch wirklich benötigte Variante, die **blind geleisteten Signaturen**, werden in §6.2.1.2.2 beschrieben.

Nicht zu verwechseln mit digitalen Signatursystemen sind sogenannte **digitale Identifikationssysteme**. Während der Empfänger einer signierten Nachricht auch einem Dritten gegenüber die Signatur (und damit evtl. die Authentikation einer Erklärung) nachweisen kann, erlaubt ein Identifikationssystem dem Empfänger lediglich, den Abgebenden einer Erklärung zum Zeitpunkt der Abgabe als Besitzer eines bestimmten Pseudonyms zu identifizieren [WeCa_79, Bras_83, FiSh_87, Simm_88]. Ein Identifikationssystem ist also ein symmetrisches Authentikationssystem für nur eine Nachricht.

Der Einfachheit halber wird der Abgebende hierzu ein (möglicherweise speziell geformtes) Pseudonym der Erklärung beilegen oder die Erklärung mit einem nur ihm und dem ihn Identifizierenden bekannten Schlüssel eines symmetrischen kryptographischen Systems verschlüsseln. Nach Empfang der Erklärung kann ein Dritter die Authentikation der Erklärung jedoch nicht mehr überprüfen (wie schon oben erwähnt), so daß digitale Identifikationssysteme aus Gründen der Rechtssicherheit zur Authentikation einer Erklärung ungeeignet erscheinen.

6.2.1.2.2 Formen der Authentikation

Je nachdem, woher die Berechtigung zur Abgabe der Erklärung bezogen wird, kann man zwischen **Eigenauthentikation** und **Fremdauthentikation** unterscheiden.

Eigenauthentikation ist dann gegeben, wenn der Abgebende sich auf eine bereits früher von ihm selbst abgegebene Erklärung beruft, z.B. bei der endgültigen Bestellung einer Ware auf die Anfrage nach einem verbindlichen Angebot.

Hier will der Erklärende also zeigen, daß beide Erklärungen von derselben Person abgegeben wurden. Dies stellt eine gewollte Verkettung verschiedener Erklärungen dar und kann dadurch erreicht werden, daß der Abgebende in beiden Erklärungen dasselbe digitale Pseudonym verwendet und die Erklärungen mit der dazu gehörenden digitalen Signatur versieht. In klassischen Systemen hätte man sowieso bei allen Geschäften denselben Namen und dieselbe Unterschrift verwendet und eventuell, um die Verkettung verschiedener Erklärungen eines Geschäftsablaufs zu erleichtern, noch eine zusätzliche Kennummer.

Fremdauthentikation ist dann gegeben, wenn der die Erklärung Abgebende die Berechtigung zur Abgabe von anderen erhalten hat.

In diesem Fall benötigt er ein Dokument, etwa ein Zeugnis oder eine Kreditwürdigkeitsbescheinigung, das aussagt, daß der Träger eines gewissen Pseudonyms zu gewissen Erklärungen berechtigt ist, und das er der abzugebenden Erklärung beifügt. Zusätzlich muß er die eigene Erklärung wie oben mit der zu diesem Pseudonym gehörenden Signatur versehen.

Der Aussteller jenes Dokumentes kann dabei evtl. selbst wieder durch weitere Dokumente authentisiert werden.

Ohne weitere Maßnahmen gäbe die Fremdauthentikation den authentisierenden Dritten, z.B. Banken, falls sie für jeden Einkauf garantieren müßten, die Möglichkeit, in Zusammenarbeit mit den Empfängern der Erklärungen ungewollte Verkettungen abzuleiten.

Dies kann durch Verwendung **umrechenbarer Autorisierungen** (credentials, [Chau_84]) verhindert werden, die gestatten, die für die Abgabe einer Erklärung notwendige Autorisierung auf ein anderes Pseudonym ausstellen zu lassen als das, welches bei der Abgabe der Erklärung verwendet wird.

Hierzu muß ohne Mitwirkung des Ausstellers die erhaltene Autorisierung in eine auf das aktuell verwendete Pseudonym lautende umgerechnet werden können. Einerseits darf niemand ohne den Willen des Erklärenden einen Zusammenhang zwischen den verwendeten Pseudonymen erkennen und andererseits der Erklärende eine erhaltene Autorisierung nur auf seine eigenen Pseudonyme umrechnen können, nicht etwa auf die eines befreundeten Benutzers.

Ein erstes, auf RSA beruhendes Verfahren zur Umrechnung von Autorisierungen wurde 1985 vorgeschlagen [Chau_85, ChEv_87]. Hierzu wird festgesetzt, daß allen möglichen Autorisierungen (z.B. einer Bank) jeweils eine bestimmte RSA–Signierfunktion zugeordnet wird. Soll auf ein Pseudonym eine Autorisierung ausgestellt werden, so heißt dies einfach, daß dieses Pseudonym mit der zu dieser Autorisierung gehörenden Signierfunktion signiert wird. Dies bedeutet natürlich, daß die möglichen Autorisierungen stark pauschaliert sein müssen.

Genügt es, eine Autorisierung genau einmal umrechnen zu können, so gestattet das Verfahren, das Pseudonym, auf das die Autorisierung umgerechnet werden soll, so zu wählen, daß es für Signaturen verwendet werden kann [Chau_89]. Möchte man eine Autorisierung aber mehrfach verwenden und sie hierzu auch mehrfach umrechnen, so ist dies zwar möglich. Die Pseudonyme, auf die die umgerechneten Autorisierungen lauten, sind dann aber zum Signieren ungeeignet, die umgerechneten Autorisierungen also nur in einem Identifikationssystem verwendbar.

Während die Anonymität des Verfahrens völlig sicher ist (im informationstheoretischen Sinne), ist die Sicherheit höchstens so groß wie die von RSA.

Ähnlich verwendet werden kann auch das spezielle, allerdings ebenfalls unbewiesene, Signatursystem aus [ChAn_90].

Ein auf einem beliebigen kryptographisch sicheren Signatursystem basierendes beweisbar kryptographisch sicheres Verfahren zur Umrechnung von Autorisierungen wurde in [Damg_88] vorgeschlagen. Aus Aufwandsgründen ist es allerdings praktisch nicht einsetzbar.

Sollte RSA gebrochen sein, aber andere sicherere asymmetrische Krypto- und Signatursysteme noch zur Verfügung stehen, so kann in Abwandlung einer in [Chau_81 Seite 86] vorgestellten Anwendung des dort behandelten Verkehrsdaten schützenden Kommunikationsnetzes folgendes Schema verwendet werden: Finden sich *n* Benutzer, die auf Pseudonyme p_1, …, p_n jeweils eine gleichlautende Autorisierung (also Signatur) erhalten wollen, so prüft die Organisation, die diese vergeben soll, ob alle *n* Benutzer berechtigt sind, die gewünschte Autorisierung zu erhalten (d.h. ob sie sie schon auf ein der Organisation vorzulegendes anderes Pseudonym ausgestellt besitzen). Dann gestattet sie ihnen, in zufälliger Reihenfolge jeweils genau eine Nachricht, nämlich das jeweilige Pseudonym, auf einem Verkehrsdaten schützendem Kommunikationsnetz, das ausschließlich für diesen Zweck (aufbauend auf einem bereits physisch vorhandenen) logisch realisiert werden muß, anonym zu veröffentlichen. Das Kommunikationsnetz garantiert also, daß niemand feststellen kann, wer welches Pseudonym beigesteuert hat. Nun signiert die Organisation alle veröffentlichten Pseudonyme.

Dazu kann jedes beliebige Signatursystem verwendet werden. Während aber das Verfahren der umrechenbaren Autorisierungen informationstheoretisch sicher einen Benutzer X unter allen anderen Benutzern verbirgt, die bis zur Verwendung der von X erhaltenen Bestätigung diese ebenfalls erhalten haben, verbirgt das zuletzt beschriebene Verfahren ihn lediglich mit der Sicherheit des logisch realisierten Verkehrsdaten schützenden Kommunikationsnetzes unter den *n* Benutzern, die die Bestätigung gleichzeitig mit ihm erhalten haben.

Eine andere Idee, die ungewollte Verkettbarkeit durch Fremdauthentikation zu verhindern, basiert auf der Annahme der Existenz **sicherer, unausforschbarer Geräte**. In ein solches können, z.B. durch Kommunikation mit anderen sicheren Geräten, Berechtigungen eines Benutzers eingetragen und auch wieder gelöscht werden. Auf Wunsch kann das sichere Gerät eine eingetragene Berechtigung etwa mittels einer systemweit prüfbaren digitalen Signatur bestätigen.

Im Gegensatz zu den normalerweise unterstellten Rechnern muß für diese Anwendung das Gerät seine Funktion aufgrund geheimer Daten, z.B. Schlüssel eines kryptographischen Systems, erbringen, die vor dem Benutzer selbst verborgen bleiben müssen. Andernfalls könnte dieser die Berechtigungen im Gerät ändern oder ein Gerät nachbauen, das den anderen Benutzern und sicheren Geräten richtig erscheint, aber mehr oder andere Berechtigungen enthält.

Nach wie vor muß es aber in seiner Funktion auch durch den Benutzer, den es „bedient“, kontrollierbar sein, so daß es sich physisch im Besitz dieses Benutzers befinden sollte. Hiervon wird auch in der Literatur zumeist ausgegangen [Riha_84, Cha9_85, Davi_85, Riha_85].

Die Existenz von Geräten, die diese beiden Forderungen zugleich erfüllen, ist höchst fraglich. Der Benutzer kann ja das Gerät, insbesondere auch während es gerade arbeitet, beliebig manipulieren und beobachten, wobei sich auch sehr aufwendige Maßnahmen lohnen können, da die Ausforschung eines einzigen Gerätes den Nachbau in sehr vielen Exemplaren erlaubt. Jede Verarbeitung von Information bewirkt aber zwangsläufig einen Energietransport innerhalb des Gerätes. Um ein Messen der Energietransporte (z.B. über elektromagnetische Abstrahlung) zu verhindern, muß das Gerät hinreichend gut abgeschirmt werden. Da ein Benutzer auch versuchen kann, sein Gerät durch zerstörendes Messen auszuforschen, muß ein sicheres Gerät zudem bemerken, wenn seine Schutzmechanismen von außen beeinträchtigt werden. In einem solchen Fall (und auch, wenn seine Funktion aufgrund eines internen Fehlers beeinträchtigt ist) muß das sichere Gerät sich selbst sofort unbrauchbar machen, d.h. seine geheimen Informationen löschen, vgl. §2.1.2.

Damit ist ein Wettlauf zwischen Konstrukteuren sicherer Geräte und der Meßtechnik festgeschrieben, den vermutlich keiner von beiden auf Dauer gewinnt.

In der Praxis werden nichtsdestotrotz zur Zeit Chipkarten als sichere Geräte eingesetzt. Mögliche Anwendungen sind Berechtigungsausweise zur Benutzung z.B. von Datenbanken oder öffentlicher Fernsprechzellen (wobei die Berechtigung anfangs auf *n*-malige Benutzung lautet und bei jeder Benutzung verringert wird) oder auch der Einsatz in digitalen Zahlungssystemen (vgl. §6.4).

Geräte, die vor ihrem Besitzer sicher sein müssen, sollten unserer Meinung nach aber so wenig wie möglich verwendet werden.

6.2.2 Andere Handlungen

Damit es sich lohnt, die zu einem Rechtsgeschäft gehörenden Erklärungen anonym abzugeben und zu empfangen, müssen auch die übrigen, im Zusammenhang damit stehenden Handlungen die Anonymität wahren.

Wenn eine solche Handlung ebenfalls im Versenden von Information über das Kommunikationsnetz besteht, z.B. dem Liefern einer Datenbankauskunft als Ware, so ist die Anonymität bereits durch das zugrundeliegende Verkehrsdaten schützende Kommunikationsnetz gesichert.

Auch die Übergabe von Geld sollte bei einem offenen System in digitaler Form erfolgen können, evtl. in der Form mehrerer aufeinanderfolgender Erklärungen (vgl. §6.4).

Hiermit sind bereits die für diejenigen Rechtsgeschäfte, die über offene Kommunikationsnetze abgewickelt werden sollen, wichtigsten Handlungen erschöpft.

Wird ein Teil der Handlungen nicht über das Kommunikationsnetz abgewickelt (z.B. materielle Ware geliefert, die vorher über das offene Kommunikationsnetz ausgewählt und bestellt wurde), so kann die Anonymität nicht vollständig erhalten werden; teilweise ist dies auch nicht wünschenswert. Im folgenden werden wir uns daher auf Erklärungen und Handlungen beschränken, die über das Kommunikationsnetz ablaufen.

6.2.3 Sicherstellung von Beweismitteln

Aufbauend auf den in §6.2.1 erläuterten Möglichkeiten zur Authentikation muß nun untersucht werden, wie genügend Mittel zum Beweis der Abgabe und des Empfangs einer Willenserklärung sichergestellt werden können.

Wie in §6.2.1, so entstehen im Vergleich zum nicht anonymen Fall auch hier keine wesentlich neuen Probleme.

Man kann bei dieser Untersuchung nach zwei Kriterien gliedern:

Zum einen kann das Ziel der Beweisführung betrachtet werden. Dieses kann sein, die Abgabe oder den Erhalt einer Erklärung zu beweisen, es kann aber auch das Gegenteil der Fall sein, d.h. die Nichtabgabe oder der Nichterhalt sollen bewiesen werden.

Das zweite Ziel kann jedoch einfach dadurch erreicht werden, daß das erste vollständig erreicht wird. Eine nicht als abgesendet oder erhalten nachweisbare Erklärung kann dann als nicht abgegeben oder nicht erhalten betrachtet werden. Das zweite Ziel wird daher im folgenden nicht explizit verfolgt.

Zum anderen kann betrachtet werden, wer ein Interesse an der Beweisführung hat, der Abgebende oder der Empfänger.

Oft wird die Tatsache, daß die Erklärung abgegeben wurde, nur für eine der beiden Parteien vorteilhaft sein, so daß nur diese überhaupt ein Interesse am Nachweis hat und Beweismittel sammeln muß.

Für den Empfänger einer Erklärung ist dieser Nachweis einfach, wenn die Erklärung Dokumentencharakter hat, d.h. eine digitale Signatur trägt. Denn dann genügt die Vorlage der Erklärung als

Nachweis dafür, daß der Inhaber des zur Signatur gehörenden digitalen Pseudonyms diese Erklärung abgegeben hat.

Darauf, daß der Abgebende einer Erklärung Beweismittel für diese Abgabe oder gar für ihren Eingang beim Empfänger sammeln muß, kann in vielen Fällen verzichtet werden, auch wenn die Tatsache, diese Erklärung abgegeben zu haben, für ihn günstig ist.

Kommt es auf den genauen Zeitpunkt der Abgabe nicht an, was gerade auf die Geschäfte des täglichen Lebens, die über offene Kommunikationsnetze abgewickelt werden sollen, zutrifft, so genügt es, wenn er diese Erklärung wiederholt, sobald ihre Abgabe bezweifelt wird, notfalls vor Gericht. Ebenso muß er keine Beweise dafür sammeln, daß er Information als Ware oder Nachrichten, die zur Übertragung digitalen Geldes gehören, geliefert hat. Im Gegensatz zum Abliefern von materiellen Waren oder Papierdokumenten kann der Abgebende die Information ja weiterhin speichern. Dem Empfänger erwachsen aus einer Doppellieferung ersichtlich keine Vorteile, denn die zweite Lieferung stellt lediglich eine Kopie der ersten dar, die er ebensogut hätte selbst erzeugen können.

Sollte der Eingang beim Empfänger doch bewiesen werden müssen, so hat man ähnliche Möglichkeiten wie bei nichtanonymen Erklärungen.

Eine Möglichkeit ist, eine signierte Quittung zu verlangen, deren Erhalt leicht nachgewiesen werden kann (s.o.). Erhält der Abgebende nicht innerhalb eines definierten Zeitraumes die erhoffte Quittung, so muß diese im Notfall sofort gerichtlich erzwungen bzw. durch ein Gericht stellvertretend ausgestellt werden können. Dazu benötigt man für diese umstrittenen Erklärungen auch die folgende Alternative.

Diese zweite Möglichkeit ist, im offenen Kommunikationsnetz sogenannte *Schwarze Bretter* einzurichten, an denen potentielle Empfänger eingangspflichtiger Erklärungen in gewissen Abständen nach solchen Erklärungen Ausschau halten müssen. Da dieses Ausschauhalten ihre Rechner übernehmen können, bedeutet dies vermutlich eine geringere Belastung als die Verpflichtung zum täglichen Leeren des Briefkastens.

Die Tatsache, daß sich eine Erklärung für einen Benutzer, der das Pseudonym p_E verwendet, an einem Schwarzen Brett befand, kann dann durch Zeugen bewiesen werden. Damit die Zeugen den Inhalt der Erklärung nicht erfahren, muß die Erklärung mit einem dem Empfänger der Erklärung bekannten Schlüssel verschlüsselt werden. Nimmt man hierzu an, daß dem Pseudonym p_E des Empfängers auf überprüfbare Art ein Schlüssel c_E eines asymmetrischen Konzelationssystems zugeordnet ist (z.B. durch eine öffentliche Schlüsselbibliothek oder indem c_E selbst als Pseudonym verwendet wird), so kann der Abgebende durch Vorlage des Pseudonyms p_E und der unverschlüsselten Erklärung (und bei indeterministischer Verschlüsselung ggf. noch der Zufallszahl, vgl. §3.1.1.1 Anmerkung 2) beweisen, daß demjenigen, der p_E als digitales Pseudonym verwendet, die Erklärung zugegangen ist. Am günstigsten ist es, eine öffentliche Stelle als Zeuge zu verwenden, die regelmäßig (z.B. täglich) ein Verzeichnis aller eingegangenen verschlüsselten Erklärungen veröffentlicht und signiert, da in diesem Fall auch der Zeuge kontrollierbar ist und der Empfänger nichts an Anonymität einbüßt.

6.2.4 Erkenntnisverfahren

Tritt eine Situation ein, in der eine der beteiligten Parteien an der Rechtmäßigkeit des erreichten Zustands zweifelt, so muß wie in einem Erkenntnisverfahren die Substanz dieses Zweifels näher untersucht werden. Der Benutzer, der dies veranlaßt, muß dazu nicht seine Identität offenbaren, sondern es kann vorerst genügen, festzustellen, ob z.B. der Inhaber eines gewissen Pseudonyms wirklich betrogen wurde.

Da auf keinen Fall im voraus eine Deanonymisierung aller möglicherweise Beteiligten vorgenommen werden sollte (dies könnte Anlaß zu Mißbrauch geben), ist zu beachten, daß nicht jeder zu einer Beteiligung am Verfahren gezwungen werden kann. Daher müssen sich alle benötigten Beweismittel im Besitz solcher Benutzer befinden, deren Beteiligung gesichert ist. Dazu gehört die Partei, die das Verfahren veranlaßte und alle nicht anonymen Beteiligten, z.B. Notare, aber auch weitere anonyme, sofern ihnen Schaden entsteht, falls sie sich nicht beteiligen. So könnte z.B. eine anonyme Datenbank, die verklagt wird, Geld erhalten, aber keine befriedigende Auskunft geliefert zu haben, gezwungen sein, die gesendete Antwort offenzulegen, wenn andernfalls davon ausgegangen würde, daß sie gar nichts gesendet hat, so daß sie verurteilt werden würde. (Dazu, daß dies wirklich eine Drohung ist, vgl. §6.2.5. Außerdem ist zu beachten, daß es innerhalb des Verkehrsdaten schützenden Kommunikationsnetzes möglich ist, der Datenbank diese Vorladung unter ihrem Pseudonym sicher zuzustellen, vgl. §6.2.3.)

6.2.5 Schadensregulierung

Wurde festgestellt, daß ein unrechtmäßiger Zustand eingetreten ist, so muß gegebenenfalls analog zur heutigen Zwangsvollstreckung wieder ein rechtmäßiger Zustand hergestellt werden.

Im allgemeinen geschieht dies durch einen Eingriff in das Vermögen eines Benutzers. Dazu sollte es groß genug sein, der Zusammenhang zwischen den ermittelten Forderungen und diesem Vermögen klar sein und Zugriff auf das Vermögen oder den speziellen Teil, auf den sich die Forderungen beziehen, möglich sein.

Ob von vornherein gesichert werden kann, daß dem Benutzer genügend Vermögen gehört, ist nicht von der Anonymität, sondern von prinzipiellen Erwägungen bezüglich der jeweiligen Rechtsgeschäfte abhängig und auch derzeit nicht immer garantiert.

Die Erfüllung der restlichen Forderungen macht den Hauptunterschied zwischen anonymen und nicht anonymen Systemen aus:

Wird keine Anonymität gewünscht, so kann der Zusammenhang über den Namen (und weitere Angaben zur eindeutigen Identifizierung) hergestellt werden. Dies bedeutet, daß die Vermögensverwaltung und die Sicherstellung von Beweismitteln völlig unabhängig voneinander organisiert werden können, indem einem einerseits alles Vermögen namentlich gehört und sich andererseits alle Beweismittel auf namentlich bekannte Personen beziehen.

Soll die Anonymität gewahrt werden, so geht diese Unabhängigkeit verloren. Dies bedeutet nicht, daß es keine Möglichkeit zur Schadensregulierung mehr gibt, sondern nur, daß die Geschäftsabläufe komplizierter werden. Bereits bei der Sicherstellung von Beweismitteln muß immer auf einen Zusammenhang mit dem Vermögen, auf das evtl. zugegriffen werden soll, geachtet werden.

Aus diesem Grunde ergibt die in diesem Kapitel durchgeführte Betrachtung der Bestandteile eines Geschäftsablaufs nicht automatisch fertige Protokolle für den ganzen Geschäftsablauf, sondern nur Hilfsmittel, die noch geschickt kombiniert werden müssen.

Die im einzelnen notwendigen Regelungen hängen vor allem davon ab, wer aufgrund des Rechtsgeschäfts für wie lange wozu verpflichtet wird.

Entsteht der Ruf nach Schadensregulierung etwa aufgrund eines großen Kredites, so ist die Voraussetzung für die Garantie, daß Vermögen zugreifbar sein wird, stets die vorherige Übergabe von Sicherheiten in Form materieller Güter, z.B. eines Grundstückes. Dies kann durch Anwenden des Autorisierungsmechanismus (vgl. §6.2.1.2.2) ebenfalls anonym erfolgen, indem z.B. ein Grundbuchamt geeignete Bestätigungen über den Wert eines Grundstücks beglaubigt und das Grundstück

sodann als belastet vermerkt, doch sind Geschäfte dieser Größenordnung eher untypisch für offene Systeme.

Weit typischer dürften Geschäfte sein, die im Kauf von Waren, insbesondere Informationen, geringen Wertes bestehen. Hier verpflichtet sich bei der derzeitigen Geschäftsabwicklung lediglich jemand, innerhalb einer kurzen Zeitspanne bei Lieferung der Ware eine gewisse kleine Menge Geldes zu bezahlen. Wegen seiner Wichtigkeit für offene digitale Kommunikationsnetze, und um beispielhaft komplette Geschäftsabläufe darzustellen, wird der Kauf einer Ware gegen geringes Entgelt in einem eigenen Abschnitt (§6.3) dargestellt.

Mit zwei Ausnahmen kann das Erbringen einer Dienstleistung, z.B. die Verwaltung eines elektronischen Briefkastens (Mailbox), wie eine Folge vieler kleiner Kaufgeschäfte betrachtet werden. Die Verwaltung einer Mailbox könnte z.B. immer dann abgerechnet werden, wenn der Benutzer diese leeren möchte. Die Übergabe des Inhalts entspricht der Lieferung einer Ware.

Die beiden angesprochenen Ausnahmen hiervon sind die Dienstleistungen, die in Anspruch genommen werden müssen, um überhaupt ein Kaufgeschäft über das offene System abwickeln zu können: die Übermittlung von Nachrichten durch das Verkehrsdaten schützende Kommunikationsnetz und die Bereitstellung und Verwaltung von Geld durch ein anonymes digitales Zahlungssystem.

Das Kommunikationsnetz kann dabei z.B. pauschal bezahlt werden oder durch Hinzufügen von Geld (digitalen „Briefmarken“) zu jeder Nachricht, vgl. §5.6.2. Im ersten Fall können nicht zahlende Benutzer vom Kommunikationsnetz ausgeschlossen werden, im zweiten Fall werden unbezahlte Nachrichten nicht weitertransportiert.

Die willkürliche Schädigung eines Benutzers durch das Kommunikationsnetz müßte, da der Betreiber des Kommunikationsnetzes nicht anonym ist, wie heute üblich außerhalb des Systems evtl. gerichtlich verfolgt werden.

Die Dienste des anonymen digitalen Zahlungssystems können ebenso behandelt werden: Die beteiligten Banken selbst können die geforderten Gebühren direkt einbehalten, die Kunden die Banken notfalls verklagen (letzteres kann unter Wahrung der Anonymität der Kunden über das Kommunikationsnetz geschehen).

6.3 Betrugssicherer Werteaustausch

Klassische Bareinkäufe in Läden könnten aus Sicht der Rechtssicherheit problemlos in völliger Anonymität durchgeführt werden, denn das räumliche Zusammensein der Geschäftspartner sichert, daß entweder Ware und Geld den Besitzer wechseln oder keines, so daß (wenn man z.B. von nachträglichen Reklamationen absieht) nie eine Schadensregulierung nötig wird.

Von solcher Gleichzeitigkeit kann man bei einem Austausch von Ware und Geld über ein Kommunikationsnetz nicht ausgehen. Es werden zeitweise immer Geschäftspartner im Vorteil sein, so daß, wenn diese zu einem geeigneten Zeitpunkt die Kommunikation abbrechen, an sie Forderungen bestehen, die gegebenenfalls durchgesetzt werden müssen.

Die folgenden zwei Abschnitte beschreiben zwei Konzepte, wie die Durchsetzbarkeit von Forderungen gesichert werden kann. Wir bevorzugen das zweite.

6.3.1 Dritte garantieren die Deanonymisierbarkeit

Das erste Konzept garantiert die mögliche Deanonymisierung des Schuldners, falls dieser sich der Schadensregulierung widersetzen möchte, so daß dann wie im nichtanonymen Fall das gesamte Vermögen des Schuldners zur Verfügung steht.

Technisch wird diese Deanonymisierung ermöglicht, indem die am Geschäft Beteiligten sich durch eine besondere Form der Fremdauthentikation ausweisen: Sie identifizieren sich je gegenüber

einem allen vertrauenswürdig erscheinenden und nicht anonymen Dritten, der ihnen oder dem Partner das Zeugnis ausstellt, notfalls den Inhaber eines bestimmten Pseudonyms identifizieren zu können (nichtöffentliches Personenpseudonym, vgl. §6.1).

Hierbei können zur Authentikation ein Dritter [Herd_85] oder eine Folge von Dritten [Chau_81] verwendet werden. Das Prinzip ist in Bild 6-3 nochmals für einen Dritten (eventuell aber verschiedene für verschiedene Benutzer) dargestellt. Seien hierzu X und Y die Benutzer und A und B die nicht anonymen Instanzen, die X bzw. Y identifizieren können, und seien

- $p_G(X,g)$ das Pseudonym, das X im Geschäft g mit Y verwenden möchte,
- $p_{G'}(Y,g)$ das Pseudonym, das Y in diesem Geschäft (aber in einer anderen Rolle als X) verwenden möchte,
- p_A bzw. p_B die öffentlichen Personenpseudonyme von A bzw. B.

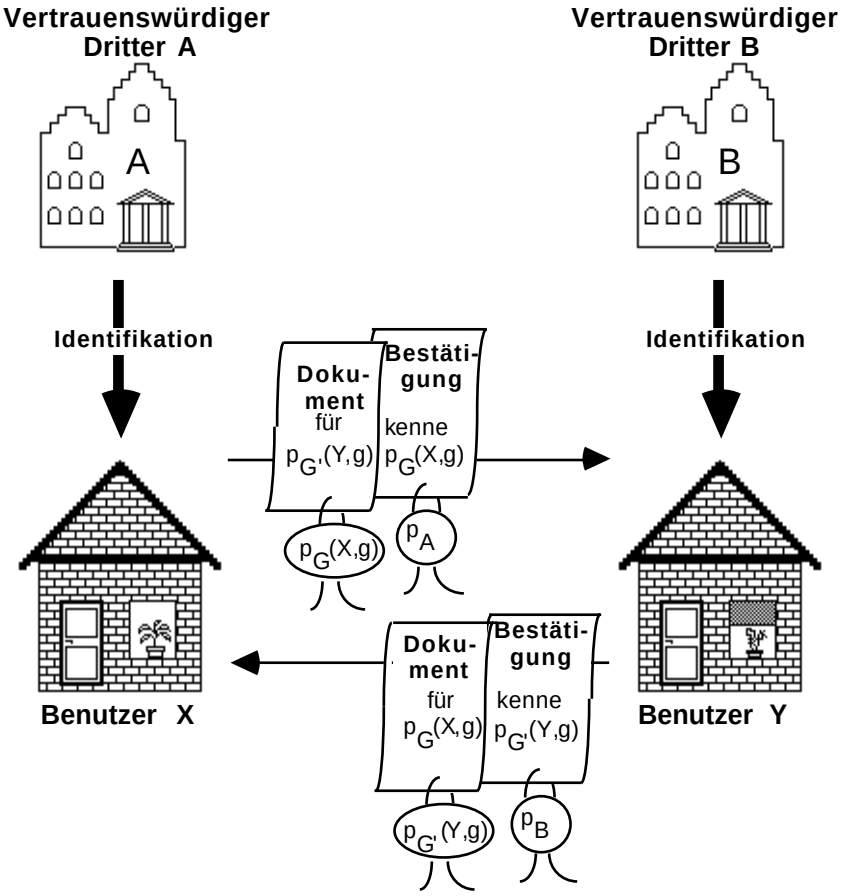


Bild 6-3: Authentisierte anonyme Erklärungen zwischen deanonymisierbaren Geschäftspartnern

Die Identifizierbarkeit von X durch A wird gewährleistet, indem X sich sein Pseudonym $p_G(X,g)$ von A signieren lassen muß, bevor er es verwenden kann. Analog muß sich Y sein Pseudonym $p_{G'}(Y,g)$ von B signieren lassen. Damit garantiert ist, daß A bei der Deanonymisierung stets die richtige Person identifiziert, muß X, bevor er von A sein Zeugnis erhält, z.B. mit seiner üblichen eigenhändigen Unterschrift oder, sicherer, mit einer zu einem öffentlichen Personenpseudonym gehörenden digitalen Signatur, die Erklärung „Das Pseudonym $p_G(X,g)$ gehört X“ unterschreiben und A aushändigen. Vertraut man dem vertrauenswürdigen Dritten nicht nur bezüglich der Anonymität, sondern auch bezüglich der Rechtssicherheit, so genügt hier auch die Verwendung eines Identifikationssystems (vgl. §6.2.1.2.1); z.B. könnte X sich durch seinen Fingerabdruck identifizieren. Analog wird bezüglich B verfahren.

Legen X und Y einander ihre signierten, also von A bzw. B bestätigten, Pseudonyme vor, bevor sie ein Rechtsgeschäft miteinander abwickeln, so wissen beide, daß bei Betrug X durch A und Y durch B identifiziert werden kann.

Im weiteren Verlauf des Geschäftes könnte es etwa geschehen, daß X sich an B wendet und beklagt, daß der Partner eine Kommunikationsbeziehung abgebrochen hat, obwohl er sie hätte fortführen müssen. X legt dazu Nachrichten von Y vor, die mit $p_{G'}(Y,g)$ signiert sind und die B beweisen, daß eine Kommunikationsbeziehung bestand. B fordert daraufhin Y auf weiterzumachen, wobei von da an alle Nachrichten von Y an X über B laufen müssen. Weigert sich Y, wird er durch B deanonymisiert und die Untersuchung wird nach herkömmlichen Verfahren wie im nicht anonymen Fall fortgesetzt.

Beschuldigt X den anderen fälschlicherweise, indem in der Folge der Erklärungen der Rest unterschlagen wird, so ist das Schlimmste, was passieren kann, daß Y den Rest der Nachrichten nochmal senden muß. Da sie diesmal über B weitergeleitet werden, kann X nicht mehr behaupten, er hätte diese Nachrichten nicht erhalten.

Entsprechendes gilt, wenn Y sich an A wendet und beklagt.

Die Vor- (+) und Nachteile (-) dieses ersten Lösungskonzeptes sind:

- Wer sollte kontrollieren, daß A und B nicht die Identität von X oder Y preisgeben, obwohl sie dazu gar nicht berechtigt sind? A und B müssen also bezüglich Vertraulichkeit absolut vertrauenswürdig sein, während dies bezüglich Betrugssicherheit nicht notwendig ist, da sie nicht die Zuordnung von Personen und Pseudonymen verfälschen können.
- Durch X oder Y können ungedeckte Schäden entstehen. Etwa könnte X bei Y eine Dienstleistung bestellen, die er auch erhält, aber zu deren Bezahlung ihm nicht das nötige Vermögen gehört. Dann kann trotz Deanonymisierung von X der Y entstandene Schaden nicht behoben werden.
- Das Verfahren legt es aus Aufwandsgründen nahe, für $p_G(X,g)$ bzw. $p_{G'}(Y,g)$ Personenpseudonyme zu verwenden, die, wie in §6.1 ausgeführt, zu einer allmählichen Deanonymisierung führen können.
+ Werden Personenpseudonyme verwendet, so ist kein Eingreifen von A und B in die einzelnen Geschäfte von X bzw. Y notwendig.

Wie die Fähigkeit zur Deanonymisierung auch auf mehrere Dritte verteilt werden kann, steht in [Chau_81 Seite 86]. Nach dem dort beschriebenen Verfahren kann der Träger des dem Partner übergebenen Pseudonyms nur durch Kooperation aller Dritter identifiziert werden. Erweiterungen dieses Schemas, um Ausfälle einiger Dritter zu tolerieren, sind in [Pfi1_85] beschrieben.

Trotz der erhöhten Anonymität durch Verwendung mehrerer Dritter hat die Deanonymisierung immer noch den Nachteil, das Vorhandensein ausreichender Vermögenswerte nicht zu garantieren.

6.3.2 Treuhänder garantiert anonymen Partnern die Betrugssicherheit

Um zu sichern, daß keine ungedeckten Schäden entstehen können, bietet sich ein sehr einfaches Verfahren an, das zudem auch ohne Deanonymisierung auskommt: das Deponieren des Geldes bei einem nicht anonymen Treuhänder, so daß Forderungen nur noch an diesen entstehen können [Pfi1_83 Seite 29-33, Waid_85, WaPf_85]. Die eigentlichen Geschäftspartner können dann völlig anonym voreinander wie auch vor dem Treuhänder sein. Sollte der Treuhänder das in ihn gesetzte Vertrauen mißbrauchen, so kann er, da er nicht anonym ist, in üblicher Weise verklagt werden, ohne daß die eigentlichen Geschäftspartner ihre völlige Anonymität aufgeben müssen.

Statt also direkt Geld und Waren untereinander auszutauschen, übergeben alle Beteiligten dem Treuhänder Informationen darüber, wieviel Geld bzw. welche Ware (Information) sie genau erhalten wollen und sodann das Geld und die Ware selbst (wobei es genau in dieser Reihenfolge geschehen

muß, denn erhält der Treuhänder zwar die Ware, nicht aber das Geld, so kann er dem Lieferanten den Schaden nicht ersetzen). Der Treuhänder prüft, ob das Erhaltene den Erwartungen der Beteiligten entspricht. Je nach dem Ergebnis der Prüfung gibt er das Erhaltene weiter oder bricht das Geschäft ab. Spätestens ab der Übergabe der Ware an den Treuhänder darf der Kunde, der die Ware bestellt hat, seine Bestellung nicht mehr stornieren.

Um Geschäfte schnell abwickeln zu können, muß dieser Treuhänder innerhalb des offenen digitalen Systems sein. Es könnte z.B. der Netzbetreiber sein, der ja auch zur Zeit ähnliche Aufgaben wahrnimmt.

Betrachtet man wieder die Benutzer X und Y, wobei willkürlich X der Kunde und Y der Lieferant einer Ware (in der Form einer Information) sei, so kann diese Idee wie in Bild 6-4 angegeben konkretisiert werden, wobei die Nummern der Dokumente die Reihenfolge der entsprechenden Erklärungen angeben. Dabei bezeichne

- $p_K(X,g)$ das Pseudonym von X als Kunde im Geschäft g,
- $p_L(Y,g)$ das Pseudonym von Y als Lieferant im Geschäft g und
- p_T das öffentliche Personenpseudonym des nicht anonymen Treuhänders T.

Die bildliche Darstellung beim Versenden des Geldes ist stark vereinfacht, da Geld natürlich nicht einfach als eine einzelne Information oder Erklärung dargestellt werden kann – man könnte es sonst durch Kopieren vermehren (vgl. §6.4).

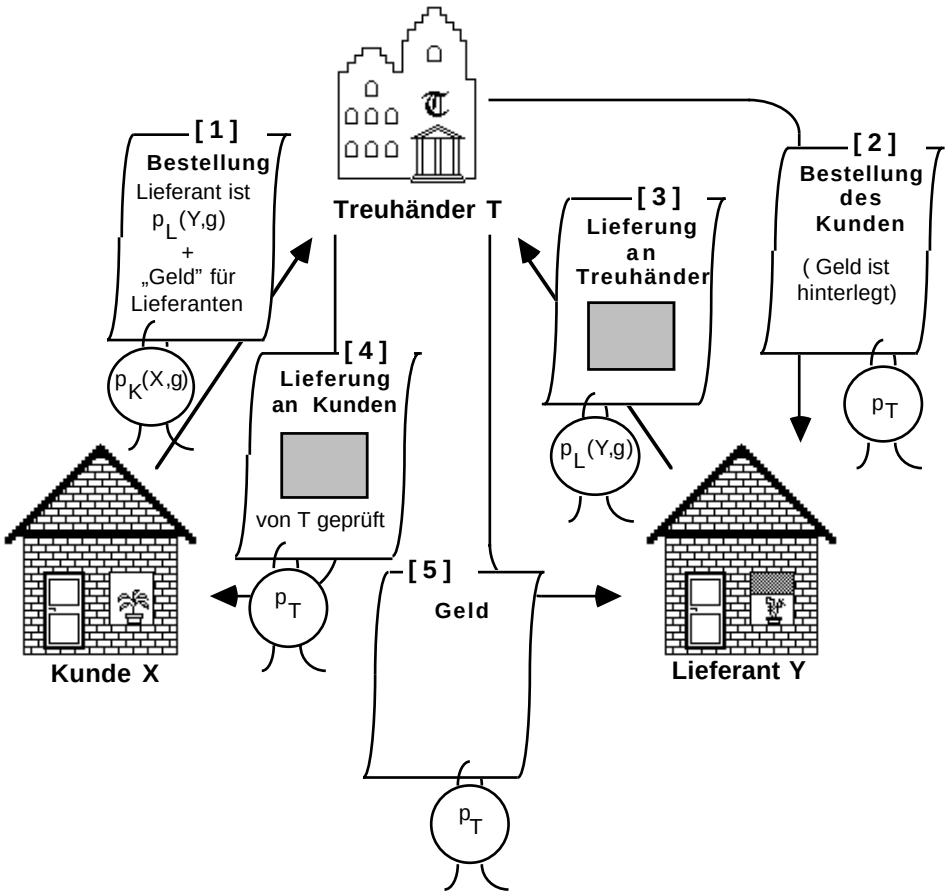


Bild 6-4: Betrugssicherheit für völlig anonyme Geschäftspartner durch aktiven Treuhänder, der Ware prüfen kann

Wie in §6.2.3 erläutert, kann die Sicherstellung von Beweismitteln für sämtliche Handlungen durch den Handelnden, also für das Deponieren und Weiterleiten bzw. Rückerstatten des Geldes und das

Liefern und Weiterleiten der Ware, entfallen und der Empfang der Bestellung (erst durch den Treuhänder, dann durch den Dienstanbieter) einfach durch Vorlage des Dokumentes bewiesen werden.

Dieses Konzept ist zwar aus Datenschutzgründen dem ersten vorzuziehen, hat aber einen Nachteil: es verlangt vom Treuhänder gewisse Prüfungen der Ware, die dieser nicht immer vornehmen kann bzw. aus Datenschutzgründen auch nicht immer vornehmen können soll.

Um diesen Nachteil zu lindern, können X und Y für die Ware eine Reklamationsfrist vereinbaren, während der der Treuhänder T zwar das erhaltene Geld einbehält, nicht aber die erhaltene Ware (Bild 6-5). Auf diese Weise muß der Treuhänder nur noch die „Echtheit“ des erhaltenen Geldes sichern, nicht aber die Ware überprüfen, also insbesondere über sie auch keine Informationen erhalten.

Ist der Kunde X mit der Ware nicht zufrieden, weil etwa eine gestellte Anfrage falsch beantwortet wurde, so kann er innerhalb der Reklamationsfrist dem Treuhänder T untersagen, das Geld weiterzugeben, und muß nun nachweisen, daß die Ware in der Tat fehlerhaft war.

Kann man sicherstellen, daß ein evtl. zu Rate zu ziehendes Gericht schnell genug arbeitet, so kann X auch durch Reklamation Y keinen größeren Schaden, etwa durch Zinsverlust, zufügen. Und selbst ein Schaden durch Zinsverlust kann vermieden werden, indem T das Geld gegen Habenzinsen anlegt und der Differenzbetrag zwischen den erzielten Haben- und den üblichen Sollzinsen sowohl von X als auch von Y als Zinskaution an T gefordert wird. Beide Zinskautionen werden vom Gericht dem Prozeßgewinner zugesprochen. Um X und Y zur Entrichtung der Zinskautionen oder einen Verzicht auf Reklamation zu zwingen, fällt bei Nichtentrichtung einer fälligen Zinskaution der Gesamtbetrag sofort der anderen Partei zu [BüPf_90].

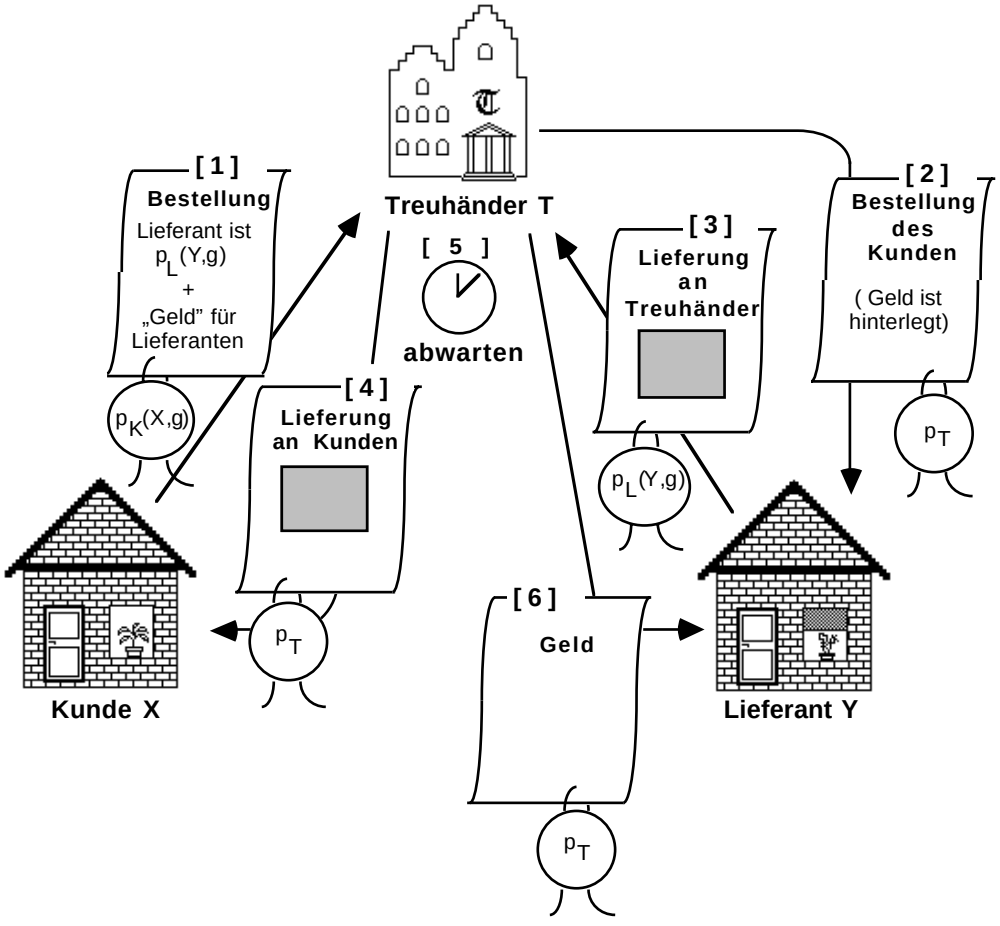


Bild 6-5: Betrugssicherheit für völlig anonyme Geschäftspartner durch aktiven Treuhänder, der Ware nicht prüfen kann

Wollen X und Y ein Geschäft „Ware gegen Ware“ abschließen, ohne daß der Treuhänder T die Waren prüfen kann oder können soll, so müssen sie es in zwei koordinierte Geschäfte „Ware gegen Geld“ zerlegen, um bei jedem Austausch die Reihenfolge „Geld vor Ware“ einhalten zu können. Die Koordination wird dadurch hergestellt, daß T weder X noch Y das Geld aushändigt, bevor nicht entweder beide ihre Zufriedenheit mit der jeweils erhaltenen Ware erklärt haben oder ein Gericht entschieden hat [BüPf_90].

Das so erweiterte Lösungskonzept des Treuhänders läßt sich wie folgt bewerten:

– Der Treuhänder muß stets aktiv in das Geschäft einbezogen werden.
+ Es ist nicht nötig, daß einer der am Geschäft direkt Beteiligten dem Treuhänder vertraut, da beide am Geschäft direkt Beteiligten den Treuhänder kontrollieren und Fehler oder Betrugsversuche durch ihn gerichtlich verfolgen lassen können. Die Existenz ausreichender Beweismittel ist gesichert, die Vollstreckbarkeit eines Anspruchs gegen den Treuhänder wie im nicht anonymen Fall absicherbar.
+ Alle Forderungen von X an Y oder umgekehrt lassen sich durch Rückgriff auf die bei T hinterlegten Werte befriedigen.
+ Die am Geschäft direkt Beteiligten können ohne Mehraufwand Transaktionspseudonyme verwenden.
+ Die Anonymität der am Geschäft direkt Beteiligten ist völlig gesichert.

Falls der Dienstanbieter nicht anonym sein will, etwa ein Zeitungsverlag, kann man natürlich gleich diesen als Treuhänder wählen, wodurch diese Abwicklungsform mit derjenigen zusammenfällt, bei der der Kaufwillige gleich bei der Bestellung bezahlt und der Dienstanbieter, wenn er nicht liefern kann oder will, das Geld zurückerstatten muß.

Alle hier vorgestellten Konzepte haben allerdings ein gemeinsames Problem offengelassen, nämlich das, wie in einem digitalen System Geld dargestellt und anonym transferiert werden kann. Dies soll im folgenden Abschnitt behandelt werden.

6.4 Anonyme digitale Zahlungssysteme

Ein Zahlungssystem soll seinen Benutzern dazu dienen, in sicherer Weise Geld zu transferieren.

Unabhängig vom jeweiligen Zahlungssystem ausgedrückt, ist Geld nichts anderes als eine rein quantitativ definierte Menge von Rechten. Um die Sicherheit und Anonymität von Zahlungssystemen betrachten zu können, muß man also nicht definieren, was bzw. wo genau in ihnen das Geld ist; im folgenden soll daher nur von Rechten gesprochen werden.

Insbesondere ist für die Sicherheit innerhalb eines Zahlungssystems ohne Belang, ob diese Rechte auf einem echten Guthaben oder (zum Teil) auf einem begrenzten Kredit beruhen, wobei natürlich ein Kredit außerhalb des Zahlungssystems hinreichend abgesichert sein muß.

Ein Zahlungssystem ist sicher (bzgl. Verfügbarkeit und Integrität), falls

- ein Benutzer erhaltene Rechte transferieren kann,
- er ein Recht nur dann verliert, wenn er hierzu den Willen hat,
- sofern ein zahlungswilliger Benutzer einen anderen Benutzer als Empfänger eindeutig bestimmt, auch nur dieser Empfänger das Recht erhält,
- er falls notwendig einen vollzogenen Transfer einem Dritten gegenüber nachweisen kann (Quittungsproblem) und
- die Benutzer auch bei Zusammenarbeit ihre Rechte an Geld nicht vermehren können.

Vertraut man nicht (nur) auf die Gutwilligkeit der Benutzer, so muß zumindest bei der Ausübung des Verfügungsrechtes in Form des Transfers das Recht nachgewiesen werden. Da hier nur Zahlungssysteme betrachtet werden, bei denen der ganze Transfer allein durch den Austausch digitaler

Nachrichten abgewickelt wird (i.allg. über ein Kommunikationsnetz) und da digitale Nachrichten beliebig kopiert werden können, die Rechte aber nach dem Transfer erlöschen müssen, genügt ein reiner Dokumentenbeweis nicht. Man benötigt daher zum Nachweis einen Zeugen, der die aktuelle Gültigkeit des Rechtes garantiert.

Um dem Zeugen seine Aufgabe zu ermöglichen, muß ihm jede Inanspruchnahme des Rechtes, das er bezeugen soll, bekannt sein, es darf also auch dann, wenn der Zahlungsempfänger dem Zahlenden vertraut, kein Übergang des Rechtes ohne Bestätigung durch den Zeugen stattfinden können.

In §6.4.1 bis §6.4.4 soll davon ausgegangen werden, daß die Benutzer in diesen Zeugen weder bezüglich Integrität noch bezüglich Vertraulichkeit volles Vertrauen zu setzen wünschen.

Tut man dies doch, etwa weil der Zeuge ein sicheres Gerät in Form einer elektronischen Brieftasche ist, so vereinfacht sich das Problem erheblich. Hierauf wird in §6.4.5 näher eingegangen.

Ein Zahlungssystem, das zwar im obigen Sinne nicht sicher ist, aber ganz ohne aktiven Zeugen auskommt, wird in §6.4.6 beschrieben.

6.4.1 Grundschema eines sicheren und anonymen digitalen Zahlungssystems

Im folgenden soll diskutiert werden, wie ein sicheres und anonymes digitales Zahlungssystem mit Zeugen realisiert werden kann.

Hierzu wird angenommen, daß ein Benutzer X an einen anderen Benutzer Y des Zahlungssystems ein Recht transferieren möchte und ein Zeuge B (für Bank) diesen Transfer bestätigt. Der Einfachheit halber soll von nur einem Zeugen ausgegangen werden. Dieser eine Zeuge soll für Falschzeugnisse haftbar sein, d.h. entstehen aufgrund seines Falschzeugnisses neue Rechte an Geld, so muß er dieses Geld bereitstellen, und weigert er sich, vorhandene Rechte zu bezeugen, so kann der davon Betroffene dies einem objektiven Dritten (z.B. einem Gericht) beweisen. Dies wird erreicht, indem man diesen Zeugen finanziell hinreichend abgesichert und nicht anonym wählt. Er übernimmt damit in gewissem Sinne die Rolle einer Bank in herkömmlichen bargeldlosen Zahlungssystemen.

Man kann davon ausgehen, daß der Zahlende X und der Empfänger Y sich bereits unter gewissen Pseudonymen (vgl. §6.1) kennen. Diese Pseudonyme sind also von außerhalb des Zahlungssystems vorgegeben und werden als schützenswert betrachtet (i.allg. Rollenpseudonyme, z.B. Kundennummer und Bezeichnung eines Dienstanbieters). Ebenso vorgegeben ist das Pseudonym des Zeugen, der aber nicht anonym sein darf (öffentliches Personenpseudonym). Außerdem ist innerhalb des Zahlungssystems durch frühere Zahlungen bereits festgelegt, unter welchem Pseudonym X sich gegenüber dem Zeugen B als Inhaber des Rechtes, das er transferieren will, ausweisen kann. Seien also

- $p_Z(X,t)$ das Pseudonym des Zahlenden X im Transfer t gegenüber dem Empfänger,
- $p_E(Y,t)$ das Pseudonym des Empfängers Y im Transfer t gegenüber dem Zahlenden,
- p_B das für viele Zahlungen gleiche Pseudonym des Zeugen B und
- $p_Z^B(X,t)$ das Pseudonym des Zahlenden X im Transfer t gegenüber dem Zeugen B.

Unter diesen Voraussetzungen ergeben sich, analog zu heutigen Überweisungen, als Protokoll zum Transfer t des Rechtes von X an Y die folgenden Ablaufschritte. Schritt [*i*+1] muß jeweils nach Schritt [*i*] stattfinden. Lediglich die Schritte [4] und [5] können in beliebiger Reihenfolge oder auch parallel ausgeführt werden.

[1] **Pseudonymwahl.** Y wählt sich ein Pseudonym $p_E^B(Y,t)$, unter dem er dem Zeugen B als Empfänger des Rechtes im Transfer t bekannt sein möchte, und teilt X mit, daß er das Recht unter diesem Pseudonym $p_E^B(Y,t)$ erhalten möchte. Entsprechend teilt X das Pseudonym $p_Z^B(X,t)$, unter dem er das Recht transferieren will, Y mit. Die notwendigen Erklärungen sind mit $p_E(Y,t)$ bzw. $p_Z(X,t)$ authentisiert.

[2] **Transferauftrag des Zahlenden.** X erteilt dem Zeugen B den Auftrag, das Recht an $p_E^B(Y,t)$ zu übertragen. Dieser Auftrag ist mit $p_Z^B(X,t)$ signiert. Als Fremdauthentikation legt X diesem Auftrag eine Autorisierung bei, die besagt, daß $p_Z^B(X,t)$ über das zu transferierende Recht verfügt und von B selbst mit p_B signiert ist. Da jeder Transfer von B beglaubigt sein muß, kann B nachprüfen, ob $p_Z^B(X,t)$ über das beglaubigte Recht tatsächlich noch verfügt oder es bereits transferiert wurde.

[3] **Bestätigung des Zeugen.** Der Zeuge B bestätigt X und Y den Transfer des Rechtes von $p_Z^B(X,t)$ auf $p_E^B(Y,t)$, wobei er sie unter diesen Pseudonymen adressiert.

[4] **Quittung für den Zahlenden.** Der Empfänger Y sendet an X eine Quittung, die nur $p_Z(X,t)$ und $p_E(Y,t)$ bezeichnet und mit $p_E(Y,t)$ authentisiert ist, und die den Erhalt des Rechtes bestätigt.

Verweigert Y die Quittung (was i.allg. nicht zu verhindern ist, da Y anonym ist), so kann X die Bestätigung des Transfers durch B (aus [3]) zusammen mit der Bestätigung von Y, das Recht unter diesem neuen Pseudonym $p_E^B(Y,t)$ empfangen zu wollen (aus [1]), als Ersatzquittung verwenden.

Genau diese Möglichkeit unterscheidet das Quittungsproblem vom allgemeinen Werteaustauschproblem, bei dem es nicht möglich ist, daß ein Dritter, etwa der Treuhänder, eines der beiden Tauschobjekte ersatzweise erzeugt.

[5] **Bestätigung für den Empfänger.** Der Zahlende X sendet an Y eine Bestätigung des Transfers, die nur $p_Z(X,t)$ und $p_E(Y,t)$ bezeichnet und mit $p_Z(X,t)$ authentisiert ist.

Auch Y kann notfalls die Bestätigung von B (aus [3]) zusammen mit der Bestätigung von X (aus [1]), das Recht an Y transferieren zu wollen, als Beweis dafür verwenden, das Recht von $p_Z(X,t)$ empfangen zu haben.

[6] **Umformen der Bestätigung.** Y wird die auf $p_E^B(Y,t)$ ausgestellte Bestätigung von B, das Recht erhalten zu haben, bei einem zukünftigen Transfer t' in Schritt [2] verwenden wollen. Um Verkettbarkeiten und damit mögliche Deanonymisierung zu vermeiden, sollte dort nicht $p_E^B(Y,t)$ als $p_Z^B(Y,t')$ verwendet werden.

Durch Verwendung der in §6.2.1.2.2 erwähnten umrechenbaren Autorisierungen (die Beschreibung einer Implementierung folgt sofort) wird erreicht, daß Y die Bestätigung auf ein neues Pseudonym umrechnen kann. Dazu muß Y allerdings bereits in Schritt [1] des Transfers t zuerst das künftige Pseudonym $p_Z^B(Y,t')$ gewählt und daraus ein zur Umrechnung geeignetes $p_E^B(Y,t)$ gebildet haben.

Implementierung (einmal) umrechenbarer Autorisierungen mittels RSA (oder: Wie nutze ich den Angriff von Davida, vgl. §3.6.3.1, zum blinden Leisten von Signaturen, vgl. §3.9.5):

Wähle Pseudonym p (= Testschlüssel eines beliebigen digitalen Signatursystems).[125]
Bilde mittels kollisionsresistenter Hashfunktion h: p,h(p).
Sei *t*, *n* der öffentliche RSA-Testschlüssel desjenigen, der die Autorisierung authentiziert, z.B. der Bank.
Wähle Zufallszahl *r* gleichverteilt in $\mathbb{Z}_n^*$. (Zur Erinnerung: Dies bedeutet $1 \le r < n$, und *r* besitzt multiplikatives Inverses mod *n*.)
Berechne $(p,h(p)) \cdot r^t \pmod n$.
Lasse dies vom Inhaber von *t* signieren.

125 Der Bezeichner p für Pseudonym hat nichts mit dem Bezeichner *p* für eine der beiden Primzahlen bei der RSA-Schlüsselgenerierung zu tun.

(Anmerkung: Ist $(p,h(p)) \in \mathbb{Z}_n^*$, so erfährt der Inhaber von t dadurch nichts über $(p,h(p))$, da r^t gleichverteilt in $\mathbb{Z}_n^*$ ist. Andernfalls benötigt der Pseudonymgenerierer die Hilfe des Inhabers von t überhaupt nicht: $ggT((p,h(p)),n)$ ergibt einen nichttrivialen Faktor von n. Damit hat der Pseudonymgenerierer RSA vollständig gebrochen, vgl. §3.1.3.1 und §3.6.1, so daß er sich sein Pseudonym selbst signieren kann.)

Erhalte $((p,h(p)) \cdot r^t)^s \equiv_n (p,h(p))^s \cdot r$

Multipliziere mit dem multiplikativen Inversen[126] von r, erhalte $(p,h(p))^s$

Das Protokoll ist in Bild 6-6 dargestellt, wobei die Authentikationen wieder als Siegel abgebildet sind.

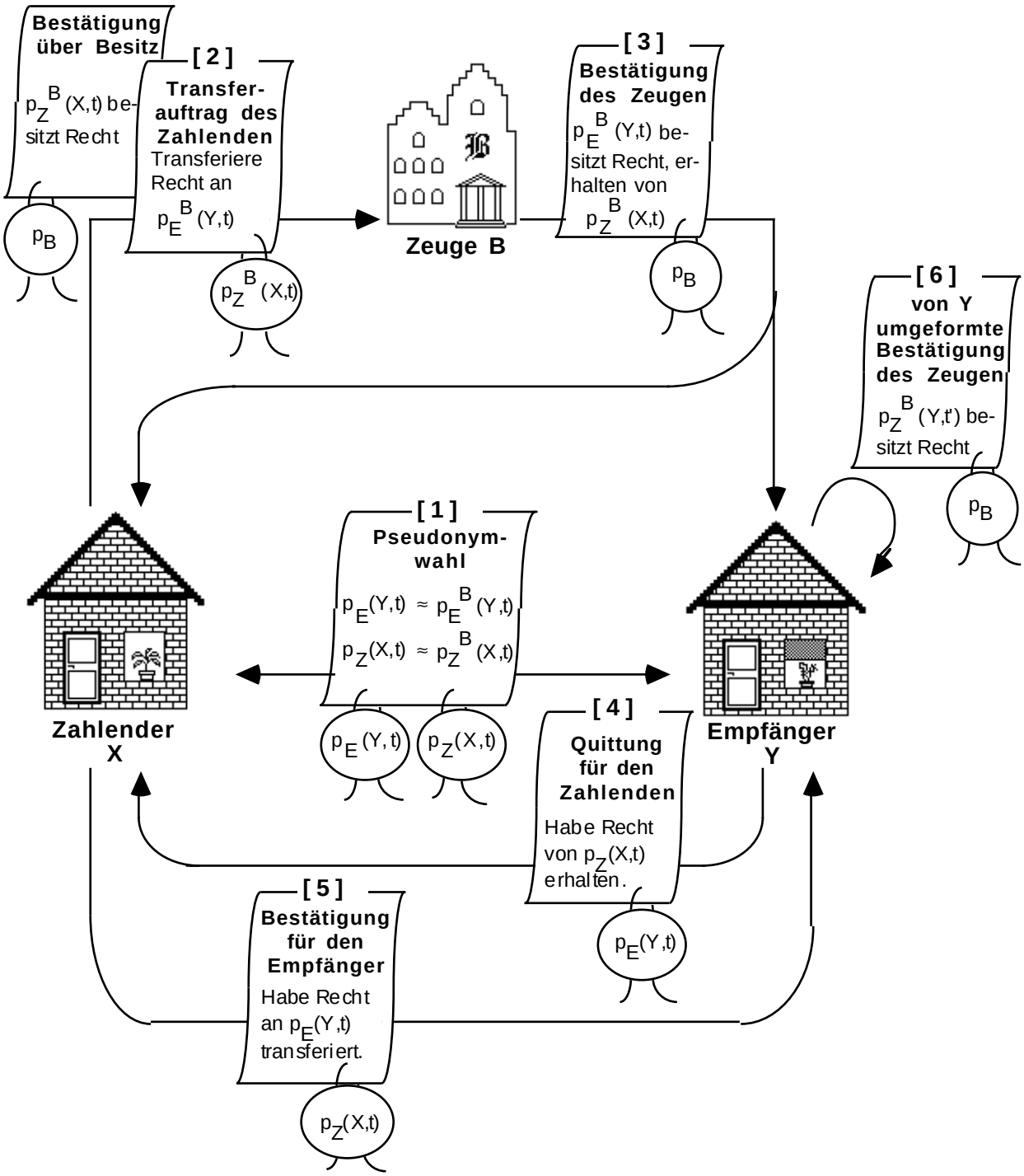


Bild 6-6: Grundschema eines sicheren und anonymen digitalen Zahlungssystems

126 Dies wird mit dem erweiterten Euklidschen Algorithmus bestimmt, der in §3.4.1.1 beschrieben ist.

Da man die in [6] erhaltene Bestätigung über den Empfang des Rechtes in einem zukünftigen Transfer verwendet, um dasselbe Recht, d.h. denselben Geldbetrag zu transferieren, ist es sinnvoll, in diesem Zahlungssystem wie bei Bargeld Rechte vorgegebener Nennwerte zu verwenden, aus denen man bei jeder Zahlung den gewünschten Betrag zusammensetzt. Natürlich muß man auch bei B Geld wechseln dürfen.

Damit man den umgeformten Bestätigungen bei ihrer Verwendung in [2] den Nennwert ansieht, verwendet B für jeden Nennwert N eine andere digitale Signatur, d.h. ein eigenes Pseudonym $p_{B,N}$.

Die Sicherheit des Protokolls ergibt sich daraus, daß am Ende eines Transfers jeder der drei Beteiligten genügend Dokumente über dessen Stattfinden hat (die er aufbewahren muß) und auch während des Transfers jeder einem objektiven Dritten stets entweder den aktuellen Zustand beweisen oder diesen überprüfbar herstellen kann, indem er die bisher von anderen erhaltenen Nachrichten vorlegt und seine eigenen, sofern deren Erhalt abgestritten wird, noch einmal sendet.

Außerdem können Forderungen innerhalb eines Transfers, die eingetrieben werden müssen, nur an den Zeugen entstehen, der nicht anonym ist.

Halten sich X und Y an das Protokoll, so ist die Anonymität des Protokolls maximal, da keiner durch den Transfer über einen anderen irgendwelche neue Information erhält: Der Zeuge erfährt bei einer Zahlung keines der Pseudonyme, die X und Y sonst verwenden, sondern nur zwei neue, die außer in dieser einen Zahlung nie mehr vorkommen. X und Y erfahren zwar voneinander auch die Pseudonyme, die sie bei dieser Zahlung gegenüber dem Zeugen verwenden, aber da diese mit nichts verkettbar sind und ohnehin klar war, daß irgendwelche Pseudonyme gegenüber dem Zeugen verwendet werden, stellt auch dies keine neue Information dar.

Daß die Anonymität des Protokolls maximal ist, heißt aber noch nicht, daß in jeder Situation starke Anonymität herrscht, denn es gibt andere Möglichkeiten zur Informationsgewinnung.

Zum einen gibt es außer den eigens gewählten Pseudonymen noch andere Kennzeichen für die Benutzer, die die Partner unabhängig vom gewählten Protokoll erfahren müssen. Hier sind das vor allem Betrag und Zeitpunkt der Zahlung. Insbesondere ist der Zahlende nur unter all denjenigen verborgen, die zu dieser Zeit ein Recht dieses Betrags haben können, was einen zusätzlichen Grund darstellt, nur wenige feste Nennwerte zuzulassen.

Zum anderen erzeugt das Zahlungssystem selbstverständlich keine zusätzliche Anonymität für Pseudonyme, die auch in anderen Situationen verwendet werden. Muß z.B. X den Transfer nachweisen, so werden dadurch die Pseudonyme $p_Z(X,t)$ und $p_E(Y,t)$ miteinander verkettet; dies ist gerade das Ziel des Nachweises.

Die Benutzer können also völlig selbst bestimmen, wieviel der ihnen ermöglichten Anonymität sie aufgeben wollen, indem sie manche Pseudonyme mehrfach verwenden oder durch Erklärungen verketten.

Hat Y in [4] die Quittung verweigert, so verwendet X als Ersatzquittung die Bestätigung des Zeugen aus [3] und die Bestätigung von Y aus [1]. Durch Verwendung der Ersatzquittung gegenüber einem Dritten (ungleich Y) erfährt der Zeuge also möglicherweise die Zuordnung von $p_Z{}^B(X,t)$ zu $p_Z(X,t)$ und von $p_E{}^B(Y,t)$ zu $p_E(Y,t)$, so daß im Verweigerungsfall die Anonymität leicht eingeschränkt wird.

Für den Fall, daß X und Y einander vertrauen und auf eine Quittung verzichten können, kann man die Bestätigung des Zeugen B an X über den Vollzug des Transfers einsparen. Die Bestätigung von Y in [1] ist aber weiterhin nötig, da X erst durch sie das Pseudonym $p_E{}^B(Y,t)$ von Y erfährt, und sie muß auch nach wie vor von Y mit $p_E(Y,t)$ authentisiert werden, damit X sicher ist, daß er sein Recht an den Richtigen transferiert. Außerdem muß in diesem Fall entweder X schon in [1] Y verbindlich mitteilen, was für eine Zahlung stattfinden wird, oder sie müssen sich am Ende noch einmal über deren Zustandekommen verständigen, da andernfalls nicht bemerkt würde, wenn beim Transfer, also entweder bei der Übertragung oder durch B, ein Fehler aufträte.

6.4.2 Einschränkung der Anonymität durch vorgegebene Konten

Es wäre denkbar, daß ein so vollständig anonymes Zahlungssystem wie das in §6.4.1 beschriebene nicht erwünscht ist, da dabei, zumindest ohne zusätzliche Vorkehrungen, niemand (insbesondere auch nicht das Finanzamt) Aussagen über das Eigentum oder Einkommen der Benutzer machen kann. Allerdings gilt das gleiche auch für gewöhnliches Bargeld, und selbst in bisherigen bargeldlosen Zahlungssystemen wird die im Prinzip vorhandene Information über Personen mit Konten bei mehreren Banken zumindest offiziell nur sehr selten zusammengeführt.

Aus diesen Gründen könnte jedem Benutzer nur die Inhabe eines einzigen Kontos (oder einer bekannten Anzahl) gestattet sein, über das alle Zahlungen abgewickelt werden müssen. Dies kann durch Nichtanonymität der Konten oder mittels umrechenbarer Autorisierungen für die Einrichtung anonymer Konten erzwungen werden. Auch für diese Situation kann man durch Abwandlung des Protokolls aus §6.4.1 ein sicheres Zahlungssystem konstruieren, das so anonym wie unter dieser Voraussetzung möglich ist.

Die Voraussetzung läßt sich dadurch ausdrücken, daß $p_Z^B(X,t)$ oder $p_E^B(Y,t)$ diesmal für alle Zahlungen eines Benutzers X bzw. Y gleich, d.h. von t unabhängig ist. Damit ist das Protokoll aus §6.4.1 nicht mehr völlig anonym, da X und Y einander in Schritt [1] unnötigerweise ihre sie eindeutig kennzeichnenden Kontonummern $p_Z^B(X,t)$ und $p_E^B(Y,t)$ mitteilen würden und B sähe, zwischen welchen zweien der festen Konten ein Transfer stattfindet.

Die Lösung ist, anonyme, bei jedem Transfer wechselnde Zwischenpseudonyme einzuführen, so daß zunächst X das Geld von seinem Konto abhebt und auf sein Zwischenpseudonym transferiert, dann unter diesem an Y unter dessen Zwischenpseudonym zahlt, und zuletzt Y das Geld wieder auf sein Konto einzahlt. Die einzelnen Transfers laufen dabei nach dem Protokoll aus §6.4.1 ab, insbesondere werden die Autorisierungen, das Recht erhalten zu haben, zwischen den Teiltransfers auf andere Pseudonyme umgerechnet, so daß eigentlich Zahlender und Empfänger je zwei zusammengehörende Zwischenpseudonyme haben. Bezeichne

- $p_K(X)$ das vorgegebene, zum Konto gehörende Pseudonym einer Person X,
- $p_{ab}(X,t)$ das Pseudonym des Zahlenden X im Transfer t, unter dem er das Geld abhebt,
- $p_{ZwZ}(X,t)$ das Zwischenpseudonym des Zahlenden X im Transfer t, d.h. dasjenige, unter dem er an Y zahlt,
- $p_{ZwE}(Y,t)$ das Zwischenpseudonym des Empfängers Y im Transfer t, d.h. dasjenige, unter dem er das Recht von X empfängt, und
- $p_{ein}(Y,t)$ das Pseudonym des Empfängers Y im Transfer t, unter dem er das Geld einzahlt.

Davon wählt also X zuerst $p_{ZwZ}(X,t)$, berechnet ein passendes $p_{ab}(X,t)$, läßt als ersten Teiltransfer das Recht von $p_K(X)$ auf $p_{ab}(X,t)$ übertragen und rechnet die Bestätigung auf $p_{ZwZ}(X,t)$ um. Bei diesem Teiltransfer können die Schritte [1], [4] und [5] des Protokolls wegfallen, da X sich nicht seine eigenen Pseudonyme mitteilen muß und auch keine Quittung des Transfers benötigt. Auch das Beilegen der Autorisierung in [2] kann unterbleiben, da B selbst den Kontostand weiß; nur im Streitfalle wird die Autorisierung benötigt.

Nun muß Y sein Pseudonym $p_{ein}(Y,t)$ wählen und ein dazu passendes $p_{ZwE}(Y,t)$ berechnen. Daraufhin wird das Recht nach dem vollständigen Protokoll aus §6.4.1 von $p_{ZwZ}(X,t)$ auf $p_{ZwE}(Y,t)$ übertragen, wobei auch die Quittungen entstehen.

Nachdem Y die Bestätigung über den Erhalt des Rechtes auf $p_{ein}(Y,t)$ umgerechnet hat, transferiert er es auf $p_K(Y)$, wiederum unter Auslassung von [1], [4], [5] und diesmal auch des Schrittes [6].

Die Dokumente, die der Zeuge B im ersten bzw. zweiten Teiltransfer als Bestätigung über den Erhalt der Zahlung ausstellt, dürfen dabei nicht gleich aussehen, denn andernfalls könnte Y die Bestätigung aus dem zweiten Teiltransfer unter Umgehung des Kontos unmittelbar für eine neue Zahlung

verwenden. Im Rahmen der umrechenbaren Autorisierungen bedeutet dies, daß B für Abhebungen und Übertragungen zwei verschiedene digitale Signaturen, also zwei Pseudonyme, verwendet. Unterscheiden kann er die Fälle anhand dessen, ob das ihm mitgeteilte Pseudonym des Zahlenden zu einem Konto gehört oder nicht.

Damit Y innerhalb einer abschätzbaren Zeit sein Recht auf $p_K(Y)$ transferieren muß (was die Voraussetzung z.B. einer jährlichen Besteuerung wäre), darf eine Autorisierung von B über eine Übertragung nicht beliebig lange zur Einzahlung berechtigen. B sollte nach Ablauf einer bestimmten Frist die zur Authentikation verwendeten Signierschlüssel tauschen und nach einer weiteren Frist, die allen Zahlungsempfängern Zeit gibt, erhaltene Rechte auf Kontopseudonyme zu transferieren, Autorisierungen, die mit dem alten Signierschlüssel authentisiert sind, nicht mehr akzeptieren.

Die Sicherheit dieses Zahlungssystems folgt aus der Sicherheit des Zahlungssystems aus §6.4.1, da es sich nur um eine spezielle Verwendung desselben handelt.

Die Anonymität ist deswegen maximal unter der Voraussetzung, daß es feste Konten gibt, weil die schützenswerten Pseudonyme $p_K(X)$, $p_K(Y)$, $p_Z(X,t)$ und $p_E(Y,t)$ durch die Zwischenpseudonyme und das Umrechnen der Autorisierungen völlig voneinander entkoppelt werden. Genauer heißt dies, daß weder Y das Pseudonym $p_K(X)$ noch X das Pseudonym $p_K(Y)$ noch B die Pseudonyme $p_Z(X,t)$ oder $p_E(Y,t)$ erfährt und daß jeder die Pseudonyme der anderen, die er schon kannte, aufgrund einer Zahlung zusätzlich nur mit neugewählten Pseudonymen verketten kann, die ansonsten nie mehr verwendet werden, also auch keine Information liefern.

Ansonsten gelten für Anonymität und Quittungen die gleichen Anmerkungen wie am Ende von §6.4.1. Dabei ist zu beachten, daß Betrag und Zeit einer Zahlung hier wesentlichere Verkettungen erlauben können als in §6.4.1, nämlich das Erkennen des Zusammenhangs zwischen den beiden an einer Zahlung beteiligten Konten. Um dies zu vermeiden, sollte zwischen den einzelnen Teiltransfers eine zufällig gewählte Zeitspanne verstreichen und der gesamte für eine Zahlung benötigte Betrag nicht auf einmal abgehoben bzw. eingezahlt werden.

6.4.3 Aus der Literatur bekannte Vorschläge

Die in §6.4.1 und §6.4.2 beschriebenen völlig anonymen und sicheren Zahlungssysteme sind durch Kombination von Elementen früher beschriebener anonymer digitaler Zahlungssysteme entstanden. Diese lassen sich aber am einfachsten und systematischsten als Abschwächungen obiger Systeme beschreiben, weshalb sie erst jetzt behandelt werden.

Von DAVID CHAUM, dem Erfinder des Mechanismus für das Umrechnen von Autorisierungen, stammt auch die Idee, diese Umrechnung in Zahlungssystemen anzuwenden, wenngleich er sie in diesem Zusammenhang anders nennt, und eine Reihe darauf beruhender verwandter anonymer Zahlungssysteme [Chau_83, Cha8_85, Chau_87, Chau_89].

Allen Vorschlägen von CHAUM ist gemeinsam, daß sie von der Existenz fester, nicht anonymer Konten ausgehen. Auch enthalten sie alle keine Quittungen; statt dessen gibt es Varianten, die, ähnlich wie in §6.3.1, Deanonymisierung von Zahlendem oder Empfänger bei Zusammenarbeit des Partners und des Zeugen erlauben.

Der andere wesentliche Unterschied von CHAUMs Zahlungssystemen zu dem in §6.4.2 beschriebenen ist, daß sie, bis auf eine Variante [Chau_89 §4.2.1], keinen Gebrauch davon machen, daß man die Zwischenpseudonyme so wählen kann, daß zu ihnen eine digitale Signatur existiert. Dies erschwert es, den Benutzern Sicherheit, insbesondere gegen den Zeugen B, zu garantieren, da der Inhaber eines Rechtes nun nicht mehr wie bei einer Überweisung eindeutig durch seine Signatur authentisiert erklären kann, wohin es transferiert werden soll. In den einfacheren Varianten wird diese Sicherheit gar nicht oder nur durch Einschränkung der Anonymität erreicht. In einer weiteren Variante [Chau_89 §4.2.2] wird zumindest Sicherheit gegen den Zeugen B erzielt, indem B dem Zahlungs-

empfänger signieren muß, daß er das Recht nur ihm gutschreiben wird, bevor ihm das Dokument über die Inhabe des Rechtes vorgelegt wird. Hierzu händigt der Zahlende das Dokument dem Empfänger aus. Die Gültigkeit der Signatur von B muß natürlich begrenzt werden, da B sich sonst für immer weigern könnte, ein Recht zu transferieren mit der Begründung, er hätte schon jemand anderem bestätigt, es ihm gutzuschreiben, nur dieser hätte das Dokument über die Inhabe noch nicht vorgelegt.

Der Verzicht auf namentliche Konten bei digitalen Zahlungssystemen kam erstmals im Zahlungssystem der anonymen Nummernkonten [Pfi1_83, WaPf_85] vor, das ansonsten genau die üblichen Überweisungen mit digitalen Signaturen nachbildet. Die Anonymität der Konten ist dabei aber im Grunde keine Eigenschaft des Zahlungssystems, da sie nur etwas über Verkettungsmöglichkeiten der im Zahlungssystem verwendeten Pseudonyme nach außen aussagt und nichts mit der Sicherheit zu tun hat (sofern Kontoüberziehung verboten ist). Allgemein wäre jedes im strengen Sinne sichere digitale Zahlungssystem, das namentliche Konten verwendet, auch mit anonymen Nummernkonten sicher.

Die Verwendung von Transaktionspseudonymen gegenüber der Bank (also $p_Z^B(X,t)$ und $p_E^B(Y,t)$ in obigem Protokoll) wurde in [Bürk_86, BüPf_87, BüPf_89] vorgeschlagen. Das dortige Zahlungssystem entspricht dem aus §6.4.1, wenn man auf das Umrechnen der Autorisierungen verzichtet. Die Sicherheit wird durch diesen Verzicht nicht eingeschränkt, jedoch die Anonymität, wenn auch nur in geringem Maße, da bei jedem Transfer t das Pseudonym $p_Z^B(X,t)$ gleich dem Pseudonym $p_E^B(X,t')$ aus einem früheren Transfer t' ist. Der Zeuge B kann also erkennen, daß es sich beide Male um dieselbe Person handelt, und falls der Zahlende W aus dem Transfer t' und der Empfänger Y aus dem Transfer t zusammenarbeiten (was besonders wahrscheinlich ist, wenn W und Y derselbe sind), können auch sie dies erkennen und dadurch die eigentlich schützenswerten Pseudonyme $p_Z(X,t)$ und $p_E(X,t')$ verketten.

Dieses System bliebe übrig, falls der Mechanismus der umrechenbaren Autorisierungen aus §6.2.1.2.2, der zur Zeit nur mit dem speziellen Krypto- bzw. Signatursystem RSA implementiert werden kann, gebrochen würde, andere Systeme für digitale Signaturen jedoch Bestand hätten. Da man von manchen anderen Systemen wenigstens beweisen kann, daß ihr Brechen ebenso schwierig ist wie das Lösen seit langem als schwierig bekannter mathematischer Probleme, ist dies nicht völlig unwahrscheinlich.

6.4.4 Einige Randbedingungen für ein anonymes Zahlungssystem

Die Verwendung eines anonymen Zahlungssystems sollte niemandem im Vergleich zu den heutigen Zahlungssystemen Nachteile bringen, insbesondere sollte ein Benutzer

- frei unter mehreren Banken als Zeugen seiner Zahlungen wählen können,
- Geld beliebig transferieren können, insbesondere auch in ein konventionelles Zahlungssystem außerhalb des offenen Systems und umgekehrt,
- Geld gegen Zinsen anonym anlegen können und
- entsprechend Geld gegen Zinsen anonym entleihen können.

Neben den Anforderungen der Benutzer an die bereitzustellenden Dienste sind aber auch die Forderungen der Banken und des Staates zu beachten, z.B. darf ein anonymes Zahlungssystem die vermögens- und einkommensabhängige Besteuerung nicht verhindern.

Hier sind insbesondere auch die Grenzen sinnvoller Anonymität zu bedenken, die in diesem Text jedoch als außerhalb der Grenzen des offenen digitalen Systems angenommen und daher nicht näher betrachtet werden. Ohne ihre Betrachtung sind jedoch sinnvolle Aussagen über Sinn und Möglichkeit z.B. einer Besteuerung trotz und in Anonymität nicht möglich.

6.4.4.1 Transfer zwischen Zahlungssystemen

In den in §6.4.1 und §6.4.2 betrachteten Zahlungssystemen wurde stets von nur einem Zeugen, also einer Bank, ausgegangen. Sowohl aus wirtschaftspolitischen als auch aus Anonymitätsgründen sollte es ein anonymes Zahlungssystem aber erlauben, daß Zahlender und Empfänger verschiedene Banken verwenden, ja sogar, daß sie verschiedene Zahlungssysteme verwenden.

Verwenden Zahlender und Empfänger beide das anonyme Zahlungssystem, aber verschiedene Zeugen, so muß in Schritt [1] des Protokolls aus §6.4.1 der Empfänger Y neben dem Pseudonym $p_E^B(Y,t)$ noch einen Zeugen seines Vertrauens, B_E bestimmen. Im weiteren teilt der Zahlende in [2] seinem Zeugen B_Z den neuen Zeugen B_E mit. Da B_Z und B_E nicht anonym voreinander sind, können diese nun hinsichtlich der Zahlung beliebig kooperieren und als ein einziger Zeuge B betrachtet werden, wobei alle Kommunikation zwischen B und X von B_Z, alle zwischen B und Y von B_E übernommen wird.

Der Transfer zwischen einem nicht anonymen und dem anonymen Zahlungssystem kann sehr einfach realisiert werden: Der Betreiber des nicht anonymen Zahlungssystems, also eine Bank, spielt die Rolle eines Mittlers, der in beiden Zahlungssystemen auftreten kann, der Transfer zwischen zwei Benutzern X und Y wird in zwei Transfers zwischen X und Bank bzw. Bank und Y aufgespalten. Auf dieselbe Weise kann ein Benutzer natürlich auch sich selbst Geld in ein anderes Zahlungssystem überweisen, ohne seine Anonymität im anonymen System zu beeinträchtigen.

6.4.4.2 Verzinsung von Guthaben, Vergabe von Krediten

In beiden in §6.4.1 und §6.4.2 betrachteten Zahlungssystemen können die Rechte, für die B als Zeuge dient, als bei der Bank B geführte Sichteinlagen betrachtet werden: Der Inhaber des Rechtes kann jederzeit darauf zugreifen, aber B kann jeden Zugriff feststellen. Damit entsteht für eine Bank nur bedingt ein Anlaß zur Verzinsung, so daß man z.B. eine Aufrechnung mit den Bankgebühren erwägen, d.h. auf eine getrennte Verzinsung und Gebührenabrechnung verzichten könnte.

Beide Zahlungssysteme können aber leicht so erweitert werden, daß Guthaben für eine bestimmte Zeit fest angelegt werden können, also eine Verzinsung für die Bank sinnvoll wird: Bei festen Konten ist dies trivial; ohne feste Konten könnte man durch Verwendung verschiedener Signaturen p_B^{z1}, p_B^{z2}, … des Zeugen unterschiedliche Fälligkeitszeitpunkte ausdrücken.

Möchte man Geldanlageformen ermöglichen, die von der Höhe des angelegten Guthabens abhängig sind, so muß man das anzulegende Guthaben für die Bank erkennbar zusammenfassen, was zweckmäßigerweise durch Verwendung des Zahlungssystems aus §6.4.2 mit festen Konten geschieht. Eine Verzinsung ist dann wie heute möglich.

Verzichtet man auf solche Anlageformen und verwendet das System aus §6.4.1, so muß man jedes einzelne Recht als eigenes Konto betrachten, wobei der Zeitpunkt der „Einrichtung“ des Kontos anhand der Signatur des Zeugen festgestellt werden kann. Ein Recht kann dann verzinst werden, indem sein Wert allmählich steigt oder indem bei jedem Transfer Zinsen ausgezahlt werden [Chau_89].

Die Vergabe und Verzinsung eines Kredites gestaltet sich im Prinzip nicht schwieriger als heute. Bezüglich der Absicherung eines Kredites gilt das bereits in §6.2.5 Gesagte: Will der Kreditnehmer anonym bleiben, so muß er der Bank gewisse Sicherheiten übergeben; identifiziert er sich hingegen gegenüber der Bank, so muß er der Anonymität wegen lediglich den Kredit einmal sich selbst überweisen (z.B. auf ein zweites Konto) und kann hernach darüber wie über ein normales Guthaben verfügen.

Die Erhebung von Gebühren für die Kontoführung und für einzelne Dienstleistungen einer Bank gestaltet sich, wie in §6.2.5 schon angedeutet, im anonymen Fall kaum komplizierter als heute: Bei festen Konten kann die Bank ihre Forderungen direkt aus dem von ihr verwalteten Guthaben

befriedigen. Geht man nicht von festen Konten aus, so müssen die Banken ihre Gebührenwünsche während des Transfers befriedigen, d.h. der Zahlende dem Transferauftrag einen „Gebührentransferauftrag“ (eine digitale „Gebührenmarke“) für die bezeugende Bank beilegen.

6.4.5 Sichere Geräte als Zeugen

Verzichtet man auf die Forderung der Anonymität vor dem Zeugen, so läßt sich ein anonymes Zahlungssystem sehr leicht realisieren: Das Protokoll aus §6.4.1 kann so abgewandelt werden, daß ein Benutzer X in allen Zahlungen, in denen er als Zahlender oder Zahlungsempfänger auftritt, die Pseudonyme $p_Z^B(X,t)$ bzw. $p_E^B(X,t')$ gleich wählt (d.h. eine feste Kontonummer als Geschäftsbeziehungspseudonym verwendet) und B die Pseudonyme $p_Z(X,t)$ und $p_E(Y,t)$ erfährt. Damit entfällt das Problem der Ersatzquittungen aus [4] und [5] sowie die Umformung der Bestätigung in Schritt [6].

Um die Annahme zu rechtfertigen, dem Zeugen hinsichtlich der Anonymität der Benutzer vertrauen zu dürfen, wird i.allg. der Zeuge als sicheres, unausforschbares Gerät gewählt (vgl. §6.2.1.2.2).

Als Einsatzmöglichkeiten bieten sich zwei Varianten an:

Die erste Variante verwendet ein einziges zentrales sicheres Gerät, das alle Transaktionen bezeugt.

Bereits aus den schon in §1.2.2 generell bemerkten Gründen ist diese Variante wenig wünschenswert. Hinzu kommt, daß gegenüber dem nicht vertrauenswürdigen Zeugen aus §6.4.1 und §6.4.2 kein entscheidender Gewinn erzielt wird. Das Protokoll wird zwar, wie oben erwähnt, stark vereinfacht, doch entlastet dies nur die Rechner, nicht die Benutzer des Zahlungssystems und eröffnet diesen auch keine neuen Möglichkeiten.

Als zweite Variante erhält jeder Benutzer ein eigenes sicheres Gerät als „elektronische Brieftasche“, die seine Transaktionen (im Namen des Zahlungssystembetreibers) bezeugt.

Der Einsatz sicherer Geräte als elektronische Brieftaschen wurde in [EvGY_84, Even_89] vorgeschlagen, war aus Gründen der (scheinbar notwendigen) Protokollierung aber noch nicht anonym. Anonymisierte Versionen des Verfahrens finden sich in [BüPf_87, BüPf_89].

Der einzige echte Vorteil der (anonymen wie nichtanonymen) elektronischen Brieftaschen ist, daß sie als einziges Zahlungssystem **offline Transaktionen** zulassen, also einem Benutzer erlauben, viele Zahlungen spontan zu leisten und zu empfangen, ohne in der Zwischenzeit mit einer zentralen Instanz kommunizieren zu müssen. Dieser Vorteil ist hier aber weniger schwerwiegend, da nur offene Systeme, denen ein Kommunikationsnetz zugrunde liegt, betrachtet werden.

Viel gravierender sind die Nachteile elektronischer Brieftaschen:

- Da Zahlungen nicht mehr über einen zentralen Zeugen abgewickelt werden, kann ein Verlust einer elektronischen Brieftasche zum Verlust bereits erhaltener Zahlungen führen. Um dies zu vermeiden, müssen relativ aufwendige Fehlertoleranzmaßnahmen ergriffen werden [WaPf_87, WaP1_87, WaPf_90].
- Die Sicherheit eines Zahlungssystems mit elektronischen Brieftaschen als Zeugen basiert entscheidend auf der Ausforschungssicherheit der Geräte. Wie schon in §6.2.1.2.2 diskutiert, ist die Existenz dauerhaft sicherer Geräte stark zu bezweifeln, die Bewertung der Ausforschungssicherheit real vorhandener, vermeintlich sicherer Geräte kaum möglich.
- Der Einsatz sicherer Geräte macht den Einsatz kryptographischer Techniken, insbesondere zur gegenseitigen Authentikation elektronischer Brieftaschen, nicht überflüssig. Damit stellen die sicheren Geräte auch keine Notalternative für den Fall dar, daß alle Verschlüsselungs- und Signatursysteme gebrochen werden sollten. Sollten jedoch nur alle asymmetrischen Konzelationssysteme und Signatursysteme gebrochen sein, so wäre in Verbindung mit einem symmetrischen kryptographischen System ein Einsatz noch sinnvoll, da die sicheren Geräte untereinander nur dies benötigen.

Aufgrund der genannten Nachteile und des für offene digitale Systeme der betrachteten Art nur geringen Vorteiles elektronischer Brieftaschen erscheint uns ein Einsatz hier als nicht angebracht.

6.4.6 Zahlungssysteme mit Sicherheit durch Deanonymisierbarkeit

In [ChFN_90] wurde ein anonymes Zahlungssystem vorgestellt, das nur eine schwächere als die bisher verwendete Sicherheitsdefinition erfüllt:

- Ein Benutzer darf ein erhaltenes Recht nur einmal weitergeben. Gibt er es dennoch mehrfach weiter, so muß er dadurch deanonymisierbar werden.

Die Grundidee ist dieselbe wie die in §6.3.1: Man hofft, daß das nach einer Deanonymisierung insgesamt zur Verfügung stehende Vermögen zur Schadensregulierung ausreicht.

Das System soll hier trotz seiner schwächeren Sicherheit kurz skizziert werden: Zum einen ist es auch nicht unsicherer als z.B. das heutige Kreditkartensystem, zum anderen stellt es für offline Zahlungen die einzige Alternative zu den fragwürdigen elektronischen Brieftaschen aus §6.4.5 dar.

Hauptunterschied zu den obigen Zahlungssystemen ist, daß statt des Zeugen B der Zahlende X selbst dem Zahlungsempfänger Y den Transfer bestätigt. Verhält sich X korrekt, so bleibt seine Anonymität gewahrt. Verhält er sich inkorrekt und transferiert das Recht nochmals an einen anderen Empfänger Y^*, so können dank eines kryptographischen Tricks mit hoher Wahrscheinlichkeit die beiden Transferbestätigungen von X so kombiniert werden, daß ihnen die Identität von X zu entnehmen ist. Wird das Recht bei B mehrfach geltend gemacht, so kann X durch B deanonymisiert werden.

Kernidee: Sei k der Sicherheitsparameter,
r_i zufällig gewählt ($1 \le i \le k$),
I Identität des die Banknote Ausgebenden,
C ein Commitment-Schema, dessen Geheimhaltung informationstheoretisch ist.

Die blind[127] signierte Banknote hat die Form

$$s_{Bank}(C(r_1), C(r_1 \oplus I), C(r_2), C(r_2 \oplus I), ..., C(r_k), C(r_k \oplus I))$$

Der Empfänger der Banknote entscheidet für i = 1,...,k jeweils stochastisch unabhängig, ob er r_i oder $r_i \oplus I$ aufgedeckt haben will, erhält also k Commitments aufgedeckt.. (Durch dieses One-time-pad ist die Identität des die Banknote Ausgebenden informationstheoretisch sicher geschützt. Die Bank hingegen ist wegen des Commitment-Schema nur komplexitätstheoretisch sicher. Dafür darf sie das Commitment-Schema und seinen Sicherheitsparameter wählen.)

Wird die Banknote an zwei brave Empfänger ausgegeben, ist die Wahrscheinlichkeit, daß für mindestens ein i bei der Bank r_i und $r_i \oplus I$ eintreffen und der Ausgebende dadurch identifiziert werden kann, in k exponentiell nahe bei 1.

Da im Transfer kein Zeuge benötigt wird, kann dieses Zahlungssystem unmittelbar für offline Zahlungen eingesetzt werden; zu diesem Zweck wurde es in [ChFN_90] auch vorgeschlagen.

Damit der Zahlende, der erfährt, welche Wahl der Zahlungsempfänger trifft, nicht einem zweiten Zahlungsempfänger (der mit dem Zahlenden zusammenarbeitet) dieselbe Information geben kann, so daß die Banknote zweimal zur Einlösung ansteht, ohne den Zahlenden zu identifizieren, hat DAVID CHAUM vorgeschlagen, daß die Identität des Zahlungsempfängers zumindest einen Teil der Entschei-

---

127 Bevor die Bank blind signiert, muß sie natürlich prüfen, ob die ihr vorgelegte Banknote richtig gebildet ist, insbesondere die Identität des die Banknote Ausgebenden enthält. Würde dies von der Bank nicht geprüft, dann könnte der Ausgebende beim Bilden keine oder eine falsche Identität einsetzen und sich so der Deanonymisierung bei Mehrfachausgabe entziehen.
Die Prüfung der richtigen Bildung erfolgt, indem nach Wahl der Bank viele der ihr zum blinden Signieren eingereichten Exemplare vom Einreicher aufgedeckt werden müssen. Nur wenn dabei kein Betrugsversuch entdeckt wird, signiert die Bank die restlichen Exemplare blind [ChFN_90].

dungen des Empfängers der Banknote festlegen soll. Dadurch soll sichergestellt werden, daß zwei verschiedene Empfänger der Banknote an mindestens einer Stelle unterschiedlich gewählt haben und dadurch die Identität des Zahlenden ermittelbar ist.

Zur Erhöhung der Sicherheit kann man es mit sicheren Geräten kombinieren: Die Benutzer müssen statt beliebiger Rechner sichere Geräte verwenden, die die mehrfache Weitergabe eines Rechtes verhindern. Das kombinierte System erfüllt solange die stärkere Sicherheitsdefinition, wie die sicheren Geräte wirklich sicher sind; ansonsten ist es ebenso sicher wie das ursprüngliche System ohne sichere Geräte.

Zahlungssysteme, die die offline Eigenschaft insofern einschränken, daß ein Zahlungsempfänger Y ein erhaltenes Recht nicht ohne vorherigen Kontakt mit der Bank weitergeben kann, sind in [ChFN_90, Cha1_89, CBHM_90] beschrieben. (Ein entsprechendes in [OkOh_89, OkOh1_89] vorgeschlagenes Zahlungssystem erwies sich laut DAVID CHAUM als fehlerhaft.)

Leseempfehlungen

Betrugssicherer Werteaustausch: AsSW_97, Baum_99

Digitale Zahlungssysteme: AJSW_97, Chau_92, PfWa2_97

Aktuelle Forschungsliteratur:

Computers & Security, Elsevier

Tagungsserien ACM Conference on Computer and Communications Security, IEEE Symposium on Security and Privacy, IFIP/Sec, VIS, ESORICS, Crypto, Eurocrypt, Asiacrypt; Tagungsbände erscheinen bei ACM, IEEE, bei Chapman & Hall London, in der Serie Datenschutz-Fachberichte des Vieweg-Verlags Wiesbaden, in der Serie LNCS des Springer-Verlags Heidelberg

Hinweise auf aktuelle Forschungsliteratur und Kommentare zu aktuellen Entwicklungen:

http://www.itd.nrl.navy.mil/ITD/5540/ieee/cipher/ oder
abonnieren unter cipher-request@itd.nrl.navy.mil

Datenschutz und Datensicherheit DuD, Vieweg-Verlag

7 Umrechenbare Autorisierungen (Credentials)

Eine Übersicht für eine umrechenbare Autorisierung wird in [Cha8_85, Chau_87] gegeben. Dort ist jeweils nicht erwähnt, daß wegen des gemeinsamen RSA-Modulus alle Organisationen die Credential-Signaturen nicht selbst leisten können, sondern von einer Signaturorganisation leisten lassen müssen. Denn wer ein Paar zueinander inverser RSA-Exponenten kennt, kann den Modulus faktorisieren [Hors_85 Seite 187, MeOV_97 Seite 287].

In meiner Übersicht ist der Fall einer festen und zu Beginn bekannten Menge von Credentials beschrieben.

Eine detaillierte Protokollbeschreibung ist in [Chau2_90] zu finden.

In [ChEv_87] ist beschrieben, wie die Prüfung, ob die Pseudonyme richtig gebildet sind, so durchgeführt werden kann, daß die Wahrscheinlichkeit, daß unentdeckt falsche darunter sind, exponentiell klein im Prüfungsaufwand wird.

In [Cha1_88] ist beschrieben, wie Credentials, die zu Beginn nicht bekannt sind, ausgestellt werden können.

In [Bura_88] und [ChAn_90] sind weitere Sorten umrechenbarer Autorisierungen beschrieben.

Leider ist mir keine Anwendung dieser rein digitalen Identitäten bekannt. Denn es reicht nicht, daß z.B. die Autorisierung „Führerschein“ nur zwischen den Pseudonymen einer Person umgerechnet werden kann. Die Autorisierung „Führerschein“ sollte auch nur das biologische Wesen nutzen können, das die Prüfung bestanden hat. Man muß das beschriebene Verfahren also für (fast?) alle Anwendungen so erweitern, daß in der Protokollphase „Pseudonyme vereinbaren“, Pseudonyme an Körpermerkmale gebunden werden. Ob es untereinander hinreichend nicht verkettbare, aber andererseits hinreichend genau meßbare Körpermerkmale gibt, ist eine Frage außerhalb der Informatik. Gibt es sie, ist die Erweiterung der beschriebenen Protokolle kanonisch.

Eine Erweiterung auf der Basis sicherer, d.h. unausforschbarer biometrischer Hardware, vgl. §6.2.1.2.2, die mit *einem* hinreichend genau meßbaren Körpermerkmal auskommt, ist in [Bleu_99] beschrieben.

8 Verteilte Berechnungsprotokolle

Gegeben sei die sehr allgemeine Aufgabe, daß n Teilnehmer T_i zusammen eine Berechnung durchführen wollen. Jeder Teilnehmer soll seinen Input I_i zur Berechnung beitragen und am Schluß seinen Output O_i erhalten, der als Funktion f_i aller Inputs spezifiziert ist. Folgende Sicherheitseigenschaften sollen erreicht werden:

Größtmögliche **Vertraulichkeit**:
Kein Teilnehmer soll über Inputs von anderen Teilnehmern mehr erfahren, als er aus seinem Output schließen kann.

Größtmögliche **Integrität**:
Kein Teilnehmer soll, nachdem er seinen Input gegeben hat, die Werte der Outputs beeinflussen können.

Größtmögliche **Verfügbarkeit**:
Kein Teilnehmer soll, nachdem er seinen Input gegeben hat, einem anderen Teilnehmer seinen Output vorenthalten können.

Alle drei Sicherheitseigenschaften sollen auch gelten, wenn sich *mehrere* Teilnehmer zusammenschließen.

Sie sollen dann nicht mehr erfahren, als was sie aus ihren gemeinsamen Outputs schließen können.

Nachdem der letzte von ihnen seinen Input gegeben hat, sollen sie die Werte der Outputs nicht mehr beeinflussen und anderen nicht mehr vorenthalten können.

Es gibt nun zwei gänzlich unterschiedliche Möglichkeiten, diese Aufgabe zu lösen:

1. Alle Teilnehmer senden ihre Inputs an einen **zentralen Rechner**, dem alle vertrauen (müssen). Dieser zentrale Rechner führt die Berechnungen durch und sendet jedem Teilnehmer seinen Output. Zwischen jedem Teilnehmer und dem zentralen Rechner sollte so **verschlüsselt** werden, daß **Konzelation und Authentikation** gewährleistet sind, vgl. Bild 8-1. Vertraut man dieser Verschlüsselung und dem zentralen Rechner sowie der Verfügbarkeit des Kommunikationsnetzes, ist das gestellte Problem gelöst. Bzgl. der Verfügbarkeit und Integrität der Outputs kann man den zentralen Rechner zumindest prinzipiell im Nachhinein kontrollieren, wenn alle Nachrichten digital signiert werden und eine Zusammenarbeit von zentralem Rechner und Teilnehmern ausgeschlossen werden kann (sonst könnten sie sich im Nachhinein die für ihre Handlungen nötigen digitalen Signaturen zuschanzen). Leider ist der zentrale Rechner bezüglich Vertraulichkeit nahezu unkontrollierbar, vgl. §1.2.2 und §2.1.
2. Die Teilnehmer führen ein **verteiltes Berechnungsprotokoll** (Multi-Party Computation Protocol) durch, vgl. Bild 8-2. Bei solch einem Protokoll gibt es keine zentrale Instanz, der alle vertrauen müssen. Aber auch hier ist, wenn auch realistischeres Vertrauen nötig:
   - Entweder muß darauf vertraut werden, daß ein Angreifer weniger als ein Drittel aller Teilnehmer umfaßt. Dann sind informationstheoretisch sichere verteilte Berechnungsprotokolle bekannt [BeGW_88, ChCD1_88].
   - Oder es muß mit einer kryptographischen Annahme gearbeitet werden. Dann kann der Anteil des Angreifers an den Teilnehmern aber beliebig sein [ChDG_88].
   - Ein neueres Protokoll erfordert nur, daß eine der beiden gerade skizzierten Restriktionen für den Angreifer zutrifft [Chau_90].

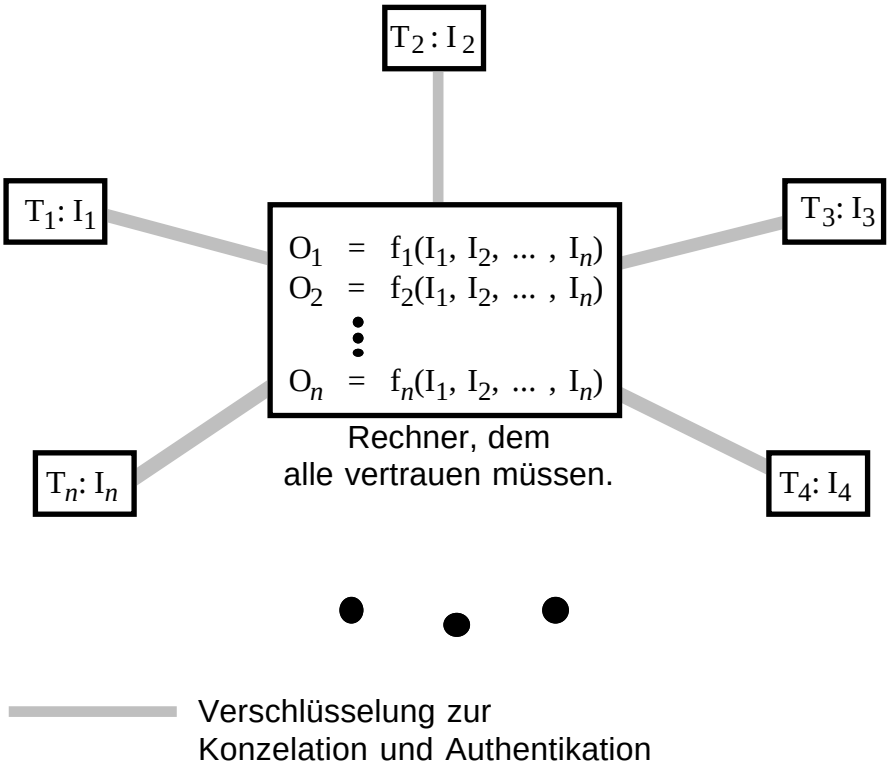


Bild 8-1: Zentralisiertes Berechnungsprotokoll zwischen Teilnehmern T_i. Jeder T_i sendet seinen Input I_i an einen zentralen Rechner, dem alle vertrauen müssen. Dieser berechnet die Funktionen f_j und sendet an jeden T_i seinen Output O_i.

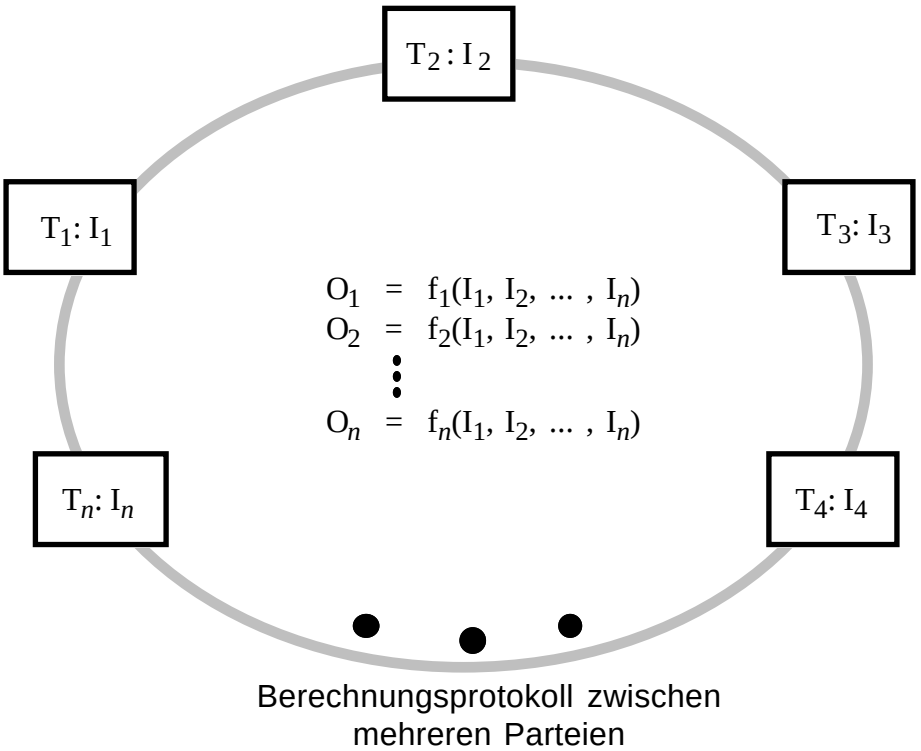


Bild 8-2: Verteiltes Berechnungsprotokoll zwischen Teilnehmern T_i. Jeder T_i bringt seinen Input I_i in die Berechnung ein und erhält seinen Output O_i.

Grob funktionieren die verteilten Berechnungsprotokolle folgendermaßen:

Die Eingaben werden mittels eines Schwellwertschemas, vgl. §3.9.6, unter den Teilnehmern verteilt.

Danach können die Teile sowohl addiert als auch multipliziert und so beliebige Berechnungen ausgeführt werden.

Die genauen Details sprengen den Rahmen dieses Skripts. Wichtig ist mir die Botschaft, daß prinzipiell *alles*, was mittels global vertrauenswürdiger zentraler Rechner berechnet werden kann, auch dezentral und ohne globales Vertrauen berechenbar ist – auch wenn der Aufwand der bisher bekannten verteilten Berechnungsprotokolle hoch ist.

9 Regulierbarkeit von Sicherheitstechnologien

Keiner von uns lebt völlig allein. Sie lesen gerade meinen Text – und von mir weiß ich es selbst. Außerdem setzen Rechnernetze wie auch mehrseitige Sicherheit voraus, daß es *mehrere* Beteiligte gibt. Und diese haben jeweils nicht nur Individualziele, sondern sie sind auch Teil unterschiedlichster Gemeinschaften. Individuen haben nicht nur legitime Rechte und Ansprüche an die Gemeinschaften, sondern Gemeinschaften auch an ihre Mitglieder. Und hier entsteht folgendes Problem:

Die vorgestellten Sicherheitstechnologien können nicht nur zur legitimen und Demokratie fördernden Herstellung mehrseitiger Sicherheit in einer offenen Gesellschaft benutzt werden, sondern auch zur Bildung von Subkulturen und deren Abschottung gegenüber außen. Im Grenzfall nutzen also auch organisiert Kriminelle die weit verbreitete Sicherheitstechnologie, um ihre Verbrechen zu planen, zu koordinieren und auszuführen. Um dem vorzubeugen, fordern manche Politiker eine Regulierung von Sicherheitstechnologie, d.h. eine Einschränkung von Herstellung, Verbreitung und sogar Benutzung von Sicherheitsmechanismen, etc.

Nun möchte ich nicht versuchen zu entscheiden, was gemeinschaftsförderlich, was gut, was in der Bekämpfung des Schlechten verhältnismäßig ist. Das sind schwierige Fragen, auf die es keine endgültigen Antworten gibt: Auch die Demokratie begann oftmals als Anliegen einer Subkultur, auch eine demokratisch regierte Supermacht wie die USA verfolgt mit der Regulierung von Sicherheitstechnologie Macht- und Wirtschaftsinteressen.

Ich werde einer einfacheren Frage nachgehen: Ist Sicherheitstechnologie überhaupt regulierbar, d.h. dienen regulatorische Eingriffe wie Export- und Importverbote, Gestaltungs- und Benutzungsvorschriften wirklich dem angestrebten Zweck, oder sind sie wirkungslos oder gar kontraproduktiv?

Wovon ich nichts halte ist, allein um **symbolische Politik** zu betreiben, den Nutzen von Sicherheitstechnologie für die Gesetzestreuen einzuschränken, ohne die Kriminellen signifikant zu stören. Dann lieber ein symbolisches Gesetz, daß böse Gedanken verbietet. Denn alle Kriminalität beginnt mit bösen Gedanken – und durch solch ein Gesetz wird wenigstens kein Guter behindert.

In den letzten 2 Jahrzehnten wurde insbesondere versucht, die Verbreitung von **sicheren Rechnern**, insbesondere sicheren Betriebssystemen, und von **Kryptographie** zu regulieren. In den USA hat beispielsweise DEC die Entwicklung eines sicheren verteilten Betriebssystems [GGKL_89, GKLR_92, Wich_93] etwa 1994 mit der Begründung eingestellt, es sei hierfür keine Exportgenehmigung zu erhalten und die Entwicklung von zwei Betriebssystemversionen für den Weltmarkt nicht lohnend. Über die Regulierung von Kryptographie wird, nachdem mindestens seit 1986 hinter verschlossenen Türen darüber diskutiert wurde [WaPP_87, Riha2_96], in den USA seit 1993 unter dem Schlagwort *Key escrow* / *Key recovery* bzw. *Clipper-Chip* und Nachfolgevorschläge und in Deutschland unter dem Schlagwort *Kryptogesetz* hierüber öffentlich diskutiert [HuPf2_96, WiHP_97, HaMö_98].

Die Verbreitung von unsicheren Rechnern ist heutzutage weit fortgeschritten, denken Sie an PCs mit Betriebssystemen ohne jede Zugriffskontrolle und entsprechend empfänglich für Trojanische Pferde via Würmer und Computer-Viren. Einen politischen Streit in diesem Gebiet gibt es folglich nicht – er wird aber entstehen, sobald sichere(re) PCs marktgängig werden.

Im Folgenden konzentriere ich mich auf die Regulierbarkeit von Kryptographie – denn hier kann durch Unkenntnis und symbolpolitischen Übereifer möglicherweise sehr großer Schaden nicht nur für einzelne, sondern gerade auch die Demokratie entstehen [Pfit_98].

Auf Grundlage der Funktionsprinzipien kryptographischer und steganographischer Verfahren (§3 und §4) werden nun die wichtigsten Anforderungen und technischen Gestaltungsmöglichkeiten diskutiert, die für die aktuelle Diskussion um eine Kryptoregulierung von Bedeutung sind.[128]

9.1 Benutzeranforderungen an Signatur- und Konzelationsschlüssel

Digitale Signatursysteme werden zur Zeit für den elektronischen Rechtsverkehr[129] und die elektronische Archivierung eingeführt. Für beide Anwendungsbereiche ergeben sich Zertifizierungsanforderungen an Signaturschlüssel, die neben den kryptographischen Eigenschaften des Signatursystems Voraussetzung für die Rechtsverbindlichkeit digital signierter Dokumente sind. So muß der Empfänger einer signierten Nachricht nicht nur prüfen können, daß Nachricht und Signatur zum öffentlichen Testschlüssel passen, sondern er muß den öffentlichen Testschlüssel auch nachweisbar einer Person zuordnen können[130]. Hierfür sollen sogenannte TrustCenter – technisch-organisatorische Einheiten, die verschiedene vertrauenswürdige, der Sicherheit dienliche Dienstleistungen erbringen sollen –, auch die Funktion einer Certification Authority wahrnehmen. Die Erzeugung der Signaturschlüsselpaare in TrustCentern[131] kann zur Voraussetzung zur Teilnahme am elektronischen Rechtsverkehr gemacht werden, und damit besteht nun die Möglichkeit, alle geheimen Signierschlüssel dort zu speichern. Die Verwendbarkeit von Signatursystemen im elektronischen Rechtsverkehr beruht aber auf der Zusicherung, daß außer dem Besitzer eines geheimen Signaturschlüssels niemand eine entsprechende Signatur erzeugen kann. Eine Zugriffsmöglichkeit auf geheime Signaturschlüssel erlaubt das Fälschen von Beweismitteln und entwertet damit die ganze Sicherheitsinfrastruktur, auf der der elektronische Rechtsverkehr beruht.

Schlüsselpaare für Konzelation, also zum Vertraulichkeitsschutz, brauchen nicht in einem rechtlich bindenden Sinne zertifiziert zu werden. Es mag sinnvoll sein, beim Aufbau einer Infrastruktur Schlüsselzertifizierungen einzusetzen, um die Authentizität der verwalteten Schlüssel zu sichern. Damit kann verhindert werden, daß in einer Sicherheitsinfrastruktur Personen unter falschen Namen öffentliche Konzelationsschlüssel bereitstellen und auf diese Weise in den Besitz vertraulicher Nachrichten gelangen. Aber die Anforderung der Benutzer an Konzelationssysteme besteht darin, daß *sie* sich *gegenseitig* der Authentizität der öffentlichen Konzelationsschlüssel sicher sind, nicht daß sie diese vor Gericht nachweisen können. Daher können Kommunikationspartner, sobald es ihnen gelungen ist, einmal Schlüssel auszutauschen[132], deren Authentizität sie vertrauen, jedes beliebige Konzelationssystem zur weiteren vertraulichen Kommunikation verwenden. Die Erzeugung von Schlüsselpaaren in sogenannten TrustCentern kann also für Konzelation nicht erzwungen werden, denn niemand ist beim Gebrauch der Konzelationsschlüssel auf Zertifikate von TrustCentern angewiesen.

---

128 Die folgenden Unterkapitel sind teilweise kopiert aus [HuPf_96, HuPf_98].

129 Das Gesetz zur digitalen Signatur (Art. 3 des Informations- und Kommunikationsdienste-Gesetzes (IuKDG)) trat zum 1. Aug. 1997 in Kraft.

130 Daher werden der öffentliche Testschlüssel und die Identität der inhabenden Person von einer *Certification Authority* digital signiert, was als *Zertifizierung* des öffentlichen Schlüssels bezeichnet wird. Um solche Zertifikate überprüfen zu können, muß der Empfänger einer signierten Nachricht eventuell weitere Zertifikate einholen. Der entstehende Zertifizierungsbaum ist beispielsweise in X.509 und in eingeschränkter Form im Gesetz zur digitalen Signatur (bzw. der zugehörigen Verordnung) beschrieben.

131 Dies ist allerdings nicht empfehlenswert, da zentrale Systemkomponenten, auf denen die Ausführung verschiedener voneinander unabhängiger Teilaufgaben konzentriert wird, immer auch Schwachpunkte einer Sicherheitsarchitektur bilden, vgl. [FJPP_95], und es vereinfacht die Zertifizierung auch nicht.

132 z.B. mit Hilfe der staatlich bereitgestellten Sicherheitsinfrastruktur, aber auch durch persönlichen Austausch von Schlüsseln.

Im folgenden argumentieren wir, daß Benutzer einen Schlüsselaustausch für eine gegenüber jedwedem Dritten vertrauliche Konzelation unter Verwendung der Sicherheitsinfrastruktur für den elektronischen Rechtsverkehr bzw. von Konzelationssystemen, bei denen die Hinterlegung der geheimen Konzelationsschlüssel vorgesehen ist, eigenständig realisieren können, ohne eine Einschränkung in ihren Anforderungen in Kauf nehmen zu müssen.

9.2 Digitale Signaturen ermöglichen den Austausch von Konzelationsschlüsseln

Jede Sicherheitsinfrastruktur für digitale Signaturen ermöglicht den sicheren Austausch von Schlüsseln für Konzelation: Nach bekannten Algorithmen [Schn_96], die bereits in allgemein verfügbaren Programmen implementiert sind (z.B. Pretty Good Privacy – PGP), kann jeder für Konzelation geeignete Schlüsselpaare generieren. Danach kann jeder seine öffentlichen Konzelationsschlüssel selbst zertifizieren, d.h. die Authentizität des Konzelationsschlüssels für andere überprüfbar machen, indem er etwa folgende Nachricht digital signiert: „Der öffentliche Schlüssel ... gehört ...“ (in Bild 9-1 die Nachricht $s_A(A,c_A)$). Eine Sicherheitsinfrastruktur für digitale Signaturen erlaubt nun jedem, die Unverfälschtheit dieser Nachricht zu prüfen[133] und sich so zu vergewissern, daß alles, was mit diesem öffentlichen Schlüssel verschlüsselt wird, nur vom Inhaber des Schlüsselpaares wieder entschlüsselt werden kann. Soll diese Verwendung eines Signatursystems ausgeschlossen werden, so besteht die einzige Möglichkeit darin, die Sicherheit der Signaturen zu untergraben, indem digital signierte Nachrichten in Umlauf gebracht werden, bei denen der Unterzeichner eben nicht der Eigentümer des Signierschlüssels ist. Damit ist natürlich jede Rechtsverbindlichkeit von Dokumenten, die in diesem System signiert wurden, hinfällig.

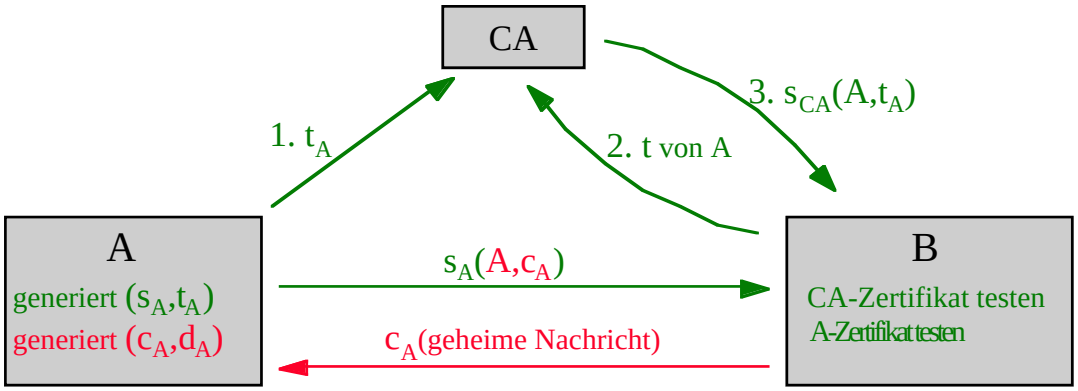


Bild 9-1: Sichere digitale Signaturen ermöglichen teilnehmerautonome sichere Konzelation

Konzelationsschlüssel, die mit Hilfe einer Sicherheitsinfrastruktur für digitale Signaturen verbreitet wurden, ermöglichen sowohl, Nachrichten vertraulich auszutauschen als auch Schlüssel für symmetrische Konzelationssysteme. Mit diesen kann dann effizienter konzeliert werden.

---

[133] indem er sowohl das Zertifikat $s_{CA}(A,t_A)$ der Certification Authority testet wie auch das Zertifikat, das sich Teilnehmer A selbst ausgestellt hat.

9.3 Key-Escrow-Systeme können zum zunächst unerkennbaren Schlüsselaustausch verwendet werden

Ähnlich wie sichere digitale Signaturen zum sicheren Austausch von Schlüsseln für Konzelation verwendet werden können, kann auch jedes Konzelationssystem, also insbesondere auch jedes Key-Escrow- und Key-Recovery-System, bei dem den Sicherheitsdiensten der Zugang zu den geheimen Schlüsseln ermöglicht werden kann, dazu benutzt werden, um Schlüssel für zusätzlich zu verwendende Konzelationssysteme zu vereinbaren (Bild 9-2). Statt einem Konzelationssystem direkt die vertraulich zu haltenden Nachrichten zu übergeben, wird zunächst ein öffentlicher Konzelationsschlüssel redundant codiert übertragen. Der Empfänger überprüft den Schlüssel, ähnlich dem Prüfen des Zertifikates bei Verwendung digitaler Signaturen. Sofern er keine Verfälschung des öffentlichen Schlüssels feststellt, verwendet er den öffentlichen Schlüssel entweder, um geheim zu haltende Nachrichten zu verschlüsseln (Bild 9-2, 2. Pfeil) oder aber zum Austausch eines weiteren Schlüssels für ein symmetrisches Konzelationssystem, mit dem dann effizienter verschlüsselt werden kann (Bild 9-2, 4. Pfeil). Bei der weiteren Kommunikation werden dann die mit dem aufgesetzten Konzelationssystem bereits verschlüsselten Nachrichten mit dem zugrundeliegenden Key-Escrow-Konzelationssystem ein weiteres Mal verschlüsselt, wie dies in Bild 9-2 mit Pfeil 3 und 4 angedeutet wird.

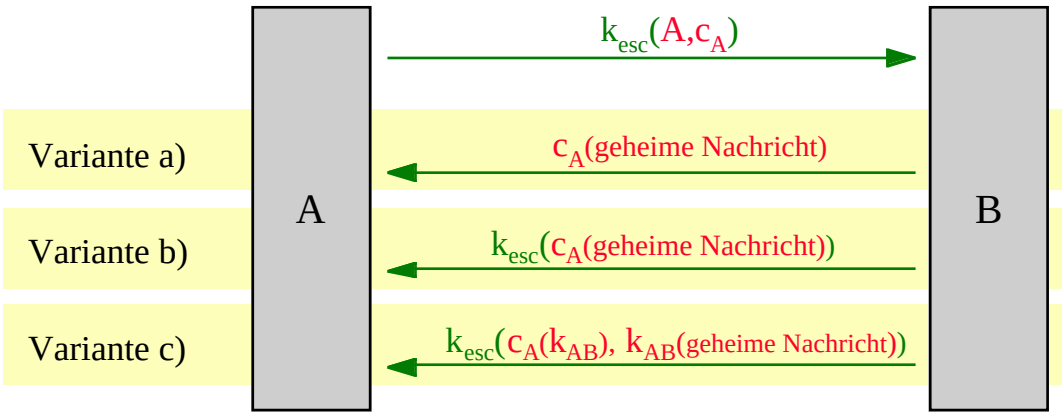


Bild 9-2: Key-Escrow-Konzelation ohne Dauerüberwachung ermöglicht Schlüsselaustausch und damit a) "normale" Konzelation, b) asym. Konzelation ohne Key-Escrow und c) sym. Konzelation ohne Key-Escrow

Ist das Key-Escrow-System so gestaltet, daß eine Entschlüsselung früherer Nachrichten nicht möglich oder nicht erlaubt ist, so braucht nicht einmal der Umweg über den öffentlichen Konzelationsschlüssel gegangen zu werden. Wie in Bild 9-3 gezeigt, kann dann direkt ein symmetrischer Schlüssel ausgetauscht werden – dies muß nur rechtzeitig geschehen.

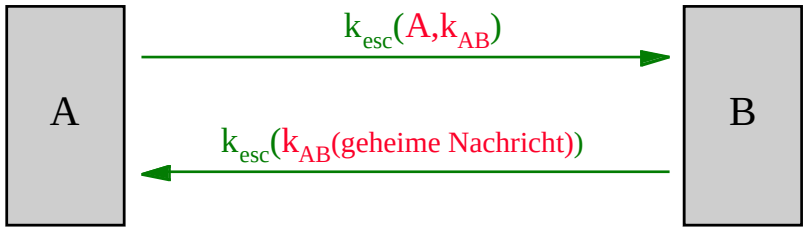


Bild 9-3: Key-Escrow-Konzelation ohne rückwirkende Entschlüsselung ermöglicht direkt symmetrische Konzelation ohne Key-Escrow

Die Verwendung zusätzlicher Verschlüsselungsmaßnahmen kann erst dann entdeckt werden, wenn die Schlüssel eines Key-Escrow-Systems herausgegeben und zum Entschlüsseln des Fernmeldeverkehrs benutzt wurden, was nach geltendem Recht eine Erlaubnis zur Überwachung des Fernmeldeverkehrs voraussetzt. Key-Escrow ist – wenn die Schlüsselherausgabe tatsächlich so zurückhaltend gehandhabt wird, wie in der öffentlichen Diskussion immer wieder betont wird – für die Ermittlungen der Sicherheitsbehörden, des Verfassungsschutzes, des Bundesnachrichtendienstes und MADs vermutlich kaum hilfreich, da gerade in den Fällen, wo Key-Escrow die Entschlüsselung von Nachrichten ermöglichen soll, höchstens die Verwendung zusätzlicher Verschlüsselungssysteme festgestellt werden kann. Daher wird neben der Einführung von Key-Escrow auch das Verbot anderer Verschlüsselungsverfahren diskutiert, was die Problematik allerdings in keiner Weise löst: Wird die Überwachung des Fernmeldeverkehrs zurückhaltend gehandhabt, so wird ein Übertreten des Kryptoverbots häufig „zu spät“ bemerkt werden. Ein ernsthafter Durchsetzungsversuch des Kryptoverbots setzt also eine weitgehende Aushöhlung des Fernmeldegeheimnisses voraus. Doch selbst dann können sich Teilnehmer steganographischer Techniken bedienen (siehe §9.5), bei denen die Verwendung eines Verschlüsselungssystems praktisch nicht nachweisbar ist.

Daneben ist zu berücksichtigen, daß alle Stellen und Personen, die Kenntnis von dem geheimen Konzelationsschlüssel eines Benutzers erhalten haben, alle zu diesem Schlüssel gehörigen Nachrichten entschlüsseln können. Im Unterschied zu traditionellen Überwachungstechniken gerät damit das Ende einer Abhöraktion diffus, da die Kenntnis eines Schlüssels nicht rückholbar ist. Die zentrale Speicherung dieser sensiblen Informationen bei Key-Escrow-Agencies und Bedarfsträgern macht diese ggf. zu einem vielversprechenden Angriffspunkt für Wirtschaftsspionage und andere kriminelle Aktivitäten.

Kurzum: Key-Escrow zur Verbrechensbekämpfung wird weitestgehend wirkungslos bleiben – erfordert aber kaum realisierbaren technischen[134] und erheblichen organisatorischen[135] Aufwand, um die Aushöhlung der Rechte der Bürger und eine Steigerung der informationellen Verletzbarkeit von Wirtschaft und Gesellschaft in vertretbaren Grenzen zu halten.

9.4 Symmetrische Authentikation ermöglicht Konzelation

Jedes Bit der zu konzelierenden Nachricht wird übertragen, indem beide möglichen Werte, 0 und 1, jeweils gefolgt von einem MAC, übertragen werden [Rive_98]. Der MAC des richtigen Bits paßt, der des falschen nicht, vgl. Bild 9-4. Da außer Sender und Empfänger niemand prüfen kann, welche

134 Wie soll eine Zugriffskontrolle in den Datenbanken der Key-Escrow-Agencies fehlerfrei realisiert werden, und gleichzeitig ein kurzfristiger Zugriff auf die Schlüssel möglich bleiben? Die Erfahrung, daß bisher *jedes* größere technische System Fehler enthielt, lehrt, daß derartige Risikozusammenballungen vermieden werden sollten.

135 Es ist erstaunlich, daß gerade die Bedarfsträger Key-Escrow fordern, deren Erfolge und Skandale doch häufig auf der Erfahrung „Jeder Mensch hat seinen Preis“ beruhen.

MACs passen und welche nicht (könnte das jemand mit einer besseren Wahrscheinlichkeit als 0,5, dann wäre dies ein Ansatz zum Brechen des Authentikationssystems), ist die Nachricht so perfekt konzeliert.

Sender *A*		**Empfänger *B***
Kennt k_{AB}		Kennt k_{AB}
Zu übertragen sei Nachricht $b_1, \ldots b_n$ mit $b_i \in \{0, 1\}$		
Berechnet $MAC_1 := code(k_{AB},b_1) \ldots MAC_n := code(k_{AB},b_n)$		
Sei $a_1, \ldots a_n$ die bitweise invertierte Nachricht.		
Wählt zufällig $MAC'_1 \ldots MAC'_n$ mit $MAC'_1 \neq code(k_{AB},a_1) \ldots MAC'_n \neq code(k_{AB},a_n)$		
Überträgt $\{(b_1, MAC_1), (a_1, MAC'_1)\} \ldots$ $\{(b_n, MAC_n), (a_n, MAC'_n)\}$	(die Mengenklammern bedeuten „zufällige Reihenfolge“) ——————————>	Probiert, ob $\{MAC_1 = code(k_{AB},b_1)$ oder $MAC'_1 = code(k_{AB},a_1)\}$ und empfängt den passenden Wert b_1 … probiert, ob $\{MAC_n = code(k_{AB},b_n)$ oder $MAC'_n = code(k_{AB},a_n)\}$ und empfängt den passenden Wert b_n

Bild 9-4: Konzelation mittels symmetrischer Authentikation nach Rivest

Rivests Vorschlag ist etwas mehr als doppelt so aufwendig wie die Lösung von Aufgabe 3-22. Er hat aber den Vorteil, daß beide Bits mit MACs jeweils in separate Nachrichten mit gleicher Sequenznummer gesteckt werden können. Dann wirft einerseits das ganz normale Authentikationsprotokoll des Empfängers die falschen Bits weg, der Empfänger muß also gar nichts implementieren. Zusätzlich muß auch der Sender die falschen Bits und MACs nicht selbst generieren und in Nachrichten packen – das kann jemand anderes tun, sofern es nur vor der Stelle, an der der Angreifer abhört, getan wird. Und um es zu tun, braucht er nicht den symmetrischen Authentikationsschlüssel – er nimmt einfach irgendwelche zufälligen MACs, die dann mit sehr großer Wahrscheinlichkeit nicht passen, vgl. Bild 9-5. Dann können sich Sender und Empfänger auch in einem Land mit totalem Konzelationsverbot trefflich herausreden: „Konzelation? Haben wir nicht gemacht, haben wir nicht gewollt, haben wir nicht mal gemerkt!“

Sender *A*

Kennt k_{AB}

Zu übertragen sei Nachricht
$b_1, ... b_n$ mit $b_i \in \{0, 1\}$

Berechnet
$MAC_1 := code(k_{AB},b_1)$ ... $MAC_n := code(k_{AB},b_n)$

Überträgt
$(1, b_1, MAC_1), ... (n, b_n, MAC_n)$

Komplementgenerierer

Hört die Nachricht $b_1, ... b_n$ ab.

Bildet $a_1, ... a_n$, die bitweise invertierte Nachricht.
Wählt zufällig $MAC'_1 ... MAC'_n$ und mischt in
den Nachrichtenstrom von Sender A
an die passenden Stellen
$(1, a_1, MAC'_1), ... (n, a_n, MAC'_n)$

Überträgt die Mischung

Abhörer

kann a_i und b_i nicht unterscheiden

Empfänger *B*

Kennt k_{AB}

normales Authentikationsprotokoll
Ignoriert Nachrichten mit falscher Sequenznr.
Ignoriert Nachrichten mit falscher Authentikat.
gibt die übrigbleibenden weiter
empfangen wird mit größter Wahrscheinlichk.
$b_1, ... b_n$

Bild 9-5: Konzelation mittels symmetrischer Authentikation ohne Teilnehmeraktivität

Wie in der Lösung von Aufgabe 3-22 angemerkt, können statt jeweils einzelner Bits mehrere Bits gleichzeitig mit einem MAC authentisiert werden. Dies erfordert aber bei l gleichzeitig authentisierten Bits pro MAC jeweils 2^l Bitwerte und MACs sowie Probierschritte des Empfängers. Damit ist dies hier nur von Nachteil, während in dem Verfahren von Aufgabe 3-22 damit auch Vorteile erzielt werden konnten.

Interessant ist die Frage: Was tun, wenn jemand, z.B. eine Strafverfolgungsbehörde, Sie zwingen kann, den Authentikationsschlüssel rauszurücken? Die Antwort lautet: Vorbeugen. Dies heißt, unter den sehr vielen Bitfolgen auch mindestens eine mit diesem „jemand“ angenehmem oder zumindest unverdächtigem Inhalt zu haben – und ihm natürlich nur diesen Schlüssel herauszurücken. Damit mehrere Bitfolgen mit „echtem“ Inhalt sich gegenseitig schützen, sind allerdings noch trickreichere Codierungen der Bitfolgen nötig, als die hier beschriebenen [Rive_98].

9.5 Steganographie: Vertrauliche Nachrichten zu finden ist unmöglich

Nachrichten können nicht nur durch kryptographische Konzelationssysteme vor unerwünschter Kenntnisnahme geschützt werden, sondern auch durch Steganographie: Die zu schützende Nachricht wird in einer anderen Nachricht, die notwendigerweise wesentlich länger ist, geeignet versteckt wie in §4 erläutert wurde. Selbst wenn Kryptographie – oder noch wesentlich allgemeiner: jede Form

unüberwachbarer Kommunikation oder Speicherung – verboten würde, ist dem Anwender nichts zu beweisen, wenn die verwendete steganographische Technik gut ist. Ist sie noch nicht gut, so kann jeder Nachweis von Steganographie vor Gericht unmittelbar in verbesserte Steganographie umgesetzt werden, vgl. §4.1.3.3.

Steganographie macht sich eine Grundtatsache der menschlichen Kommunikation zunutze, nämlich daß die gleiche Nachricht für unterschiedliche Empfänger eine unterschiedliche Bedeutung haben kann. In den modernen digitalen Kommunikationsnetzen können hocheffiziente Verfahren realisiert werden, mit denen Benutzer, ohne sich persönlich zu treffen und ohne daß es von Dritten nachgewiesen werden kann, Nachrichten austauschen können oder vereinbaren, wie sie zukünftig auszutauschende Nachrichten interpretieren wollen. Natürlich ist es möglich und zur Erhöhung der Sicherheit empfehlenswert, die zu schützende Nachricht vor dem steganographischen Verstecken noch zu verschlüsseln.

Die bereits verfügbaren und geplanten datenintensiven interaktiven Telekommunikationsanwendungen aus dem Multi-Media-Bereich bieten sich als Trägersysteme für steganographische Techniken geradezu an.

9.6 Maßnahmen bei Schlüsselverlust

Der Verlust[136] von Schlüsseln, die zum Schutz der Vertraulichkeit oder zur Integritätssicherung bei der *Übertragung* von Nachrichten verwendet werden, stellt für den Benutzer kein ernstliches Problem dar: Können etwa die mit Hilfe eines Konzelationssystems geschützten Daten nicht entschlüsselt werden, so generiert der Empfänger ein neues Schlüsselpaar und teilt dem Sender seinen neuen öffentlichen Konzelationsschlüssel authentisiert mit. Danach verschlüsselt der Sender die Daten unter Zuhilfenahme des neuen Schlüssels noch einmal und überträgt sie erneut. Diese Methode ist weit weniger aufwendig als jede sichere Form des Schlüsselbackups[137] und weitaus sicherer als Key-Escrow- und Key-Recovery-Systeme. Key-Escrow- und Key-Recovery-Systeme halten die geheimen Schlüssel für gesetzlich festgelegte Bedarfsträger, Sicherheitsbehörden und Nachrichtendienste vor, um ggf. eine Überwachung des Fernmeldeverkehrs zu ermöglichen, als technische Hilfe bei Schlüsselverlust sind sie überflüssig.

Gleiches gilt hinsichtlich Schlüsseln für Signaturen, vgl. Bild 9-6. Verliert der Inhaber seinen geheimen Signierschlüssel, kann er nicht mehr passend zu seinem öffentlichen Schlüssel signieren. Er kann sich aber jederzeit ein neues Schlüsselpaar generieren und den neuen öffentlichen Schlüssel zertifizieren lassen. Dieser muß den Empfängern von zukünftig signierten Nachrichten samt Zertifikat mitgeteilt werden.

Der Verlust öffentlicher Schlüssel ist praktisch nicht relevant, da ihre Öffentlichkeit beliebig weit verbreitete Speicherung gewährleistet.

Bei Schlüsseln für den Schutz der Vertraulichkeit oder der symmetrischen Authentikation längerfristig gespeicherter Daten[138] führt ein Verlust des geheimen Entschlüsselungs- bzw. Prüfschlüssels in der

---

136 Relevant für unsere Betrachtungen ist die Situation, daß Schlüssel vollständig verloren sind. Falls die Schlüssel in ungefugte Hände geraten können, sind weitere geeignete Maßnahmen zu ergreifen, etwa das Führen von Sperrlisten durch die Stellen, die Schlüssel zertifiziert haben. Bevor Teilnehmer Schlüssel verwenden, müssen sie bei der Stelle, die sie zertifiziert hat, nachfragen, ob der Schlüssel nicht gesperrt ist.

137 Hierzu kann jeder seine eigenen geheimen Schlüssel mit einem Schwellwertschema (§3.9.6) in Teile zerlegen und einzelne Teile an ihm vertrauenswürdig scheinende Freunde, Organisationen etc. weitergeben. Zur Rekonstruktion ist dann eine vom Teilnehmer vor der Zerlegung gewählte Mindestzahl an Teilen nötig. Bei geschickter Wahl dieser Mindestzahl können einerseits wenige doch nicht vertrauenswürdige Teilverwahrer den geheimen Schlüssel nicht rekonstruieren, andererseits verhindert der Verlust einzelner Teile nicht die Rekonstruktion.

138 etwa für Archivierungszwecke, z.B. in Grundbuchämtern oder im Gesundheitswesen.

Regel zum Verlust der Daten bzw. ihrer Prüfbarkeit. Hier muß der Schlüsselinhaber also ggf. Vorsorgemaßnahmen wie Schlüsselbackup oder Key-Escrow einplanen.

Schlüsselbackup und Key-Escrow sind nur für Anwendungen sinnvoll, bei denen Daten nicht auf anderem Wege wiederbeschafft werden können, da die Sicherheit der geheimen Schlüssel in jedem Falle sinkt, wenn außer dem Schlüsseleigentümer noch weitere Personen oder Institutionen im Besitz (von Teilen) des geheimen Schlüssels sind.

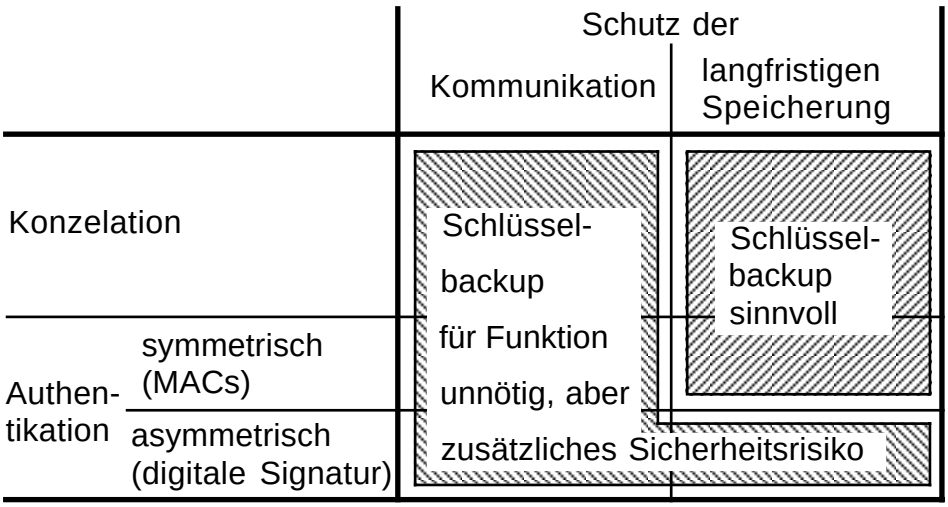


Bild 9-6: Wo ist Schlüsselbackup sinnvoll, wo unnötig und ein zusätzliches Sicherheitsrisiko

Für die Archivierung längerfristig gespeicherter Daten können also Schlüsselbackupmechanismen sinnvoll sein, die allerdings den Bedürfnissen der Benutzer Rechnung tragen sollten. So sollte jedes Schlüsselbackup technisch und organisatorisch so gestaltet werden, daß der Inhaber des Schlüssels von der Rekonstruktion des Schlüssels zumindest sofort erfährt, wenn nicht an ihr mitwirken muß. Nur so bleibt er über die Sicherheit seiner Daten im Bilde. Dies ist eine wichtige erste Verteidigungslinie gegen Mißbrauch [AABB_97 Kap. 2.3]. Gerade diese Forderung erfüllen Key-Escrow-Systeme nicht.

Benutzern von Verschlüsselung Key-Escrow-Systeme mit dem Argument nahe bringen zu wollen, damit sei auch ihr Schlüsselbackup-Problem gelöst, ist also unseriös: Zwei unabhängige Probleme, denen unterschiedliche Schutzinteressen zugrunde liegen, müssen auseinandergehalten und einzeln gelöst werden.

Eine vertiefte Darstellung der Möglichkeiten von Key Recovery und ihrer jeweiligen Risiken ist in [Weck_98] zu finden. [Weck1_98] enthält die sich daraus ergebenden Empfehlungen.

9.7 Schlußfolgerungen

Kryptographie ermöglicht jedem, Einzelnen wie Organisationen, Vertraulichkeit und Integrität seiner Informationen auch in Räumen ohne physische Kontrolle effizient selbst zu schützen. Geschieht dies nicht, so können beispielsweise beim Internet weiterhin Tausende neugieriger Systemverwalter den e-mail-Verkehr mitlesen und Geheimdienste die elektronische Kommunikation weiterhin sehr bequem für Zwecke der Wirtschaftsspionage auswerten. Ohne sichere Kryptographie kann Information im Netz beliebig manipuliert werden.

Die dezentrale Natur globaler digitaler Kommunikationsnetze hat den Nationalstaaten Kontroll- und Schutzmöglichkeiten entzogen. Kryptographie gibt den Einzelnen die Möglichkeit zurück, ihre

Informationen selbst zu schützen. In einer Informationsgesellschaft geht Kryptographie deshalb alle an, da alle von ihrer (Un)Sicherheit betroffen sind!

Der Einsatz von Kryptographie wird deutlich sicherer, wenn Benutzer die für Kryptographie nötigen Schlüssel von ihren Geräten selbst erzeugen lassen. Das oft gehörte Argument, die Schlüsselgenerierung gehöre zum Zwecke der Verbrechensbekämpfung zusammen mit einer Aufbewahrung auch der geheimen Schlüssel zu den Aufgaben von staatlich autorisierten TrustCentern, ist aus den dargelegten technischen Gründen nicht haltbar. Das Argument, dies diene sogar den Interessen des Nutzers, entpuppt sich bei genauer Betrachtung als unzutreffend. Entsprechendes gilt auch für Key-Escrow.

Eine Regulierung kryptographischer Techniken zur Konzelation mit dem Ziel der Verbrechensbekämpfung ist nicht sinnvoll, da eine Kryptoreglementierung nicht wirksam durchsetzbar ist[139], aber die Sicherheitsinteressen von Bürgern, Wirtschaft und Gesellschaft empfindlich beeinträchtigt. Die im Signaturgesetz vorgenommene fördernde Regulierung von Kryptographie zur Integritätssicherung und Beweiseignung digitaler Dokumente ist zu begrüßen.

Leseempfehlungen

Zur Vertiefung von §9.6: AABB_98

Diskussion zur Kryptokontroverse in Deutschland

Wortprotokolle des Wiesbadener Forums Datenschutz am 19. Juni 1997: HaMö_98

Eine aktuelle Pressesammlung:

Nicolas Reichelt: Verhindert ein Kryptographie-Gesetz; http://www.crypto.de/#press

Zu rechtlichen Regelung(sversuch)en der Kryptographie:

European Cryptography Resources; http://www.modeemi.cs.tut.fi/~avs/eu-crypto.html

Bert-Jaap Koops: Crypto Law Survey; http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm

Ulf Möller: Rechtliche Situation, politische Diskussion; http://www.thur.de/ulf/krypto/verbot.html

139 Das in [HaMö_98 Seite 90] vorgebrachte Argument, es gehe bei einer Kryptoreglementierung gar nicht um die Benutzer von PCs, sondern von Telefonen, ist aus technischen Gründen unhaltbar: Etwa im Jahre 2007 werden auch die portablen Geräte wie Handys programmierbar sein oder die programmierbaren Multimedia-Workstations tragbar. Dann können sie wie die heutigen PCs vom Benutzer bequem zur Verschlüsselung mit PGP auch aller seiner Telefongespräche verwendet werden [HaMö_98 Seite 112f].

10 Entwurf mehrseitig sicherer IT-Systeme

Ziel dieses Schlußkapitels ist, das bisher Vermittelte zusammenfassend und integrierend unter der Frage zu ordnen: Wie entwerfe ich ein mehrseitig sicheres IT-System? Nach allem Gesagten wird ein wesentlicher Aspekt sein: Wie entwerfe ich ein durch Verteilung mehrseitig sicheres System? Aus der Not des Entwurfes sicherer verteilter Systeme soll eine Tugend werden.

<<< noch zu schreiben >>>

Bezeichner und Notationen

a	Länge der Ausgabeeinheit
$\Leftrightarrow$	(davorstehende Aussage) ist äquivalent zu (danach stehender Aussage)
A	vom Angreifer kontrollierte Station
B	Bank
b	Tiefe des Referenzenbaums bei GMR
	Blocklänge einer Blockchiffre
	verfügbare Bandbreite zur Signalisierung (in §5.5.2 wird b = 12 kbit/s angenommen)
$\|p\|$	Betrag von *p*
	Länge von *p* in Bits
	Anzahl der Elemente der Menge *p*
c	Chiffrierschlüssel eines asymmetrischen Konzelationssystems
CAN	Controller Area Network
C_i	28-Bit-Block (des Schlüssels von DES) (i = 1, ..., 16)
code	Codieralgorithmus
const	Konstante
CRA	Chinesischer Restealgorithmus
d	Dechiffrierschlüssel eines asymmetrischen Konzelationssystems
D_i	28-Bit-Block (des Schlüssels von DES) (i = 1, ..., 16)
δ	Abstand vom rechten Rand der Rückkopplungseinheit
$\in$	Element von
$\in_R$	gleichverteilt zufällig (Random) gewählt aus, d.h. $x \in_R M$ bedeutet, daß das Element x zufällig aus der Menge M gewählt wird
ent	Entschlüsselungsalgorithmus
ε	Fehlerwahrscheinlichkeit
E	Expansionspermutation (von DES)
E	Ereignis
	Empfänger
$\exists$	es existiert ein
exp(..)	Exponentialfunktion, d.h. $\exp(x) = e^x \approx 2{,}718^x$
f	Faktor

f(..)	Funktion
	Verschlüsselungsfunktion von DES
F	Faktorisierungsalgorithmus bzw. probabilistischer polynomialer Algorithmus
$\forall$	für alle
G	Gruppe
	gerufener Partner
	Geheimnis
	Größe
	Großrechner
g	Generator einer zyklischen Gruppe
gen	Schlüsselgenerierungsalgorithmus
ggT	größter gemeinsamer Teiler
G^*	ungerichteter Graph
h	Hashfunktion
H	Head
H	Untergruppe
IP	Eingangspermutation (Initial Permutation) von DES
IP^{-1}	Ausgangspermutation von DES
IT-System	informationstechnisches System
$\times$	Kreuzprodukt von Mengen
K	Schlüsselmenge
	Kollisionsfinde-Algorithmus
K	Klartextblock
K_i	Teilschlüssel von DES (i = 1, ..., 16)
k	geheimer Schlüssel
kgV	kleinste gemeinsame Vielfache
km	Kilometer
L	linke Hälfte eines DES-Blocks
LAN	Local Area Network
L	Diskrete-Logarithmen-Ziehalgorithmus bzw. probabilistischer polynomialer Algorithmus
l	Länge eines Schlüssels
	Sicherheitsparameter

ld	Logarithmus dualis, Logarithmus zur Basis 2
ln	natürlicher Logarithmus
LS_i	zyklischer Linksshift (in Iterationsrunde *i* bei DES) (i = 1, ..., 16)
m	Meter
m	Nachricht (message)
MAC	message authentication code (als Bezeichner eines Nachrichtenfeldes)
MAC	message authentication code (als Bezeichner des Wertes eines Nachrichtenfeldes)
N	Nachricht
N	Nennwert einer Zahlung
$\mathbb{N}$	Menge der natürlichen Zahlen = {1, 2, 3, 4, ...}
$\mathbb{N}_0$	Menge der natürlichen Zahlen einschließlich 0 = {0, 1, 2, 3, 4, ...}
\	ohne, d.h. Mengensubtraktion
O(..)	Order-of-magnitude: eine Funktion f(*n*) heißt größenordnungsmäßig höchstens so schnell wachsend wie eine Funktion g(*n*), gesprochen „f ist ein groß O von g“, notiert f(*n*) = O(g(*n*)), wenn es Konstanten *const* und n_0 gibt, so daß für alle $n \geq n_0$ gilt: $\|f(n)\| \leq const \cdot \|g(n)\|$
o.B.d.A.	ohne Beschränkung der Allgemeinheit
p	Primzahl
p	Pseudonym
P	Prädiktor
P	Permutation (von DES)
§	Abschnitt, Kapitel
PC	Personal Computer
PC-1	Permuted Choice 1 (bei DES)
PC-2	Permuted Choice 2 (bei DES)
PGP	Pretty Good Privacy
$\phi(n)$	Eulersche ϕ-Funktion: Speziell für *p*, *q* prim und $p \neq q$ gilt: $\phi(p \cdot q) = (p-1)(q-1)$
$\pi(x)$	Anzahl der Primzahlen $\leq x$
präf(•)	präfixfreie Abbildung
Q	Polynom
q	Primzahl
r	Zufallszahl (verwendet als Blendungsfaktor) (random number)
$\mathcal{R}$	Rate-Algorithmus
R	Schlüsselregister

	Referenz einer GMR-Signatur
	rufender Partner
	Rolle
R	Rolle in einer Transaktion
	rechte Hälfte eines DES-Blocks
$\mathbb{R}$	Menge der reellen Zahlen
$\mathcal{S}$	Menge der Schlüsseltexte
S	Schlüsseltext
	Schlüsseltextblock
	Schlüsseltextzeichen
	Sender
	Signatur
S	steganographischer Algorithmus „Verbergen“
S^{-1}	Umkehroperation zu S, d.h. ein Algorithmus, der die eingebetteten Daten extrahiert
S_i	Substitutionsbox (von DES) (i = 1, ..., 8)
s	Sekunde
s	Signierschlüssel eines digitalen Signatursystems
	Startwert eines Pseudozufallsbitfolgengenerators
Sig	Signatur
sign	Signieralgorithmus
$a \mid b$	a teilt b (ohne Rest), d.h. $\exists\, c \in \mathbb{Z}: b = ac$
t	Testschlüssel eines digitalen Signatursystems
T	Tail
T	Teilnehmerstation
test	Testalgorithmus
$\approx$	ungefähr gleich
ver	Verschlüsselungsalgorithmus
W	Wurzelziehalgorithmus
W	Wahrscheinlichkeitsverteilung
$W(x)$	Wahrscheinlichkeit von x
$W(x\|s)$	bedingte Wahrscheinlichkeit von x, wenn man s kennt
w	Wurzel

X	Menge der Klartexte
X	Person
x	Klartext
	Klartextzeichen
XOR	bitweises XOR = bitweise Addition mod 2
z	Zufallszahl
	zufällige Bitkette
Z	Schlüsselverteilzentrale
	Zufallsvariable
$\mathbb{Z}$	Menge der ganzen Zahlen = {..., -4, -3, -2, -1, 0, 1, 2, 3, 4, ...}
$\mathbb{Z}_n$	Restklassen(ring) mod n
$\mathbb{Z}_n^*$	multiplikative Gruppe der zu n teilerfremden Restklassen mod n
$\lceil x \rceil$	kleinste ganze Zahl z mit $z \geq x$
$\lfloor y \rfloor$	größte ganze Zahl z mit $z \leq y$

Literatur

AABB_97 Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffery I. Schiller, Bruce Schneier: The Risks of Key Recovery, Key Escrow, and Trusted Third Party Encryption; Final Report – 27 May 1997; http://theory.lcs.mit.edu/~rivest/publications.html.

AABB_98 Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffery I. Schiller, Bruce Schneier: The Risks of Key Recovery, Key Escrow, and Trusted Third Party Encryption; Update June 1998; http://www.cdt.org/crypto/risks98.

AbJe_87 Marshall D. Abrams, Albert B. Jeng: Network Security: Protocol Reference Model and the Trusted Computer System Evaluation Criteria; IEEE Network 1/2 (1987) 24-33.

ACGS_84 Werner Alexi, Benny Chor, Oded Goldreich, Claus P. Schnorr: RSA/Rabin Bits are 0.5 + 1/poly(log N) Secure; 25th Symposium on Foundations of Computer Science (FOCS) 1984, IEEE Computer Society, 1984, 449-457.

ACGS_88 Werner Alexi, Benny Chor, Oded Goldreich, Claus P. Schnorr: RSA and Rabin functions: Certain parts are as hard as the whole; SIAM J. Comput. 17/2 (1988) 194-209.

Adam2_92 John A. Adam: Threats and countermeasures; IEEE spectrum 29/8 (1992) 21-28.

AJSW_97 N. Asokan, Phillipe A. Janson, Michael Steiner, Michael Waidner: The State of the Art in Electronic Payment Systems; Computer 30/9 (1997) 28-35.

Akl_83 Selim G. Akl: Digital Signatures: A Tutorial Survey; Computer 16/2 (1983) 15-24.

Alba_83 A. Albanese: Star Network With Collision-Avoidance Circuits; The Bell System Technical Journal 62/3 (1983) 631-638.

Alke_88 Horst Alke: DATENSCHUTZBEHÖRDEN: Alles registriert – nur beim Autotelefon? Registrierung durch die DBP beim "normalen Telefon"; Vollspeicherung beim Autotelefondienst; Datenschutz und Datensicherung DuD /1 (1988) 4-5.

AlSc_83 Bowen Alpern, Fred B. Schneider: Key exchange Using 'Keyless Cryptography'; Information Processing Letters 16 (1983) 79-81.

Amst_83 Stanford R. Amstutz: Burst Switching -- An Introduction; IEEE Communications Magazine 21/8 (1983) 36-42.

Ande4_96 Ross Anderson: Stretching the Limits of Steganography; Information Hiding, LNCS 1174, Springer-Verlag, Berlin 1996, 39-48.

AnKu_96 Ross Anderson, Markus Kuhn: Tamper Resistance - a Cautionary Note; 2nd USENIX Workshop on Electronic Commerce, 1996, 1-12.

AnLe_81 T. Anderson, P. A. Lee: Fault Tolerance - Principles and Practice; Prentice Hall, Englewood Cliffs, New Jersey, 1981.

AnPe_98 Ross J. Anderson, Fabien A. P. Petitcolas: On the Limits of Stenography; IEEE Journal on Selected Areas in Communications 16/4 (1998) 474-481.

AsSW_97 N. Asokan, Matthias Schunter, Michael Waidner: Optimistic Protocols for Fair Exchange; 4th ACM Conference on Computer and Communications Security, Zürich, April 1997, 6-17.

Aßma_88 Ralf Aßmann: Effiziente Software-Implementierung von verallgemeinertem DES; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe, Abgabe Februar 1989; ursprünglich als Studienarbeit "Effiziente MC 68000 Assembler-Implementierung von verallgemeinertem DES" in 1988 begonnen.

BaGo_84 Friedrich L. Bauer, Gerhard Goos: Informatik, Eine einführende Übersicht; Zweiter Teil; Dritte Auflage völlig neu bearbeitet und erweitert von F. L. Bauer; Heidelberger Taschenbücher Band 91; Springer-Verlag, Berlin 1984.

BaHS_92 Dave Bayer, Stuart Haber, W. Scott Stornetta: Improving the Efficiency and Reliability of Digital Time-Stamping; Sequences '91: Methods in Communication, Security, and Computer Sciences, Springer-Verlag, Berlin 1992, 329-334.

Bara_64 Paul Baran: On Distributed Communications: IX. Security, Secrecy, and Tamper-Free Considerations; Memorandum RM-3765-PR, August 1964, The Rand Corporation, 1700 Main St, Santa Monica, California, 90406 Reprinted in: Lance J. Hoffman (ed.): Security and Privacy in Computer Systems; Melville Publishing Company, Los Angeles, California, 1973, 99-123.

Baum_99 Birgit Baum-Waidner: Haftungsbeschränkung der digitalen Signatur durch einen Commitment Service; in: Patrick Horster (Hrsg.): Sicherheitsinfrastrukturen; DuD-Fachbeiträge Vieweg, Wiesbaden 1999, 65-80.

BCKK_83 Werner Bux, Felix H. Closs, Karl Kuemmerle, Heinz J. Keller, Hans R. Mueller: Architecture and Design of a Reliable Token-Ring Network; IEEE Journal on Selected Areas in Communications 1/5 (1983) 756-765.

BDFK_95 Andreas Bertsch, Herbert Damker, Hannes Federrath, Dogan Kesdogan, Michael J. Schneider: Erreichbarkeitsmanagement; PIK Praxis der Informationsverarbeitung und Kommunikation /4 (1995) 231-234.

BeBE_92 Charles H. Bennett, Gilles Brassard, Artur K. Ekert: Quantum Cryptography; Scientific American 267/4 (1992) 26-33.

BeBR_88 Charles H. Bennett, Gilles Brassard, Jean-Marc Robert: Privacy amplification by public discussion; SIAM J. Comput. 17/2 (1988) 210-229.

BeEG_86 F. Belli, K. Echtle, W. Görke: Methoden und Modelle der Fehlertoleranz; Informatik Spektrum 9/2 (1986) 68-81.

BeFG_89 Wilfried Beller, Jürgen Frößl, Thomas Giesler: Spezifikation und Implementierung eines erweiterten DES-Algorithmus als VENUS-Standardzellenchip; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe (Betreuer: Oliver Haberl, Thomas Kropf), 1989.

BeGW_88 Michael Ben-Or, Shafi Goldwasser, Avi Wigderson: Completeness theorems for non-cryptographic fault-tolerant distributed computation; 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 1-10.

BeRo_95 Mihir Bellare, Phillip Rogaway: Optimal Asymmetric Encryption; Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 92-111.

Ber1_83 Thomas A. Berson: Long Key Variants of DES; Crypto '82, Plenum Press, New York 1983, 311-313.

BFD1_95 BfD–Info 1: Bundesdatenschutzgesetz – Text und Erläuterung; 3. Auflage August 1995 – herausgegeben vom Bundesbeauftragten für den Datenschutz, PF 200112, D-53131 Bonn; dort kostenlos anforderbar.

BFD5_98 BfD–Info 5: Datenschutz und Telekommunikation; 2. Auflage September 1998 – herausgegeben vom Bundesbeauftragten für den Datenschutz, PF 200112, D-53131 Bonn; dort kostenlos anforderbar.

BGJK_81 P. Berger, G. Grugelke, G. Jensen, J. Kratzsch, R. Kreibich, H. Pichlmayer, J. P. Spohn: Datenschutz bei rechnerunterstützten Telekommunikationssystemen; Bundesministerium für Forschung und Technologie Forschungsbericht DV 81-006 (BMFT-FB-DV 81-006), Institut für Zukunftsforschung GmbH Berlin, September 1981.

BGMR_85 Michael Ben-Or, Oded Goldreich, Silvio Micali, Ronald L. Rivest: A Fair Protocol for Signing Contracts; LNCS 194, Automata, Languages and Programming (ICALP), 12th Colloquium, Nafplion, Greece, July 15-19, 1985, Wilfried Brauer (ed.), Springer-Verlag, Heidelberg, 43-52.

BiSh3_91 Eli Biham, Adi Shamir: Differential Cryptanalysis of DES-like Cryptosystems; Journal of Cryptology 4/1 (1990) 3-72.

BiSh4_91 Eli Biham, Adi Shamir: Differential Cryptanalysis of the Full 16-round DES; Technical Report no. 708, December 1991, Computer Science Department, Technion, Haifa, Israel.

BiSh_93 Eli Biham, Adi Shamir: Differential Cryptanalysis of the Data Encryption Standard; Springer-Verlag, New York 1993.

BlBS_86 L. Blum, M. Blum, M. Shub: A Simple Unpredictable Pseudo-Random Number Generator; SIAM J. Comput. 15/2 (1986) 364-383.

Bleu_90 Gerrit Bleumer: Vertrauenswürdige Schlüssel für ein Signatursystem, dessen Brechen beweisbar ist; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe 1990.

Bleu_99 Gerrit Bleumer: Biometrische Ausweise – Schutz von Personenidentitäten trotz biometrischer Erkennung; Datenschutz und Datensicherheit 23/3 (1999) 155-158.

BlFM_88 Manuel Blum, Paul Feldman, Silvio Micali: Non-interactive zero-knowledge and its applications (extended abstract); 20th Symposium on Theory of Computing (STOC)1988, ACM, New York 1988, 103-112.

BlMi_84 Manuel Blum, Silvio Micali: How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits; SIAM J. Comput. 13/4 (1984) 850-864.

BlPW_91 Gerrit Bleumer, Birgit Pfitzmann, Michael Waidner: A remark on a signature scheme where forgery can be proved; Eurocrypt '90, LNCS 473, Springer-Verlag, Berlin 1991, 441-445.

BMI_89 Bundesministerium des Inneren: Rahmenkonzept zur Gewährleistung der Sicherheit bei Anwendung der Informationstechnik; Datenschutz und Datensicherung DuD /6 (1989) 292-299.

BMTW_84 Toby Berger, Nader Mehravari, Don Towsley, Jack Wolf: Random Multiple-Access Communication and Group Testing; IEEE Transactions on Communications 32/7 (1984) 769-779.

Bött_89 Manfred Böttger: Untersuchung der Sicherheit von asymmetrischen Kryptosystemen und MIX-Implementierungen gegen aktive Angriffe; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe 1989.

Bras_83 Gilles Brassard: Relativized Cryptography; IEEE Transactions on Information Theory 29/6 (1983) 877-894.

Bras_88 Gilles Brassard: Modern Cryptology - A Tutorial; LNCS 325, Springer-Verlag, Berlin 1988.

Brau_82 Ewald Braun: BIGFON - der Start für die Kommunikationstechnik der Zukunft; SIEMENS telcom report 5/2 (1982) 123-129.

Brau_83 Ewald Braun: BIGFON - Erprobung der optischen Breitbandübertragung im Ortsnetz; SIEMENS telcom report 6/2 (1983) 52-53.

Brau_84 Ewald Braun: Systemversuch BIGFON gestartet; SIEMENS telcom report 7/1 (1984) 9-11.

Brau_87 E. Braun: BIGFON; Informatik-Spektrum 10/4 (1987) 216-217.

BrDL_91 Jørgen Brandt, Ivan Damgård, Peter Landrock: Speeding up Prime Number Generation; AsiaCrypt '91 - Abstracts, 265-269.

BüPf_87 Holger Bürk, Andreas Pfitzmann: Value Transfer Systems Enabling Security and Unobservability; Interner Bericht 2/87, Fakultät für Informatik, Universität Karlsruhe 1987.

BüPf_89 Holger Bürk, Andreas Pfitzmann: Digital Payment Systems Enabling Security and Unobservability; Computers & Security 8/5 (1989) 399-416.

BüPf_90 Holger Bürk, Andreas Pfitzmann: Value Exchange Systems Enabling Security and Unobservability; Computers & Security 9/8 (1990) 715-721.

Bura_88 Axel Burandt: Informationstheoretisch unverkettbare Beglaubigung von Pseudonymen mit beliebigen Signatursystemen; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe, Mai 1988.

Bürk_86 Holger Bürk: Digitale Zahlungssysteme und betrugssicherer, anonymer Wertetransfer; Studienarbeit am Institut für Informatik IV, Universität Karlsruhe, April 1986.

CBHM_90 David Chaum, Bert den Boer, Eugène van Heyst, Stig Mjølsnes, Adri Steenbeek: Efficient offline electronic checks; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 294-301.

Cha1_83 David Chaum: Blind Signature System; Crypto '83, Plenum Press, New York 1984, 153.

Cha1_84 David Chaum: A New Paradigm for Individuals in the Information Age; 1984 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Washington 1984, 99-103.

Cha1_88 D. Chaum: Blinding for unanticipated signatures; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 227-233.

Cha1_89 David Chaum: Online Cash Checks; Eurocrypt '89; Houthalen, 10.-13. April 1989, A Workshop on the Theory and Application of Cryptographic Techniques, 167-170.

Cha3_85 David Chaum: The Dining Cryptographers Problem. Unconditional Sender Anonymity; Draft, received May 13, 1985.

Cha3_87 David Chaum: Security without Identification: Card Computers to make Big Brother Obsolete; Draft, received 24.6.1987.

Cha8_85 David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044.

Cha9_85 David Chaum: Cryptographic Identification, Financial Transaction, and Credential Device; United States Patent, Patent Number 4,529,870; Date of Patent: Jul. 16, 1985, Filed Jun. 25, 1982.

ChAn_90 David Chaum, Hans van Antwerpen: Undeniable signatures; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 212-216.

Chau_81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.

Chau_83 David Chaum: Blind Signatures for untraceable payments; Crypto '82, Plenum Press, New York 1983, 199-203.

Chau_84 David Chaum: Design Concepts for Tamper Responding Systems; Crypto '83, Plenum Press, New York 1984, 387-392.

Chau_85 David Chaum: Showing credentials without identification. Signatures transferred between unconditionally unlinkable pseudonyms; Abstracts of Eurocrypt '85, April 9-11, 1985, Linz, Austria, IACR, General Chairman: Franz Pichler, Program Chairman: Thomas Beth.

Chau_87 David Chaum: Sicherheit ohne Identifizierung; Scheckkartencomputer, die den Großen Bruder der Vergangenheit angehören lassen; Informatik-Spektrum 10/5 (1987) 262-277; Datenschutz und Datensicherung DuD /1 (1988) 26-41.

Chau_88 David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability; Journal of Cryptology 1/1 (1988) 65-75.

Chau_89 David Chaum: Privacy Protected Payments – Unconditional Payer and/or Payee Untraceability; SMART CARD 2000: The Future of IC Cards, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20. 10. 1987, North-Holland, Amsterdam 1989, 69-93.

Chau_90 David Chaum: The Spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 591-602.

Chau_92 David Chaum: Achieving Electronic Privacy; Scientific American (August 1992) 96-101.

Chau_95 David Chaum: Designated Confirmer Signatures; Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 86-91.

Chau2_90 David Chaum: Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms; Auscrypt '90, LNCS 453, Springer-Verlag, Berlin 1990, 246-264.

ChCD1_88 David Chaum, Claude Crépeau, Ivan Damgård: Multiparty unconditional secure protocols; 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 11-19.

ChDG_88 David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

ChEv_87 David Chaum, Jan-Hendrik Evertse: A secure and privacy-protecting protocol for transmitting personal information between organizations; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 118-167.

ChFN_90 D. Chaum, A. Fiat, M. Naor: Untraceable Electronic Cash; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 319-327.

ChRo_91 David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 206-214.

Clar_88 A. J. Clark: Physical protection of cryptographic devices; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 83-93.

Clem_85 Rudolf Clemens: Die elektronische Willenserklärung - Chancen und Gefahren; Neue Juristische Wochenschrift NJW /324 (1985) 1998-2005.

Cles_88 Wolfgang Clesle: Schutz auch vor Herstellern und Betreibern von Informationssystemen; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, Juni 1988.

ClPf_91 Wolfgang Clesle, Andreas Pfitzmann: Rechnerkonzept mit digital signierten Schnittstellenprotokollen erlaubt individuelle Verantwortungszuweisung; Datenschutz-Berater 14/8-9 (1991) 8-38.

CoBi_95 David A. Cooper, Kenneth P. Birman: Preserving Privacy in a Network of Mobile Computers; 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995, 26-38.

CoBi1_95 David A. Cooper, Kenneth P. Birman: The design and implementation of a private message service for mobile computers; Wireless Networks 1 (1995) 297-309.

Cohe_84 Fred Cohen: Computer Viruses, Theory and Experiments; Proceedings of the 7th National Computer Security Conference, 1984, National Bureau of Standards, Gaithersburg, MD; USA, 240-263.

Cohe_87 Fred Cohen: Computer Viruses – Theory and Experiments; Computers & Security 6/1 (1987) 22-35.

Cohe1_89 Fred Cohen: Computational Aspects of Computer Viruses; Computers & Security 8/3 (1989) 325-344.

Cohe2_92 F. B. Cohen: A Formal Definition of Computer Worms and Some Related Results; Computers & Security 11/7 (1992) 641-652.

Copp_92 Don Coppersmith: DES and differential cryptanalysis; appeared in an IBM internal newsgroup (CRYPTAN FORUM).

Copp2_94 D. Coppersmith: The Data Encryption Standard (DES) and its strength against attacks; IBM Journal of Research and Development 38/3 (1994) 243-250.

Cove_90 Minutes of the First Workshop on Covert Channel Analysis: Overview, Introduction and Welcomes, Retrospective, Worked Examples, Covert Channel Examples, Research Panel, Vendor Panel, Policy Panel, General Discussion, Research Working Group Discussion, Results of the Research Working Group, Policy Working Group Discussion, Results of the Policy Working Group, Summary Discussion; CIPHER Newsletter of the TC on Security & Privacy, IEEE Computer Society (Special Issue, July 1990) 1-35.

CrSh_98 Ronald Cramer, Victor Shoup: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack; Crypto '98, LNCS 1462, Springer-Verlag, Berlin 1998, 13-25.

CTCPEC_92 Canadian System Security Centre; Communications Security Establishment; Government of Canada: The Canadian Trusted Computer Product Evaluation Criteria; April 1992, Version 3.0e.

CZ_98 Internet-Rechner knacken DES-Code; Alte Entschlüsselungszeit ist halbiert; Computer Zeitung Nr. 10, 5. März 1998, 7.

Damg_88 Ivan Bjerre Damgård: Collision free hash functions and public key signature schemes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.

Damg_92 Ivan Damgård: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks; Crypto '91, LNCS 576, Springer Verlag, Berlin 1992, 445-456.

DaPa_83 D. W. Davies, G. I. P. Parkin: The Average Cycle Size of the Key Stream in Output Feedback Encipherment; Cryptography; Proceedings, Burg Feuerstein 1982, Edited by Thomas Beth; LNCS 149, Springer-Verlag, Heidelberg, 1983, 263-279.

DaPr_84 D. W. Davies, W. L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer; John Wiley & Sons, New York 1984.

DaPr_89 D. W. Davies, W. L. Price: Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer; (2nd ed.) John Wiley & Sons, New York 1989.

Davi_85 Danald Watts Davies: Apparatus and methods for granting access to computers; UK Patent Application, Application Nr. 8503481, Date of filing 11 Feb. 1985, Application published 4 Spt. 1985.

Denn_82 Dorothy Denning: Cryptography and Data Security; Addison-Wesley Publishing Company, Reading 1982; Reprinted with corrections, January 1983.

Denn_84 Dorothy E. Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems; Communications of the ACM 27/4 (1984) 388-392.

Denn_85 Dorothy E. Denning: Commutative Filters for Reducing Inference Threats in Multilevel Database Systems; Proceedings of the 1985 Symposium on Security and Privacy, April 22-24, 1985, Oakland, California, IEEE Computer Society, 134-146.

DES_77 Specification for the Data Encryption Standard; Federal Information Processing Standards Publication 46 (FIPS PUB 46), January 15, 1977.

Dier3_91 Rüdiger Dierstein: Programm-Manipulationen: Arten, Eigenschaften und Bekämpfung; Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR), Zentrale Datenverarbeitung, D-8031 Oberpfaffenhofen; Institutsbericht IB 562/6, Juli 1986, Neufassung 1991.

DiHe_76 Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654.

DiHe_79 Whitfield Diffie, Martin E. Hellman: Privacy and Authentication: An Introduction to Cryptography; Proceedings of the IEEE 67/3 (1979) 397-427.

DINISO8372_87 DIN ISO 8372: Informationsverarbeitung – Betriebsarten für einen 64-bit-Blockschlüsselungsalgorithmus;

Dobb_98 Hans Dobbertin: Cryptanalysis of MD4; Journal of Cryptology 11/4 (1998) 253-271.

DuD_86 DuD: AKTUELL; Datenschutz und Datensicherung DuD /1 (1986) 54-56.

DuDR2_92 DuD Report: Dark Avenger Mutation Engine; Datenschutz und Datensicherung DuD 16/8 (1992) 442-443.

DuD3_99 Schwerpunktheft Biometrie; Datenschutz und Datensicherheit 23/3 (1999).

Eber_98 Ulrich Eberl: Der unverlierbare Schlüssel; Forschung und Innovation – Die SIEMENS-Zeitschrift für Wissenschaft und Technik /1 (1998) 14-19.

EcGM_83 Klaus Echtle, Winfried Görke, Michael Marhöfer: Zur Begriffsbildung bei der Beschreibung von Fehlertoleranz-Verfahren; Universität Karlsruhe, Fakultät für Informatik, Institut für Informatik IV, Interner Bericht Nr. 6/83 Mai 1983.

Echt_90 Klaus Echtle: Fehlertoleranzverfahren; Studienreihe Informatik, Springer-Verlag, Heidelberg 1990.

EFF_98 "EFF DES cracker" machine brings honesty to crypto debate; Press Release July 17, 1998; http://www.eff.org/pub/Privacy/Crypto_misc/DESCracker/HTML/19980716_eff_descracker_pressrel.html und http://www.eff.org/descracker/.

ElGa_85 Taher ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms; IEEE Transactions on Information Theory 31/4 (1985) 469-472.

EMMT_78 W. F. Ehrsam, S. M. Matyas, C. H. Meyer, W. L. Tuchman: A cryptographic key management scheme for implementing the Data Encryption Standard; IBM Systems Journal 17/2 (1978) 106-125.

EnHa_96 Erin English, Scott Hamilton: Network Security Under Siege; The Timing Attack; Computer 29/3 (1996) 95-97.

Even_89 Shimon Even: Secure Off-Line Electronic Fund Transfer Between Nontrusting Parties; SMART CARD 2000: The Future of IC Cards, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20. 10. 1987, North-Holland, Amsterdam 1989, 57-66.

EvGL_85 Shimon Even, Oded Goldreich, Abraham Lempel: A Randomized Protocol for Signing Contracts; Communications of the ACM 28/6 (1985) 637-647.

EvGY_84 S. Even, O. Goldreich, Y. Yacobi: Electronic Wallet; 1984 International Zurich Seminar on Digital Communications, Applications of Source Coding, Channel Coding and

Secrecy Coding, March 6-8, 1984, Zurich, Switzerland, Swiss Federal Institute of Technology, Proceedings IEEE Catalog Nr. 84CH1998-4, 199-201.

FaLa_75 David J. Farber, Kenneth C. Larson: Network Security Via Dynamic Process Renaming; Fourth Data Communications Symposium, 7-9 October 1975, Quebec City, Canada, 8-13-8-18.

Fede_98 Hannes Federrath: Vertrauenswürdiges Mobilitätsmanagement in Telekommunikationsnetzen. Dissertation, TU Dresden, Fakultät Informatik, Februar 1998.

Fede_99 Hannes Federrath: Sicherheit mobiler Kommunikation; DuD-Fachbeiträge, Vieweg, Wiesbaden 1999.

FeJP_96 Hannes Federrath, Anja Jerichow, Andreas Pfitzmann: Mixes in mobile communication systems: Location management with privacy; Information Hiding, First International Workshop, Cambridge, UK, May/June 1996, LNCS 1174, Springer-Verlag, Heidelberg 1996, 121-135.

FeMa_98 Hannes Federrath, Kai Martius: Mehrseitig sicherer Web-Zugriff; KES Zeitschrift für Kommunikations- und EDV-Sicherheit 14/4 (1998) 10-12.

FeMa1_98 Hannes Federrath, Kai Martius: Anonymität und Authentizität im World Wide Web; ITG-Fachbericht 153, VDE-Verlag, Berlin, Offenbach, 1998, 91-101.

FePf_97 Hannes Federrath, Andreas Pfitzmann: Bausteine zur Realisierung mehrseitiger Sicherheit; in: G. Müller, A. Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 83-104.

FePf_99 Hannes Federrath, Andreas Pfitzmann: Stand der Sicherheitstechnik; in: Kubicek et al (Hrsg.): "Multimedia@Verwaltung", Jahrbuch Telekommunikation und Gesellschaft 1999, Hüthig, 124-132.

FGJP_98 Elke Franz, Andreas Graubner, Anja Jerichow, Andreas Pfitzmann: Einsatz von dummies im Mixnetz zum Schutz der Kommunikationsbeziehungen; in: K. Bauknecht, A. Büllesbach, H. Pohl, S. Teufel (Hrsg.): Sicherheit in Informationssystemen, Proc. der Fachtagung SIS '98, Univ. Hohenheim, März 1998, Hochschulverlag AG an der ETH Zürich, 65-94.

FiSh_87 Amos Fiat, Adi Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 186-194.

FJKP_95 Hannes Federrath, Anja Jerichow, Dogan Kesdogan, Andreas Pfitzmann: Security in Public Mobile Communication Networks; Proc. of the IFIP TC 6 International Workshop on Personal Wireless Communications, Prag 1995, 105-116.

FJKPS_96 Hannes Federrath, Anja Jerichow, Dogan Kesdogan, Andreas Pfitzmann, Otto Spaniol: Mobilkommunikation ohne Bewegungsprofile; it+ti 38/4 (1996) 24-29; Nachgedruckt in: G. Müller, A. Pfitzmann (Hrsg.), Mehrseitige Sicherheit in der Kommunikationstechnik; Addison-Wesley-Longman 1997, 169-180.

FJMP_97 Hannes Federrath, Anja Jerichow, Jan Müller, Andreas Pfitzmann: Unbeobachtbarkeit in Kommunikationsnetzen; GI-Fachtagung Verläßliche IT-Systeme (VIS'97), DuD-Fachbeiträge, Vieweg 1997, 191-210.

FJMPS_96 Elke Franz, Anja Jerichow, Steffen Möller, Andreas Pfitzmann, Ingo Stierand: Computer Based Steganography: How it works and why therefore any restrictions on cryptography are nonsense, at best; Information Hiding, First International Workshop, Cambridge, UK, May/June 1996, LNCS 1174, Springer-Verlag, Heidelberg 1996, 7-21.

FJPP_95 Hannes Federrath, Anja Jerichow, Andreas Pfitzmann, Birgit Pfitzmann: Mehrseitig sichere Schlüsselerzeugung; in: P. Horster (Hrsg.); Trust Center; Proc. der Arbeitskonferenz Trust Center 95; DuD-Fachbeiträge, Vieweg, Wiesbaden 1995, 117-131.

FMSS_96 S.M. Furnell, J.P. Morrisey, P.W. Sanders, C.T. Stockel: Applications of keystroke analysis for improved login security and continuous user authentication; 12th IFIP International Conference on Information Security (IFIP/Sec '96), Chapman & Hall, London 1996, 283-294.

FO_87 FO: Ein Fall für Prometheus; ADAC motorwelt /4 (1987) 50-53.

Folt_87 Hal Folts: Open Systems Standards; IEEE Network 1/2 (1987) 45-46.

FoPf_91 Dirk Fox, Birgit Pfitzmann: Effiziente Software-Implementierung des GMR-Signatursystems; Proc. Verläßliche Informationssysteme (VIS'91), März 1991, Darmstadt, Informatik-Fachberichte 271, Springer-Verlag, Heidelberg 1991, 329-345.

FrJP_97 Elke Franz, Anja Jerichow, Andreas Pfitzmann: Systematisierung und Modellierung von Mixen; GI-Fachtagung Verläßliche IT-Syteme (VIS´97), DuD-Fachbeiträge, Vieweg, 171-190.

FrJW_98 Elke Franz, Anja Jerichow, Guntram Wicke: A Payment Scheme for Mixes Providing Anonymity; IFIP/GI Working Conference "Trends in Electronic Commerce", June 3-5, 1998, Hamburg, LNCS 1402, Springer-Verlag, Heidelberg 1998, 94-108.

FrPf_98 Elke Franz, Andreas Pfitzmann: Einführung in die Steganographie und Ableitung eines neuen Stegoparadigmas; Informatik-Spektrum 21/4 (1998) 183-193.

GaJo_80 Michael R. Garey, David S. Johnson: Computers and Intractability - A Guide to the Theory of NP-Completeness; 2nd Printing, W.H. Freeman and Company, New York 1980.

GGKL_89 Morrie Gasser, Andy Goldstein, Charlie Kaufman, Butler Lampson: The Digital Distributed System Security Architecture; Proc. 12th National Computer Security Conference 1989, 305-319.

GGKM_99 Eran Gabber, Phillip B. Gibbons, David M. Kristol, Yossi Matias, Alain Mayer: Consistent, Yet Anonymous, Web Access with LPWA; Communications of the ACM 42/2 (1999) 42-47.

GGPS_97 G. Gattung, R. Grimm, U. Pordesch, M.J. Schneider: Persönliche Sicherheitsmanager in der virtuellen Welt, in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 182.

GiLB_85 David K. Gifford, John M. Lucassen, Stephen T. Berlin: The Application of Digital Broadcast Communication to Large Scale Information Systems; IEEE Journal on Selected Areas in Communications 3/3 (1985) 457-467.

GiMS_74 E. N. Gilbert, F. J. Mac Williams, N. J. A. Sloane: Codes which detect deception; The Bell System Technical Journal 53/3 (1974) 405-424.

GIPr1_92 Präsidiumsarbeitskreis "Datenschutz und Datensicherung" der Gesellschaft für Informatik: Stellungnahme zu den Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik (ITSEC) V1.2; Datenschutz-Berater 15/4 (1992) 7-12; Datenschutz und Datensicherung DuD 16/5 (1992) 233-236; Informatik-Spektrum 15/4 (1992) 221-224.

GKLR_92 M. Gasser, C. Kaufman, J. Linn, Y. Le Roux, J. Tardo: DASS: Distributed Authentication Security Service; Education and Society, R. Aiken (ed.), Proc. 12th IFIP World Computer Congress 1992, Information Processing 92, Band II, Elsevier Science Publishers B.V. (North-Holland), 1992, 447-456.

GoGM_86 O. Goldreich, S. Goldwasser, S. Micali: How to construct random functions; Journal of the ACM 33/4 (1986) 792-807.

GoKi_86 Shafi Goldwasser, Joe Kilian: Almost All Primes Can be Quickly Certified; 18th Symposium on Theory of Computing (STOC) 1986, ACM, New York 1986, 316-329.

GoMR_88 Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

GoMT_82 Shafi Goldwasser, Silvio Micali, Po Tong: Why and How to Establish a Private Code on a Public Network; 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 134-144.

GoRS_99 David Goldschlag, Michael Reed, Paul Syverson: Onion Routing for Anonymous and Private Internet Connections; Communications of the ACM 42/2 (1999) 39-41.

Groß_91 Michael Groß: Vertrauenswürdiges Booten als Grundlage authentischer Basissysteme; Proc. Verläßliche Informationssysteme (VIS'91), März 1991, Darmstadt, Informatik-Fachberichte 271, Springer-Verlag, Heidelberg 1991, 190-207.

HaMö_98 Reiner Hamm, Klaus Peter Möller (Hrsg.): Datenschutz durch Kryptographie – ein Sicherheitsrisiko? Forum Datenschutz Band 6, Nomos Verlagsgesellschaft Baden-Baden 1998.

Hast_86 Johan Håstad: On Using RSA with low exponent in a public key network; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 403-408.

Hast_88 Johan Håstad: Solving simultaneous modular equations of low degree; SIAM J. Comput. 17/2 (1988) 336-362.

HaSt_91 Stuart Haber, W. Scott Stornetta: How to Time-Stamp a Digital Document; Journal of Cryptology 3/2 (1991) 99-111.

HeKW_85 Franz-Peter Heider, Detlef Kraus, Michael Welschenbach: Mathematische Methoden der Kryptoanalyse; DuD-Fachbeiträge 8, Friedr. Vieweg & Sohn, Braunschweig 1985.

HePe_93 E. van Heyst, T. P. Pedersen: How to make efficient Fail-stop signatures; Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 366-377.

Herd_85 Siegfried Herda: Authenticity, Anonymity and Security in OSIS. An Open System for Information Services; Proceedings der 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, München, Oktober 1985, herausgegeben von P.P.Spies, IFB 113, Springer-Verlag, Berlin 1985, 35-50.

Höck_85 Gunter Höckel: Untersuchung der Datenschutzeigenschaften von Ringzugriffsmechanismen; Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, August 1985.

Home_?? Homer: Ilias; Diogenes Taschenbuch detebe 20779, übersetzt aus dem Altgriechischen von Heinrich Voß, herausgegeben von Peter Von der Mühll; Diogenes Verlag, Zürich 1980.

Hopk_89 John Hopkinson: Information Security – "The Third Facet"; Proc. 1st Annual Canadian Computer Security Conference, 30 January - 1 February, 1989, Ottawa, Canada, Hosted by The System Security Centre, Communications Security Establishment, Government of Canada, 109-129.

HöPf_85 Gunter Höckel, Andreas Pfitzmann: Untersuchung der Datenschutzeigenschaften von Ringzugriffsmechanismen; 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, IFB 113, Springer-Verlag, Berlin 1985, 113-127.

Horg_85 John Horgan: Thwarting the information thieves; IEEE spectrum 22/7 (1985) 30-41.

Hors_85 Patrick Horster: Kryptologie; Reihe Informatik/47, Herausgegeben von K. H. Böhling, U. Kulisch, H. Maurer, Bibliographisches Institut, Mannheim, 1985.

HuPf_96 Michaela Huhn, Andreas Pfitzmann: Technische Randbedingungen jeder Kryptoregulierung; DuD, Datenschutz und Datensicherheit, Vieweg-Verlag 20/1 (1996) 23-26; Leicht erweitert in: G. Müller, A. Pfitzmann (Hrsg.), Mehrseitige Sicherheit in der Kommunikationstechnik; Addison-Wesley-Longman 1997, 497-506.

HuPf2_96 Michaela Huhn, Andreas Pfitzmann: Krypto(de)regulierung; DANA, Datenschutz-Nachrichten; Deutsche Vereinigung für Datenschutz 19/6 (1996) 4-13.

HuPf_98 Michaela Huhn, Andreas Pfitzmann: Kryptographie und Internet; in Claus Leggewie, Christa Maar (Hrsg.), Bollmann Verlag, Mai 1998.

ISO7498-2_87 ISO: Information processing systems – Open Systems Interconnection Reference Model – Part 2: Security architecture; DRAFT INTERNATIONAL STANDARD ISO/DIS 7498-2; ISO TC 97, Submitted on 1987-06-18.

ISO7498-2_89 ISO: Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security architecture; INTERNATIONAL STANDARD ISO 7498-2; 1989.

ISO8731-1_87 ISO: Banking – Approved algorithms for message authentication – Part 1: DEA; ISO International Standard, First edition 01.06.1987.

ITG_87 ITG-Empfehlungen: ISDN-Begriffe; Nachrichtentechnische Zeitschrift ntz 40/11 (1987) 814-819.

ITSEC2_91 European Communities - Commission: ITSEC: Information Technology Security Evaluation Criteria; (Provisional Harmonised Criteria, Version 1.2, 28 June 1991) Office for Official Publications of the European Communities, Luxembourg 1991 (ISBN 92-826-3004-8).

ITSEC2d_91 Europäische Gemeinschaften - Kommission: Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik (ITSEC); (Vorläufige Form der harmonisierten Kriterien, Version 1.2, 28. Juni 1991) Amt für amtliche Veröffentlichungen der Europäischen Gemeinschaften, Luxemburg 1991 (ISBN 92-826-3003-X).

JMPP_98 Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Real-Time Mixes: A Bandwidth-Efficient Anonymity Protocol; IEEE Journal on Selected Areas in Communications, Special Issue "Copyright and privacy protection", 16/4 (May 1998) 495-509.

Kahn_96 David Kahn: The History of Steganography; Information Hiding, First International Workshop, Cambridge, UK, May/June 1996, LNCS 1174, Springer-Verlag, Heidelberg 1996, 1-5.

Kais_82 Wolfgang Kaiser: Interaktive Breitbandkommunikation; Nutzungsformen und Technik von Systemen mit Rückkanälen; Telecommunications, Veröffentlichungen des Münchner Kreis, Band 8, Springer-Verlag, Berlin 1982.

KaPe_00 Stefan Katzenbeisser; Fabien A. P. Petitcolas (Eds.): Information Hiding Techniques for Steganography and Digital Watermarking; Artech House Computer Security Series; Artech House Boston, London 2000.

Karg_77 Paul A. Karger: Non-Discretionary Access Control for Decentralized Computing Systems; Master Thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139, May 1977, Report MIT/LCS/TR-179.

KeMM_83 Heinz J. Keller, Heinrich Meyr, Hans R. Müller: Transmission Design Criteria for a Synchronous Token Ring; IEEE Journal on Selected Areas in Communications 1/5 (1983) 721-733.

Ken1_81 Stephen T. Kent: Security Requirements and Protocols for a Broadcast Scenario; IEEE Transactions on Communications 29/6 (1981) 778-786.

Kesd_99 Dogan Kesdogan: Anonyme Kommunikation in offenen Umgebungen; Dissertation, RWTH-Aachen, 1999.

KFBG_90 T. Kropf, J. Frößl, W. Beller, T. Giesler: A Hardware Implementation of a Modified DES-Algorithm; Microprocessing and Microprogramming 30 (1990) 59-65.

Klei_75 Leonard Kleinrock: Queueing Systems - Volume I: Theory; John Wiley and Sons, New York 1975.

KlPi_97 Herbert Klimant, Rudi Piotraschke: Informationstheoretische Bewertung steganographischer Konzelationssysteme; GI-Fachtagung Verläßliche IT-Systeme (VIS'97), DuD-Fachbeiträge, Vieweg 1997, 225-232.

KlPS_96 Herbert Klimant, Rudi Piotraschke, Dagmar Schönfeld: Informations- und Kodierungstheorie; B.G. Teubner Verlagsgesellschaft Stuttgart, Leipzig 1996.

Knut_97 Donald E. Knuth: The Art of Computer Programming, Vol. 2: Seminumerical Algorithms (3rd ed.); Addison-Wesley, Reading 1997.

Koch_98 Paul Kocher: Differential Power Analysis; http://www.cryptography.com/dpa, Juni 1998.

Köhl_86 Helmut Köhler: Die Problematik automatisierter Rechtsvorgänge, insbesondere von Willenserklärungen (Teil I); Datenschutz und Datensicherung DuD /6 (1986) 337-344.

Kran_86 Evangelos Kranakis: Primality and Cryptography; Wiley-Teubner Series in Computer Science, B. G. Teubner, Stuttgart 1986.

Krat_84 Herbert Krath: Stand und weiterer Ausbau der Breitbandverteilnetze; telematica 84, 18.-20. Juni 1984, Stuttgart, Kongreßband Teil 2 Breitbandkommunikation, Herausgeber: W. Kaiser, VDE-Verlag GmbH, Neue Mediengesellschaft Ulm mbH, 3-17.

LaMa_92 X. Lai, J. L. Massey: Hash functions based on block ciphers; Eurocrypt '92, Extended Abstracts, 24.-28. Mai 1992, Balatonfüred, Ungarn, 53-66.

Lamp_73 Butler W. Lampson: A Note on the Confinement Problem; Communications of the ACM 16/10 (1973) 613-615.

LaSP_82 Leslie Lamport, Robert Shostak, Marshall Pease: The Byzantine Generals Problem; ACM Transactions on Programming Languages and Systems 4/3 (1982) 382-401.

Leib_99 Otto Leiberich: Vom diplomatischen Code zur Falltürfunktion – Hundert Jahre Kyptographie in Deutschland; Spektrum der Wissenschaft Juni 1999, 26-34.

LeMa1_89 Arjen K. Lenstra, Mark S. Manasse: Factoring – where are we now?; IACR Newsletter, A Publication of the International Association for Cryptologic Research, 6/2 (1989) 4-5.

LeMa1_90 Arjen K. Lenstra, Mark S. Manasse: Factoring by electronic mail; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 355-371.

LiFl_83 John O. Limb, Lois E. Flamm: A Distributed Local Area Network Packet Protocol for Combined Voice and Data Transmission; IEEE Journal on Selected Areas in Communications 1/5 (1983) 926-934.

Lips_81 John D. Lipson: Elements of algebra and algebraic computing; Addison-Wesley Publishing Company, Advanced Book Program; Reading, Mass. 1981.

Lüne_87 Heinz Lüneburg: Kleine Fibel der Arithmetik; Lehrbücher und Monographien zur Didaktik der Mathematik, Band 8, BI Wissenschaftsverlag, Mannheim 1987.

LuPW_91 Jörg Lukat, Andreas Pfitzmann, Michael Waidner: Effizientere fail-stop Schlüsselerzeugung für das DC-Netz; Datenschutz und Datensicherung DuD 15/2 (1991) 71-75.

LuRa_89 Michael Luby, Charles Rackoff: A Study of Password Security; Journal of Cryptology 1/3 (1989) 151-158.

Mann_85 Andreas Mann: Fehlertoleranz und Datenschutz in Ringnetzen; Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, Oktober 1985.

MaPf_87 Andreas Mann, Andreas Pfitzmann: Technischer Datenschutz und Fehlertoleranz in Kommunikationssystemen; Kommunikation in Verteilten Systemen, GI/NTG-Fachtagung, Aachen 1987, IFB 130, Springer-Verlag, Heidelberg 1987, 16-30; Überarbeitung in: Datenschutz und Datensicherung DuD 11/8 (1987) 393-405.

Marc_88 Eckhard Marchel: Leistungsbewertung von überlagerndem Empfangen bei Mehrfachzugriffsverfahren mittels Kollisionsauflösung; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, April 1988.

MeMa_82 Carl H. Meyer, Stephen M. Matyas: Cryptography - A New Dimension in Computer Data Security; (3rd printing) John Wiley & Sons, 1982.

MeOV_97 Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: Handbook of Applied Cryptography; CRC Press, Boca Raton 1997.

Merr_83 Michael John Merritt: Cryptographic Protocols; Ph. D. Dissertation, School of Information and Computer Science, Georgia Institute of Technology, February 1983.

Meye_81 Meyers Lexikonredaktion: Meyers Großes Taschenlexikon in 24 Bänden (Lizenzausgabe „Farbiges Großes Volkslexikon“, Mohndruck, Gütersloh); B.I.-Taschenbuchverlag, Mannheim 1981.

Meye_87 Meyers Lexikonredaktion: Meyers Großes Taschenlexikon in 24 Bänden; 2. neu bearbeitete Auflage, B.I.-Taschenbuchverlag, Mannheim 1987.

Mill3_94 Benjamin Miller: Vital signs of identity; IEEE spectrum 31/2 (1994) 22-30.

MöPS_94 Steffen Möller, Andreas Pfitzmann, Ingo Stierand: Rechnergestützte Steganographie: Wie sie funktioniert und warum folglich jede Reglementierung von Verschlüsselung unsinnig ist; Datenschutz und Datensicherung DuD 18/6 (1994) 318-326.

MüPf_97 Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik; Addison-Wesley, Bonn 1997.

NaYu_90 Moni Naor, Moti Yung: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks; Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC), May 14-16, 1990, Baltimore-Maryland, ACM Press, 427-437.

NeKe_99 Heike Neumann, Volker Kessler: Formale Analyse von kryptographischen Protokollen mit BAN-Logik; Datenschutz und Datensicherheit DuD 23/2 (1999) 90-93.

Nied_87 Arnold Niedermaier: Bewertung von Zuverlässigkeit und Senderanonymität einer fehlertoleranten Kommunikationsstruktur; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, September 1987.

NRC_99 National Research Council: Trust in Cyberspace; National Academy Press, 1999; http://books.nap.edu/html/trust/index.htm

OkOh_89 Tatsuaki Okamoto, Kazuo Ohta: Divertible Zero Knowledge Interactive Proofs and Commutative Random Self-Reducibility; Eurocrypt '89; Houthalen, 10.-13. April 1989, A Workshop on the Theory and Application of Cryptographic Techniques, 95-108.

OkOh1_89 Tatsuaki Okamoto, Kazuo Ohta: Disposable Zero-knowledge Authentications and Their Applications to Untraceable Electronic Cash; Crypto '89, August 20-24 1989, Abstracts, 443-458.

Papa_86 Stavros Papastavridis: Upper and Lower Bounds for the Reliability of a Consecutive-k-out-of-n:F System; IEEE Transactions on Reliability 35/5 (1986) 607-610.

Paul_78 Wolfgang J. Paul: Komplexitätstheorie; Teubner Studienbücher Informatik, Band 39, B. G. Teubner Verlag, Stuttgart 1978.

Pede_96 Torben P. Pedersen: Electronic Payments of Small Amounts; Security Protocols 1996, LNCS 1189, Springer-Verlag, Berlin 1997, 59-68.

PfAß1_90 Andreas Pfitzmann, Ralf Aßmann: More Efficient Software Implementations of (Generalized) DES; Interner Bericht 18/90, Fakultät für Informatik, Universität Karlsruhe 1990.

PfAß_93 Andreas Pfitzmann, Ralf Aßmann: More Efficient Software Implementations of (Generalized) DES; Computers & Security 12/5 (1993) 477-500.

Pfi1_83 Andreas Pfitzmann: Ein dienstintegriertes digitales Vermittlungs-/Verteilnetz zur Erhöhung des Datenschutzes; Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 18/83, Dezember 1983.

Pfi1_85 Andreas Pfitzmann: How to implement ISDNs without user observability - Some remarks; Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 14/85.

Pfit_83 Andreas Pfitzmann: Ein Vermittlungs-/Verteilnetz zur Erhöhung des Datenschutzes in Bildschirmtext-ähnlichen Neuen Medien; GI '83 13. Jahrestagung der Gesellschaft für Informatik 3. bis 7. Oktober 1983, Universität Hamburg, IFB 73, Springer-Verlag Heidelberg, 411-418.

Pfit_84 Andreas Pfitzmann: A switched/broadcast ISDN to decrease user observability; 1984 International Zurich Seminar on Digital Communications, Applications of Source Coding, Channel Coding and Secrecy Coding, March 6-8, 1984, Zurich, Switzerland, Swiss Federal Institute of Technology, Proceedings IEEE Catalog Nr. 84CH1998-4, 183-190.

Pfit_85 Andreas Pfitzmann: Technischer Datenschutz in diensteintegrierenden Digitalnetzen - Problemanalyse, Lösungsansätze und eine angepaßte Systemstruktur; Proceedings der 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, München, Oktober 1985, herausgegeben von P.P.Spies, IFB 113, Springer-Verlag, Berlin 1985, 96-112.

Pfit_86 Andreas Pfitzmann: Die Infrastruktur der Informationsgesellschaft: Zwei getrennte Fernmeldenetze beibehalten oder ein wirklich datengeschütztes errichten?; Datenschutz und Datensicherung DuD /6 (1986) 353-359.

Pfit_89 Andreas Pfitzmann: Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz; Universität Karlsruhe, Fakultät für Informatik, Dissertation, Feb. 1989, IFB 234, Springer-Verlag, Heidelberg 1990.

Pfit_93 Andreas Pfitzmann: Technischer Datenschutz in öffentlichen Funknetzen; Datenschutz und Datensicherung DuD 17/8 (1993) 451-463.

Pfit8_96 Birgit Pfitzmann: Digital Signature Schemes — General Framework and Fail-Stop Signatures; LNCS 1100, Springer-Verlag, Berlin 1996.

Pfit_98 Andreas Pfitzmann: Sicherheitskonzepte im Internet und die Kryptokontroverse; in Reiner Hamm, Klaus Peter Möller (Hrsg.), Datenschutz durch Kryptographie – ein Sicherheitsrisiko? Nomos Verlagsgesellschaft Baden-Baden 1998, 51-85.

PfPf_90 Birgit Pfitzmann, Andreas Pfitzmann: How to Break the Direct RSA-Implementation of MIXes; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 373-381.

PfPf_92 Andreas Pfitzmann, Birgit Pfitzmann: Technical Aspects of Data Protection in Health Care Informatics; Advances in Medical Informatics, J. Noothoven van Goor and J. P. Christensen (Eds.), IOS Press, Amsterdam 1992, 368-386.

PfPW_88 Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Datenschutz garantierende offene Kommunikationsnetze; Informatik-Spektrum 11/3 (1988) 118-142.

PfPW1_89 Andreas Pfitzmann, Birgit Pfitzmann, Michael Waidner: Telefon-MIXe: Schutz der Vermittlungsdaten für zwei 64-kbit/s-Duplexkanäle über den (2•64 + 16)-kbit/s-Teilnehmeranschluß; Datenschutz und Datensicherung DuD /12 (1989) 605-622.

PfWa_86 Andreas Pfitzmann, Michael Waidner: Networks without user observability -- design options; Eurocrypt '85, LNCS 219, Springer-Verlag, Berlin 1986, 245-253; Überarbeitung in: Computers & Security 6/2 (1987) 158-166.

PfWa_91 Birgit Pfitzmann, Michael Waidner: Fail-stop-Signaturen und ihre Anwendung; Proc. Verläßliche Informationssysteme (VIS'91), März 1991, Darmstadt, Informatik-Fachberichte 271, Springer-Verlag, Heidelberg 1991, 289-301.

PfWa2_97 Birgit Pfitzmann, Michael Waidner: Strong Loss Tolerance of Electronic Coin Systems; ACM Transactions on Computer Systems 15/2 (1997) 194-213.

PoKl_78 G. J. Popek, C. S. Kline: Design Issues for Secure Computer Networks; Operating Systems, An Advanced Course, Edited by R. Bayer, R. M. Graham, G. Seegmüller; LNCS 60, 1978; Nachgedruckt als Springer Study Edition, 1979; Springer-Verlag, Heidelberg, 517-546.

PoSc_97 U. Pordesch, M.J. Schneider: Sichere Kommunikation in der Gesundheitsversorgung, in: Günter Müller, Andreas Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 68.

PPSW_95 Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Vertrauenswürdiger Entwurf portabler Benutzerendgeräte und Sicherheitsmodule; Proc. Verläßliche Informationssysteme (VIS'95), Vieweg 1995, 329-350.

PPSW_97 Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Trusting Mobile User Devices and Security Modules; Computer 30/2 (1997) 61-68.

PSWW_96 Andreas Pfitzmann, Alexander Schill, Guntram Wicke, Gritta Wolf, Jan Zöllner: 1. Zwischenbericht SSONET, http://mephisto.inf.tu-dresden.de/RESEARCH/ssonet/reports.html, 31.08.96, Seite 34.

PSWW_98 Andreas Pfitzmann, Alexander Schill, Andreas Westfeld, Guntram Wicke, Gritta Wolf, Jan Zöllner: A Java-based distributed plattform for multilateral security; IFIP/GI Working Conference "Trends in Electronic Commerce", June 3-5, 1998, Hamburg, LNCS 1402, Springer-Verlag, Heidelberg 1998, 52-64.

PWP_90 Birgit Pfitzmann, Michael Waidner, Andreas Pfitzmann: Rechtssicherheit trotz Anonymität in offenen digitalen Systemen; Datenschutz und Datensicherung DuD 14/5-6 (1990) 243-253, 305-315.

Rann_94 Kai Rannenberg: Recent Development in Information Technology Security Evaluation – The Need for Evaluation Criteria for multilateral Security; in Richard Sizer, Louise Yngström, Henrik Kaspersen und Simone Fischer-Hübner: Security and Control of Information Technology in Society; Proceedings of the IFIP TC9/WG 9.6 Working Conference August 12-17,1993, North-Holland, Amsterdam 1994, 113-128.

Rann_97 Kai Rannenberg: Kriterien und Zertifizierung Mehrseitiger IT-Sicherheit. Dissertation, Universität Freiburg, Wirtschaftswissenschaftliche Fakultät, 1997.

RaPM_96 Kai Rannenberg, Andreas Pfitzmann, Günter Müller: Sicherheit, insbesondere mehrseitige IT-Sicherheit; it+ti 38/4 (1996) 7-10; leicht überarbeitet in G. Müller, A. Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik; Addison-Wesley, Bonn 1997, 21-29.

RDFR_97 Martin Reichenbach, Herbert Damker, Hannes Federrath, Kai Rannenberg: Individual Management of Personal Reachability in Mobile Communication. in Louise Yngström, Jan Carlsen: Information Security in Research and Business; Proceedings of the IFIP TC11 13th International Information Security Conference (SEC '97): 14-16 May 1997, Copenhagen, Denmark; Chapman & Hall, London 1997, 163-174.

Rede_84 Helmut Redeker: Geschäftsabwicklung mit externen Rechnern im Bildschirmtextdienst; Neue Juristische Wochenschrift NJW /42 (1984) 2390-2394.

Rede_86 Helmut Redeker: Rechtliche Probleme von Mailbox-Systemen; GI - 16. Jahrestagung II, "Informatik-Anwendungen - Trends und Perspektiven"; Berlin, Oktober 1986, Proceedings herausgegeben von G. Hommel und S. Schindler, IFB 127, Springer-Verlag, Heidelberg 1986, 253-266.

Rei1_86 Peter O'Reilly: Burst and Fast-Packet Switching: Performance Comparisons; IEEE INFOCOM '86, Fifth Annual Conference "Computers and Communications Integration - Design, Analysis, Management", April 8-10, 1986, Miami, Florida, 653-666.

ReRu_98 Michael K. Reiter, Aviel D. Rubin: Crowds: Anonymity for Web Transactions; ACM Transactions on Information and System Security 1/1 (1998) 66-92.

ReRu_99 Michael K. Reiter, Aviel D. Rubin: Anonymous Web Transactions with Crowds; Communications of the ACM 42/2 (1999) 32-38.

RiDe_99 Andreas Rieke, Thomas Demuth: Janus: Server-Anonymität im World Wide Web; in: Patrick Horster (Hrsg.): Sicherheitsinfrastrukturen; DuD-Fachbeiträge, Vieweg, Wiesbaden 1999, 297-307.

Riel_99 Herman te Riele: Factorization of a 512-bits RSA key using the Number Field Sieve; e-mail of Aug. 26, 1999.

Riha_84 Karl Rihaczek: Datenverschlüsselung in Kommunikationssystemen - Möglichkeiten und Bedürfnisse; DuD-Fachbeiträge 6, Friedr. Vieweg & Sohn, Braunschweig 1984.

Riha_85 Karl Rihaczek: Der Stand von OSIS; Datenschutz und Datensicherung DuD /4 (1985) 213-217.

Riha2_96 Karl Rihaczek: Die Kryptokontroverse: das Normungsverbot; Datenschutz und Datensicherheit DuD 20/1 (1996) 15-22.

Rive_98 Ronald L. Rivest: Chaffing and Winnowing: Confidentiality without Encryption; MIT Lab for Computer Science, March 22, 1998; http://theory.lcs.mit.edu/~rivest/chaffing.txt

Roch_87 Edouard Y. Rocher: Information Outlet, ULAN versus ISDN; IEEE Communications Magazine 25/4 (1987) 18-32.

RoSc_96 A. Roßnagel, M. J. Schneider: Anforderungen an die mehrseitige Sicherheit in der Gesundheitsversorgung und ihre Erhebung, Informationstechnik und technische Informatik it+ti 1996, 15.

Rose_85 K. H. Rosenbrock: ISDN - Die folgerichtige Weiterentwicklung des digitalisierten Fernsprechnetzes für das künftige Dienstleistungsangebot der Deutschen Bundespost; GI/NTG-Fachtagung "Kommunikation in Verteilten Systemen - Anwendungen, Betrieb und Grundlagen -", 11.-15. März 1985, Tagungsband 1, D. Heger, G. Krüger, O. Spaniol, W. Zorn (Hrsg.), IFB 95, Springer-Verlag Heidelberg, 202-221.

RSA_78 R. L. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.

Rul1_87 Christoph Ruland: Datenschutz in Kommunikationssystemen; DATACOM Buchverlag, Pulheim, 1987.

RWHP_89 Alexander Roßnagel, Peter Wedde, Volker Hammer, Ulrich Pordesch: Die Verletzlichkeit der "Informationsgesellschaft"; Sozialverträgliche Technikgestaltung Band 5, Westdeutscher Verlag, Opladen 1989.

Schn_96 Bruce Schneier: Applied Cryptography: Protocols, Algorithms, and Source Code in C; 2nd Ed., John Wiley & Sons, New York 1996.

Schö_84 Helmut Schön: Die Deutsche Bundespost auf ihrem Weg zum ISDN; Zeitschrift für das Post- und Fernmeldewesen /6 1984.

Schö_86 Helmut Schön: ISDN und Ökonomie; Jahrbuch der Deutschen Bundespost 1986.

ScS1_84 Christian Schwarz-Schilling (ed.): ISDN - die Antwort der Deutschen Bundespost auf die Anforderungen der Telekommunikation von morgen; Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Bonn, 1984.

ScS1_86 Christian Schwarz-Schilling (ed.): Chance und Herausforderung der Telekommunikation in den 90er Jahren; Heft 4 aus der Schriftenreihe über Konzepte und neue Dienste der Telekommunikation des Bundesministers für das Post- und Fernmeldewesen, Bonn, 1986.

ScSc_73 A. Scholz, B. Schoeneberg: Einführung in die Zahlentheorie; (5. Aufl.) Sammlung Göschen, de Gruyter, Berlin 1973.

ScSc_83 Richard D. Schlichting, Fred B. Schneider: Fail-Stop Processors: An Approach to Designing Fault-Tolerant Computing Systems; ACM Transactions on Computer Systems 1/3 (1983) 222-238.

ScSc_84 Christian Schwarz-Schilling (ed.): Konzept der Deutschen Bundespost zur Weiterentwicklung der Fernmeldeinfrastruktur; Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Stab 202, Bonn, 1984.

ScSc_86 Christian Schwarz-Schilling (ed.): Mittelfristiges Programm für den Ausbau der technischen Kommunikationssysteme; Herausgeber: Der Bundesminister für das Post- und Fernmeldewesen, Bonn, 1986.

Sedl_88 H. Sedlak: The RSA cryptography processor; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 95-105.

SFJK_97 Reiner Sailer, Hannes Federrath, Anja Jerichow, Dogan Kesdogan, Andreas Pfitzmann: Allokation von Sicherheitsfunktionen in Telekommunikationsnetzen; in: G. Müller, A. Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 325-357.

Sha1_49 C. E. Shannon: Communication Theory of Secrecy Systems; The Bell System Technical Journal 28/4 (1949) 656-715.

Sham_79 Adi Shamir: How to Share a Secret; Communications of the ACM 22/11 (1979) 612-613.

Shan_48 C. E. Shannon: A Mathematical Theory of Communication; The Bell System Technical Journal 27 (1948) 379-423, 623-656.

SIEM_87 SIEMENS: Internationale Fernsprechstatistik 1987; Stand 1. Januar 1986; SIEMENS Aktiengesellschaft N ÖV Marketing, Postfach 70 00 73, D-8000 München 70, Mai 1987.

SIEM_88 SIEMENS: Vermittlungsrechner CP 113 für ISDN; SIEMENS telcom report 11/2 (1988) 80.

Silv_91 Robert D. Silverman: Massively Distributed Computing and Factoring Large Integers; Communications of the ACM 34/11 (1991) 95-103.

Sim3_88 Gustavus J. Simmons: A Survey of Information Authentication; Proceedings of the IEEE 76/5 (1988) 603-620.

Simm_88 G. J. Simmons: Message authentication with arbitration of transmitter/receiver disputes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 151-165.

SmRe_90 Gene Smarte, Andrew Reinhardt: 1975-1990, 15 Years of Bits, Bytes, and other Great Moments; Byte 15/9 (1990) 369-400.

Steg_85 H. Stegmeier: Einfluß der VLSI auf Kommunikationssysteme; GI/NTG-Fachtagung "Kommunikation in Verteilten Systemen - Anwendungen, Betrieb und Grundlagen -", 11.-15. März 1985, Tagungsband 1, D. Heger, G. Krüger, O. Spaniol, W. Zorn (Hrsg.), IFB 95, Springer-Verlag Heidelberg, 663-672.

Stel_90 D. Stelzer: Kritik des Sicherheitsbegriffs im IT-Sicherheitsrahmenkonzept; Datenschutz und Datensicherung DuD14/10 (1990) 501-506.

SuSY_84 Tatsuya Suda, Mischa Schwartz, Yechiam Yemini: Protocol Architecture of a Tree Network with Collision Avoidance Switches; Links for the Future; Science, Systems & Services for Communications, P. Dewilde and C. A. May (eds.); Proceedings of the International Conference on Communications - ICC '84, Amsterdam, The Netherlands, May 14-17, 1984, IEEE, Elsevier Science Publishers B. V. (North-Holland), 423-427.

Sze_85 Daniel T. W. Sze: A Metropolitan Area Network; IEEE Journal on Selected Areas in Communications 3/6 (1985) 815-824.

Tane_81 Andrew S. Tanenbaum: Computer Networks; Prentice-Hall, Englewood Cliffs, N. J., 1981.

Tane_88 Andrew S. Tanenbaum: Computer Networks; 2nd ed., Prentice-Hall, Englewood Cliffs 1988.

Tane_96 Andrew S. Tanenbaum: Computer Networks; 3rd ed., Prentice-Hall, New Jersey 1996.

Tele_98 Telemediarecht – Telekommunikations- und Multimediarecht; Beck-Texte im dtv, 1998.

Tele_00 Deutsche Telekom: Das kleine Lexikon (Beilage zu „Der Katalog. Telekommunikation für zu Hause und unterwegs.“; Frühjahr/Sommer 2000, 8.

ThFe_95 Jürgen Thees, Hannes Federrath: Methoden zum Schutz von Verkehrsdaten in Funknetzen; Proc. Verläßliche Informationssysteme (VIS'95), Vieweg 1995, 181-192.

Thom_84 Ken Thompson: Reflections on Trusting Trust; Communications of the ACM 27/8 (1984) 761-763.

Thom_87 Karl Thomas: Der Weg zur offenen Kommunikation; nachrichten elektronik+telematik net special "ISDN – eine Idee wird Realität"; Sondernummer Oktober 87, R. v. Decker's Verlag, 10-17.

ToWo_88 Martin Tompa, Heather Woll: How to Share a Secret with Cheaters; Journal of Cryptology 1/2 (1988) 133-138.

VaVa_85 Umesh V. Vazirani, Vijay V. Vazirani: Efficient and Secure Pseudo-Random Number Generation (extended abstract); Crypto '84, LNCS 196, Springer-Verlag, Berlin 1985, 193-202.

VoKe_83 Victor L. Voydock, Stephen T. Kent: Security Mechanisms in High-Level Network Protocols; ACM Computing Surveys 15/2 (1983) 135-171.

VoKe_85 Victor L. Voydock, Stephen T. Kent: Security in High-level Network Protocols; IEEE Communications Magazine 23/7 (1985) 12-24.

Waer_67 B.L. van der Waerden: Algebra II; Heidelberger Taschenbücher 23, Springer-Verlag, Berlin 1967, 5. Auflage.

Waer_71 B.L. van der Waerden: Algebra I; Heidelberger Taschenbücher 12, Springer-Verlag, Berlin 1971, 8. Auflage.

Waid_85 Michael Waidner: Datenschutz und Betrugssicherheit garantierende Kommunikationsnetze. Systematisierung der Datenschutzmaßnahmen und Ansätze zur Verifikation der Betrugssicherheit; Diplomarbeit am Institut für Informatik IV, Universität Karlsruhe, August 1985, Interner Bericht 19/85 der Fakultät für Informatik.

Waid_90 Michael Waidner: Unconditional Sender and Recipient Untraceability in spite of Active Attacks; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 302-319.

Walk_87 Bernhard Walke: Über Organisation und Leistungskenngrößen eines dezentral organisierten Funksystems; Kommunikation in Verteilten Systemen; Anwendungen, Betrieb, Grundlagen; GI/NTG-Fachtagung, Aachen, Februar 1987, IFB 130, herausgegeben von N. Gerner und O. Spaniol, Springer-Verlag, Heidelberg, 578-591.

WaP1_87 Michael Waidner, Birgit Pfitzmann: Anonyme und verlusttolerante elektronische Brieftaschen; Interner Bericht 1/87 der Fakultät für Informatik, Universität Karlsruhe 1987.

WaPf_85 Michael Waidner, Andreas Pfitzmann: Betrugssicherheit trotz Anonymität. Abrechnung und Geldtransfer in Netzen; 1. GI Fachtagung Datenschutz und Datensicherung im Wandel der Informationstechnologien, IFB 113, Springer-Verlag, Berlin 1985, 128-141; Überarbeitung in: Datenschutz und Datensicherung DuD /1 (1986) 16-22.

WaPf_87 Michael Waidner, Birgit Pfitzmann: Verlusttolerante elektronische Brieftaschen; 3rd International Conference on Fault-Tolerant Computing-Systems, IFB 147, Springer-Verlag, Berlin 1987, 36-50; Überarbeitung in: Datenschutz und Datensicherung DuD /10 (1987) 487-497.

WaPf_89 Michael Waidner, Birgit Pfitzmann: Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks; Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 5/89, März 1989.

WaPf_90 Michael Waidner, Birgit Pfitzmann: Loss-Tolerance for Electronic Wallets; Proceedings 20th International Symposium on Fault-Tolerant Computing (FTCS 20), Newcastle upon Tyne (UK), 140-147.

WaPf1_89 Michael Waidner, Birgit Pfitzmann: The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability; Universität Karlsruhe 1989.

WaPP_87 Michael Waidner, Birgit Pfitzmann, Andreas Pfitzmann: Über die Notwendigkeit genormter kryptographischer Verfahren; Datenschutz und Datensicherung DuD /6 (1987) 293-299.

WeCa_79 Mark N. Wegman, J. Lawrence Carter: New Classes and Applications of Hash Functions; 20th Symposium on Foundations of Computer Science (FOCS) 1979, IEEE Computer Society, 1979, 175-182.

WeCa_81 Mark N. Wegman, J. Lawrence Carter: New Hash Functions and Their Use in Authentication and Set Equality; Journal of Computer and System Sciences 22 (1981) 265-279.

Weck_98 Gerhard Weck: Key Recovery – Möglichkeiten und Risiken; Informatik-Spektrum 21/3 (1998) 147-157.

Weck1_98 Gerhard Weck: Key Recovery – Empfehlungen; Informatik-Spektrum 21/3 (1998) 157-158.

Wein_87 Steve H. Weingart: Physical Security for the microABYSS System; Proc. 1987 IEEE Symp. on Security and Privacy, April 27-29, 1987, Oakland, California, 52-58.

West_97 Andreas Westfeld: Steganographie am Beispiel einer Videokonferenz; in: G. Müller, A. Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik, Addison-Wesley-Longman 1997, 507-525.

Wich_93 Peer Wichmann: Die DSSA Sicherheitsarchitektur für verteilte Systeme; Datenschutz und Datensicherung DuD 17/1 (1993) 23-27.

Wien_90 Michael J. Wiener: Cryptanalysis of Short RSA Secret Exponents; IEEE Transaction on Information Theory 36/3 (1990) 553-558.

WiHP_97 Guntram Wicke, Michaela Huhn, Andreas Pfitzmann, Peter Stahlknecht: Kryptoregulierung; Wirtschaftsinformatik 39/3 (1997) 279-282.

ZFPW_97 Jan Zöllner, Hannes Federrath, Andreas Pfitzmann, Andreas Westfeld, Guntram Wicke, Gritta Wolf: Über die Modellierung steganographischer Systeme; GI-Fachtagung Verläßliche IT-Systeme (VIS'97), DuD-Fachbeiträge, Vieweg 1997, 211-223.

Zimm_93 Philip Zimmermann: PGP – Pretty Good Privacy, Public Key Encryption for the Masses, User's Guide; Volume I: Essential Topics, Volume II: Special Topics; Version 2.2 - 6 March 1993.

ZSI_89 Zentralstelle für Sicherheit in der Informationstechnik (Hrsg.): IT-Sicherheitskriterien; Kriterien für die Bewertung der Sicherheit von Systemen der Informationstechnik (IT); 1. Fassung vom 11.1.1989; Köln, Bundesanzeiger 1989.

ZSI_90 Zentralstelle für Sicherheit in der Informationstechnik (Hrsg.): IT-Evaluationshandbuch; Handbuch für die Prüfung der Sicherheit von Systemen der Informationstechnik (IT); 1. Fassung vom 22.2.1990; Köln, Bundesanzeiger 1990.

Aufgaben

Aufgaben zur Einführung

1-1 Zusammenhang zwischen räumlicher Verteilung und Verteilung bzgl. Kontroll- und Implementierungsstruktur

Warum sind räumlich verteilte informationstechnische Systeme (IT-Systeme) praktisch ausschließlich auch bzgl. ihrer Kontroll- und Implementierungsstruktur verteilt realisiert?

(Zur Erläuterung: *Kontrollstruktur* = Welche Instanz(en) erteilen welchen anderen Instanzen Aufträge, Handlungsanweisungen, etc.; überzeugen sich, ob und ggf. wie sie ausgeführt wurden. *Implementierungsstruktur* = Wie sind die Instanzen implementiert, z.B. viele auf einem Rechner, z.B. einzelne Instanzen auf mehreren kooperierenden Rechnern, usw.)

1-2 Vor- und Nachteile eines *verteilten* Systems bzgl. Sicherheit

a) Welche Vor- und Nachteile bzgl. Sicherheit bietet ein verteiltes System?
b) Welche Sicherheitseigenschaften werden durch räumliche Verteilung, welche eher durch Verteilung bzgl. Kontroll- und Implementierungsstruktur unterstützt?

1-3 Vor- und Nachteile eines *offenen* Systems bzgl. Sicherheit

a) Welche Vor- und Nachteile bzgl. Sicherheit bietet ein offenes System?
b) Wie kombinieren sich die Vor- und Nachteile bzgl. Sicherheit bei verteilten offenen Systemen?

1-4 Vor- und Nachteile eines *diensteintegrierenden* Systems bzgl. Sicherheit

a) Welche Vor- und Nachteile bzgl. Sicherheit bietet ein diensteintegrierendes System?
b) Wie kombinieren sich die Vor- und Nachteile bzgl. Sicherheit bei verteilten offenen diensteintegrierenden Systemen?

1-5 Vor- und Nachteile eines *digitalen* Systems bzgl. Sicherheit

a) Welche Vor- und Nachteile bzgl. Sicherheit bietet ein digitales System? Unterscheiden Sie zwischen digitaler Übertragung/Speicherung einerseits und rechnergesteuerter Vermittlung, rechnergestützter Verarbeitung andererseits. Was ändert sich, wenn Rechner frei programmiert werden können (Programm im RAM statt im ROM)?
b) Welche Zusammenhänge bestehen zwischen digitalen Systemen und den Eigenschaften verteilt, offen und diensteintegrierend?

1-6 Anforderungsdefinition für vier beispielhafte Anwendungen im medizinischen Bereich

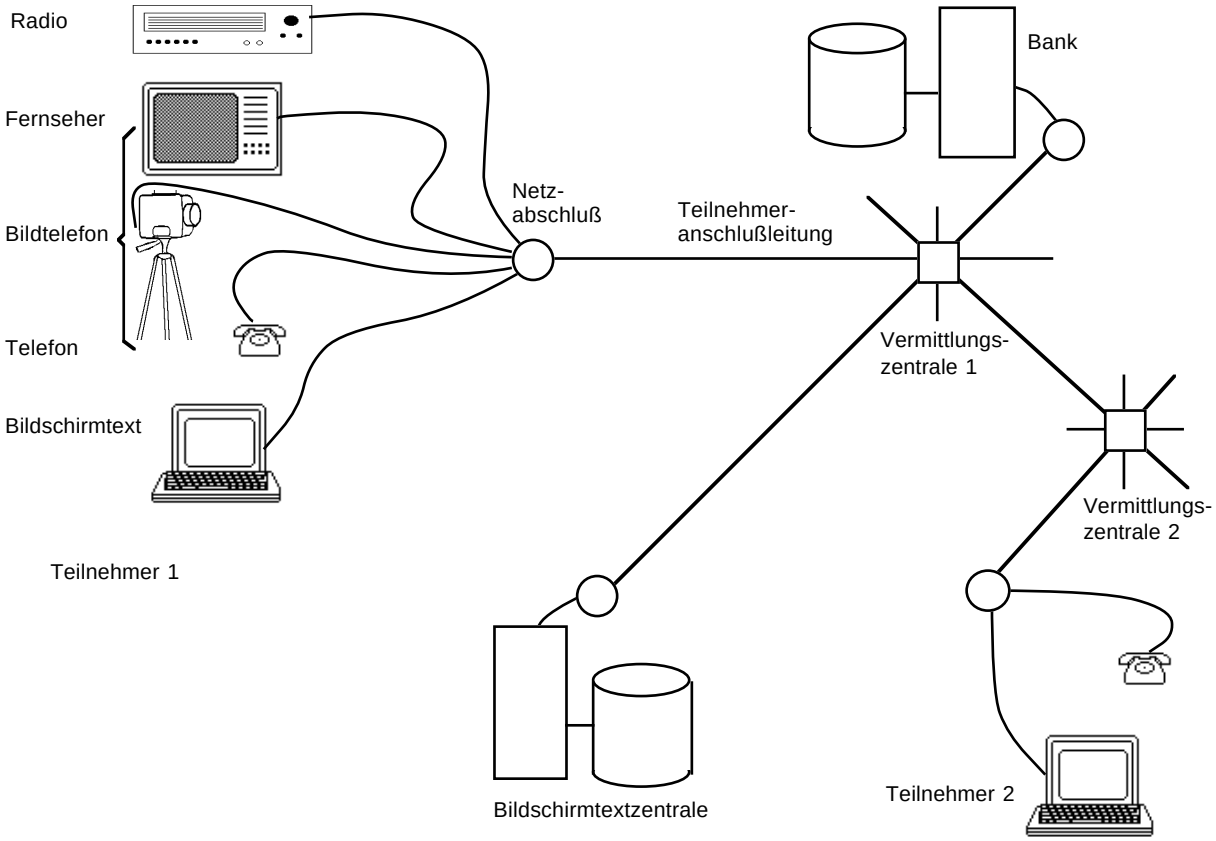
Erstellen Sie möglichst vollständige Schutzziele für folgende Anwendungen

a) Krankengeschichten vieler Patienten werden in einer Datenbank gespeichert und von behandelnden Ärzten weltweit zugegriffen.
b) Während medizinischen Operationen werden per Bildtelefon externe Experten konsultiert, um ihren Rat einzuholen. Was entstehen für zusätzliche Anforderungen, falls das Operationsteam vor Ort auf den Rat angewiesen ist?
c) Patienten sollen in ihrer Privatwohnung medizinisch überwacht werden, damit sie einerseits weniger lange im Krankenhaus verweilen müssen, andererseits etwa bei Bewußtlosigkeit schnell und automatisch Hilfe gerufen wird.
d) Medizinische und psychologische Faktendatenbanken werden von Patienten abgefragt, um bequemer, ausführlicher und aktueller als durch Bücher möglich informiert zu werden und den Ärzten Zeit zu sparen (die sie sich häufig sowieso nicht nehmen würden).

Sind Sie sich der Vollständigkeit Ihrer Schutzziele sicher?

1-7 Wo sind ab wann Rechner zu erwarten?

Kennzeichnen Sie in Bild 1-1 durch Eintrag der Zahl *n* die Stellen, an denen Sie ab dem Jahr *n* einen Rechner erwarten. Kennzeichnen Sie durch Anhängen von /<Rollenbezeichnung>, wer diesen Rechner progammiert hat und programmieren kann. Was bedeutet dies für die Sicherheit?



1-8 Defizite juristischer Regelungen

Warum reichen juristische Regelungen z.B. für Rechtssicherheit und Datenschutz nicht aus?

1-9 Alternative Definitionen/Charakterisierungen mehrseitiger Sicherheit

In §1.4 wird eine Definition/Charakterisierung mehrseitiger Sicherheit gegeben. Da es sich hierbei eher um eine umgangssprachliche Charakterisierung als eine strenge Definition eines mathematisch/natur-/ingenieurwissenschaftlichen Sachverhalts handelt: Denken Sie sich mindestens 3 alternative Definitionen/Charakterisierungen aus oder sammeln Sie solche aus der Literatur. Sie können fündig werden in: [Cha8_85, Chau_92, PWP_90, MüPf_97, FePf_99].

Aufgaben zu Sicherheit in einzelnen Rechnern und ihre Grenzen

2-1 Angreifer außerhalb der Weisheit der Schulbücher

Denken Sie sich Angreifer aus, die nicht, wie in §2.1.1 gefordert, den derzeit bekannten physischen Naturgesetzen unterliegen und vor denen Sie Sicherheit prinzipiell nicht erreichen können.

2-2 (Un)Abhängigkeit von Vertraulichkeit, Integrität, Verfügbarkeit am Beispiel Identifikation

In §1.2.1 wurde zwischen den Schutzzielen Vertraulichkeit, Integrität und Verfügbarkeit unterschieden. Diskutieren Sie anhand des Beispiels Identifikation (§2.2.1), ob „sichere“ IT-Systeme zu erwarten sind, für die zumindest bzgl. eines der drei Schutzziele keinerlei Anforderungen bestehen.

2-3 Sende Zufallszahl, erwarte Verschlüsselung – geht's auch umgekehrt?

In §2.2.1 wird als kleines Protokollbeispiel zur Identifikation angegeben:

Generiere Zufallszahl, sende sie;

Erwarte Verschlüsselung der Zufallszahl. Prüfe.

Kann dieses Protokoll auch umgekehrt ausgeführt werden:

Generiere Zufallszahl, verschlüssele sie, sende verschlüsselte Zufallszahl;
Erwarte entschlüsselte Zufallszahl. Prüfe.

Was ändert sich durch diese Umkehrung? (Als Hilfe: Welche Annahmen über die Zufallszahlenerzeugung müssen Sie jeweils machen?)

2-4 Notwendigkeit der Protokollierung bei Zugriffskontrolle

a) Protokollierungsdaten (Sekundärdaten) sind genauso personenbezogene Daten und damit ggf. vor unbefugter Kenntnisnahme zu schützen wie die Primärdaten. Die Katze beißt sich also in den Schwanz, denn nun müßten wir wieder den Zugriff auf die Sekundärdaten protokollieren (Tertiärdaten), usw. Warum ist das Problem trotz Rekursivität befriedigend gelöst?
b) Was sollte man anstreben?
c) Bzgl. der Autorisierung gilt Ähnliches wie bei a) für die Protokollierungsdaten geschildert: Um zu regeln, wer Rechte vergeben darf (Autorisierung), müssen selbst wieder Regeln vergeben (und durchgesetzt) werden, wer dies darf, usw. Auch hier schnappt die Katze verdächtig nach ihrem eigenen Schwanz. Gilt hier auch bzgl. der Lösung Ähnliches wie bei a)?

2-5 Begrenzung des Erfolges verändernder Angriffe

In §1.2.5 wurde erwähnt, daß verändernde Angriffe zwar prinzipiell erkannt, ihr Erfolg aber nicht vollständig verhindert werden kann. Welche Maßnahmen müssen ergriffen werden, um verändernde Angriffe möglichst wirkungslos zu machen? Überlegen Sie insbesondere, wie Sie Backups für Daten und Programme organisieren würden.

2-6 Wirklich kein verborgener Kanal mehr?

Einige um die Geheimhaltung ihrer Pläne zur Landesverteidigung extrem besorgte Militärs beschließen nach Lektüre der ersten zwei Kapitel des Skripts: Nie und nimmer werden wir in Zukunft darauf vertrauen, daß gelieferte Geräte keine Trojanischen Pferde enthalten, die unsere Geheimnisse schon vor dem Verteidigungsfall nach außen geben wollen. Deshalb werden wir diese Geräte in hermetisch geschirmte Räume stellen und alle Kanäle nach außen schließen, indem Leitungen nach außen in Friedenszeiten durch physische Schalter unterbrochen sind (Begründung klar!), der Stromverbrauch unseres Rechenzentrums konstant ist (damit nicht darüber Information nach außen fließen kann) und natürlich geben wir auch unsere Geräte nicht zur Reparatur an den Hersteller (denn auch in ihnen könnte Information gespeichert sein). Sie melden dem Verteidigungsminister: Endlich entkommt unsern Rechnern in Friedenszeiten kein Bit mehr! Haben sie Recht?

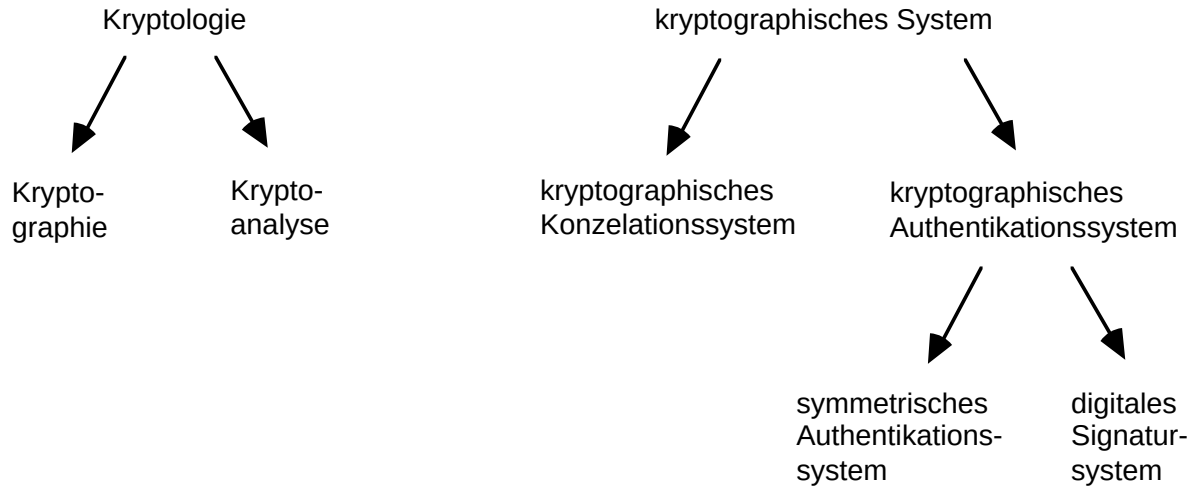
2-7 Perspektivwechsel: Das diabolische Endgerät

a) Die Sicherheit jedes Beteiligten kann nicht größer sein, als sein Endgerät, d.h. das Gerät, mit dem er direkt interagiert, für ihn sicher ist. Denken Sie sich als Advocatus Diaboli ein Endgerät mit möglichst vielen Eigenschaften aus, die die Sicherheit seines Benutzers untergraben, das er aber trotzdem benutzen mag.
b) Vergleichen Sie ihren Vorschlag von a) mit den PCs, die Sie im Frühjahr 1999 angeboten bekommen.

Aufgaben zu Kryptologische Grundlagen

3-0 Begriffe

Im nachfolgenden Bild sind Begriffsbäume dargestellt. Erläutern Sie den inhaltlichen Aufbau der Begriffsbäume und die Vor- und Nachteile, die kurzen Begriffe *Kryptosystem* bzw. *Verschlüsseln* und *Entschlüsseln* allgemein für kryptographische Systeme zu verwenden oder spezieller nur für kryptographische Konzelationssysteme.



3-1 Vertrauensbereiche für asymmetrisches Konzelationssystem und digitales Signatursystem

In Bild 3-1 sind die Vertrauens- und Angriffsbereiche so eingezeichnet, daß über die Schlüsselgenerierung keine Aussage gemacht wird. Klar ist, daß auch sie in einem Vertrauensbereich stattfinden muß. Ob in dem des Verschlüsselers, des Entschlüsselers oder in einem dritten bleibt offen und sollte je nach Anwendung auch verschieden sein:

Bei Verschlüsselung eines lokalen Speichermediums sollte Schlüsselgenerierung, Verschlüsselung und Entschlüsselung etwa im selben Gerät stattfinden – dann muß der Schlüssel dies Gerät nie verlassen, kann also bzgl. Vertraulichkeit und Integrität optimal geschützt werden. (Es sei daran erinnert, daß Verschlüsselung nichts zur Verfügbarkeit beiträgt und daß ggf. von den auf dem Medium gespeicherten Daten Backup-Kopien des Klartextes oder sowohl von Schlüsseltext als auch von Entschlüsselungsschlüssel gemacht werden müssen.)

Bei direktem Schlüsselaustausch zwischen Verschlüsseler und Entschlüsseler außerhalb des betrachteten Kommunikationsnetzes, etwa via Diskette, sollte der Schlüssel in einem der beiden Vertrauensbereiche generiert werden.

Bei Schlüsselaustausch innerhalb des Kommunikationsnetzes gemäß Bild 3-3 kann die Schlüsselgenerierung der k_i durch den Verschlüsseler, den Entschlüsseler oder die Schlüsselverteilzentrale erfolgen. Es ist völlig gleich, in welchem der drei Vertrauensbereiche dies geschieht, denn während des Ablaufs der Schlüsselverteilung liegt der Schlüssel sowieso in jedem Vertrauensbereich unverschlüsselt vor.

Alles bisher Gesagte gilt natürlich nicht nur für symmetrische Konzelation, sondern auch für symmetrische Authentikation.

Und nun zur Frage: Wie würden Sie die Vertrauensbereiche bei asymmetrischen Konzelationssystemen und digitalen Signatursystemen einzeichnen? Tun Sie es – und begründen Sie bitte Ihre Entscheidung.

3-2 Weshalb Schlüssel?

In §3.1.1 wird unterstellt, daß kryptographische Systeme Schlüssel benutzen.

a) Ginge es auch ohne Schlüssel? Ggf. wie?
b) Was wären die Nachteile?
c) Was sind aus Ihrer Sicht also die Hauptvorteile der Benutzung von Schlüsseln?

3-2a Schlüsselabgleich bei teilnehmerautonomer Schlüsselgenerierung?

In Aufgabe 3-1 wird erläutert, warum die Schlüsselgenerierung bei asymmetrischen kryptographischen Systemen in dem Gerät stattfinden sollte, in dem der geheime Teil des Schlüsselpaares verwendet wird.

Hin und wieder wird hiergegen eingewendet: Da die erzeugten Schlüsselpaare eindeutig sein müssen (offensichtlich besonders wichtig bei digitalen Signatursystemen), können sie nicht teilnehmerautonom dezentral erzeugt werden, denn dabei lassen sich Schlüsselpaar-Doubletten (zwei Teilnehmer erzeugen unabhängig voneinander das gleiche Schlüsselpaar) nicht völlig ausschließen. Um dies zu erreichen, sollen sogenannte TrustCenter – technisch-organisatorische Einheiten, die verschiedene vertrauenswürdige, der Sicherheit dienliche Dienstleistungen erbringen sollen –, auch die Funktion der (koordinierten!) Schlüsselgenerierung wahrnehmen.

Was halten Sie von diesem Einwand und Vorschlag?

3-3 Steigerung der Sicherheit durch Gebrauch von mehreren kryptographischen Systemen

Was können Sie tun, wenn Sie mehrere kryptographische Systeme haben, die auf unterschiedlichen Sicherheitsannahmen beruhen, Sie aber keinem der kryptographischen Systeme (bzw. keiner der Sicherheitsannahmen) allein vertrauen wollen? Unterscheiden Sie die Schutzziele Konzelation (d.h. Sie haben mehrere Konzelationssysteme) und Authentikation (d.h. Sie haben mehrere Authentikationssysteme). Welchen Berechnungs- und Speicher- bzw. Übertragungsaufwand verursacht die Maßnahme? (Letzteres bei Konzelation nur ungefähr.)

3-4 Schlüsselaustauschprotokolle

a) Alle beschriebenen Schlüsselaustauschprotokolle gehen davon aus, daß die zwei Teilnehmer, die einen Schlüssel austauschen wollen, Schlüssel mit einer *gemeinsamen* Schlüsselverteilzentrale bzw. einem *gemeinsamen* öffentlichen Schlüsselregister ausgetauscht haben – bei Schlüsselverteilzentralen Schlüssel eines symmetrischen Kryptosystems und bei öffentlichen Schlüsselregister Schlüssel eines asymmetrischen Kryptosystems.
   Was ist zu ändern, falls dies nicht der Fall ist? Beispielsweise könnten die zwei Teilnehmer in unterschiedlichen Ländern wohnen und Schlüssel jeweils nur mit Schlüsselverteilzentralen bzw. öffentlichen Schlüsselregistern ihres Landes ausgetauscht haben.

b) *Optional:* Wie sollte Ihre Lösung für den Austausch symmetrischer Schlüssel gestaltet werden, wenn die Teilnehmer in ihrem Land jeweils mehrere Schlüsselverteilzentralen verwenden wollen, um den Schlüsselverteilzentralen jeweils möglichst wenig vertrauen zu müssen?

c) Bei symmetrischen kryptographischen Systemen wurde die Parallelschaltung von Schlüsselverteilzentralen vorgeführt und begründet. Ist eine Parallelschaltung auch bei asymmetrischen kryptographischen Systemen sinnvoll? Wie sollte sie erfolgen? Was leistet sie?

d) *Optional:* Gibt es ein Schlüsselaustauschprotokoll, das trotz folgenden Angreifermodells sicher ist: Entweder erhält der Angreifer eine passive Mithilfe aller Schlüsselverteilzentralen, d.h. sie verraten ihm alle ausgetauschten Schlüssel, oder (exklusiv!) der Angreifer ist komplexitätstheoretisch unbeschränkt, d.h. er kann jedes asymmetrische Konzelationssystem (und auch jedes normale digitale Signatursystem) brechen. Zusätzlich ist natürlich unterstellt, daß der Angreifer alle Kommunikation im Netz abhört. Wie sieht ein Schlüsselaustauschprotokoll ggf. aus? Welches kryptographische System muß man nach dem Schlüsselaustausch ggf. zum Zwecke der Konzelation, welches zum Zwecke der Authentikation verwenden?

e) Was passiert, wenn ein Angreifer beim Austausch symmetrischer Schlüssel verändernd angreift? Dies könnte geschehen, indem er Nachrichten auf den Leitungen verändert oder indem eine Schlüsselverteilzentrale inkonsistente symmetrische Schlüssel verteilt.

Was sollte getan werden, um Teilnehmer hiergegen zu schützen? Behandeln sie zunächst den Fall einer Schlüsselverteilzentrale, danach den mehrerer Schlüsselverteilzentralen in Serie (beispielsweise bei mehrfach hierarchischer Schlüsselverteilung, vgl. Aufgabenteil a)) und danach den Fall mehrerer Schlüsselverteilzentralen parallel (vgl. §3.1.1.1).

f) Aktualität von Schlüsseln: Inwieweit und wie kann sichergestellt werden, daß ein Angreifer Teilnehmer nicht dazu bringen kann, alte (und beispielsweise kompromittierte), inzwischen durch neue ersetzte Schlüssel zu verwenden? (Unterstellt ist, daß es sich jeweils durchaus um Schlüssel des „richtigen" Teilnehmers handelt.)

g) Austausch öffentlicher Schlüssel ohne öffentliche Schlüsselregister: anarchischer Schlüsselaustausch à la PGP

Nehmen wir an, wir können keine öffentlichen Schlüsselregister benutzen (sei es, daß Staat oder Telekommunikationsanbieter keine betreiben, sei es, daß manche Bürger zentral-hierarchischen öffentlichen Schlüsselregistern nicht vertrauen wollen, sei es, daß es zwar öffentliche Schlüsselregister gibt, diese aber auch gleich die Schlüsselpaare erzeugen – ein Schuft, wer hinter dieser Fürsorge etwas Böses vermutet). Trotzdem (oder gerade deswegen) sollen in einer offenen Teilnehmergruppe öffentliche Schlüssel ausgetauscht werden.

Zur Wiederholung: Worauf muß beim Austausch der öffentlichen Schlüssel geachtet werden?

Auf wen könnte die Funktion des vertrauenswürdigen öffentlichen Schlüsselregisters verteilt werden?

Wie erfolgt dann der Schlüsselaustausch?

Was ist zu tun, wenn nicht jeder jedem gleich viel vertraut?

Was ist zu tun, wenn ein Teilnehmer entdeckt, daß sein geheimer Schlüssel anderen bekannt wurde?

Was sollte ein Teilnehmer tun, wenn sein geheimer Schlüssel zerstört wurde?

optional: Was ist zu tun, wenn ein Teilnehmer entdeckt, daß er eine falsche Zuordnung Teilnehmer <—> öffentlicher Schlüssel zertifiziert hat?

Kann man das in dieser Teilaufgabe entwickelte Verfahren mit dem aus Teilaufgabe b) kombinieren?

3-4a Welche Sorte von Trust ist für welche Funktion eines TrustCenters nötig?

Manche (wie wir gleich sehen werden: naive) Autoren weisen sogenannten TrustCentern folgende Funktionen zu:

Schlüsselgenerierung: Das Erzeugen von Schlüssel(paare)n für Teilnehmer.

Schlüsselzertifizierung: Das Zertifizieren des Zusammenhangs von öffentlichem Schlüssel und Teilnehmer, vgl. §3.1.1.2.

Verzeichnisdienst: Bereithalten der zertifizierten öffentlichen Schlüssel zur Abfrage für beliebige andere Teilnehmer.

Sperrdienst: Sperren von öffentlichen Schlüsseln, u.a. auf Wunsch des Schlüsselinhabers, d.h. Ausstellen einer signierten Erklärung, daß (und ab wann, s.u.) der öffentliche Schlüssel gesperrt ist.

Zeitstempeldienst: Digitales Signieren von vorgelegten Nachrichten inkl. aktuellem Datum und aktueller Zeit.

Diskutieren Sie, welche Sorten von Vertrauen der Teilnehmer dies jeweils erfordert. Sinnvoll könnte etwa die Unterscheidung nach **blindem Vertrauen** (Teilnehmer hat keinerlei Möglichkeit zu prüfen, ob sein Vertrauen gerechtfertigt ist) und **überprüfendem Vertrauen** (Teilnehmer kann überprüfen, ob sein Vertrauen gerechtfertigt ist) sein. Wie hängen diese Eigenschaften mit den in §1.2.5 beschriebenen Angreifermodellen, insbesondere der Unterscheidung in *beobachtende* und *verändernde Angriffe* zusammen?

3-5 Hybride Verschlüsselung

Sie wollen große Dateien austauschen und diesen Austausch – aus Effizienzgründen – mit einem hybriden kryptographischen System sichern. Wie verfahren Sie, wenn es Ihnen

a) nur auf Konzelation,
b) auch auf Authentikation,
c) nur auf Authentikation

ankommt?

Betrachten Sie zunächst nur den Fall 1, daß ein Sender *S* einem Empfänger *E* eine Datei zukommen läßt, danach den Fall 2, daß *S* zeitlich nacheinander mehrere Dateien an *E* schickt und schließlich den Fall 3, daß *E* antwortet, d.h. eine Datei an *S* schickt.

3-6 Situationen aus der Praxis (zur Systematik der kryptographischen Systeme)

Geben Sie zu Situationen an, was für kryptographische Systeme Sie einsetzen würden (Authentikation/Konzelation; symmetrisch/asymmetrisch) und wie Sie die Schlüsselverteilung vornehmen würden. Auch könnten Sie eine grobe Einschätzung angeben, wie sicherheits- und zeitkritisch das System ist. (Z.B. Vernam-Chiffre in Hardware: 10 Gbit/s, in Software 500 Mbit/s; informationstheoretisch sichere Authentikationscodes in Hardware 5 Gbit/s, in Software 100 Mbit/s; DES in Hardware: 200 Mbit/s, in Software bis 10 Mbit/s, je nach PC; RSA in Hardware: 200 kbit/s, in Software 6000 bit/s (GMR, s^2-mod-n-Generator vermutlich ähnlich).)

a) Andreas schaut sich in Hildesheim Wohnungen an. Er will Birgit e-mail mit den Vor- und Nachteilen der Wohnungen schicken und eine Entscheidung, ob er einen Mietvertrag unterschreiben soll, zurückerhalten. (Beide sind so heiser, daß sie nicht telefonieren können.)
b) David hat wieder eine geniale Idee für ein neues kryptographisches System und will sie ein paar vertrauenswürdigen Kryptographen zuschicken (File-Transfer). Leider lauern in den meisten Rechenzentren weniger geniale Kryptographen, die Davids Ideen klauen wollen.
c) Ein sicheres Betriebssystem schreibt Daten auf eine Wechselplatte heraus. Es soll beim Wiedereinlesen sicher sein, daß die Daten nicht verändert wurden.
d) Ein Rechnerfan möchte einen neuen Rechner anhand eines digitalen Katalogs mit einer digitalen Nachricht bestellen.
e) Eine Maschinenbaufirma faxt einer anderen eine unverbindliche Anfrage, ob sie ein ganz bestimmtes Teil fertigen könnte und zu welchem Preis.

3-7 Vernam-Chiffre

a) Rechnen Sie ein kleines Beispiel, wobei wir hierfür und für das Folgende vereinbaren, daß die bearbeiteten Bits bzw. Zeichen von links nach rechts geschrieben werden.
b) In der Vorlesung wurde die Sicherheit der Vernam-Chiffre bewiesen, wenn modulo 2 addiert wird. Führen Sie den entsprechenden Beweis für Addition in beliebigen Gruppen.
c) Entschlüsseln Sie den Schlüsseltext 01011101, der mit dem Schlüssel 10011011 verschlüsselt wurde. Ist das Ergebnis eindeutig?
d) Ein Angreifer fängt den Schlüsseltext 01011101 01011101 01011101 ab und möchte das dritte Bit von rechts des zugehörigen Klartextes invertieren. Was muß er tun, wenn modulo 2 gerechnet wird? Was muß er tun, wenn modulo 4, modulo 8 bzw. modulo 16 gerechnet wird, also jeweils 2, 3 bzw. 4 Bits „gemeinsam“ addiert werden? In welchem Sinne ist die Aufgabe für modulo 4 und 16 lösbar, in welchem nicht?
e) Warum gibt es bei der Vernam-Chiffre keinen adaptiven aktiven Angriff auf den *Schlüssel*?
f) Es gibt 16 Funktionen von {0, 1} x {0, 1} —> {0, 1}. Welche davon eignen sich als Verschlüsselungs- bzw. Entschlüsselungsfunktion für die binäre Vernam-Chiffre? Geben Sie zunächst allgemeine notwendige und hinreichende Bedingungen für die Funktionen beliebiger Vernam-Chiffren an, und wenden Sie diese danach auf den binären Fall an.

g) Ein jüngerer Kryptograph hat folgende Idee zur Senkung des Schlüsselaustauschaufwands: Statt einem zufälligen Schlüssel verwende ich einen Teil eines Buches, den auch mein Partner hat. Dann brauchen wir uns nur noch auf Buch, Seite und Zeile des Beginns zu einigen und können dann modulo Alphabetgröße addieren bzw. subtrahieren, so lange das Buch reicht. Welchen Rat und warum geben Sie als älterer Kryptograph Ihrem jüngeren Freund?

h) In manchen Ländern darf man zwar verschlüsselte Nachrichten schicken, muß aber den Schlüssel aufbewahren und auf Wunsch einer Strafverfolgungsbehörde o.ä. aushändigen. Hat das bei einer Vernam-Chiffre Sinn? (Unterstellt wird natürlich, daß die Strafverfolgungsbehörde alle Schlüsseltexte abhört, in richtiger Reihenfolge speichert und Sendern und Empfängern zuordnen kann.)

i) Wie lange kann man mit der informationstheoretisch sicheren Vernam-Chiffre

Textkommunikation (Tippgeschwindigkeit 40 bit/s),
Sprachkommunikation (mit der normalen ISDN-Codierung: 64 kbit/s) bzw.
Hochauflösende Bewegtbildkommunikation (140 Mbit/s)

verschlüsseln, wenn man eine

3,5"-Diskette (1,4 MByte),
optische 5,25"-Platte (600 MByte),
Video-8-Streamertape (5 GByte),
3,5"-Diskette (120 MByte),
DVD-RAM einseitig (2,6 GByte),
Dual-Layer-DVD beidseitig (17 GByte)

voller Zufallsbits ausgetauscht hat?

Zur Einordnung: Die ersten 2 Medien sind Standardtechnologie mindestens seit 1990 und massenhaft verbreitet. Das Video-8-Streamertape (5 GByte) ist seit etwa 1993 verbreitet, allerdings ist das Schreib-/Lesegerät nie ein Massenartikel geworden – es wurde aber laufend verbessert: 1998 passen auf das gleiche Medium mittels kompatibler Schreib-/Lesegeräte 20 GByte. 3,5"-Diskette (120 MByte) und DVD-RAM einseitig (2,6 GByte) sind 1998 dabei, den Massenmarkt zu erobern und Nachfolger der ersten 2 Medien zu werden. Dual-Layer-DVD beidseitig (17 GByte) ist bisher kein Produkt für die Speicherung von Daten durch den Endnutzer, sondern das, was mit der Technologie von 1998 auf eine DVD gepreßt werden kann. Es dient als Schätzung, was im Jahre 2000 dem Endnutzer zur Speicherung zur Verfügung steht.

3-7a Symmetrisch konzelierter Message Digest = MAC ?

a) Hin und wieder hört man folgendes Rezept zur Generierung von Message Authentication Codes (MACs):

Bilden Sie nach einem öffentlich bekannten Verfahren eine Prüfziffer zur Nachricht und verschlüsseln Sie die Prüfziffer mit einem symmetrischen Konzelationssystem.
Übertragen Sie die Nachricht mit der verschlüsselten Prüfziffer.
Der Empfänger entschlüsselt die Prüfziffer und prüft, ob sie paßt, indem er mit dem öffentlich bekannten Verfahren zur erhaltenen Nachricht die Prüfziffer berechnet. Sind beide Prüfziffern gleich, wird die Nachricht als authentisch betrachtet.

Was halten Sie von diesem Rezept?

(Als Tip: Nehmen sie das sicherstmögliche öffentliche Verfahren zur Bildung von Prüfziffern, nämlich eine kollisionsresistente Hashfunktion, und das sicherste symmetrische Konzelationssystem, die Vernam-Chiffre, und überlegen Sie dann, ob die Kombination sicher ist.)

b) Was halten Sie von folgender Verbesserung: Statt den Hashwert mittels des symmetrischen Schlüssels zu verschlüsseln, wird der symmetrische Schlüssel mit gehasht (so beispielsweise im Transport Layer Security (TLS) Protokoll). Falls Sie der Meinung sind, daß dies nicht für

alle kollisionsresistenten Hashfunktionen eine sichere Konstruktion ist: Welche zusätzliche Eigenschaft muß die kollisionsresistente Hashfunktion haben? (Als Startpunkt zum Nachdenken: Könnte der Hashwert dem Angreifer etwas über den symmetrischen Schlüssel verraten?)

c) Falls die in b) gegebene Konstruktion für symmetrische Authentikation für eine konkrete Hashfunktion sicher ist: Können Sie dann aus dieser Hashfunktion ein symmetrisches Konzelationssystem bauen? (Das wäre spannend, denn weitverbreitet ist im Geheimdienstbereich die Ansicht: Gute Hashfunktionen dürfen auch Gegnern bekannt werden, während gute Konzelationssysteme vor ihm tunlichst geheimgehalten werden müssen. In Deutschland beispielsweise beziehen sich die wissenschaftlichen Publikationen des BSI im Bereich Kryptographie ausschließlich auf Hashfunktionen, vgl. z.B. [Dobb_98]).

3-8 Authentikationscodes

a) Rechnen Sie ein kleines Beispiel zur Authentikation von 4 Bits mit dem in der folgenden Tabelle gegebenen Authentikationscode (= Bild 3-15). H (Head) und T (Tail) sind jeweils der Klartext.

Schlüssel \ Text mit MAC	H,0	H,1	T,0	T,1
00	H	–	T	–
01	H	–	–	T
10	–	H	T	–
11	–	H	–	T

b) Warum ist der Authentikationscode informationstheoretisch sicher? (Sie können den Beweis aus der Vorlesung fortsetzen oder zusammenfassen, oder zunächst den Code wesentlich einfacher darstellen.)

c) Wenn man diesen Code verwendet, um eine Nachricht aus mehreren Bits zu authentisieren (wie in Teil a) oder Beispiel 2 der Vorlesung), kann ein Angreifer dann die Nachricht unbemerkt ändern, indem er nur die Reihenfolge der Bits mitsamt ihren MACs vertauscht?

d) Rechnen Sie ein kleines Beispiel zur Konzelation und Authentikation von 4 Bits mit dem in der folgenden Tabelle gegebenen Konzelations- und Authentikationscode. (Er bewirkt gleichzeitig sowohl Konzelation als auch Authentikation.)
Warum ist der folgende Konzelations- und Authentikationscode bei beliebigen Angriffen auf die Authentikation, aber nur bei beobachtenden auf die Konzelation informationstheoretisch sicher? (Es sei unterstellt, daß der Angreifer bei einem verändernden Angriff erfährt, ob sein Angriff auf die Authentikation erfolgreich war oder nicht.)

Schlüssel \ Schlüsseltext	00	01	10	11
00	H	–	T	–
01	T	–	–	H
10	–	T	H	–
11	–	H	–	T

e) *Optional:* Modifizieren Sie den angegebenen Konzelations- und Authentikationscode so, daß er auch bzgl. Konzelation verändernden Angriffen widersteht. Ist ihr System noch effizienter, als wenn man einfach einen Authentikationscode und eine Vernam-Chiffre verwendet?

f) Die angegebenen Authentikationscodes haben zwei Nachteile: Sie arbeiten beide auf einzelnen Bits (H oder T), so daß an jedes Bit ein MAC gehängt werden muß, was für viele Anwendungen nicht effizient ist. Außerdem werden Fälschungsversuche nur mit der Wahrscheinlichkeit 0,5 erkannt und abgewiesen. Wünschen würde man sich eine Wahrscheinlichkeit sehr nahe bei 1.

Der folgende Authentikationscode von Mark N. Wegman und J. Lawrence Carter [WeCa_81] hat diese Nachteile nicht: Nachrichten m seien Elemente eines großen endlichen Körpers K (z.B. Addition und Multiplikation modulo einer großen Primzahl, die allen bekannt ist). Schlüssel seien Paare (a, b) von Elementen von K (und werden nur einmal verwendet). MACs werden so gebildet:

$$MAC := a + b \bullet m$$

Beweisen Sie: Beobachtet ein Angreifer m und MAC, so kann er zu einer anderen Nachricht m' den passenden Wert MAC' nicht zielgerichteter als durch pures Raten bilden. Seine Erfolgswahrscheinlichkeit ist also maximal $1/|K|$.

3-9 Mathematische Geheimnisse

a) (*Nur grob*): Primzahlerzeugungen werden oft so programmiert: Nur am Anfang wird eine ungerade Zufallszahl p gewählt. Wenn diese nicht prim ist, wird in 2-er-Schritten weitergegangen.

Ist es ganz klar, ob Systeme, die auf der Faktorisierungsannahme beruhen, in diesem Fall noch sicher sind?

(Falls jemand das genauer machen wollte, sollte er eine Cramersche Vermutung verwenden, die $\pi(x + \log^2 x) - \pi(x) > 0$ besagt. Dabei bedeute $\pi(y)$ die Anzahl der Primzahlen bis zum Maximum y.)

b) Jemand programmiert – wegen des in a) erwähnten Verfahrens – seine Primzahlerzeugung versehentlich so, daß auch, nachdem eine Primzahl p gefunden wurde, für die Suche der zweiten Primzahl q ab der akuten Stelle in 2-er-Schritten weitergegangen wird. Wie faktorisieren Sie dann sein Produkt n?

c) Berechnen Sie mit Square-and-Multiply: $3^{19} \bmod 101$. (Manchmal hilft: $100 \equiv -1$.)

d) Berechnen Sie $3^{123} \bmod 77$ mit Hilfe des kleinen Fermatschen Satzes.

e) Berechnen Sie mit dem erweiterten Euklidschen Algorithmus das Inverse von 24 mod 101.

f) Wie ist das mit dem Inversen von 21 mod 77?

g) Berechnen sie $y \bmod 77$ mit $y \equiv 2 \bmod 7 \wedge y \equiv 3 \bmod 11$, und z mit $z \equiv 6 \bmod 7 \wedge z \equiv 10 \bmod 11$.

h) Testen Sie, ob die folgenden Restklassen quadratische Reste sind (Rechnen auch mit negativen Restklassen vereinfacht die Arbeit): $13 \bmod 19$, $2 \bmod 23$, $-1 \bmod 89$.
Falls ja, und falls der einfache Algorithmus aus dem Skript anwendbar ist, ziehen Sie damit die Wurzel.

i) Ziehen Sie alle vier Wurzeln aus $58 \bmod 77$. (Sie kennen das Geheimnis $77 = 7 \bullet 11$.)

j) Nutzen Sie, daß $2035^2 \equiv 4 \bmod 10379$, um 10379 zu faktorisieren.
(Für Liebhaber: Wie erzeugt man so eine Aufgabe von Hand?)

k) Sei $n = p \bullet q$, wobei p und q prim sind, $p \neq q$ und $p \equiv q \equiv 3 \bmod 4$, und sei x ein quadratischer Rest modulo n. Zeigen Sie, daß von den 4 Wurzeln von x modulo n genau eine selbst wieder ein quadratischer Rest ist.

l) Zeigen Sie: Wenn die Faktorisierungsannahme falsch wäre, dann wäre auch die Quadratische-Reste-Annahme falsch.

3-10 Welche Angriffe sind auf welche kryptographischen Systeme zu erwarten?

a) Ein Notar versieht beliebige ihm vorgelegte Dokumente mit einem Zeitstempel und signiert sie digital, damit jeder die notarielle Beurkundung des jeweiligen Dokuments prüfen kann. Welche kryptographische Systemklasse muß er verwenden, welchen Angriffen muß das verwendete kryptographische System standhalten?

b) Eine Zeitung erhält von ihren Korrespondenten Artikel *vertraulich* zugeschickt. Welche kryptographischen Systeme können verwendet werden, welchen Angriffen müssen sie jeweils standhalten? (Wir ignorieren, daß im praktischen Leben Zeitungsartikel auch authentisch sein sollten.)

3-11 Wie lange sind Daten zu schützen? Dimensionierung kryptographischer Systeme; Reaktion auf falsche Dimensionierung; Öffentlichen Schlüssel eines Signatursystems oder asymmetrischen Konzelationssystems veröffentlichen?

a) Welcher Zusammenhang besteht zwischen der Zeit, die Daten mittels kryptographischer Systeme geschützt werden sollen, und der Dimensionierung der dazu verwendeten kryptographischen Systeme? Was bedeutet dies für Konzelationssysteme für personenbezogene medizinische Daten? Und für digitale Signatursysteme, wenn Signaturen ein Leben lang gültig bleiben sollen?

b) Was ist bei Konzelationssystemen, was bei digitalen Signatursystemen zu tun, wenn absehbar wird, daß sie demnächst gebrochen werden können? (Hinweis: Vergessen Sie nicht festzulegen, wie mit zu schwach verschlüsselten bzw. signierten Daten verfahren werden soll.)

c) Nehmen wir an, die authentische Weitergabe von öffentlichen Schlüsseln sei aufwendig, vgl. z.B. Aufgabe 3-4 c), so daß Sie es sich nur leisten können, entweder einen öffentlichen Schlüssel eines asymmetrischen Signatursystems oder einen öffentlichen Schlüssel eines asymmetrischen Konzelationssystems weiterzugeben. Welchen geben Sie weiter und warum?

d) Ist Ihre für c) vermutlich unter dem Aspekt der Funktionalität kryptographischer Protokolle gegebene Antwort auch im Lichte der Lösung von b) richtig?

3-12 Kryptopolitik durch Wahl einer geeigneten Schlüssellänge?

Um Staaten Verbrechensbekämpfung (und auch Spionage, aber darüber wird weniger offen geredet) zu erleichtern, Unternehmen und BürgerInnen aber trotzdem einen „ausreichenden“ Schutz der Vertraulichkeit ihrer Daten durch kryptographische Konzelationssysteme zu ermöglichen, wird oftmals vorgeschlagen, die Schlüssellänge geeignet festzulegen: So lang, daß neugierige NachbarInnen und Konkurrenzfirmen den Aufwand des Brechens nicht leisten können, der finanziell und deshalb auch rechenpowermäßig weitaus potentere Staat dies aber kann. Halten Sie dieses Vorgehen für vernünftig oder prinzipiell für aussichtslos? Begründen Sie bitte Ihre Antwort. (Als Hintergrundinformation: Staaten wie die USA und die Bundesrepublik Deutschland haben zwar bisher keine juristischen Einschränkungen der Herstellung und des inländischen Verkaufs von kryptographischen Produkten, ihre Ausfuhr aber unter einen Genehmigungsvorbehalt gestellt. Die USA etwa erteilen 1996 Ausfuhrgenehmigungen nur, falls die Schlüssellänge symmetrischer Konzelationssysteme maximal 40 Bit ist. In der Bundesrepublik ist hierzu offiziell nichts bekannt – das Vorgehen dürfte jedoch ähnlich sein.)

3-13 Authentikation und Konzelation in welcher Reihenfolge?

Für viele Anwendungen sind sowohl Vertraulichkeit als auch Integrität durch kryptographische Systeme zu sichern. In welcher Reihenfolge sollten Konzelation und Authentikation erfolgen? Macht es einen Unterschied, ob symmetrische oder asymmetrische kryptographische Systeme verwendet werden? (Für die gesamte Aufgabe wird angenommen, daß Konzelation und Authentikation in derselben gerätetechnischen Umgebung ablaufen. Also ist beides aus Sicht des Nutzers gleich vertrauenswürdig implementiert. Ebenso kann die Benutzungsschnittstelle unabhängig von

der Reihenfolge gleich gestaltet werden, beispielsweise wird beim Authentisieren immer auch der Klartext angezeigt.)

3-14 s^2-mod-n-Generator

a) Rechnen Sie ein kleines Beispiel für den s^2-mod-n-Generator, verwendet als symmetrisches Konzelationssystem. Vorschlag: Der Schlüssel sei: n=77 und Startwert s=2, der Klartext sei 0101_2.

b) Entschlüsseln Sie die Nachricht 1001_2, s_4=25, die mit dem s^2-mod-n-Generator als asymmetrischem Konzelationssystem verschlüsselt wurde. Ihr geheimgehaltener Dechiffrierschlüssel lautet p=3, q=11. {Wenden Sie beim Entschlüsseln nur Algorithmen an, die auch für $|p| = |q| \geq 300$ noch effizient durchführbar sind. Das Durchprobieren aller möglichen Startwerte z.B. zeigt nur, daß Sie verstanden haben, wie der s^2-mod-n-Generator bei viel zu kleiner Wahl der Schlüssellänge *gebrochen* werden kann. Entsprechendes gilt für Weiterquadrieren und Beobachten, wie sich der Zyklus schließt. Zeigen sollen Sie, daß Sie verstanden haben, wie *entschlüsselt* wird.}

c) Was passiert, wenn man den s^2-mod-n-Generator als symmetrisches Konzelationssystem verwendet und den Schlüsseltext über eine Leitung ohne Fehlertoleranz sendet, und durch einen physischen Fehler ein Bit kippt? Und wenn ein Bit vollständig verlorengeht?

d) Können sie mit dem s^2-mod-n-Generator ein symmetrisches Authentikationssystem fabrizieren, das weniger Schlüsselaustausch braucht als ein echter Authentikationscode? Wie geht das ggf.? Wie sicher ist Ihr System?

e) Wieviel Schlüsselaustausch ist nötig, wenn man mit dem s^2-mod-n-Generator als symmetrischem Konzelationssystem mehrere Nachrichten hintereinander senden will? Was muß jeder speichern?

f) *Optional:* Erreicht der s^2-mod-n-Generator irgendwann einmal den inneren Zustand $s_i = 1$, so sind auch alle inneren Folgezustände 1, da $1^2 = 1$. Also sind alle ab da ausgegebenen Bits 1. Es ist klar, daß dies nur „sehr selten“ vorkommen darf, sonst wäre der s^2-mod-n-Generator kryptographisch unsicher. Berechnen Sie die Wahrscheinlichkeit, daß der s^2-mod-n-Generator nach i Schritten den inneren Zustand 1 erreicht hat. (Tip: Es ist kaum etwas zu berechnen, Sie müssen nur auf ein einfaches Argument kommen.)

3-15 GMR

a) Ihr geheimer Schlüssel sei $p = 11$, $q = 7$. Signieren Sie die Nachricht $m = 01$ an der Referenz $R = 28$. (Die Länge der Nachrichten sei immer 2 bit.) Geht dies bei $R = 28$? Wenn nein, nehmen Sie statt dessen $R = 17$.

b) Nehmen Sie an, insgesamt sollen 8 Nachrichten signiert werden. Welche Teile des Referenzenbaumes und der zugehörigen Signaturen muß der Signierer bei der vierten Nachricht neu erzeugen? Und bei der fünften? Welche Teile werden verschickt; was muß der Empfänger testen? (Am besten ein Bild malen und einkringeln.)

c) Skizzieren sie den Beweis für die in §3.5.3 genannte Kollisionsresistenz der Familie der $f_{präf(m)}$. Wo in Ihrem Beweis wird die präfixfreie Codierung benötigt?

3-16 RSA

a) Rechnen Sie ein kleines Beispiel für RSA als Konzelationssystem.

b) Bilden Sie ein kleines Beispiel für RSA als Signatursystem.

c) Wie werden die Klartexte 0 und 1 verschlüsselt, wie werden die Schlüsseltexte 0 und 1 entschlüsselt? Was lernen wir daraus?

d) Wie würden Sie aus RSA ein indeterministisch verschlüsselndes Konzelationssystem machen? Wie viele zufällige Bits brauchen Sie? Warum reichen z.B. 8 zufällige Bits nicht? Warum sollte man z.B. die Uhrzeit nicht als zufällige Bits nehmen?

e) *Optional:* Sehen Sie, ob Ihr Vorschlag aus d) sicher gegen aktive Angriffe ist, vgl. §3.6.3? Was wäre ggf. gegen aktive Angriffe zu tun? Wie sähe dann das Nachrichtenformat aus?

f) *Optional:* Wie würden Sie aus RSA ein indeterministisch signierendes Signatursystem machen? Was könnte das bringen? Könnte das gar Probleme schaffen?

g) Wie wächst der bei RSA pro verschlüsseltem Bit erforderliche Rechenaufwand in Abhängigkeit des Sicherheitsparameters l, wenn die in §3.4.1.1 beschriebenen Algorithmen zur Implementierung verwendet werden? Unterscheiden Sie als Grenzfälle, ob es sich um extrem kurze oder extrem lange Nachrichten handelt.

h) Skizzieren Sie, wieso die geheime Operation bei RSA, also das Signieren bzw. Entschlüsseln, etwa vier mal so schnell ist, wenn man modulo p und q einzeln rechnet, als wenn man modulo n rechnet.

i) Um welchen Faktor wird die öffentliche Operation bei RSA schneller, wenn Sie statt einem zufälligen Exponenten den Exponenten $2^{16}+1$ nehmen?

j) Was halten Sie von folgender Änderung der Schlüsselgenerierung von RSA (entnommen [MeOV_97 Seite 287]): Statt $\phi(n) = (p-1) \cdot (q-1)$ wird das kleinste gemeinsame Vielfache von $p-1$ und $q-1$ genommen, d.h. kgV($p-1$, $q-1$)? Ändert sich dadurch etwas an der Sicherheit von RSA? Was ändert sich dadurch am Rechenaufwand?

k) *Optional:* Rechnen Sie ein kleines Beispiel für den Angriff von Johan Håstad.

l) Beispielsweise in [MeOV_97 Seite 287] finden Sie einen Beweis dafür, daß vollständiges Brechen von RSA, d.h. das Finden des geheimen Schlüssels d, äquivalent zum Faktorisieren des Modulus ist. Warum sagt dieser Beweis leider nichts Wesentliches über die Sicherheit von RSA aus?

3-17 DES

a) Ein einfaches, DES-artiges kryptographisches System sei gegeben durch: Blocklänge 6; 2 Iterationsrunden, keine Eingangspermutation; f sei definiert durch $f(R, K_i) = R \bullet K_i \bmod 8$; K sei einfach (K_1, K_2).

   Der Schlüssel sei 011010. Verschlüsseln Sie den Klartext 000101 und entschlüsseln Sie ihn wieder.

b) Wie würden Sie S-Boxen in Software realisieren, um möglichst schnell verschlüsseln zu können?

c) *Optional:* Und wie die 32-bit-Permutationen?

d) Klartext-Schlüsseltext-Angriff: Wenn Sie einige Klartextblock-Schlüsseltextblock-Paare kennen, die mittels DES mit dem gleichen Schlüssel verschlüsselt wurden, wie und mit welchem Aufwand (gemessen in Ver- bzw. Entschlüsselungen von DES) können Sie den Schlüssel ermitteln?

e) gewählter Klartext-Schlüsseltext-Angriff: Ändert sich gegenüber d) etwas, wenn sie zusätzlich einen Klartextblock wählen dürfen und den zugehörigen Schlüsseltextblock erhalten? Wie hoch ist nun der Aufwand?

f) Wie könnten Sie DES brechen, wenn DES keine Substitutionen, sondern nur Permutationen und Additionen-modulo-2 enthielte? Überlegen Sie, ob Ihr Angriff allgemeiner für jede lineare Blockchiffre funktioniert?

3-18 Betriebsarten

a) Mit CBC konnte man Vertraulichkeit erreichen oder Authentikation: ersteres, indem man genau die Schlüsseltextblöcke überträgt, letzteres, indem man den Klartext und nur den letzten Schlüsseltextblock überträgt. Man könnte denken, beides ginge gleichzeitig, indem man einfach den Schlüsseltext überträgt: Einerseits wäre Geheimhaltung gesichert, andererseits

könnte der Empfänger den Klartext berechnen und hätte also auch sämtliche Information für die Authentikation zur Verfügung. Wieso kann das prinzipiell nicht genügen?

b) Zeigen Sie mit einem einfachen Angriff, daß das Schema immer noch unsicher ist, wenn man an jeden Klartext vor dem Verschlüsseln hinten einen Block aus lauter Nullen anhängt, um wenigstens etwas Redundanz zu haben.

c) Betrachten Sie die Betriebsart PCBC in §3.8.2.5. Zeigen Sie, daß Ihr Angriff dort nicht mehr funktioniert (wenn man dort genauso vorgeht, also einen Block aus lauter Nullen anhängt und dann den Schlüsseltext überträgt und sowohl Konzelation als auch Authentikation will).

d) Untersuchen Sie für die in §3.8.2 angegebenen Betriebsarten, was ein Angreifer, der Gelegenheiten zu *adaptiven aktiven Angriffen* hat, erreichen kann. Nehmen Sie hierzu an, daß er die verwendete Blockchiffre nicht brechen kann. {Wie üblich wird unterstellt, daß der Angreifer die verwendeten kryptographischen Systeme kennt. Insbesondere kennt er also die Länge der verwendeten Blockchiffre und die jeweils verwendete Betriebsart.}

e) Sie möchten mit ECB, CBC (sei es zur Konzelation, sei es zur Authentikation) oder PCBC arbeiten, wissen aber nicht, ob Sie Klartexte *passender* Länge bekommen (d.h. Klartextlänge ist durch die Blocklänge ohne Rest teilbar). Der oftmals gegebene Rat, "dann füllen wir eben mit Nullen auf die Blocklänge auf." befriedigt Sie nicht, denn es soll einen Unterschied machen, ob die Nachricht "Ich schulde Dir DM 10" oder "Ich schulde Dir DM 1000000" ankommt. Entwerfen Sie eine Codierung, die
   1. Klartexte beliebiger Länge auf solche passender Länge abbildet,
   2. eindeutig zu invertieren ist und
   3. deren Längenexpansion minimal ist.

f) OFB gestattet leider weder wahlfreien Zugriff noch Parallelisierbarkeit, denn der Zustand des Schieberegisters muß jeweils von vorne berechnet werden. Entwerfen Sie eine naheliegende Modifikation von OFB, die wahlfreien Zugriff und Parallelisierbarkeit erlaubt.

3-18a Spiegelangriff

Nehmen wir an, wir wollen Paßworte beim LOGIN durch etwas besseres ersetzen, haben aber dumme Terminals und ein unsicheres Netz zwischen Terminals und Großrechner. Dafür hat jeder Teilnehmer einen „Taschenrechner“, der eine sichere symmetrische Blockchiffre (vgl. Fußnote in §2.2.1) implementiert, allerdings keinen elektrischen Anschluß an das Terminal besitzt. Deshalb muß aller Informationstransfer via Display und Tastatur über den Teilnehmer laufen. Was tun Sie, wenn Sie annehmen dürfen, daß der Großrechner selbst sicher ist? Welche Angriffe werden durch das von Ihnen entworfene System nicht toleriert?

3-19 Ende-zu-Ende-Verschlüsselung

Welche Probleme ergeben sich, wenn jemand Ende-zu-Ende-Verschlüsselung von einem Bürorechner aus betreiben will, also einem Rechner, zu dem nicht nur er selbst Zugang hat? Was sehen Sie für Lösungsansätze?

3-20 Ende-zu-Ende-Verschlüsselung ohne vorher ausgetauschten Schlüssel

Unser Kryptographenpaar Alice und Bob ist – wie immer – getrennt und diesmal ist Alice auf einer Flugreise.

a) Und wieder einmal ist das Fluggepäck verloren gegangen, Schlüssel und Kryptogeräte also nicht verfügbar. Wie können die beiden halbwegs vertraulich und authentisch kommunizieren? (Ein Tip: Sie müssen annehmen, daß der Angreifer nicht immerzu überall sein kann.)

b) Das Fluggepäck wurde wiedergefunden und zurückgegeben. Aber irgendwer wollte die Schlüssel in den Kryptogeräten ausforschen, die diese also gelöscht haben. Wie können Alice und Bob ihre Sicherheit nun verbessern?

3-21 Identifikation und Authentikation mittels asymmetrischem Konzelationssystem

Wie kann ein Teilnehmer *T*, dessen öffentlicher Schlüssel c_T eines asymmetrischen Konzelationssystems einem Teilnehmer *U* bekannt ist, von *U* identifiziert werden, vgl. §2.2.1?

a) Entwerfen Sie ein kleines kryptographisches Protokoll. (Die Existenz solch eines Protokolls ist erstaunlich, da asymmetrische Konzelationssysteme nur mit dem Ziel *Vertraulichkeit* im Sinn definiert wurden. Aber es gibt tatsächlich das gesuchte Protokoll.)

b) Gegen welche Angriffe muß das asymmetrische Konzelationssystem in Ihrem kleinen kryptographischen Protokoll sicher sein?

c) Können Sie Ihr kleines kryptographisches Protokoll so verbessern, daß es nicht nur *T* identifiziert, sondern auch Nachrichten N_i authentisiert? (Als Ansporn: Es gibt solch ein Protokoll.) Ist Ihr Authentikations„system“ symmetrisch oder asymmetrisch, d.h. entspricht es gar digitalen Signaturen? Was ist der Nachteil gegenüber „normalen“ Authentikationssystemen?

d) Muß in Ihrem verbesserten kryptographischen Protokoll das asymmetrische Konzelationssystem noch genauso starken aktiven Angriffen standhalten wie in b)?

3-22 Konzelation mittels symmetrischem Authentikationssystem

Wie kann ein Teilnehmer *A*, der mit einem Teilnehmer *B* nur ein symmetrisches Authentikationssystem inkl. ausgetauschtem Schlüssel k_{AB} gemein hat, mit diesem konzeliert kommunizieren? Bitte nehmen Sie nicht einfach den vertraulich ausgetauschten Schlüssel k_{AB} für ein symmetrisches Konzelationssystem, es geht tatsächlich nur mit einem Authentikationssystem. (Die Existenz solch eines Protokolls ist erstaunlich, da symmetrische Authentikationssysteme nur mit dem Ziel *Integrität* im Sinn definiert wurden. Aber es gibt tatsächlich das gesuchte Protokoll.)

3-23 Diffie-Hellman Schlüsselaustausch

a) *Kernidee:* Was ist die Kernidee des Diffie-Hellman Schlüsselaustauschs?

b) *Anonymität:* Diffie-Hellman Schlüsselaustausch, wie er in §3.9.1 beschrieben wurde, unterscheidet sich von asymmetrischen Konzelationssystemen gemäß §3.1.1.1 darin, ob der Sender gegenüber dem Empfänger der verschlüsselten Nachricht anonym bleiben kann oder nicht.

Bei asymmetrischen Konzelationssystemen ist dies trivialerweise der Fall: Der Sender besorgt sich den öffentlichen Konzelationsschlüssel, verschlüsselt die Nachricht damit, schickt sie an den Empfänger. Der Empfänger entschlüsselt die Nachricht völlig unabhängig davon, wer sie verschlüsselt hat.

Bei Diffie-Hellman Schlüsselaustausch ist das anders: Für die symmetrische Ver- und Entschlüsselung der Nachricht wird jeweils der geheime, der Paarbeziehung der Beteiligten zugeordnete Schlüssel verwendet. Also scheint erstmal keine Anonymität des Senders gegenüber dem Empfänger möglich zu sein.

Überlegen Sie sich eine Modifikation des Diffie-Hellman Schlüsselaustauschs, bei der Anonymität des Senders gegenüber dem Empfänger möglich ist. Und was ist zu tun, wenn der Sender anonym eine Antwort erhalten will?

c) *Individuelle Parameterwahl:* Diffie-Hellman Schlüsselaustausch, wie er in §3.9.1 beschrieben wurde, unterscheidet sich von asymmetrischen Konzelationssystemen gemäß §3.1.1.1 darin, daß sicherheitskritische Parameter (konkret: *p* und *g*) allgemein bekannt und deshalb im einfachsten Fall für alle Teilnehmer gleich sind. Es stellt sich sofort die Frage, wer sie wählt und ob derjenige hierdurch eine besondere Machtposition erringen kann. Denn es ist nicht

kontrollierbar, ob er *p* und *g zufällig* wählt oder irgendwie speziell so, daß das Ziehen diskreter Logarithmen für ihn besonders leicht ist. Läßt man eine fremde Instanz *p* und *g* wählen, so benötigt man also eine stärkere Sicherheitsannahme als die Diskrete-Logarithmus-Annahme. Alternativ können viele Instanzen *p* und *g* gemeinsam wählen, aber das erfordert auch wieder ein kryptographisches Protokoll.

Überlegen Sie sich im Rahmen dieser Aufgabe, ob Sie den Diffie-Hellman Schlüsselaustausch so modifizieren können, daß jeder Teilnehmer alle sicherheitskritischen Parameter selbst wählt.

d) *DSA (DSS) als Basis:* Am 19. Mai 1994 wurde vom US-amerikanischen „National Institute of Standards and Technology (NIST)“, der Nachfolgeorganisation des National Bureau of Standards, das 1977 DES standardisierte, ein Standard für digitale Signaturen innerhalb aller und mit allen öffentlichen Stellen der USA endgültig gesetzt: der **Digital Signature Standard (DSS)**, gegeben durch den **Digital Signature Algorithm (DSA)** [Schn_96 Seite 483-495]. Die uns interessierenden Schritte bei der Schlüsselgenerierung, -zertifizierung und -veröffentlichung gemäß DSS sind:

1. Öffentlich (und möglicherweise für alle Teilnehmer gleich) ist eine große Primzahl *p* und eine im wesentlichen zufällig bestimmte Zahl $g \in \mathbb{Z}_p^*$.
2. Jeder Teilnehmer wählt eine Zahl *x* von etwa 159 Bit Länge zufällig und hält sie als Teil seines Signierschlüssels geheim.
3. Jeder Teilnehmer berechnet $g^x \bmod p$, welches als Teil seines Testschlüssels zertifiziert und veröffentlicht wird.

Reicht dies, um Diffie-Hellman Schlüsselaustausch durchzuführen?

3-24 Blind geleistete Signaturen mit RSA

Rechnen Sie ein kleines Beispiel für blind geleistete Signaturen mit RSA. Sie können Rechenaufwand sparen, indem Sie auf Aufgabe 3-16 b) aufbauen.

3-25 Schwellwertschema

Zerlegen Sie das Geheimnis $G = 13$ so in $n = 5$ Teile, daß $k = 3$ nötig sind, um es zu rekonstruieren. Verwenden Sie die kleinstmögliche Primzahl *p* und als zufällige Koeffizienten $a_1 = 10$ und $a_2 = 2$. (In der Praxis darf *p* keinesfalls in so starker Abhängigkeit von *G* gewählt werden. Diese Wahl von *G* soll Ihnen nur das Rechnen erleichtern und zeigen, daß Sie die Bedingungen an *p* in Bezug auf *G* verstanden haben.)

Rekonstruieren Sie das Geheimnis unter Verwendung der Teile (1, q(1)), (3, q(3)) und (5, q(5)).

3-26 Beweisbar sicherer Gebrauch mehrerer Konzelationssysteme?

Nachdem Sie wissen, daß es informationstheoretisch sichere, effiziente Schwellwertschemata gibt, lohnt es sich, sich nochmals Aufgabe 3-3 zuzuwenden. Wie kombinieren Sie mehrere Konzelationssysteme so, daß das resultierende Gesamtsystem beweisbar mindestens so sicher ist wie das sicherste verwendete Konzelationssystem? (Dabei dürfen Sie voraussetzen, daß die Konzelationssysteme für gleichverteilt zufällige Klartexte – wie sie beispielsweise bei minimaler Längencodierung entstehen – entworfen sind und daß „Sicherheit des Konzelationssystems“ Sicherheit bei gleichverteilt zufälligen Klartexten bedeutet. Ebenfalls dürfen Sie voraussetzen, daß die Teile, die das informationstheoretisch sichere, effiziente Schwellwertschemata generiert, aus Sicht der Konzelationssysteme gleichverteilt zufällige Klartexte sind.)

Vergleichen Sie den Aufwand der beweisbar sicheren Kombination mit der in Aufgabe 3-3 beschriebenen.

Vergleichen Sie die durch die Kombination jeweils erreichte Sicherheit.

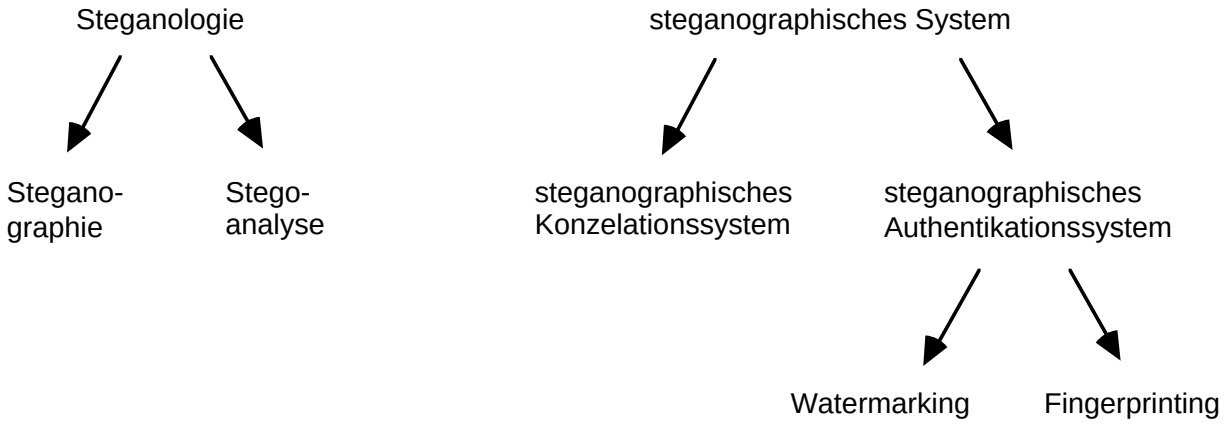
Fällt Ihnen eine besonders einfache Implementierung eines Schwellwertschemas für genau diese Anwendung ein?

Aufgaben zu Steganographische Grundlagen

4-1 Begriffe

Im nachfolgenden Bild sind Begriffsbäume dargestellt, die analog zu denen der Kryptographie, vgl. Aufgabe 3-0, gebildet sind. Neu gebildet – und gewöhnungsbedürftig – ist der Begriff *Steganologie*. Ich war versucht, den kürzeren Begriff *Stegologie* zu nehmen, habe mich aber doch für die längere, sprachlich korrekte Form entschieden. Manche Autoren nennen das Gebiet Steganologie und damit insbesondere auch den Bereich Steganographie *information hiding*. Information hiding zerfällt bei Ihnen in die Bereiche *steganography* (das, was ich steganographische Konzelation nenne) und *copyright marking* (das, was ich steganographische Authentikation nenne). Ähnlich wie statt *Kryptoanalyse* hin und wieder auch im Deutschen *Kryptanalyse* gesagt wird, führt die englische Begriffsbildung *steganalysis* dazu, im Deutschen auch den kürzeren Begriff *Steganalyse* zu verwenden.

Erläutern Sie den inhaltlichen Aufbau der Begriffsbäume und die Vor- und Nachteile, den kurzen Begriff *Stegosystem* allgemein für steganographische Systeme zu verwenden oder spezieller nur für steganographische Konzelationssysteme.



4-2 Eigenschaften der Ein- und Ausgaben kryptographischer und steganographischer systeme

a) Zeichnen Sie ein Schema (Blockschaltbild), das die allgemeinen Parameter (alle Ein- und Ausgabewerte) eines kryptographischen Konzelationssystems enthält. Worin unterscheiden sich symmetrische und asymmetrische Kryptographie? Welche Annahmen über die Eingaben darf ein gutes Kryptosystem machen, welche Vorgaben über seine Ausgaben muß es erfüllen?

b) Zeichnen Sie ein Schema (Blockschaltbild), das die allgemeinen Parameter (alle Ein- und Ausgabewerte) eines steganographischen Konzelationssystems enthält. Welche Annahmen über die Eingaben darf ein gutes Stegosystem machen, welche Annahmen über seine Ausgaben muß es erfüllen?

c) Worin bestehen also die Hauptunterschiede zwischen kryptographischen und steganographischen Konzelationssystemen?

4-3 Macht gute Steganographie Verschlüsselung überflüssig?

Warum ist es bei Benutzung eines sicheren steganographischen Konzelationssystems eigentlich unnötig, die geheime Nachricht vor der Einbettung zu verschlüsseln? Und warum würden Sie es in der Praxis doch tun?

4-4 Beispiele steganographischer Systeme

Zählen Sie einige Beispiele steganographischer Systeme auf. Wie beurteilen Sie die Sicherheit der von Ihnen aufgezählten Systeme?

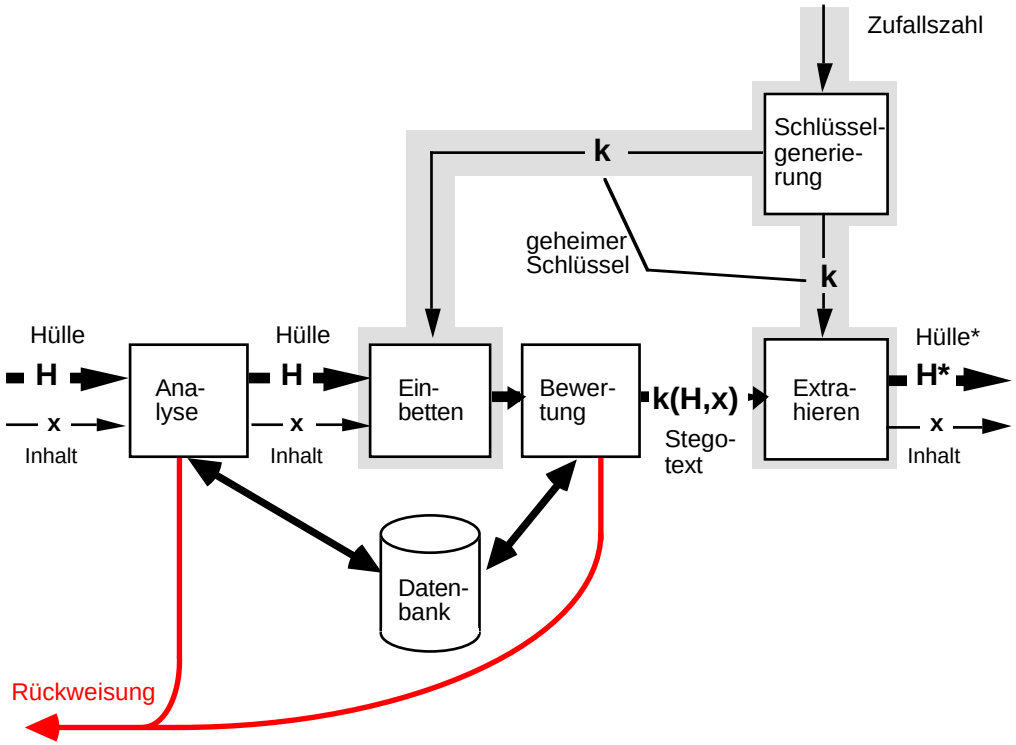
4-4a Modellierung steganographischer Systeme

Eine Gruppe Studies diskutiert über die in den Bildern 4-2, 4-3, 4-4 und 4-5 gezeigte Modellierung steganographischer Systeme:

Da wurde Wesentliches vergessen, sagt *Anja Vorausschau*. Vor dem Einbetten brauchen wir noch eine **Analyse**, die mit einer Datenbank aller bisher verwendeten Hüllen zusammenarbeiten muß. Nur so können wir vermeiden, daß in ungeeignete Hüllen eingebettet wird.

Richtig, pflichtet *Berta Schauzurück* bei. Aber noch besser wäre es, die Analyse hinter die Einbettung zu ziehen – nennen wir diesen Algorithmus **Bewertung**. Die Aufgabe von Bewertung ist zu prüfen, ob der Stegotext unauffällig ist. Wenn ja, wird er ausgegeben. Wenn nein, wird er zurückgewiesen.

Streitet Euch nicht, bemerkt *Caecilie Grundmann*. Wir nehmen einfach sowohl Analyse vorher wie auch Bewertung nach der Einbettung. Das ist der allgemeine Fall und wir sind uns ja wohl einig: Sicher ist sicher! Ich zeichne Euch das mal auf:



Dörte Reinhardt zieht die Augenbrauen hoch und widerspricht: Nein, ich finde keinen Eurer Verbesserungsvorschläge gut. Das ursprüngliche Modell reicht.

Wem geben Sie recht – und vor allem: Wie begründen Sie, wem Sie recht geben?

4-5 Steganographische Konzelation mit öffentlichen Schlüsseln?

Ross Anderson beschreibt in [Ande4_96, AnPe_98 Seite 480] folgendes Verfahren für steganographische Konzelation:

> Gegeben eine Hülle, in die überhaupt irgendwelcher Schlüsseltext eingebettet werden kann. Dann gibt es üblicherweise eine Rate, mit der dessen Bits eingebettet werden können, ohne daß dies der Stegoanalytiker bemerkt. ... Alice kennt Bobs öffentlichen Chiffrierschlüssel. Sie kann ihre geheimzuhaltende Nachricht nehmen, mit Bobs öffentlichem Chiffrierschlüssel

verschlüsseln, und den resultierenden Schlüsseltext einbetten. Jeder mögliche Empfänger versucht dann einfach, jede extrahierte Nachricht zu entschlüsseln, und nur Bob wird Erfolg haben. In der Praxis könnte der mit dem öffentlichen Schlüssel verschlüsselte Wert ein Kontrollblock, bestehend aus einem Sitzungsschlüssel und einigem Füllmaterial, sein. Der Sitzungsschlüssel würde zum Betrieb eines konventionellen steganographischen Systems verwendet.

Ross Anderson nennt dies Steganographie mit öffentlichen Schlüsseln (public key steganography). Schließen Sie sich dieser Begriffsbildung an? Bitte begründen Sie Ihre Entscheidung.

4-6 Steigerung der Sicherheit durch Gebrauch von mehreren steganographischen Systemen

Was können Sie tun, wenn Sie mehrere steganographische Systeme haben, die auf unterschiedlichen Sicherheitsannahmen beruhen, Sie aber keinem der steganographischen Systeme (bzw. keiner der Sicherheitsannahmen) allein vertrauen wollen? Unterscheiden Sie die Schutzziele Konzelation (d.h. Sie haben mehrere Konzelationssysteme) und Authentikation (d.h. Sie haben mehrere Authentikationssysteme).

4-7 Krypto- und steganographische Konzelation kombiniert?

Steganographische Konzelation überdeckt die Funktionalität kryptographischer Konzelation, so daß, wenn die steganographische Konzelation als sicher vorausgesetzt werden kann, keine Kombination mit kryptographischer Konzelation sinnvoll ist. Ist eine Kombination sinnvoll und wie sollte sie ggf. erfolgen, falls Zweifel an der Sicherheit der steganographischen Konzelation bestehen?

4-8 Krypto- und steganographische Authentikation in welcher Reihenfolge?

Steganographische Authentikation schützt das Werk nicht gegen unerkannte kleine Verfälschungen, eine digitale Signatur tut dies zwar, kann aber von jederman leicht entfernt werden. Können Sie beides kombinieren? Ggf. wie?

4-9 Stego-Kapazität von komprimierten und nichtkomprimierten Signalen

Kann in komprimierte oder nichtkomprimierte Signale prozentual mehr unentdeckbar eingebettet werden? Unterscheiden Sie bitte zwischen verlustfreier- und verlustbehafteter Kompression und begründen Sie Ihre Antworten.

Aufgaben zu Sicherheit in Kommunikationsnetzen

5-0 Quantifizierung von Unbeobachtbarkeit, Anonymität und Unverkettbarkeit

In §5.1.2 werden Definitionen von Unbeobachtbarkeit, Anonymität und Unverkettbarkeit gegeben und betont, daß deren Vertrauenswürdigkeit vom unterlegten Angreifermodell und ggf. einer Klasseneinteilung abhängt. Es wird jeweils definiert, daß Unbeobachtbarkeit, Anonymität und Unverkettbarkeit perfekt sind, wenn die relevanten Wahrscheinlichkeiten für den Angreifer vor und nach seinen Beobachtungen gleich sind. Versuchen Sie, Unbeobachtbarkeit, Anonymität und Unverkettbarkeit zu quantifizieren, d.h. jeweils zwischen den Extremen:

- Der Angreifer erfährt nichts – der Angreifer erfährt alles und
- Der Angreifer weiß nichts – der Angreifer weiß alles

sinnvolle a) Kenngrößen bzw. b) Abstufungen zu finden.

5-1 Ende-zu-Ende- oder Verbindungs-Verschlüsselung

Aufgrund der Enthüllungen über das massenhafte Mithören von Geheimdiensten (z.B. Stasi) des Fernmeldeverkehrs beschließt die Leitung eines Konzerns, in Zukunft allen Fernmeldeverkehr zwischen den Geschäftsstellen zu verschlüsseln. Wie kann, wie sollte dies geschehen? Handelt es

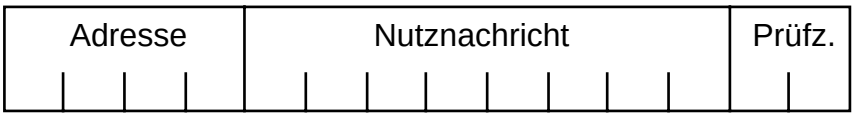
sich bei Ihren Maßnahmen eher um Verbindungs- oder eher um Ende-zu-Ende-Verschlüsselung? Oder empfehlen Sie gar, beides zu verwenden?

5-2 Erst verschlüsseln und dann fehlertolerant codieren oder umgekehrt?

In vielen Kommunikationsnetzen werden fehlererkennende und ggf. sogar fehlerkorrigierende Codes eingesetzt, die an das dumme Fehlerverhalten der Kommunikationskanäle angepaßt sind. Je nach Kanal ist damit zu rechnen, daß eher Nullen zu Einsen oder eher Einsen zu Nullen werden, daß Fehler eher einzeln oder in Bündeln auftreten, etc. Sollte man in Netzen mit angepaßter fehlertoleranter Codierung erst verschlüsseln und danach fehlertolerant codieren, oder ist die umgekehrte Reihenfolge vorzuziehen? Begründen Sie Ihre Antwort.

5-3 Behandlung von Adreß- und Fehlererkennungsfeldern bei Verschlüsselung

Gegeben sei die Klartext-Nutznachricht 00110110, die an die Adresse 0111 geschickt werden soll (angegeben ist in dieser Aufgabe jeweils die Binärcodierung). Das übliche Nachrichtenformat bestehe aus einer 4-Bit-Adresse, einer 8-Bit-Nutznachricht und einem 2-Bit-Prüfzeichen zur Erkennung von Übertragungsfehlern in Adresse oder Nutznachricht.



Das 2-Bit-Prüfzeichen werde durch zweibitweise Addition mod 4 von Adresse und Nutznachricht gebildet. Für obige Adresse und Klartext-Nutznachricht ergäbe sich als Prüfzeichen also 10, insgesamt also die Nachricht 0111 00110110 10.

a) Obige Nachricht werde mit einem One-time-pad Ende-zu-Ende-verschlüsselt. Das One-time-pad sei 0110 0111 1111 0001 1010 1000 1101 ... Wie lautet die Ende-zu-Ende-verschlüsselte Nachricht, wenn modulo 2 One-time-pad-verschlüsselt wird?

b) Obige Nachricht werde mit einem One-time-pad Verbindungs-verschlüsselt. Das One-time-pad sei 0011 1010 0111 0011 0001 1101 0111 ... Wie lautet die Verbindungs-verschlüsselte Nachricht, wenn modulo 2 One-time-pad-verschlüsselt wird?

c) Obige Nachricht werde mit obigen One-time-pads Ende-zu-Ende- und Verbindungs-verschlüsselt. Wie lautet die Ende-zu-Ende- und Verbindungs-verschlüsselte Nachricht, wenn modulo 2 One-time-pad-verschlüsselt wird?

5-4 Verschlüsselung bei verbindungsorientierten und verbindungslosen Kommunikationsdiensten

Bei Kommunikationsdiensten wird zwischen *verbindungsorientierten* und *verbindungslosen* Kommunikationsdiensten unterschieden. Erstere garantieren, daß Nachrichten in derselben Reihenfolge empfangen wie gesendet werden, letztere garantieren dies nicht. Müssen Sie dies bei der Wahl des Verschlüsselungsalgorithmus und seiner Betriebsart berücksichtigen? Falls ja, worauf müssen Sie achten?

5-4a Abfragen und Überlagern

a) Rechnen Sie ein kleines Beispiel zum Verfahren „Abfragen und Überlagern“.

b) Warum muß zwischen Teilnehmer und Servern verschlüsselt werden? Muß auch die Antwort der Server an den Teilnehmer verschlüsselt werden?

c) Wie implementieren Sie „Abfragen und Überlagern“ unter Verwendung üblicher kryptographischer Hilfsmittel, wenn die Übertragungsbandbreite vom Teilnehmer zu den Servern knapp ist? Wie, wenn die Übertragungsbandbreite von den Servern zum Teilnehmer, nicht aber der Server untereinander knapp ist? Können Sie Ihre beiden Lösungen geschickt kombinieren?

d) Was halten Sie von folgendem Vorschlag: Um die Server zu verwirren, bildet der Teilnehmer einen zusätzlichen Abfragevektor vollkommen zufällig, sendet ihn an einen weiteren Server, erhält dessen Antwort und ignoriert diese. Keiner der verwendeten s+1 Server weiß, ob er als normaler Server oder als weiterer, ignorierter Server verwendet wird.

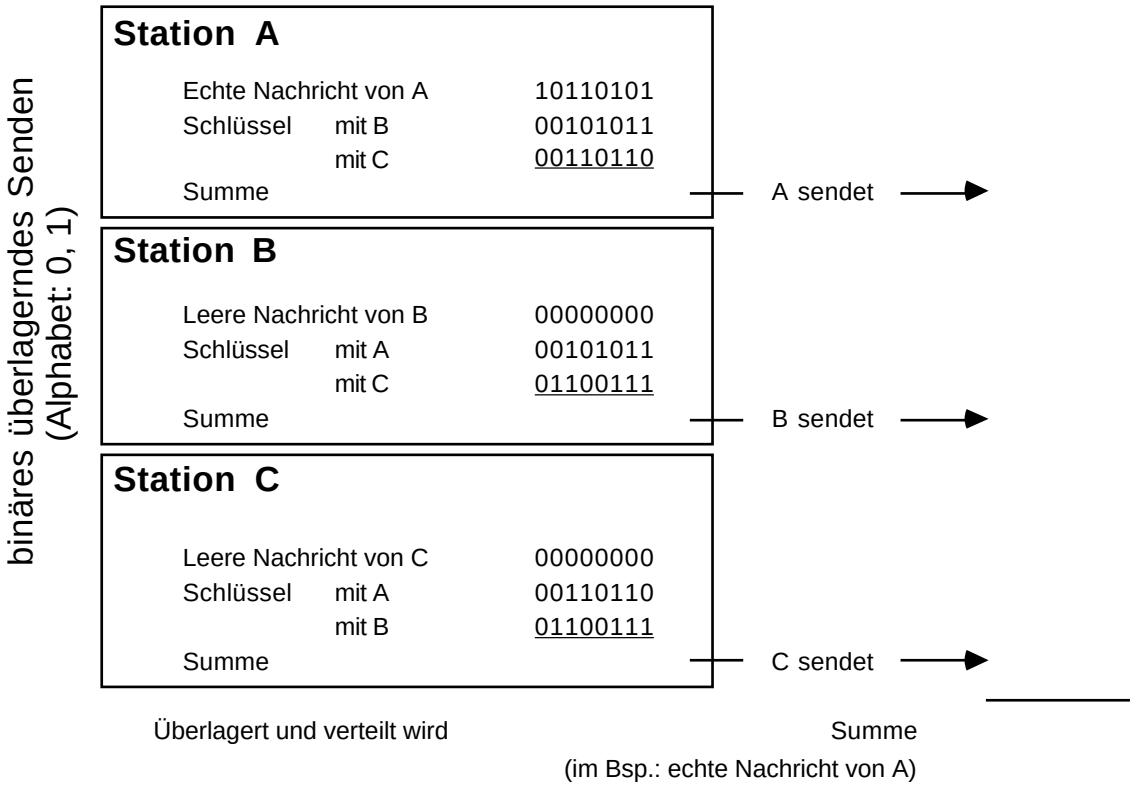
5-4b Vergleich „Verteilung“ und „Abfragen und Überlagern“

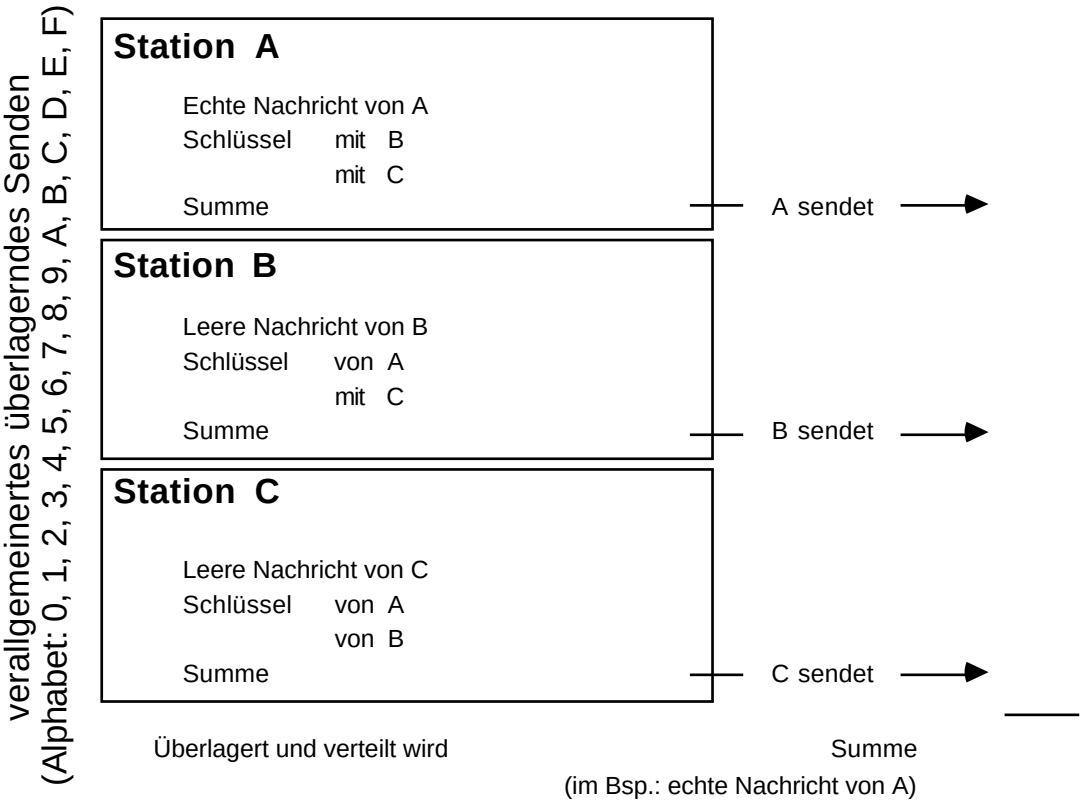
a) Vergleichen Sie „Verteilung“ und „Abfragen und Überlagern“. Überlegen Sie insbesondere, wie offene implizite Adressen bei „Verteilung“ bei „Abfragen und Überlagern“ äußerst effizient implementiert werden können.

b) Können Sie sich Kommunikationsdienste vorstellen, wo „Abfragen und Überlagern“ nötig ist, da „Verteilung“ nicht anwendbar ist?

5-5 Überlagerndes Senden: Ein Beispiel

Gegeben sei das im folgenden Bild gezeigte Beispiel für überlagerndes Senden. Tragen Sie die fehlenden Werte im oberen Teilbild ein, in dem modulo 2 gerechnet wird. Rechnen Sie danach das gleiche Beispiel modulo 16 im unteren Teilbild. Welche Werte müssen/dürften Sie, bis auf die hexadezimale Codierung, aus dem oberen Teilbild abschreiben? Warum?





5-6 Überlagerndes Senden: Bestimmen einer passenden Schlüsselkombination zu einer alternativen Nachrichtenkombination

Wie müssen im Beispiel der vorherigen Aufgabe die Schlüssel lauten, damit bei gleichen lokalen Summen und Ausgaben (und natürlich damit auch bei gleicher globaler Summe) die Station A die leere Nachricht 00000000 und die Station B die echte Nachricht 01000101 gesendet hat? Welche

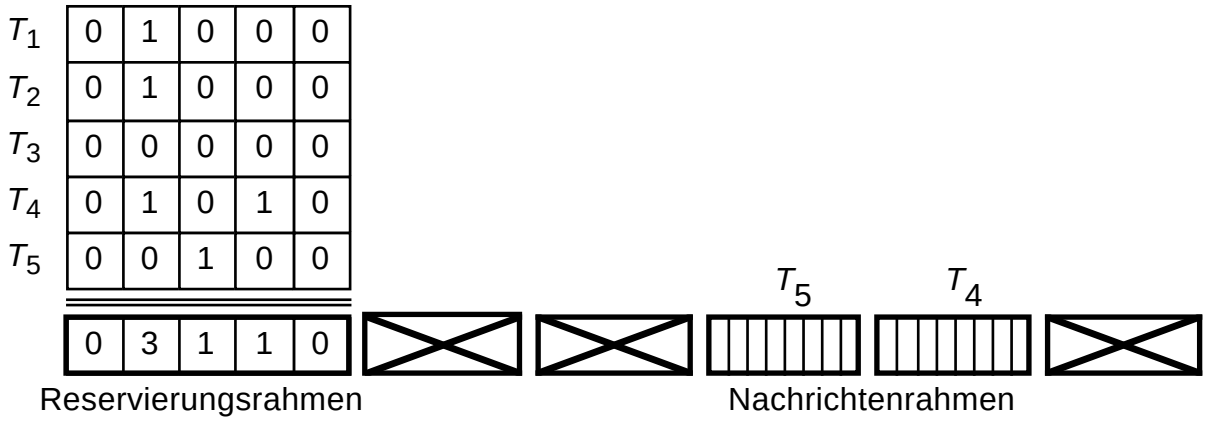
echte Nachricht muß Station C dann gesendet haben? Ist die Schlüsselkombination eindeutig? Wird sie es, wenn zusätzlich vorgegeben wird, daß der Schlüssel zwischen A und B unverändert bleiben soll, etwa weil ihn der Angreifer kennt? (A hat diesen Schlüssel B auf einer Diskette gegeben, die B, nachdem er ihren Inhalt auf seine Festplatte kopiert hat, nicht etwa physisch vernichtet oder zumindest vielmals physisch beschrieben und physisch gelöscht hat, sondern er hat sie nur gelöscht, d.h. das Betriebssystem seines Rechners hat den Schlüssel als gelöscht gekennzeichnet. Danach hat er die fast neue Diskette mit wenigen Daten bespielt und einem freundlichen Menschen, der unerwartet auch als Angreifer agiert, gegeben.) Wie lauten in diesem Fall die anderen Schlüssel?

Hinweis: Wenn Sie diese Aufgabe nur bzgl. Addition modulo 2 rechnen, sollten Sie auch dort zwischen + und – unterscheiden. In den Ergebnissen zur Geltung kommt der Unterschied natürlich nur bei größerem Modulus, im Beispiel also nur bei Addition modulo 16.

5-7 Mehrfachzugriffsverfahren für additive Kanäle, beispielsweise überlagerndes Senden

a) Reservierungsverfahren

Im folgenden Bild ist gezeigt, wie Reservierung im Nachrichtenrahmen Kollisionen von Nachrichtenrahmen vermeidet, wenn nach der Reservierung nur von solchen Stationen Nachrichtenrahmen genutzt werden, deren Reservierung in einer 1 als Summe resultierte. Was würde im Beispiel passieren, wenn die Reservierungsrahmen nicht modulo einer größeren Zahl, sondern modulo 2 überlagert würden?



b) Paarweises überlagerndes Empfangen

In einem DC-Netz, in dem paarweises überlagerndes Empfangen nicht vorgesehen ist, einigen sich zwei Freunde, Sparsi und Knausri, durch paarweises überlagerndes Empfangen doppelt so viel aus ihrem Anteil der Bandbreite für ihre Kommunikation herauszuholen. Was müssen sie als erstes in Erfahrung bringen? Vielleicht ist Ihnen das folgende Beispiel eine Hilfe: Sparsi hat 0110 1100 gesendet und als Summe 1010 0001 empfangen. Er fragt sich: Was hat Knausri an mich gesendet? Wie lautet die Antwort, wenn das überlagernde Senden modulo 2 stattfindet, wie bei modulo 256?

{Wir unterstellen, daß das Mehrfachzugriffsverfahren Sparsi und Knausri keine Schwierigkeiten bereitet. Beispielsweise möge es erlauben, daß einer von beiden einen (Simplex-)Kanal reserviert, den beide dann für paarweises überlagerndes Empfangen nutzen.}

c) Globales überlagerndes Empfangen: Kollisionsauflösungsalgorithmus mit Mittelwertbildung

Die Teilnehmerstationen 1 bis 5 senden die Nachrichten 5, 13, 9, 11, 3. Wie erfolgt die Kollisionsauflösung mit Mittelwertbildung und überlagerndem Empfangen? Das lokale Entscheidungskriterium der Teilnehmerstationen sei $Informationseinheit \leq \lfloor Ø \rfloor$. Benutzen Sie die Notation von Bild 5-14 des Skripts.

Wie erfolgt sie, wenn die Teilnehmerstationen 5, 13, 11, 11, 3 senden?

5-8 Schlüssel-, Überlagerungs- und Übertragungstopologie beim Überlagernden Senden

a) Warum kann die Schlüsseltopologie unabhängig von Überlagerungs- und Übertragungstopologie festlegt werden?
b) Warum sollten Überlagerungs- und Übertragungstopologie aufeinander abgestimmt werden?

5-9 Abstimmung der Überlagerungs- und Übertragungsalphabete beim Überlagernden Senden

Mittels welchem „Trick“ konnte ein sehr großes Überlagerungsalphabet (nützlich etwa für Kollisionsauflösungsalgorithmus mit Mittelwertbildung) und eine minimale Verzögerungszeit, d.h. ein minimales Übertragungsalphabet, gleichzeitig erreicht werden?

5-10 Vergleich „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ und „Überlagerndes Senden“

a) Vergleichen Sie die Senderanonymitäts-Konzepte „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ und „Überlagerndes Senden“ bzgl. Stärke der berücksichtigten Angreifer, Aufwand, Flexibilität. Was ist Ihrer Meinung nach das elegantere, schönere Konzept? Warum?
b) Decken die Konzepte auch Empfängeranonymität ab? Wie ist ggf. das Zusammenspiel zwischen Sender- und Empfängeranonymität?

5-11 Vergleich „Überlagerndes Senden“ und „umcodierende MIXe“

Vergleichen Sie die Konzepte „Überlagerndes Senden“ und „umcodierende MIXe“ bzgl. Schutzziel, Stärke der berücksichtigten Angreifer, Aufwand, Flexibilität. Was ist Ihrer Meinung nach das praktikablere Konzept? In welcher Hinsicht und warum?

5-12 Reihenfolge der Grundfunktionen von MIXen

Inwieweit ist die Reihenfolge der Grundfunktionen eines MIXes in Bild 5-22 zwangsläufig?

5-13 Reicht Ausgabenachrichten von MIXen gleichlang?

Was halten Sie von folgendem Vorschlag (gemäß der Maxime des Altbundeskanzlers Dr. Helmut Kohl: Entscheidend ist, was hinten rauskommt): MIXe können auch Eingabenachrichten unterschiedlicher Länge zusammen mixen. Hauptsache, alle zugehörigen Ausgabenachrichten haben gleiche Länge.

5-14 Keine Zufallszahlen --> MIXe sind überbrückbar

Erläutern Sie, wie ein Angreifer MIXe überbrücken kann, wenn das direkte Umcodierungsschema für Senderanonymität (§5.4.6.2) ohne Zufallszahlen verwendet wird.

5-15 Individuelle Wählbarkeit des symmetrischen Konzelationssystems bei MIXen?

MIXe veröffentlichen ihren öffentlichen Chiffrierschlüssel und legen damit jeweils auch das zugehörige asymmetrische Konzelationssystem fest. In §5.4.6.1 (symmetrische Konzelation durch ersten und/oder letzten MIX) und §5.4.6.3 (indirektes Umcodierungsschema) werden zwei Anwendungen von symmetrischer Konzelation bei MIXen vorgestellt. Sollte dort jeweils jeder

Benutzer des MIX-Netzes das verwendete symmetrische Konzelationssystem frei wählen können oder sollte es der MIX jeweils vorgeben?

5-15a Weniger Speicheraufwand beim Generierer von anonymen Rückadressen

In §5.4.6.3 ist beschrieben, wie anonyme Rückadressen gebildet werden können. Aufwendig ist hierbei für den Generierer, daß er sich jeweils unter dem eindeutigen Namen *e* der Rückadresse alle symmetrischen Schlüssel solange merken muß, bis die Rückadresse verwendet wird. Dies kann sehr lange sein und ist deshalb umständlich. Überlegen Sie sich, wie es der Generierer von Rückadressen vermeiden kann, sich symmetrische Schlüssel, ja selbst eindeutige Namen merken zu müssen.

5-16 Warum längentreue Umcodierung?

Erläutern Sie, wie ein Angreifer die MIXe in Bild 5-26 unten überbrücken kann. (Es ist trivial, aber drücken Sie es ruhig trotzdem mal präzise aus.)

5-17 Minimal längenexpandierendes längentreues Umcodierungsschema für MIXe

a) Beschreiben Sie kurz den Kommunikations- und Verschlüsselungsaufwand des MIX-Netzes.

b) Sie haben ein gegen adaptive aktive Angriffe sicheres asymmetrisches Konzelationssystem und ein effizienteres symmetrisches Konzelationssystem zur Verfügung. Beide arbeiten (nach der Schlüsselwahl) auf Blöcken fester Länge. Entwerfen Sie ein beim Sender die Nutznachricht minimal längenexpandierendes und während den Umcodierungsschritten der MIXe längentreues Umcodierungsschema für MIXe.

Leiten Sie der Einfachheit halber das minimal längenexpandierende Umcodierungsschema aus dem in §5.4.6.5 für symmetrische Konzelationssysteme mit den Eigenschaften $k^{-1}(k(x)) = x$ und $k(k^{-1}(x)) = x$, d.h. nicht nur Ver- und Entschlüsselung, sondern auch Ent- und Verschlüsselung sind zueinander invers, angegebenen her, vgl. Bild 5-31.

5-18 Brechen der direkten RSA-Implementierung von MIXen

Was ist die *Kernidee* des in §5.4.6.8 beschriebenen Angriffs auf die direkte RSA-Implementierung von MIXen?

5-19 Wozu MIX-Kanäle?

a) Warum und wozu sind MIX-Kanäle nötig?

b) Was begrenzt ihren Einsatz bzw. führt zur „Weiterentwicklung“ der Zeitscheibenkanäle?

5-19a Freie MIX-Reihenfolge bei fester Anzahl benutzter MIXe?

Ronny Freeroute und Andi Kaskadius streiten mal wieder darüber, ob freie Wahl der MIXe und ihrer Benutzungsreihenfolge vorzuziehen ist – oder stattdessen sogenannte MIX-Kaskaden, d.h. MIXe und ihre Reihenfolge sind fest vorgegeben. Entgegen dem üblichen Diskussionsschema

Ronny Freeroute: Frei wählbare MIX-Reihenfolgen sind besser, da so jede(r) in seiner Wahl frei ist (Vertrauen, Verwendbarkeit von MIXen am Wege) und, wenn nur wenige MIXe angreifen, größere Anonymitätsgruppen entstehen.

Andi Kaskadius: MIX-Kaskaden sind vorzuziehen, da so bei maximal vielen angreifenden MIXen die größtmögliche Anonymitätsgruppe entsteht sowie Durchschnitte von Anonymitätsgruppen leer sind oder aber immer die ganze Anonymitätsgruppe enthalten

sagt eines Tages ein blasser Ronny: „Was ist, wenn der Angreifer maximal viele MIXe kontrolliert und weiß, daß jede(r) bei freier Wahl der MIX-Reihenfolge eine feste Anzahl MIXe benutzt. Dann hat der Angreifer gute Chancen, auch den nichtangreifenden MIX zu überbrücken.“ Was meint Ronny, d.h. wie könnte ein einfacher Angriff auf ein MIX-Netz mit freier Routenwahl und von Teilnehmern fest gewählter Routenlänge aussehen? Und was würden Sie dagegen tun?

5-20 Wie sichern, daß MIXe Nachrichten von genügend vielen unterschiedlichen Sendern erhalten?

Wie zu Beginn von §5.4.6.1 begründet, müssen MIXe in jedem Schub Informationseinheiten von genügend vielen Sendern bearbeiten – sind sie von zu wenigen, ist die Gefahr zu groß, daß einer von ihnen von den wenigen andern auch über die MIXe beobachtet werden kann.

Wie kann das Einhalten der Forderung „Informationseinheiten von genügend vielen Sendern“ beim

a) ersten MIX einer MIX-Kaskade, wie bei
b) hinteren, d.h. nicht ersten, MIXen einer MIX-Kaskade

gesichert werden? Andernfalls könnte beispielsweise der erste MIX in seinem Ausgabeschub alle Informationseinheiten bis auf eine durch von ihm selbst erzeugte Informationseinheiten ersetzen und so die nicht ersetzte Informationseinheit trotz aller weiteren MIXe verfolgen.

5-21 Koordinations-Protokoll bei MIXen: Wozu und wo?

Bei welchen Verschlüsselungsschemata ist ein Koordinations-Protokoll zwischen MIXen notwendig? Begründen Sie Ihre Antwort.

5-22 Grenzziehung der Verantwortungsbereiche – Funktionsumfang der Netzabschlüsse

Warum sollten sich die nötigen Vertrauensbereiche von Teilnehmer und Netzbetreiber möglichst wenig überlappen? Erläutern Sie dies für alle vorgestellten Schutzmaßnahmen in Kommunikationsnetzen (Bild 5-48 ist Ihnen dabei vielleicht eine Hilfe).

5-26 Mehrseitig sicherer Web-Zugriff

Machen Sie sich Gedanken über mehrseitig sicheren Web-Zugriff: Welche Sicherheitsprobleme bestehen? Genauer: Welcher Beteiligte hat welche Schutzinteressen? Wenden Sie danach die bisher vermittelten Konzepte auf diese neue Problemstellung an. {Mit dieser größer angelegten Aufgabe soll Ihre Neugierde geweckt und das Vermittelte nochmals etwas geübt werden. Es ist deshalb ok, wenn Sie sich nach der Problemanalyse und ersten eigenen Überlegungen im Web umschauen, welche Konzepte es so gibt. Und auch ich werde mir erlauben, Sie in der Lösung auf Literatur zu verweisen.}

5-27 Firewalls

Viele Firmen und Behörden kontrollieren der Verkehr zwischen den öffentlichen Netzen, beispielsweise dem *Internet*, und ihrem firmen- bzw. behördeninternen Netz, beispielsweise ihrem LAN oder *Intranet*, durch sogenannte Firewalls (der Name ist dem Brandschutz in Gebäuden entlehnt und bedeutet dort Brandschutzmauer, d.h. eine Mauer, die die Ausbreitung von Bränden verhindern oder zumindest verzögern soll). Halten Sie dies für sinnvoll? Was genau bringt das? Wo sind die Grenzen? Kann das Zugangskontrolle und Zugriffskontrolle zu bzw. in den Arbeitsplatzrechnern ersetzen?

Aufgaben zu Werteaustausch und Zahlungssysteme

6-1 Electronic Banking – Komfortversion

a) Was halten Sie von folgender Papierbanking-Komfortversion: Der Kunde erhält von der Bank fertige Formulare, in die nur noch der Betrag einzusetzen ist. Insbesondere wird dem Kunden die Mühe erspart, eine Unterschrift zu leisten.
b) Unterscheidet sich diese Papierbanking-Komfortversion von der heute üblichen Electronic Banking Version?
c) Ändert sich dies, wenn Kunden von ihrer Bank je eine Chipkarte mit fertig geladenen Schlüsseln für MACs oder digitale Signaturen erhalten?

6-2 Personenbezug und Verkettbarkeit von Pseudonymen

Warum bieten Rollenpseudonyme mehr Anonymität als Personenpseudonyme und Transaktionspseudonyme mehr Anonymität als Geschäftsbeziehungspseudonyme?

6-3 Eigenauthentikation vs. Fremdauthentikation

Erläutern Sie Eigenauthentikation und Fremdauthentikation sowie den Unterschied zwischen ihnen. Geben Sie typische Beispiele an. Warum ist die Unterscheidung zwischen Eigenauthentikation und Fremdauthentikation für die Gestaltung mehrseitig sicherer IT-Systeme wichtig?

6-4 Diskussion der Sicherheitseigenschaften von digitalen Zahlungssystemen

Die in §6.4 genannten Sicherheitseigenschaften berücksichtigen nur Verfügbarkeit und Integrität von Zahlungsvorgängen, also nur das absolut Notwendige. Erstellen Sie eine Liste der aus Ihrer Sicht notwendigen Schutzziele, geordnet nach Vertraulichkeit, Integrität und Verfügbarkeit.

6-5 Wie weit erfüllen konkrete digitale Zahlungssysteme Ihre Schutzziele?

Prüfen Sie Ihnen bekannte digitale Zahlungssysteme gegen die von Ihnen in Aufgabe 6-4 erarbeiteten Schutzziele. Beispiele könnten

Telefonbanking und

Homebanking Computer Interface (HBCI) http://www.bdb.de/Verband/Intern.htm#Punkt2

sein.

Aufgaben zu Regulierbarkeit von Sicherheitstechnologien

9-1 Digitale Signaturen bei „Symmetrische Authentikation ermöglicht Konzelation“

Können in dem Verfahren „Symmetrische Authentikation ermöglicht Konzelation“ (§9.4) digitale Signaturen statt symmetrischer Authentikationscodes eingesetzt werden? Wenn ja, wie? Wenn nein, warum nicht?

9-2 „Symmetrische Authentikation ermöglicht Konzelation“ vs. Steganographie

Was ist „Symmetrische Authentikation ermöglicht Konzelation“ (§9.4) und Steganographie (§4) gemeinsam? Was unterscheidet sie?

9-3 Kryptoregulierung durch Verbot frei programmierbarer Rechner?

Aufgrund der Verfahren aus §9.1 - §9.6 ist klar, daß eine Kryptoregulierung nicht greift, solange frei programmierbare Rechner verfügbar sind. Nehmen wir an, *alle* Staaten einigten sich darauf, die Produktion von frei programmierbaren Rechnern zu verbieten und es gelänge sogar, dieses Verbot durchzusetzen. Nehmen wir an, die Jahrzehnte, in denen bereits verkaufte frei programmierbare Rechner noch funktionieren, wären verstrichen – es gibt also keine frei programmierbare Rechner mehr. Ist dann eine Kryptoregulierung durchsetzbar? Noch genauer gefragt: Ist dann eine Kryptoregulierung durchsetzbar, die es erlaubt, ab einer Überwachungsgenehmigung den E-mail-Verkehr eines Verdächtigen zu entschlüsseln – aber nicht vorher?

Lösungen

Lösungen zur Einführung

1-1 Zusammenhang zwischen räumlicher Verteilung und Verteilung bzgl. Kontroll- und Implementierungsstruktur

Räumlich verteilte IT-Systeme erlauben üblicherweise Eingaben an mehreren Stellen. Diese Eingaben haben (werden sie nicht ignoriert) Auswirkungen auf den Systemzustand. Wollte man eine zentrale Kontrollstruktur mittels einer Instanz mit globaler Systemsicht realisieren, müßte ihr jede Eingabe mitgeteilt und erst dann der Zustandsübergang vollzogen werden. Da die Geschwindigkeit der Übermittlung von Information durch die Lichtgeschwindigkeit (etwa 300000 km/s) begrenzt ist und heutige Rechner Verarbeitungsgeschwindigkeiten jenseits von 10^7 Befehl/s erreichen, wird bereits ab

$$\frac{300000 \text{ km/s}}{10^7 \text{ Befehl/s}} = 30 \text{ m/Befehl},$$

also einer Übertragungsentfernung von nur 30 m, potentiell mehr Zeit für die Informationsübertragung als für die Informationsverarbeitung benötigt.[140] Da die Verarbeitungsgeschwindigkeit weiter steigt (wozu bekanntlich die Strukturgrößen kleiner werden), es sich bei der Lichtgeschwindigkeit aber um eine Naturkonstante handelt, wird dieses Argument immer wichtiger.

Physische Wartung ist in einem räumlich verteilten IT-System besonders aufwendig, insbesondere wenn sie direkt nach einem Ausfall erfolgen muß. Bezüglich ihrer Kontroll- und Implementierungsstruktur verteilt realisierte IT-Systeme bieten, da bei Ausfall einzelner Subsysteme nicht das ganze IT-System ausfällt, die Möglichkeit, Wartungsmaßnahmen zu einem geeigneten Zeitpunkt später durchzuführen.

Weitere Argumente finden Sie bei der Lösung der Aufgabe 1-2.

1-2 Vor- und Nachteile eines *verteilten* Systems bzgl. Sicherheit

a) Vorteile:

Vertraulichkeit kann dadurch unterstützt werden, daß jeder seine Daten in seiner physischen Verfügungsgewalt behalten kann – Ende der informationellen Fremdbestimmung: Es ist nicht mehr nötig, die eigenen Daten wegzugeben und auf das Beste zu hoffen.

Integrität kann dadurch unterstützt werden, daß jeder seine Daten vor jedem Zugriff anderer selbst vorgelegt bekommt und ihre Richtigkeit bestätigen oder ihre Falschheit belegen kann. Dieses Vorlegen muß nicht unbedingt dem Menschen gegenüber geschehen, sein PC kann so programmiert werden, daß er dies stellvertretend leistet. Außerdem kann Protokollierung verteilt erfolgen, so daß eine Fälschung oder Unterdrückung der Protokollierung erschwert wird.

Verfügbarkeit kann dadurch unterstützt werden, daß es z.B. für Sprengstoffanschläge oder logische Bomben keine zentralen und alle Daten gefährdenden Angriffspunkte mehr gibt.

Nachteile:

---

140 Hier kann mensch einwenden, es wären nicht die Befehle der Rechner, sondern die Befehle ihrer Benutzer, der Menschen, zu betrachten, wodurch sich auch bei hektischer Bedienung ein deutlich anderes Bild ergäbe. Aus meiner Sicht ist dies aber nicht angemessen: Da es nicht darum geht, ob aus der Sicht der Benutzer es sich um ein verteiltes System handelt, sondern um die Implementierungsstruktur, also mindestens eine Betrachtungsebene tiefer, müssen schon Maschinenoperationen als Einheiten genommen werden.

Ein weiterer Einwand könnte nun sein, daß nicht Maschinenbefehle, sondern etwa Plattenzugriffe als Operationen genommen werden müssen, da erst sie den Zustandswechsel persistent (dauerhaft) machen und vorher aus Gründen der Fehlertoleranz sowieso rückgesetzt werden können muß. Dies ist zwar eine richtige Sicht bzgl. der Systemgestaltung – nur wurde dabei genau eine verteilte Kontroll- und Implementierungsstruktur einzelner Rechner bzw. Prozesse eingeführt.

Vertraulichkeit kann dadurch gefährdet werden, daß Daten weitgehend frei bewegt werden können, was eine Kontrolle erschwert und sogar in Frage stellt, daß es zu jedem Zeitpunkt eine juristisch verantwortliche Instanz, im Datenschutzrecht „speichernde Stelle“ genannt, gibt. *Integrität* und *Verfügbarkeit* können dadurch gefährdet werden, daß für Fehler und Angriffe mehr Ansatzpunkte bestehen. Man denke an Verbindungsleitungen oder gar Funkstrecken, die physisch schwer zu schützen sind, oder an die vielen Rechner, die größtenteils an nicht überwachten Plätzen stehen.

b) Räumliche Verteilung unterstützt besonders die Verfügbarkeit trotz physischer Gewalt. Verteilung bzgl. Kontroll- und Implementierungsstruktur unterstützt besonders Vertraulichkeit und Integrität.

1-3 Vor- und Nachteile eines *offenen* Systems bzgl. Sicherheit

a) Vorteile:
Offene Systeme können leichter zu kooperierenden verteilten Systemen verbunden werden. Dadurch können die oben ausgeführten Vorteile eines verteilten Systems bzgl. Sicherheit leichter erreicht werden. Insbesondere wird durch das kompatible äußere Verhalten offener (Sub-)Systeme der Aufbau eines Gesamtsystems aus bzgl. Entwerfer, Produzent und Wartungsdienst diversitären (Sub-)Systemen erleichtert. Wie in §1.2.2 erläutert, kann Diversität die Sicherheit stark erhöhen.
Nachteile:
Offene Systeme können leichter zu kooperierenden verteilten Systemen verbunden werden. Dadurch können die oben ausgeführten Nachteile eines verteilten Systems bzgl. Sicherheit leichter erreicht werden. Insbesondere wird ein Benutzer eines offenen Rechnernetzes nicht wissen, wer alles in welcher Rolle an diesem offenen System welche Rechte (oder auch illegale Möglichkeiten) hat.

b) Sie verstärken sich gegenseitig, d.h. alles für verteilte Systeme oder offene Systeme Gesagte gilt für verteilte offene Systeme besonders.

1-4 Vor- und Nachteile eines *diensteintegrierenden* Systems bzgl. Sicherheit

a) Vorteile:
Diensteintegrierende Systeme dürfen, da ihre Kosten auf mehrere Dienste verteilt werden können, mehr kosten. Dies erlaubt relativ mehr Aufwand bzgl. Sicherheit. Dies kann einerseits einen sorgfältigeren Entwurf und dessen genauere Prüfung (z.B. auf Trojanische Pferde) ermöglichen. Andererseits kann mit mehr Redundanz zur Steigerung der Verfügbarkeit gearbeitet werden, etwa zusätzlichen alternativen Übertragungsleitungen oder gar Funkstrecken in Kommunikationsnetzen. Auch sind die ziemlich aufwendigen Verfahren zum Schutz der Anonymität von Sender und/oder Empfänger, vgl. §5, möglicherweise eher realisierbar.
Nachteile:
Wird in einem diensteintegrierenden System eine Sicherheitseigenschaft verletzt, so dürfte dies gleich mehrere Dienste betreffen. Der Schaden ist also besonders groß. Dies ist katastrophal, wenn es keine Ersatzsysteme mehr gibt, es sich also um ein total diensteintegrierendes (= alternativloses) System handelt, vgl. [RWHP_89, Pfit_89 Seite 213ff].
Zugang zum diensteintegrierenden System muß grundsätzlich auch Teilnehmern gestattet sein, die nur einen Teil der angebotenen Dienste nutzen. Für die nichtgenutzten Dienste stellen diese Teilnehmer dann ein unnötiges Sicherheitsrisiko dar.

b) Diensteintegration eskaliert unabhängig vom Systemtyp alles Gesagte, insbesondere auch alles über verteilte offene Systeme Gesagte.

1-5 Vor- und Nachteile eines *digitalen* Systems bzgl. Sicherheit

a) Digital übertragene/gespeicherte Information kann, wenn ein Angreifer einen gewissen Minimalaufwand zu treiben gewillt ist, perfekt (= verlustlos und ohne Spuren zu hinterlassen) kopiert und spurlos geändert werden. Dies eskaliert also Probleme bzgl. Vertraulichkeit und Integrität, sofern nicht die in §3 beschriebenen, gerade durch digitale Übertragung und Speicherung ermöglichten Verschlüsselungsmaßnahmen ergriffen werden. Verfügbarkeit wird durch digitale Übertragung/Speicherung im allgemeinen unterstützt.
Rechnergesteuerte Vermittlung und rechnergestützte Verarbeitung werfen einerseits, u.a. wegen der Bedrohung durch Trojanische Pferde, besondere Probleme bzgl. Vertraulichkeit, Integrität und Verfügbarkeit auf. Andererseits ermöglicht ihre große Leistungsfähigkeit andernfalls nicht durchführbare Schutzmaßnahmen, so daß bei geeigneter Systemgestaltung dieser Nachteil mehr als aufgewogen wird. Ist man sich der Korrektheit und Vollständigkeit der Schutzmaßnahmen sicher, sollten sie (und soweit bzgl. der Nutzfunktionen gleiches gegeben ist auch sie) lieber in Hardware als frei speicherprogrammierbar realisiert werden. Merke: Softwaregestützter Schutz ist meist soft!

b) Aus den unter a) angeführten Gründen unterstützt die Eigenschaft digital die Konstruktion verteilter Systeme.
Solange ein offenes System nicht vollständig, insbesondere abschließend, spezifiziert ist, ist rechnergesteuerte Vermittlung und/oder rechnergestützte Verarbeitung (beides frei programmierbar) notwendig, um die bestehenden Systeme ohne nennenswerten physischen Aufwand zukünftigen Entwicklungen (Spezifikationen) anpassen zu können.
Aus Aufwandsgründen müssen beim heutigen Stand der Technik diensteintegrierende Systeme digitale Systeme sein.

1-6 Anforderungsdefinition für vier beispielhafte Anwendungen im medizinischen Bereich

a) *Verfügbarkeit*: mittelkritisch, solange es noch lokale Papieroriginale gibt, denn selbst bei Ausfall ist die Situation genau wie heute.
Integrität: kritisch, denn niemand will auf Dauer zusätzlich in den Papierdokumenten nachschauen. Ebenso wird sie niemand von Hand aktualisieren oder auch nur regelmäßig ausdrucken wollen.
Vertraulichkeit gegen fremde Ärzte: kritisch; gegen Außenstehende: extrem kritisch.
Dem Patienten sind ggf. Berichtigungs- und Löschrechte einzuräumen, deren internationale Harmonisierung und Durchsetzung schwierig sein dürfte.

b) *Verfügbarkeit*: unkritisch, solange das Operationsteam vor Ort auf den Rat nicht angewiesen ist, andernfalls extrem kritisch.
Integrität: Kleine Integritätsverletzungen, z.B. einzelne Pixel falsch, sind unkritisch, größere Verletzungen, z.B. völlig falsche Bilder, wären kritisch. Damit der Patient (oder seine Rechtsnachfolger!) ärztliche Kunstfehler nachweisen können, muß auch nach Jahren noch nachgewiesen werden können, wer was gesehen und folglich welchen Rat gegeben hat. Insbesondere muß hierzu eine gerichtsverwertbare Identifikation (vgl. §2.2.1) des externen Experten erfolgen.
Vertraulichkeit: Vertraulichkeit der Bilder: wenig kritisch; wird zu Beginn die Krankengeschichte übertragen, muß der Nachrichteninhalt natürlich vor unbefugter Kenntnisnahme geschützt werden. Der Schutz des Senders oder Empfängers ist überflüssig.
Ausführlicher in [PfPf_92 §4.2].

c) *Verfügbarkeit*: mittelkritisch, da Versagen „nur“ zur heutigen Situation führt. Sofern Patienten zu Hause gelassen werden, die andernfalls im Krankenhaus wären: kritisch.

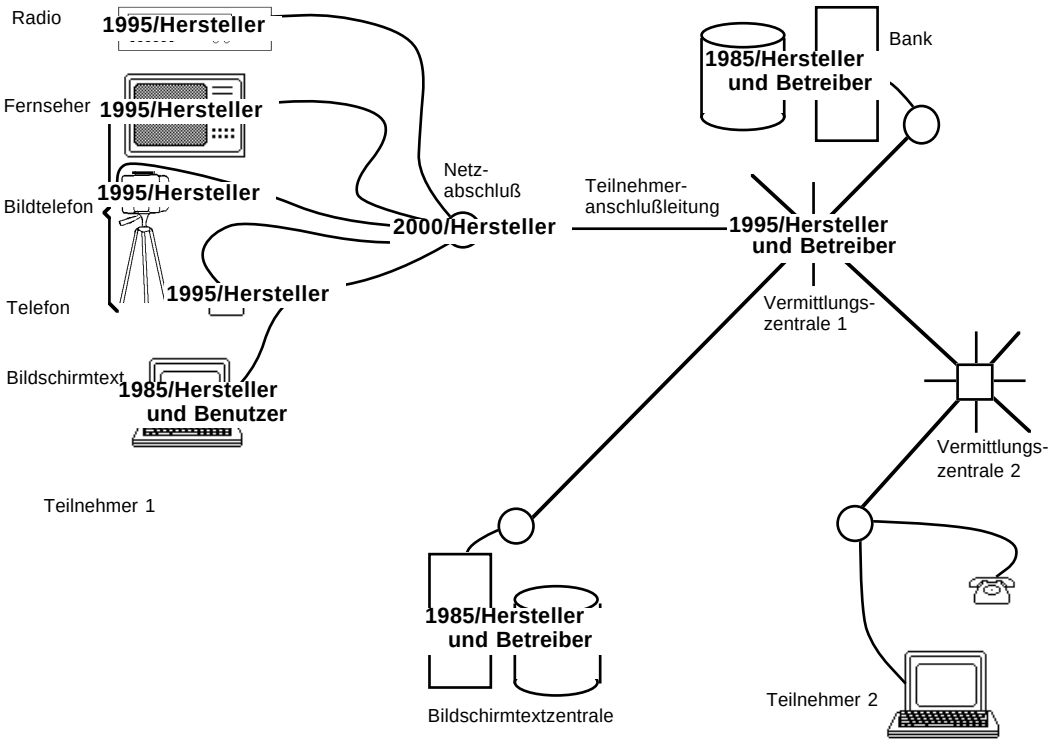
Integrität: falscher Alarm ist unkritisch, sollte aber nicht zu oft ausgelöst werden. Das Gegenteil: siehe unter Verfügbarkeit;
Vertraulichkeit eines Alarms ist wenig kritisch, alle anderen Nachrichteninhalte und auch die Patienten als Sender/Empfänger müssen geschützt werden.
Ausführlicher in [PfPf_92 §4.3].

d) *Verfügbarkeit*: unkritisch;
Integrität: nicht sehr kritisch, da in kritischen Fällen sowieso ein Arzt in die Behandlung eingeschaltet werden muß. In jedem Fall sollte beim Zugriff eine Identifikation der Faktendatenbank erfolgen und geklärt sein, wer für den Inhalt der Faktendatenbank verantwortlich ist.
Werden Ärzte in ihrem Urteil von solch einer Faktendatenbank abhängig, steigen die Anforderungen an die Verfügbarkeit und Integrität erheblich.
Vertraulichkeit des Nachrichteninhalts und selbst der Abfrage müssen durch Anonymität gesichert werden.
Ausführlicher in [PfPf_92 §4.1].

Natürlich bin ich mir der Vollständigkeit meiner Schutzziele nicht sicher!

1-7 Wo sind ab wann Rechner zu erwarten?

Die Zahlen geben das Jahr an, in dem eine nennenswerte Verbreitung der rechnergestützten Komponenten (jeweils innerhalb ihrer Komponentenklasse) erfolgte oder für das sie prognostiziert wird. In nächster Zukunft müssen also auch Vermittlungszentralen, „harmlose“ Teilnehmerendgeräte und später auch Netzabschlüsse als mögliche Angreifer betrachtet werden.



1-8 Defizite juristischer Regelungen

Die Antwort besteht aus einer Kombination der Argumente von §1.2.3 und §1.2.5:

Ein Verbot ist nur dann wirkungsvoll, wenn seine Einhaltung mit angemessenem Aufwand *überprüft* (bei beobachtenden Angriffen prinzipiell nicht möglich) und durch Strafverfolgung gesichert und der ursprüngliche Zustand durch Schadensersatz *wiederhergestellt* werden kann. Beides ist bei IT-Systemen leider nicht gegeben.

„Datendiebstahl“ allgemein, speziell das direkte Abhören von Leitungen oder Kopieren von Daten aus Rechnern, ist kaum feststellbar, da sich an den Originaldaten nichts ändert. Ebenso ist, wie oben erwähnt, das Installieren Trojanischer Pferde kaum festzustellen und erst recht nicht die unerlaubte Weiterverarbeitung von Daten, die man legal (oder auch illegal) erhalten hat.

Die Wiederherstellung des ursprünglichen Zustands müßte bei „Datendiebstahl“ vor allem darin bestehen, alle entstandenen Daten zu löschen. Man ist aber nie sicher, ob nicht noch weitere Kopien existieren. Außerdem können sich Daten im Gedächtnis von Menschen festsetzen, wo das Löschen besser nicht angestrebt werden sollte.

Es sei erwähnt, daß zur Überprüfung der Einhaltung von Datenschutzbestimmungen man zumindest bzgl. des Vertraulichkeitsaspektes gegenüber der prüfenden Instanz (etwa einem Datenschutzbeauftragten bzw. seinen Mitarbeitern) Ausnahmen zulassen muß.

Dual zum Problem des „Datendiebstahls“ gibt es das Problem des Datenverlustes: Verlorene Daten sind evtl. nicht wiederzubeschaffen, so daß auch hier der ursprüngliche Zustand nicht wiederhergestellt werden kann.

Es sind also zusätzliche wirkungsvollere Maßnahmen nötig.

Anmerkung: Für ein nationale Grenzen überschreitendes IT-System ist ein internationaler Konsens (etwa ausgedrückt durch Recht) und seine internationale Durchsetzung (etwa durch internationale Gerichtsbarkeit und Exekutivgewalt) nötig. Während es nationale Grenzen überschreitende IT-Systeme in Fülle gibt, liegen internationaler Konsens und internationale Durchsetzung im argen.

1-9 Alternative Definitionen/Charakterisierungen mehrseitiger Sicherheit

The large-scale automated transaction systems of the near future can be designed to protect the privacy and maintain the security of both individuals and organizations. ... Individuals stand to gain in increased convenience and reliability; improved protection against abuses by other individuals and by organizations, a kind of equal parity with organizations, and, of course, monitorability and control over how information about themselves is used. ... the advantages to individuals considered above apply in part to organizations as well. ... Since mutually trusted and tamper-resistant equipment is not required with the systems described here, any entry point to a system can be freely used; users can even supply their own terminal equipment and take advantage of the latest technology. [Cha8_85]

Meanwhile, in a system based on representatives and observers, organizations stand to gain competitive and political advantages from increased public confidence (in addition to the lower costs of pseudonymous record-keeping). And individuals, by maintaining their own cryptographically guaranteed records and making only necessary disclosures, will be able to protect their privacy without infringing on the legitimate needs of those with whom they do business. [Chau_92]

Ausgehend von der zunehmenden Bedeutung der Abwicklung von Rechtsgeschäften über offene digitale Systeme wird die Forderung abgeleitet, diese Systeme so zu gestalten, daß ihre Benutzung unbeobachtbar durch Unbeteiligte und anonym vor Beteiligten stattfinden, aber dennoch die notwendige Rechtssicherheit garantiert werden kann. [PWP_90]

The term multilateral security is therefore used here to describe an approach aiming at a balance between the different security requirements of different parties. In particular, respecting the different security requirements of the parties involved implies renouncing the commonly used precondition, that the parties have to trust each other, and, especially, renouncing the precondition, that the subscribers have to place complete trust into the service providers. Consequently, each party must be viewed as a potential attacker on the other and the safeguards have to be designed accordingly. [RDFR_97]

Mehrseitige Sicherheit bedeutet die Berücksichtigung der Sicherheitsanforderungen aller beteiligten Parteien. [Rann_97]

Mehrseitige Sicherheit bedeutet die Einbeziehung der Schutzinteressen *aller* Beteiligten sowie das Austragen daraus resultierender Schutzkonflikte beim Entstehen einer Kommunikationsverbindung. [FePf_97]

Mehrseitige Sicherheit bedeutet die Einbeziehung der Schutzinteressen *aller* Beteiligten sowie das Austragen daraus resultierender Schutzkonflikte beim Entstehen einer Kommunikationsverbindung. Die Realisierung von mehrseitiger Sicherheit führt nicht zwangsläufig dazu, daß die Interessen aller Beteiligten erfüllt werden. Möglicherweise offenbart sie sogar gegensätzliche, unvereinbare Interessen, die den Beteiligten bisher nicht bewußt waren, da Schutzziele explizit formuliert werden. Sie führt jedoch zwangsläufig dazu, daß die Partner einer mehrseitig sicheren Kommunikationsbeziehung in einem ausgewogenen Kräfteverhältnis bzgl. Sicherheit miteinander interagieren. [Fede_98]

Mehrseitige Sicherheit ist *Sicherheit mit minimalen Annahmen über andere*:

- Jeder Beteiligte hat Sicherheitsinteressen.
- Jeder Beteiligte kann seine Interessen formulieren.
- Konflikte werden erkannt und Lösungen ausgehandelt.
- Jeder Beteiligte kann seine Sicherheitsinteressen in den ausgehandelten Lösungen durchsetzen. [FePf_99]

... new developments, e.g. the rising need and demand for multilateral security, which cover the requirements of users and usees as well as those of IT system owners and operators. ...
The TCSEC had a strong bias on the protection of system owners and operators only. This bias is slowly losing strength, but can still be seen in all criteria published afterwards. Security of users and usees, especially of users of telecommunication systems is not considered. Therefore techniques, providing bi- or multilateral security, e.g. those protecting users in a way privacy regulations demand it, cannot be described properly using the current criteria. [Rann_94]

Mehrseitige Sicherheit in der Kommunikationstechnik betont den gleichwertigen Schutz aller an einer Kommunikation beteiligten Partner. Die Schutzziele der Privatheit und der Verbindlichkeit werden als wichtige Größen im Hinblick auf persönliche Entfaltung des Menschen und auf verbindliches soziales Handeln betrachtet. [PoSc_97]

Während in den klassischen Sicherheitsmodellen die Sicht des Netzbetreibers im Vordergrund steht, berücksichtigt die mehrseitige Sicht auch und gerade die Sicht der Benutzer. Neben dem Schutz, den die Netze ihren Benutzern bieten, werden die Benutzer auch Mechanismen zum Selbstschutz benötigen, um die Abhängigkeit von anderen zu reduzieren. Damit schützen sich Telekooperationspartner nicht nur gegen Angriffe aus dem Netz, sondern sie wahren auch ihre Interessen gegeneinander, so wie Käufer und Verkäufer Geld und Quittung austauschen, manchmal sogar unter notariellen Augen. [GGPS_97]

Mehrseitige Sicherheit in der Telekommunikation ist ein anwenderbezogenes Merkmal. Der geforderte Schutz gilt letztlich den einzelnen Menschen und Organisationen, die sich zur Bewältigung ihres beruflichen und privaten Alltags moderner Telekommunikationstechnik bedienen. Sicherheit dient aber nicht nur den Kommunikationspartnern selbst, sondern auch all denjenigen, die mit den Partnern oder mit dem jeweiligen Kommunikationsinhalt in Beziehung stehen oder mit der Bereitstellung der Kommunikationsmittel zu tun haben ... [RoSc_96]

Mehrseitige Sicherheit: Wo bisher bei der Einführung neuer Kommunikationsmedien und -dienste die *Sicherheit des Betreibers* (z.B. gegen unberechtigte Benutzung und Eintreibung der angefalle-

nen Kosten) im Vordergrund stand, soll jetzt auch die *Sicherheit des Benutzers* eines neuen Dienstes Beachtung finden. [BDFK_95]

Es erscheint durchaus legitim zu verlangen, daß Nutzer in der Wahrung ihrer Sicherheitsinteressen durch das System gleichberechtigt zu unterstützen sind. Wenn man allen Nutzern zubilligt, daß ihre Sicherheit nicht von der Gutwilligkeit anderer abhängt, sondern nur auf eigenem Verhalten aufbaut, kann man ein System, daß dieses Konzept verwirklicht, mehrseitig sicher nennen. [PSWW_96]

Mehrseitige Sicherheit bedeutet die Gewährleistung von Moral im informationstechnischen Sinn. [ein Student im WS 1999/2000]

Lösungen zu Sicherheit in einzelnen Rechnern und ihre Grenzen

2-1 Angreifer außerhalb der Weisheit der Schulbücher

Zunächst fällt einem ein *Angreifer unbeschränkter Kapazität* bzgl. Rechenleistung, Zeit und Speicher ein. Wir werden jedoch im folgenden sehen, daß man gegen solch einen komplexitätstheoretisch unbeschränkten Angreifer mit entsprechenden informationstheoretischen Verfahren sehr wohl Sicherheit (genauer: Vertraulichkeit und Integrität) erreichen kann.

Schwieriger tut man sich mit Angreifern, die Eigentümer unbekannter (technischer, physika–lischer, chemischer, etc.) Neuentwicklungen sind, z.B. Meßgeräte, Sprengstoffe, etc.

Ganz kritisch wird es bzgl. „Eigentümern“ bisher nicht bekannter physischer Phänomene: neue Arten von Strahlung, Feldern etc.; Oder auf den Menschen bezogen: Übermenschliche Sinneswahrnehmungen (Röntgenblick, Hellsehen) und Fähigkeiten (Durch-die-Wand-gehen-können, Unverletzbarkeit, Telekinese). Nicht zur Eröffnung einer parapsychologischen Diskussion, sondern als warnenden Hinweis, daß perfekte Sicherheit nur in einer bzgl. Erkenntnis *geschlossenen* Welt möglich ist, wir aber aufgrund unserer wachsenden Welterkenntnis von einer *offenen* Welt ausgehen müssen, sind die folgenden Beispiele gedacht:

Unbefugter Informationsgewinn: Gegen einen Angreifer mit hellseherischen Fähigkeiten hilft bzgl. Vertraulichkeit weder Schirmung, vgl. §2.1, noch Kryptographie, vgl. §3.

Unbefugte Modifikation von Informationen: Weder hilft gegen einen Angreifer mit telekinetischen Fähigkeiten (auf der passenden Strukturebene) bzgl. Integrität Schirmung, vgl. §2.1, noch hilft gegen einen Angreifer mit hellseherischen und telekinetischen Fähigkeiten Kryptographie, vgl. §3.

Unbefugte Beeinträchtigung der Funktionalität: Gegen einen Angreifer mit telekinetischen Fähigkeiten hilft bzgl. Verfügbarkeit weder Schirmung noch (endliche) Fehlertoleranz.

2-2 (Un)Abhängigkeit von Vertraulichkeit, Integrität, Verfügbarkeit am Beispiel Identifikation

Es sind keine „sicheren“ IT-Systeme zu erwarten, für die auch nur bzgl. eines der drei Schutzziele keinerlei Anforderungen bestehen:

- Bestehen keinerlei Anforderungen bzgl. *Verfügbarkeit*, dann braucht überhaupt kein IT-System realisiert zu werden. Damit sind dann alle seine (vorhandenen) Informationen integer und vertraulich.
- Bestehen keinerlei Anforderungen bzgl. *Integrität*, dann kann das IT-System Menschen nicht unterscheiden, da seine zu ihrer Unterscheidung nötige Information unbefugt modifiziert werden kann. Gleiches gilt für die Information zur Unterscheidung anderer IT-Systeme.
- Bestehen keinerlei Anforderungen bzgl. *Vertraulichkeit*, dann kann sich das IT-System anderen Instanzen gegenüber nicht als es selbst ausweisen, denn Angreifer könnten alle dafür nötige Information vom IT-System erhalten. (Die umgekehrte Argumentation geht

nicht so glatt durch: „Da das IT-System keinerlei Information vertraulich halten kann, kann es andere Instanzen nicht erkennen, denn Angreifer könnten alle Information erhalten, anhand derer das IT-System andere Instanzen unterscheiden will." Die Argumentation scheint zwar schlüssig zu sein, jedoch könnte das IT-System die Information, die es von der anderen Instanz erwartet, vor dem Abspeichern einer kollisionsresistenten Hashfunktion (vgl. §3.6.4 und §3.8.3) unterwerfen und nur das Ergebnis dieser Hashfunktion abspeichern und später mit dem entsprechend berechneten Wert der Eingabe vergleichen.)

2-3 Sende Zufallszahl, erwarte Verschlüsselung – geht's auch umgekehrt?

Der Unterschied ist subtil:

In der normalen Variante muß über die Zufallszahlenerzeugung nur angenommen werden, daß sich Zufallszahlen nicht (mit relevanter Wahrscheinlichkeit) wiederholen.

In der umgekehrten Variante muß *zusätzlich* angenommen werden, daß die Zufallszahlenerzeugung für Angreifer unvorhersagbar ist. Das sollte zwar bei Zufallszahlenerzeugung, wie der Name suggeriert, der Fall sein, aber Vorsicht ist die Mutter der Porzelankiste.

Der Unterschied wird an einem extremen Beispiel am deutlichsten: Die normale Variante ist sicher, wenn als Zufallsgenerator einfach ein Zähler genommen wird, der nach jeder „Zufallszahl" inkrementiert wird. Mit dieser Implementierung der Zufallszahlenerzeugung wäre die umgekehrte Variante des Protokolls vollkommen unsicher.

Für Interessierte: In [NeKe_99] finden Sie neben einer ausführlicheren Erklärung dieses Beispiels auch eine lesenswerte Einführung in einen Formalismus (BAN-Logik), mit dem solche Protokolle untersucht werden können.

2-4 Notwendigkeit der Protokollierung bei Zugriffskontrolle

a) Das Vertraulichkeitsproblem nimmt mit jeder Stufe der „Rekursion" ab, da die personenbezogenen Daten immer weniger (hoffentlich!) und auch weniger sensibel werden. Oder anders herum gesagt: Das Problem ist nicht wirklich rekursiv, da der Zugriff auf die Sekundärdaten natürlich wesentlich strengeren und anderen Sicherheitsbestimmungen unterliegen kann, als der Zugriff auf die Primärdaten. So könnte man z.B. die Sekundärdaten direkt ausdrucken und hätte damit die Möglichkeit der nachträglichen Manipulation durch räumlich entfernte Personen ausgeschlossen. Weiterhin ist es nicht sinnvoll, die Sekundärdaten zu ausführlich erstellen zu lassen, sie sollten also wirklich weniger werden. Schließlich kann der Personenkreis, der auf Sekundärdaten zugreifen *darf*, auf wenige besonders vertrauenswürdige Personen beschränkt werden. Hoffentlich beschränkt dies auch den Personenkreis, der zugreifen *kann*. Führt man Tertiärdaten, so kann man diese ggf. von anderen Datenschutzbeauftragten auswerten lassen als die Sekundärdaten. Dann kontrollieren sich die Kontrolleure gegenseitig. Unter der Annahme nicht manipulierbarer Protokolle könnte man so bei n Stufen ein zeitlich gestaffeltes $2n$-Augenprinzip (in Anlehnung an das übliche – zeitlich nicht gestaffelte – 4-Augen-Prinzip) realisieren.

b) Anstreben sollte man eine möglichst weitgehende Festlegung, wer unter welchen Umständen was darf, um so die Rechte im IT-System nicht zu großzügig einräumen zu müssen. Dann kann der Umfang der Protokollierung reduziert werden: So wenig Protokollierung wie möglich, aber so viel wie unbedingt nötig.

c) Hier nimmt die „Sensibilität" nicht mit jeder Stufe ab, sondern eher zu: Die Festlegung, wer Rechte vergeben darf, ist mindestens so sicherheitskritisch wie die Durchsetzung der Einhaltung der Rechte selbst. Gleiches wie bei a) gilt jedoch in der Hinsicht, daß die Vergabe von Rechten wesentlich strengeren und anderen Sicherheitsbestimmungen unterliegen kann, als die Nutzung von Rechten, die Vergabe der von Rechten zur Rechtevergabe wiederum abermals strengeren Sicherheitsbestimmungen, und so weiter und so fort.

2-5 Begrenzung des Erfolges verändernder Angriffe

Verändernde Angriffe sollten möglichst a) *schnell erkannt* und alle unbefugten Änderungen möglichst b) *präzise lokalisiert* werden. Danach sollte möglichst (weitgehend) c) der *Zustand* hergestellt werden, der *ohne den verändernden Angriff* vorliegen würde.

a) Neben allgemein anwendbaren informationstechnischen Maßnahmen wie
   - **fehlererkennende Codierung** bei Übertragung und Speicherung,
   - **Überprüfung der Authentizität aller Nachrichten** vor deren Bearbeitung (also Authentikationscodes oder digitale Signaturen, vgl. §3), damit gefälschte Kommandos in Nachrichten erst gar nicht ausgeführt werden, ggf. Gleiches bzgl. Speicherinhalten, die aus ungeschützten Bereichen wieder eingelesen werden, z.B. von Floppy Disks,

   gibt es für jede einzelne Anwendung spezifische Plausibilitätstests. Beispielsweise können sich Menschen in der Bundesrepublik bisher mit allerhöchstens 1000 km/h bewegen. Dies kann dazu verwendet werden zu entdecken, wenn sich ein Angreifer für jemand anderes ausgibt, der sich kurz vorher von woanders gemeldet hatte.

b) Hier hilft **Protokollierung**, wer wann worauf zugegriffen hat, vgl. §2.2.3.

c) Hier muß zwischen Hardware und Daten (inkl. Programme) unterschieden werden.

   Hardware muß entweder sehr schnell repariert oder gar neu beschafft werden können (Verträge mit den möglichen Lieferanten für Notfälle und eine Versicherung zur Deckung der Kosten sind hier eine große Hilfe) oder es muß Alternativen wie Backup-Rechenzentrum, alternatives Kommunikationsnetz etc. geben. Der Ersatz von Hardware ist also vor allem ein finanzielles Problem.

   Für **Daten (inkl. Programme)** werden **Backups** benötigt. Hier besteht folgendes Dilemma: Einerseits sollte ein möglichst aktueller Zustand wiederhergestellt werden, damit möglichst wenig Arbeit verloren geht. Andererseits sollte, wenn unbefugte Veränderungen nicht sofort erkannt wurden, genügend weit in die Vergangenheit zurückgesetzt werden, daß auf jeden Fall ein Zustand ohne die unbefugten Veränderungen erreicht wird. Was tun? Das Backup-System muß es erlauben, die Weite des Rücksetzens weitestgehend dynamisch festzulegen. Dies kann geschehen, indem beispielsweise täglich ein **komplettes Backup** angelegt wird, wöchentlich eins, monatlich eins und jährlich eins. Das tägliche Backup kann dann wöchentlich überschrieben werden, das wöchentliche monatlich und das jährliche bitte nie - so viele Backup-Streamertapes sollte sich Ihre Firma schon leisten können. Eine Alternative ist, nur hin und wieder ein komplettes Backup anzulegen, dafür aber ein **Logfile** zu schreiben, auf dem alle Änderungen (inkl. der genauen Werte) vermerkt sind. Anhand des aktuellen Zustands und des Logfiles kann dann beliebig in die Vergangenheit zurückgegangen werden – genau wie anhand eines alten Zustands und des Logfiles auch bis in die Gegenwart vorwärts gegangen werden kann. Bitte denken Sie aber auch daran, öfters ein Backup des Logfiles zu machen. Natürlich können Sie all dies auch kombinieren. Die aktuellen Kosten von Backup-Speichermedien und -Geräten, ihre Leistung und Ihre Datenbestände inkl. Änderungsrate bestimmen, wo für Sie ein guter Kompromiß liegt.

2-6 Wirklich kein verborgener Kanal mehr?

Leider nein! Sollten unsere Militärs ihre Rechner wirklich benötigen, gibt es folgenden verborgenen Kanal: Immer wenn ein *Rechner ausfällt*, werden unsere Militärs einen neuen bestellen. Tun sie das in den auf den Ausfall folgenden 24 Stunden und wird als plausibel angenommen, daß Rechner alle 10 Jahre (= 3652,5 Tage unter Einschluß von im Mittel 2,5 Schaltjahrtagen) ausfallen, so besteht ein verborgener Kanal von immerhin fast (bei Annahme eines perfekten Rechners, der außer zum Zwecke der Nutzung des verborgenen Kanals nicht ausfällt: genau)

$$\mathrm{ld}\ 3652{,}5\ \mathrm{bit},$$

d.h. mehr als 11 bit in 10 Jahren.

Natürlich kann man, vermutlich zu Recht, einwenden, daß etwa *illoyales Personal* einen verborgenen Kanal weit größerer Kapazität darstellt. Mir ging es aber um Informations*technik* und einen üblicherweise übersehenen Aspekt.

Übrigens kann auch *loyales Personal* einen verborgenen Kanal weit größerer Bandbreite darstellen: Nehmen wir an, daß ein Trojanisches Pferd im Rechner die Video-Ansteuerung der Terminals beeinflussen und damit die Bildqualität subtil verschlechtern kann. Dann braucht ein Angreifer nur zu beobachten, ob das Personal die hermetisch geschirmten Räume mit geröteten Augen verläßt oder wann es besonders zahlreich Augenärzte aufsucht.

2-7 Perspektivwechsel: Das diabolische Endgerät

Die Eigenschaften für a) stehen jeweils vorne gefolgt von den Motiven des Beteiligten, solch ein Gerät zu benutzen. Die Schlußfolgerung für die Umsetzung steht hinter dem Folgerungspfeil. Schlagworte, die sie im PC Handbuch finden können, sind kursiv.

Verwendung des Endgerätes für möglichst viele Zwecke gleichzeitig, damit es in seinen inneren Abläufen möglichst undurchschaubar wird. Die Motivation für den Benutzer, dies zu tolerieren, besteht drin, daß er so seine Daten für viele Zwecke gemeinsam verwalten und gegenseitig nutzen kann und daß das Endgerät so ein günstiges Preis-Leistungsverhältnis suggeriert. —> dienste-integrierendes Endgerät

Das Endgerät sollte als nützlich, chick und fortschrittlich, möglichst sogar unvermeidbar gelten. —> diensteintegrierendes Endgerät, chick, klein, bunt, kommunikativ ... vorgegeben beispielsweise durch Rahmenverträge für große Firmen und öffentliche Stellen.

Das Endgerät sollte durch möglichst Viele flexibel änderbar sein. —> programmierbares Endgerät; Software möglichst sogar dynamisch nachladbar: *PC, PDA; ActiveX*

Die Verantwortung für Änderungen sollte möglichst jemand haben, der nicht wirklich versteht, was er tut, dies aber nicht unbedingt selbst merkt, für sein Tun aber gute Gründe hat. Ideal wäre der Benutzer selbst. Dann können andere hinterher sagen: Selbst schuld ... (Eine subtile Variante hiervon ist die Vorinstallation bei Kauf durch den Verkäufer nach den Wünschen des Käufers.)

Möglichst viel Rechenleistung und Speicherplatz, die für undurchsichtige Zwecke verwendet werden, damit die Betriebsmittelbelegung (universeller) Trojanischer Pferde nicht auffällt.

Da Trojanische Pferde nicht auf Anhieb alles richtig machen, sollten Systemabstürze als normal gelten.

Undurchsichtige, umfangreiche Programme, deren Quellcode nicht bekannt ist.

Proprietäre, möglichst nicht öffentlich dokumentierte, undurchsichtige Datenformate statt öffentlich dokumentierten oder gar standardisierten Formaten. Also *WORD*-Format statt *RTF*.

Weder Zugangs- noch Zugriffskontrolle (damit jedes Programm jedes andere Programm beliebig beeinflussen kann), schon gar nicht Zugriffskontrolle nach dem Prinzip Least Privilege. —> *Windows 98* und *MacOS* haben keinerlei Zugangs- und Zugriffskontrolle, *Windows NT* und *Linux* keine Zugriffskontrolle nach dem Prinzip Least Privilege.

Das Endgerät sollte möglichst universelle, bidirektionale Schnittstellen haben, damit es der Benutzer mit allem Möglichen leicht verbinden kann. Besonders lukrativ erscheinen öffentliche offene Netze, z.B. das —> *Internet*: Einerseits für das dynamische Nachladen von Software, z.B. aktive Inhalte im Web, andererseits damit Angreifer häufig und unbemerkt Zugriff auf ihre universellen Trojanischen Pferde im Endgerät erhalten können.

Dem Benutzer nicht bewußte, zumindest nicht verdächtige Weitergabe von Geheimnissen nach außen: Inhalte, Identität des Benutzers, ersatzweise des Gerätes. —> Übertragung von unver-

schlüsselten Daten über offene Netze, z.B. das Internet. Hineincodieren des Ersteller(geräte)s jedes Objektes in die digitale Repräsentation des Objektes, z.B. die *DCE UUID* bei *OLE* (Microsoft nennt sie *GUID = Globally Unique Identifier*). Geräteidentitäten, wie z.B. Ethernetadressen oder Prozessoridentitäten (*Pentium III*), untergraben die Anonymität der Benutzer, da sie unbemerkt weitergegeben werden (können).

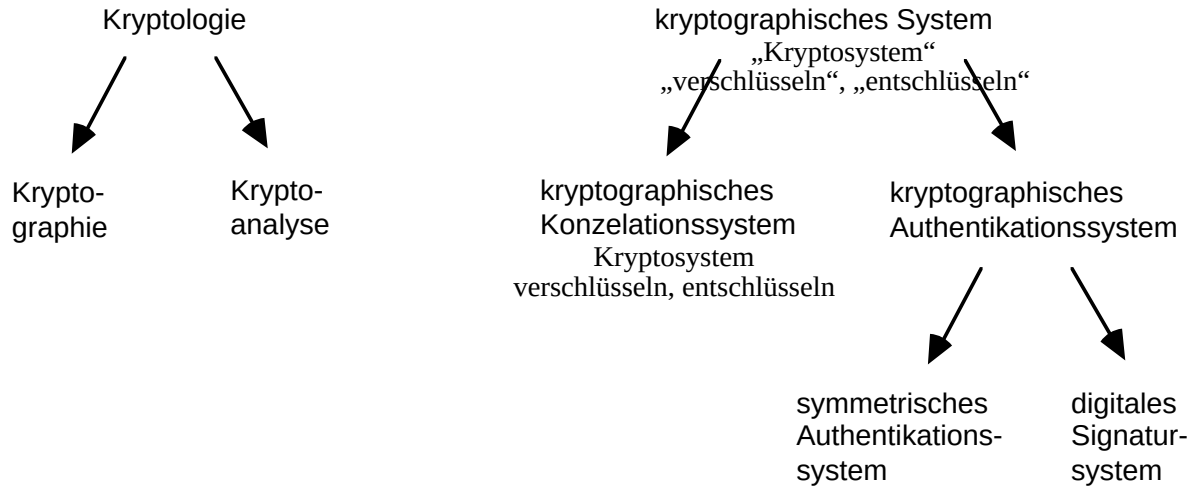
Lösungen zu Kryptologische Grundlagen

3-0 Begriffe

Kryptologie umfaßt sowohl Kryptographie (die Wissenschaft von den Algorithmen der „gutartigen" Benutzer) wie auch Kryptoanalyse (die Wissenschaft von den Algorithmen der Angreifer auf kryptographische Systeme). Kenntnisse im Bereich Kryptoanalyse sind notwendig, um die Sicherheit kryptographischer Systeme und ihrer Anwendung beurteilen zu können.

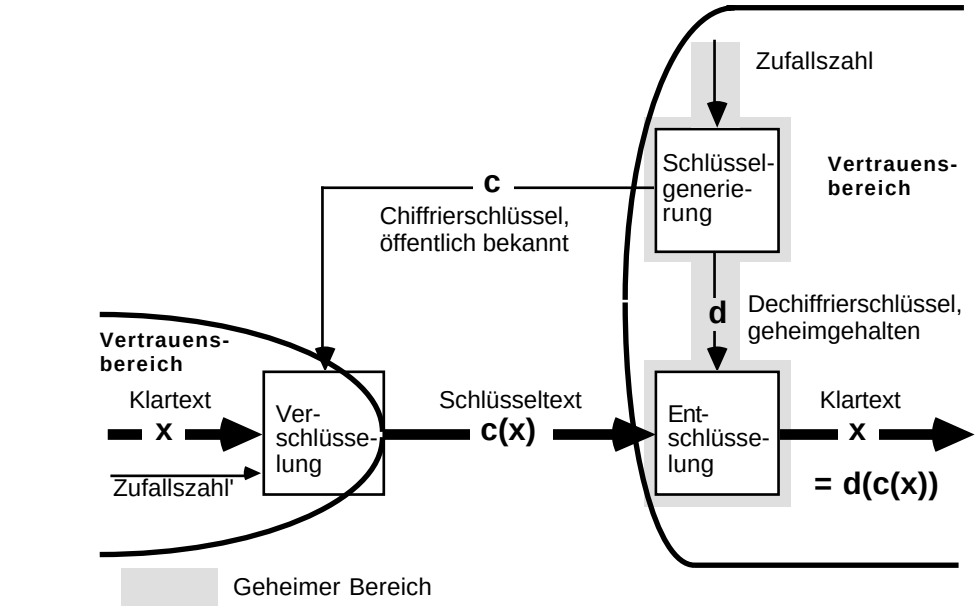
Kryptographische Systeme sind entweder Konzelationssysteme (Schutzziel: Vertraulichkeit) oder Authentikationssysteme (Schutzziel: Integrität). Authentikationssysteme gibt es symmetrisch (und damit ungeeignet als Beweismittel) oder asymmetrisch und damit geeignet als Beweismittel gegenüber Dritten. Wegen dieser wichtigen Zusatzeigenschaft werden asymmetrische Authentikationssysteme mit dem klingenden Namen digitale Signatursysteme bezeichnet.

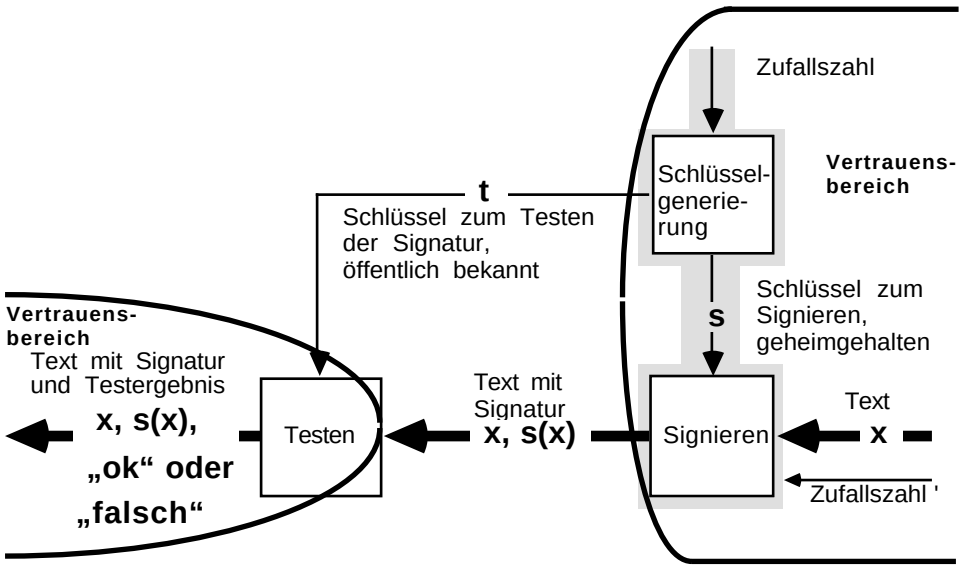
Der Vorteil, die Begriffe *Kryptosystem*, *verschlüsseln* und *entschlüsseln* oben rechts im Begriffsbaum anzusiedeln, ist, kurze übliche Begriffe allgemein verwenden zu können. Der Nachteil hiervon ist, daß die Unterscheidung zwischen Konzelation und Authentikation begrifflich in den Hintergrund rückt, so daß ich diese Begriffe nur für kryptographische Konzelationssysteme benutze – allgemeiner verwende ich diese Begriffe nur mit Anführungszeichen. Mein Begriffsbaum sieht also folgendermaßen aus:



3-1 Vertrauensbereiche für asymmetrisches Konzelationssystem und digitales Signatursystem

Der geheimzuhaltende Schlüssel kann dann bzgl. Vertraulichkeit und Integrität optimal geschützt werden, wenn er in dem Gerät, sprich Vertrauensbereich, generiert wird, in dem er auch verwendet wird. Es ergeben sich folgende beiden Bilder:





Eine Backup-Kopie des Signierschlüssels dürfte nie nötig sein, denn Signaturen bleiben auch bei Zerstörung des Signierschlüssels testbar und damit gültig. Der Signierschlüssel sollte das Signiergerät also unter keinen Umständen verlassen (können). Sollte er nicht mehr verfügbar sein, wird halt ein neues Schlüsselpaar generiert, der öffentliche Testschlüssel zertifiziert – und weiter kann's gehn. Gleich sollte verfahren werden, wenn asymmetrische Konzelation für Kommunikation eingesetzt wird: Nach erneuter Schlüsselgenerierung und erneutem Schlüsselaustausch wird die Nachricht einfach noch mal verschlüsselt und noch mal übertragen. Bei Verwendung asymmetrischer Konzelation für die Speicherung von Daten sind ggf. entweder Backup-Kopien der Daten im Klartext oder der verschlüsselten Daten und des Dechiffrierschlüssels nötig.

3-2 Weshalb Schlüssel?

a) Es geht auch ohne Schlüssel: Für symmetrische Kryptographie müssen zwei Teilnehmer dann jeweils einen geheimen, nur ihnen bekannten (und kryptographisch guten) Algorithmus austauschen. Für asymmetrische Kryptographie müßte jeweils ein öffentlicher Algorithmus publiziert werden, aus dem dann der zugehörige geheime nicht hergeleitet werden kann.

b) Die Nachteile sind zahlreich und viele davon sind schwerwiegend:
   – Von wem erhalten die Teilnehmer gute kryptographische Algorithmen? Schließlich wollen auch Nicht-Kryptographen vertraulich und integer kommunizieren können, ohne daß

jemand anderes, auch nicht der den Algorithmus oder das Algorithmenpaar erfindende Kryptograph, die Sicherheit brechen kann. Folglich müßte der erfindende Kryptograph beseitigt oder zumindest in Isolationshaft genommen werden, denn er kennt den geheimen Algorithmus. Während im Altertum so etwas vorgekommen sein soll, verbietet es sich, seit allen Menschen, selbst Kryptographen, Menschenrechte zuerkannt werden.

- Wie soll die Sicherheit des kryptographischen Algorithmus analysiert werden? Jeder, der einen symmetrischen Algorithmus analysiert, ist genauso ein Sicherheitsrisiko wie der Erfinder. Bei asymmetrischen Algorithmen ist zumindest jeder, der zur Überprüfung der Sicherheit auch den geheimen Algorithmus erfährt, ein Sicherheitsrisiko.
- Eine Implementierung der kryptographischen Systeme in Software, Firmware oder gar Hardware ist nur möglich, wenn der Benutzer in der Lage ist, sie selbst durchzuführen. Andernfalls gilt für den Implementierer Gleiches wie für den Algorithmenerfinder.
- In öffentlichen Systemen, wo potentiell jeder mit jedem gesichert kommunizieren können möchte, wären nur Softwareimplementierungen mit Algorithmen- statt Schlüsselaustausch möglich – Firmware ist maschinenabhängig und Hardwareaustausch erfordert physischen Transport. Selbst bei Softwareimplementierungen besteht der Nachteil, daß die bei symmetrischen kryptographischen Systemen vertraulich und integer und bei asymmetrischen Systemen integer auszutauschende Bitmenge erheblich größer ist: Algorithmen (sprich: Programme) sind üblicherweise länger als Schlüssel.
- Eine Normung von kryptographischen Algorithmen wäre aus technischen Gründen unmöglich.

c) Die Hauptvorteile der Benutzung von Schlüsseln sind: Die kryptographischen Algorithmen können öffentlich bekannt sein. Dies ermöglicht
   + eine breite öffentliche Validierung ihrer Sicherheit,
   + ihre Normung,
   + beliebige Implementierungsformen,
   + Massenproduktion sowie
   + eine Validierung auch der Sicherheit der Implementierung.

3-2a Schlüsselabgleich bei teilnehmerautonomer Schlüsselgenerierung?

Der Einwand gegen teilnehmerautonome dezentrale Schlüsselgenerierung ist falsch (s.u.). Folglich ist der Vorschlag unnötig und – bekanntermaßen – bzgl. der Vertraulichkeit der geheimen Schlüssel sehr gefährlich.

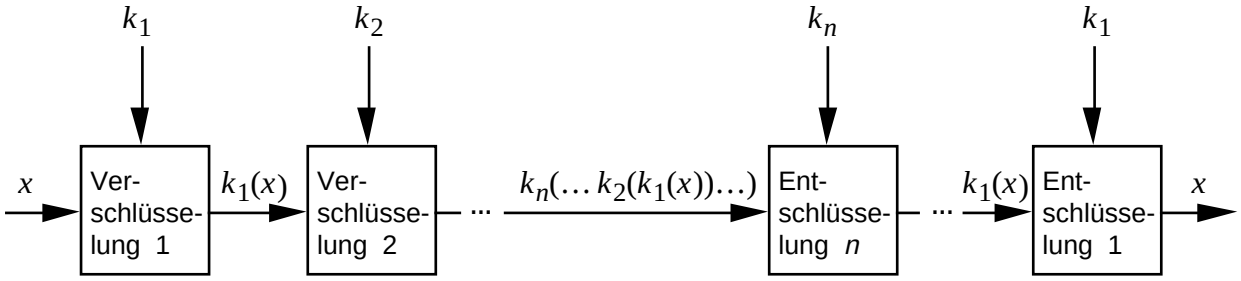
Grundsätzlich gilt: Natürlich kann nicht völlig ausgeschlossen werden, daß bei der Erzeugung der Zufallszahl, die die Schlüsselgenerierung parametrisiert, dieselbe Zufallszahl mehrfach auftritt und dann auch der geheime Teil des Schlüsselpaares mehrfach generiert werden kann. Die Schlußfolgerung, deshalb müsse die Schlüsselgenerierung koordiniert ablaufen, ja gar von sogenannten TrustCentern durchgeführt werden, ist aber unsinnig. Tritt dieselbe Zufallszahl mehrfach auf, dann ist entweder der Zufallsgenerator schlecht oder die Zufallszahl zu kurz. In beiden Fällen hilft es nicht, die Zufallszahl nur beim ersten mal zu verwenden und bei wiederholtem Auftreten zu verwerfen. Denn wer sollte einen Angreifer daran hindern, mit dem gleichen Zufallsgenerator Zahlen gleicher Länge zu erzeugen und zu hoffen, daß er Glück hat? Da dies nicht verhindert werden kann, hilft nur: Zufallsgenerator gut wählen und validieren sowie Zufallszahl lang genug wählen. Alles andere ist Kosmetik, die vom eigentlichen Problem ablenkt.

Und nebenbei: Falls überhaupt Schlüssel verglichen werden, dann genügt es, *öffentliche* Schlüssel zu vergleichen. Die Schlüsselgenerierung kann also selbst dann teilnehmerautonom bleiben – der Schlüsselvergleich erfolgt als Teil der Schlüsselzertifizierung.

3-3 Steigerung der Sicherheit durch Gebrauch von mehreren kryptographischen Systemen

Beim Schutzziel **Konzelation** kann man die kryptographischen Systeme **in Serie** verwenden, d.h. der Klartext wird mit dem Konzelationssystem 1 verschlüsselt, danach der resultierende Schlüsseltext mit Konzelationssystem 2, usw. Z.B. bei einem symmetrischen System in der Kurzschreibweise aus Bild 3-2, wenn k_i der Schlüssel des i-ten Systems ist: $S := k_n(\ldots k_2(k_1(x))\ldots)$. Man muß natürlich für jedes System einen unabhängigen Schlüssel gewählt und die Reihenfolge festgelegt haben.

Entschlüsselt wird in umgekehrter Reihenfolge, d.h. zuletzt mit Konzelationssystem 1, was wieder den Klartext liefert. Im Beispiel $x = k_1^{-1}(k_2^{-1}(\ldots k_n^{-1}(S)\ldots))$.



Natürlich können auch asymmetrische Konzelationssysteme auf diese Weise verwendet werden. Auch ein gemischter Einsatz von symmetrischen und asymmetrischen Konzelationssystemen ist möglich. Im Beispiel können hierzu für jedes einzelne i, $1 \le i \le n$, jeweils die k_i links durch c_i und das k_i rechts durch d_i ersetzt werden, $1 \le i \le n$.

Die beschriebene Konstruktion geht natürlich nur, wenn der Schüsseltextraum von Konzelationssystem i ein Teilraum des Klartextraumes des Konzelationssystems i+1 ist. Dies läßt sich in der Praxis meist problemlos erreichen. (Genauer später in der Vorlesung: §3.8.1 Block- und Stromchiffren sowie §3.8.2 Betriebsarten für Blockchiffren). Haben Klartext und Schlüsseltexte bei allen n Systemen die gleiche Länge, verändert diese Maßnahme den Speicher- bzw. Übertragungsaufwand nicht. (Meistens wird er aber wegen Blockung ein bißchen wachsen.) Der *Berechnungsaufwand* ist die *Summe aller Berechnungsaufwände* der einzelnen Konzelationssysteme (allerdings auf den evtl. etwas verlängerten Nachrichten).

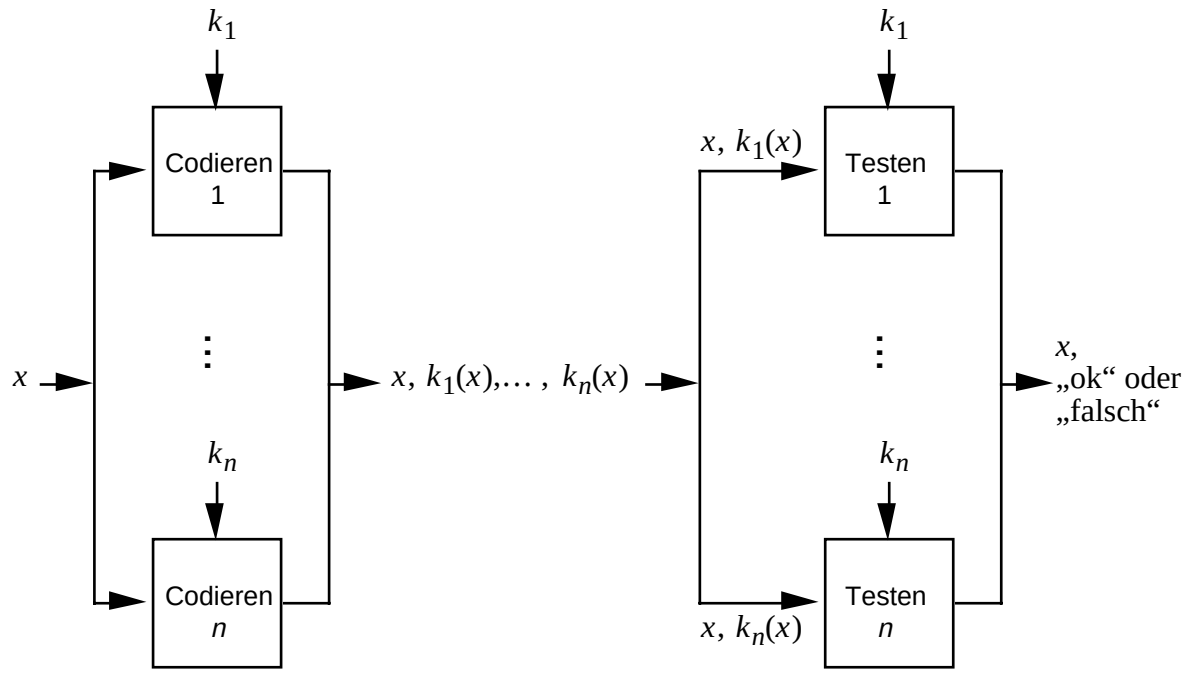
Nebenbei: Das durch die Verwendung mehrerer Konzelationsysteme in Serie entstehende Konzelationssystem heißt **Produktchiffre** der verwendeten Konzelationsysteme.

Beim Schutzziel **Authentikation** kann man die kryptographischen Systeme **parallel** verwenden, d.h. mit jedem System wird ein Authentikator der ursprünglichen Nachricht x gebildet, und alle werden hintereinander an x gehängt (mit Trennzeichen o.ä.). (Z.B. gemäß Bild 3-6: x, $k_1(x),\ldots, k_n(x)$.) Der Empfänger prüft jeden Authentikator getrennt mit dem passenden System; alle müssen stimmen.

Natürlich können auch digitale Signatursysteme auf diese Weise verwendet werden, ebenso ist ein gemischter Einsatz möglich. Im Beispiel können hierzu für jedes einzelne i, $1 \le i \le n$, jeweils die k_i links durch s_i und das k_i rechts durch t_i ersetzt werden.

Auch hier müssen die verwendeten Schlüssel unabhängig voneinander gewählt werden.

Berechnungsaufwand und *Speicher- und Übertragungsaufwand* für den Authentikator sind *genau die Summe* derer aus den einzelnen Systemen. Die Berechnung kann problemlos parallel erfolgen, so daß bei geeigneter Implementierung das Gesamtsystem fast so schnell wie das langsamste verwendete Authentikationssystem arbeitet.



Sowohl beim Schutzziel Konzelation als auch beim Schutzziel Authentikation ist bei Softwareimplementierung der kryptographischen Systeme natürlich noch zusätzlich Speicheraufwand für die jeweiligen Programme nötig.

Zur **Beweisbarkeit** der beiden Konstruktionen:

Systeme parallel zur Authentikation ist trivial zu beweisen, da sich an der Benutzung jedes einzelnen Systems, insbesondere der Verteilung von Nachrichten und Schlüsseltexten/MACs überhaupt nichts ändert.

Systeme in Serie zur Konzelation ist leider überhaupt nicht zu beweisen. Die Verteilung der Eingaben von „Verschlüsselung 2“ bis „Verschlüsselung *n*“ wird durch die Konstruktion geändert – statt dem Klartext wird Schlüsseltext verschlüsselt. Dies könnte nun so unglücklich geschehen, daß die Verschlüsselungen 2 bis *n* zur Sicherheit nichts beitragen, das System in Serie also nur so sicher wie „Verschlüsselung 1“ ist. In der Praxis dürfte dies nur der Fall sein, wenn „Verschlüsselung 2“ bis „Verschlüsselung *n*“ zielgerichtet so „unglücklich“ entworfen werden. (In Aufgabe 3-26 finden Sie eine in einem gewissen Sinne beweisbar sichere, allerdings aufwendigere Konstruktion zum Gebrauch mehrerer Konzelationssysteme.)

Umgekehrt liefert der Beweis für *Systeme parallel zur Authentikation* auch, daß die Konstruktion *nur* so sicher ist, wie es schwer ist, alle verwendeten Systeme zu brechen. Und ebenfalls umgekehrt scheitert der Beweis für *Systeme in Serie zur Konzelation* hauptsächlich daran, daß die Zwischenergebnisse, d.h. die Schlüsseltexte zwischen den verwendeten Konzelationssystemen, dem Angreifer nicht vorliegen – gerade das macht seine Aufgabe aber normalerweise ungleich schwieriger als die, *nur* alle verwendeten Konzelationssysteme zu brechen.

Diskussion *falscher* Lösungsvorschläge:

Um die Nachricht mit mehreren Konzelationssystemen zu verschlüsseln und so den Klartext vertraulich zu halten, obwohl einzelne Konzelationssysteme unsicher sind, wird die Nachricht in so viele Abschnitte aufgeteilt, wie Konzelationssysteme zur Verfügung stehen, und jeder Abschnitt mit einem anderen Konzelationssystem verschlüsselt. *Nachteil*: Die Wahrscheinlichkeit, daß ein Angreifer zumindest Abschnitte der Nachricht entschlüsseln kann, wird durch dies Vorgehen eher erhöht als erniedrigt.

Um die Nachricht mit mehreren Authentikationssystemen zu authentizieren, wird mit dem ersten Authentikationssystem zur Nachricht ein 1. Prüfteil gebildet. Mit dem zweiten Authenti-

kationssystem wird dieser 1. Prüfteil authentiziert, und so der 2. Prüfteil gebildet. Mit dem dritten Authentikationssystem wird der 2. Prüfteil authentiziert, usw. *Nachteil*: Kann ein Angreifer das erste Authentikationssystem brechen, indem er zum 1. Prüfteil eine andere passende Nachricht findet, sind alle anderen Authentikationssysteme hiergegen wirkungslos (denn der 1. Prüfteil wird für die andere passende Nachricht ja nicht geändert, der ganze Rest kann also unverändert bleiben). Außerdem läßt sich die Berechnung der Prüfteile hier nicht so schön parallelisieren.

Um Schlüsselaustauschaufwand zu sparen, wird, sofern möglich, der gleiche Schlüssel für mehrere kryptographische Systeme genommen. *Nachteil*: In vielen Fällen ist das resultierende System nicht stärker, manchmal sogar schwächer als das schächste der verwendeten Systeme. Dies ist besonders leicht bzgl. Authentikation zu zeigen: Nehmen wir an, alle Systeme verwenden den gleichen Schlüssel. Dann ist klar, daß ein vollständiges Brechen auch nur eines Systems das vollständige Brechen des Gesamtsystems ermöglicht. Erlauben manche Systeme das effektive Gewinnen von Information über den Schlüssel, so kann, obwohl keines für sich allein genug Information für einen erfolgreichen Angriff zu gewinnen erlaubt, die Summe der einzelnen Informationen hierfür ausreichen.

Diskussion eines *ungeschickten* Lösungsvorschlags:

Es wird mit Authentikationssystem i ein Authentikator nicht nur unter die Nachricht x gebildet, sondern auch unter die Prüfteile 1 bis i-1. Diese Lösung ist zwar sicher, aber unnötig aufwendig:

1. Die Berechnung der Prüfteile erfordert in der Regel umso mehr Rechenaufwand, je länger die zu authentizierende Zeichenkette ist.
2. Die Berechnung der Prüfteile ist nicht mehr einfach parallelisierbar, da Prüfteil i von Prüfteil i-1 abhängt.

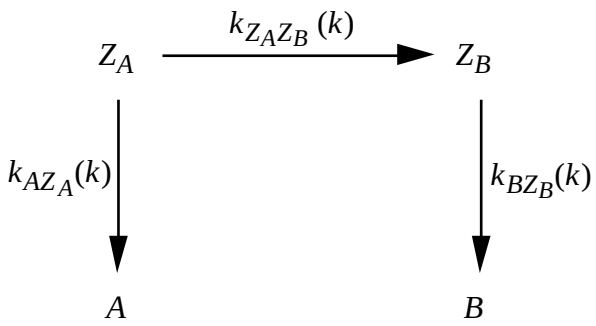
3-4 Schlüsselaustauschprotokolle

a) Die Funktion *einer* Schlüsselverteilzentrale muß von *mehreren* gemeinsam erbracht werden. Dazu müssen entweder
   - alle Schlüsselverteilzentralen direkt Schlüssel voneinander kennen, oder
   - die Schlüsselverteilzentralen benötigen wiederum eine Ober-Schlüsselverteilzentrale, usw.

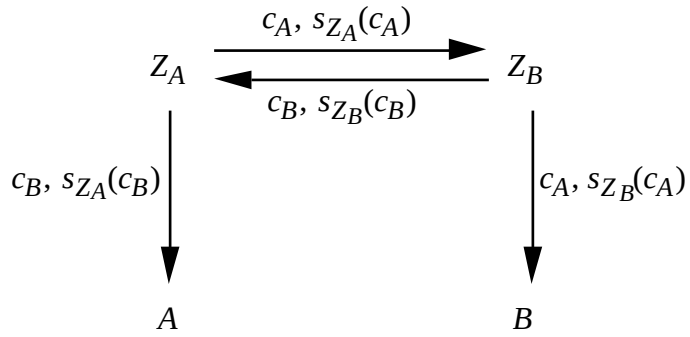
   Im zweiten Fall müssen zunächst die zwei Schlüsselverteilzentralen einen Schlüssel austauschen, was genau wie bisher zwischen zwei Teilnehmern geht.

   Im folgenden nehmen wir also an, die zwei Schlüsselverteilzentralen Z_A und Z_B, mit denen die beiden Teilnehmer A bzw. B einen Schlüssel ausgetauscht haben, hätten nun gemeinsame Schlüssel.

   Symmetrisches System: Jetzt kann eine der beiden Schlüsselverteilzentralen einen Schlüssel generieren und der anderen vertraulich mitteilen (z.B. generiert die Schlüsselverteilzentrale Z_A den Schlüssel k und schickt ihn an Z_B, verschlüsselt mit $k_{Z_AZ_B}$ (den sie ja nun gemeinsam haben). Nun teilt jede „ihrem“ Teilnehmer den Schlüssel vertraulich mit, wie bisher (d.h. Z_A schickt $k_{AZ_A}(k)$ an A, und Z_B schickt $k_{BZ_B}(k)$ an B).

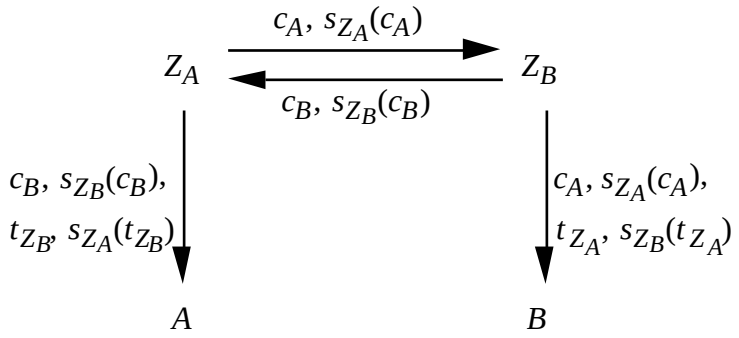


Authentikation von öffentlichen Schlüsseln: Der öffentliche Schlüssel jedes Teilnehmers wird von „seiner" Zentrale authentisiert an die Zentrale des anderen geschickt (z.B. schickt Z_A den Schlüssel c_A an Z_B, signiert mit s_{Z_A}). Die Zentrale des anderen prüft die Authentisierung und authentisiert den öffentlichen Schlüssel ggf. ihrerseits, damit der Empfänger diese zweite Authentisierung prüfen kann (z.B. prüft Z_B mit t_{Z_A}, den sie ja nun kennt, und signiert dann c_A mit s_{Z_B}, was B dann mit t_{Z_B} prüfen kann).



Der jeweils resultierende Schlüsselaustausch ist in allen Fällen natürlich höchstens so vertrauenswürdig wie der schwächste verwendete Schritt, d.h. das schwächste Schlüsselaustausch-Subprotokoll.

Anmerkung: Im Fall, daß die Authentisierung von öffentlichen Schlüsseln durch digitale Signatursysteme erfolgt, wäre es eine Alternative, daß die Zentrale des anderen Teilnehmers nicht den öffentlichen Schlüssel des ersten Teilnehmers, sondern den öffentlichen Schlüssel von dessen Zentrale authentisiert. Dann kann der Teilnehmer die Authentisierung des öffentlichen Schlüssels des anderen Teilnehmers selbst prüfen.



Die Vorteile dieser Alternative sind, daß

+ sich Zentralen weniger merken müssen (in der vorherigen Alternative muß sich z.B. Z_B das Zertifikat c_A, $S_{Z_A}(c_A)$ merken, dann nur dann kann Z_B die Ausstellung des Zertifikats c_A, $S_{Z_B}(c_A)$ rechtfertigen),
+ klar ist, wer was geprüft hat (und, je nach rechtlichem Kontext, wer wofür haftet),
+ für die Teilnehmer erkennbar ist, über wie viele (und ggf. wie vertrauenswürdige und in welcher Weise haftende) Stufen die Authentisierung des öffentlichen Schlüssels erfolgte und
+ deshalb diese Alternative allgemeiner anwendbar ist, vgl. Aufgabenteil g).

Der Nachteil dieser Alternative ist, daß

- Teilnehmer mehr Signaturen überprüfen müssen, um sich von der Authentizität eines Schlüssels zu überzeugen.

Aus meiner Sicht überwiegen die Vorteile bereits heute deutlich – und sie werden immer mehr überwiegen, je leistungsfähiger die Rechentechnik der Teilnehmer wird.

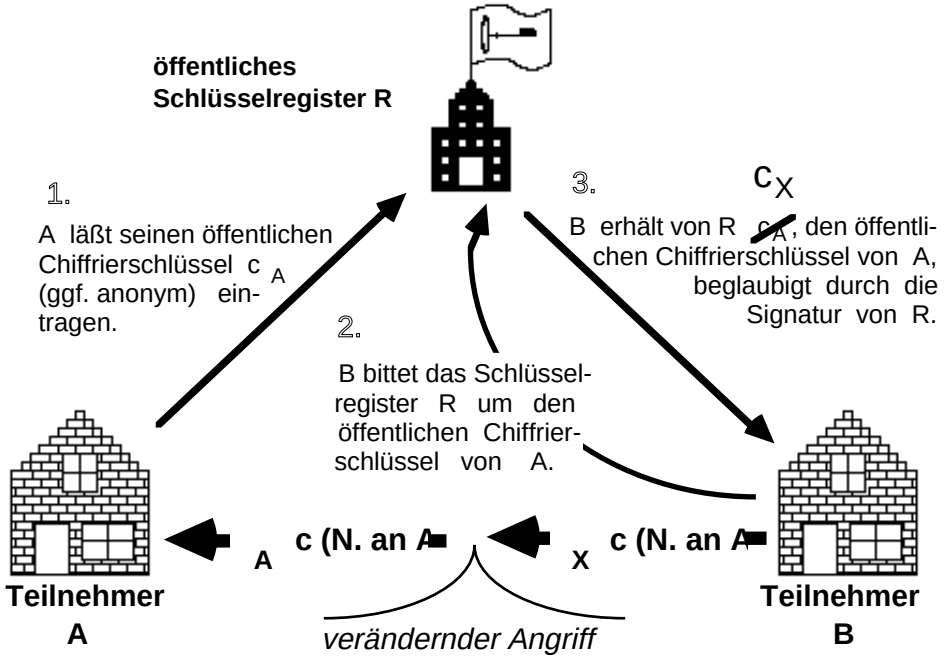
b) *Optional:* Vermutlich sieht Ihre Lösung so aus, daß mehrere Teilschlüssel jeweils über zwei Schlüsselverteilzentralen (eine in jedem Land) ausgetauscht und von den Teilnehmern modulo 2 addiert werden. Das ist besser als den Schlüssel gleich als Ganzes auszutauschen, hat aber den Nachteil, daß nicht alle Schlüsselverteilzentralen eines Landes zusammenarbeiten müssen, um den Schlüssel zu erhalten. Es genügt, wenn jeweils eine der zwei Zentralen kooperiert. Dies bedeutet nun, daß die lokale Entscheidung der Teilnehmer, welches die Schlüsselverteilzentralen in ihrem Land sind, die keinesfalls alle kooperieren, ausgehebelt wird. Wenn wir unterstellen, daß alle Schlüsselverteilzentralen untereinander Schlüssel ausgetauscht haben, dann gibt es eine bessere Lösung, die die lokalen Entscheidungen der beiden Teilnehmer respektiert:

In einer ersten Phase tauscht einer der Teilnehmer über alle seine Schlüsselverteilzentralen Teilschlüssel mit jeder Schlüsselverteilzentrale des anderen Teilnehmers aus, die der Teilnehmer und die Schlüsselverteilzentrale des anderen Teilnehmers jeweils modulo 2 addieren und so jeweils einen geheimen Schlüssel erhalten. Hat der Teilnehmer *a* Schlüsselverteilzentralen und der andere Teilnehmer *b*, dann werden also *b* mal jeweils *a* Teilschlüssel, insgesamt also *a*•*b* Teilschlüssel ausgetauscht und daraus *b* Schlüssel gewonnen.

In einer zweiten Phase werden diese *b* geheimen Schlüssel mit den Schlüsselverteilzentralen des anderen Teilnehmers sowie die zwischen dem anderen Teilnehmer und seinen Schlüsselverteilzentralen ausgetauschten Schlüssel verwendet, um wie in Bild 3-3 gezeigt zwischen den Teilnehmern *b* Teilschlüssel auszutauschen (die natürlich nichts mit den Teilschlüsseln in der ersten Phase zu tun haben dürfen).

c) Ja, eine Parallelschaltung ist auch bei asymmetrischen kryptographischen Systemen sinnvoll. Es bestehen nämlich Möglichkeiten für **verändernde Angriffe** eines einzelnen (oder weniger) Schlüsselregisters *R*: Dieses könnte falsche Schlüssel weitergeben, zu denen es selbst die geheimen Schlüssel kennt. Wie in den folgenden Bildern gezeigt, könnte es dann einerseits unbemerkt die Vertraulichkeit von Nachrichten brechen, indem es Nachrichten umschlüsselt, und andererseits Nachrichten unbemerkt fälschen, indem es sie „umsigniert“ oder ganz neue signiert.

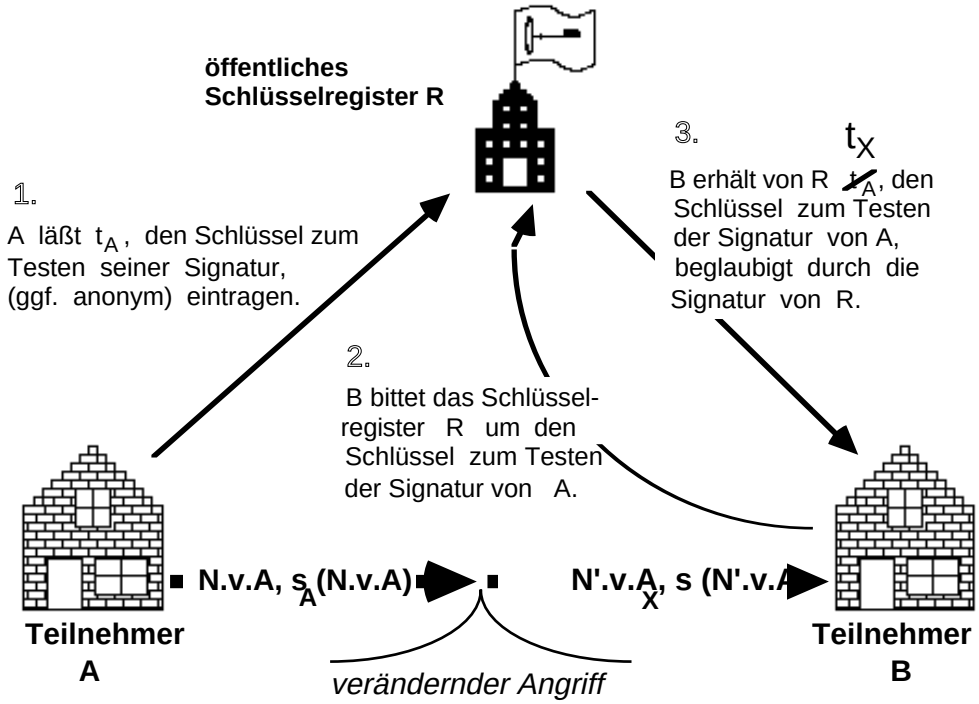
Verändernder Angriff auf die Vertraulichkeit durch Umschlüsseln von Nachrichten



Ablauf des verändernden Angriffs:

0. Angreifer nennt B falschen öffentlichen Schlüssel c_X, dessen zugehöriger inverser d_X dem Angreifer bekannt ist.
1. Angreifer fängt Nachricht ab.
2. Angreifer entschlüsselt Nachricht und nimmt sie zur Kenntnis. Er verschlüssele sie mit c_A.
3. Angreifer sendet umgeschlüsselte Nachricht (damit B nicht am Ausbleiben der Antwort von A etwas merkt).

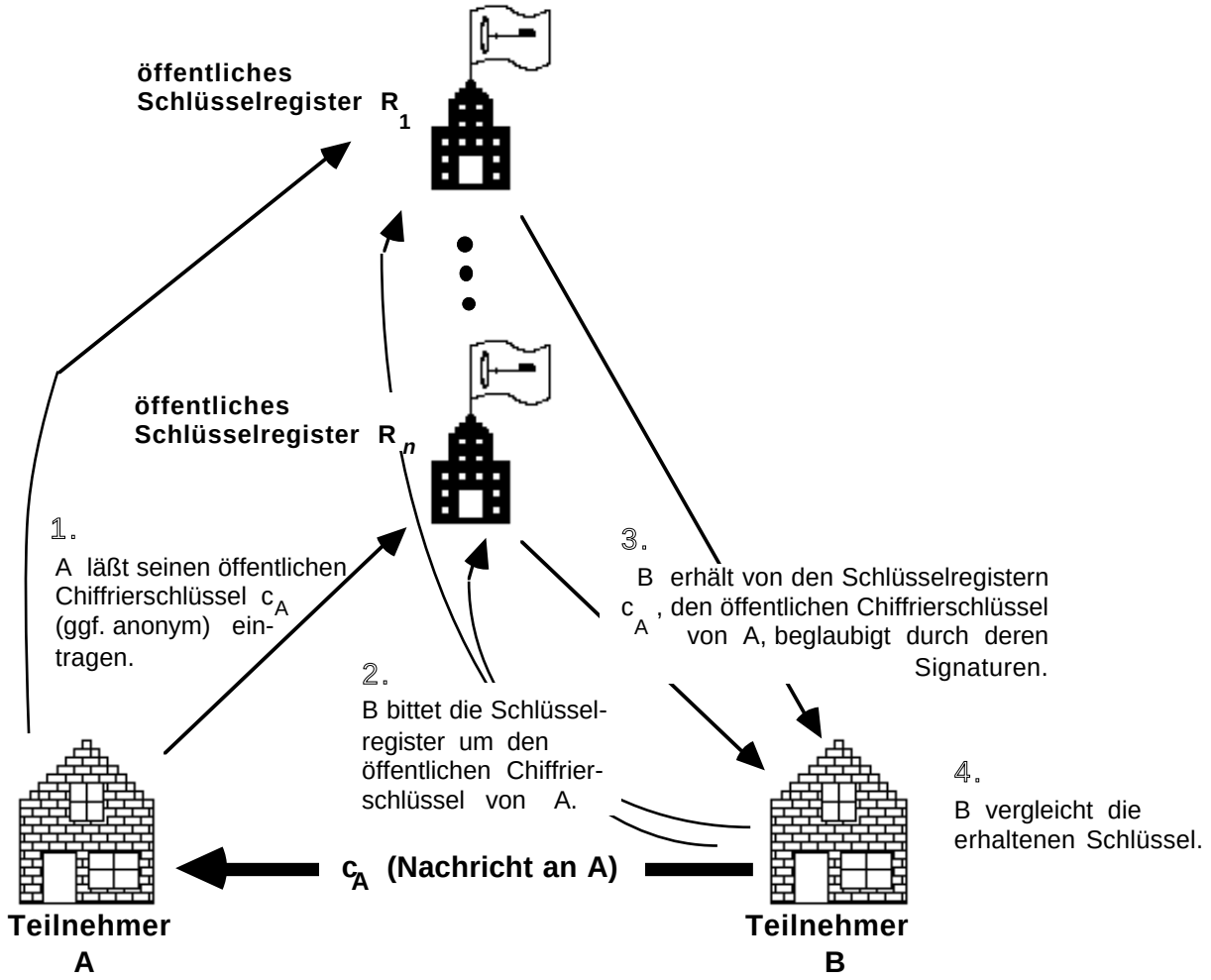
Verändernder Angriff auf die Authentizität durch Umsignieren von Nachrichten



Ablauf des verändernden Angriffs:

0. Angreifer nennt B falschen öffentlichen Testschlüssel t_X, dessen zugehöriger Schlüssel zum Signieren s_X dem Angreifer bekannt ist.
1. (Optional:) Angreifer fängt Nachricht ab. (Er kann auch eine ganz neue generieren.)
2. Angreifer ändert ggf. Nachricht. Er signiert sie mit s_X.
3. Angreifer sendet die mit s_X signierte Nachricht.

Die Parallelschaltung erfolgt so: Jeder Teilnehmer A[141] gibt seinen öffentlichen Schlüssel mehreren öffentlichen Schlüsselregistern bekannt. Jeder Teilnehmer *B* fordert den öffentlichen Schlüssel eines anderen Teilnehmers *A* von allen diesen Schlüsselregistern an und vergleicht die erhaltenen Schlüssel. Sind sie nicht alle gleich, ist entweder das Signatursystem gebrochen (Fälschung auf dem Weg zum Teilnehmer *B*), oder mindestens eines der Schlüsselregister betrügt, oder der Teilnehmer *A* hat nicht allen Schlüsselregistern denselben Schlüssel genannt. Letzteres können die „ehrlichen“ Schlüsselregister ausschließen, indem sie einander von Schlüsseleintragungen informieren. (Wiederum kann man aber nicht alle Fälle unterscheiden, d.h. man hat Verfügbarkeit nicht garantiert.)



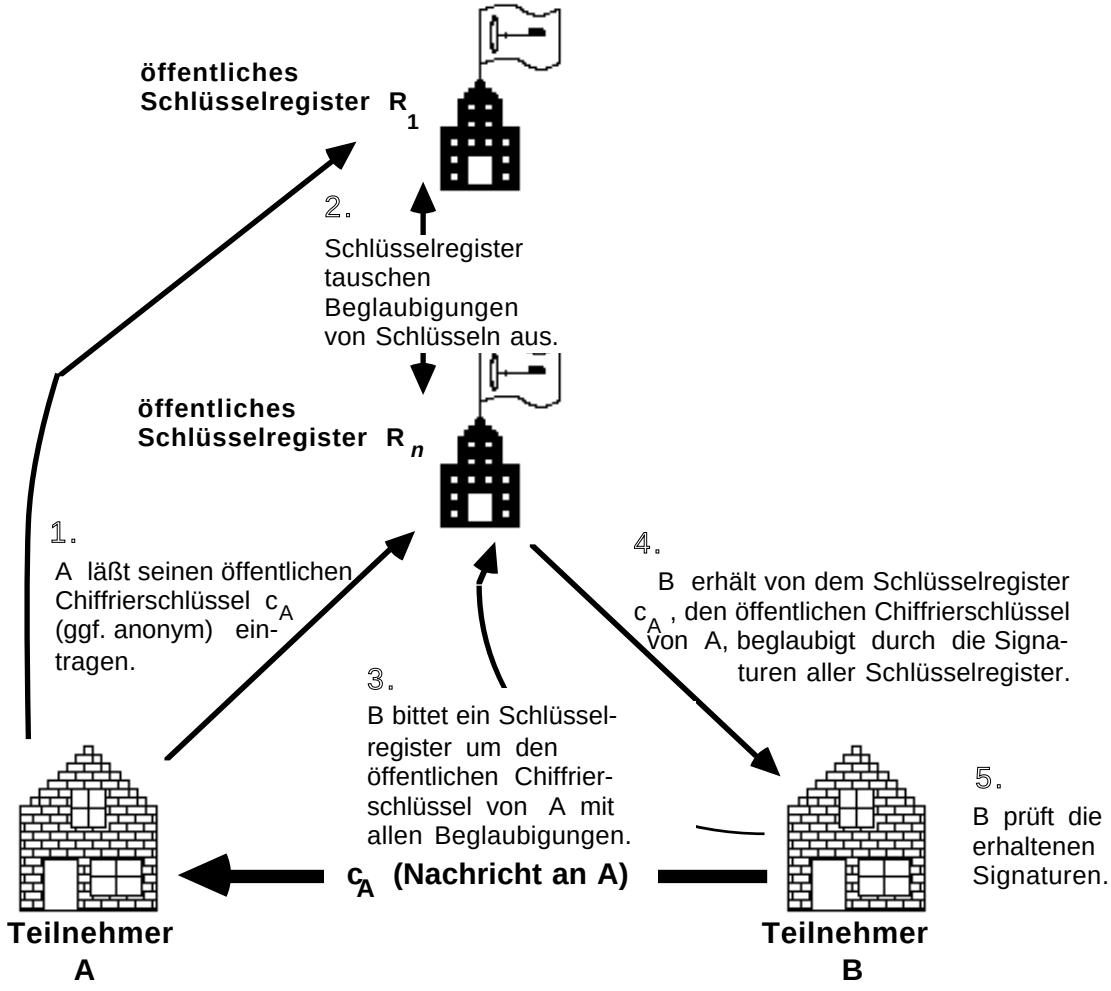
Um zumindest einige Fälle unterscheiden zu können, sollte ein Teilnehmer, der einem Schlüsselregister seinen öffentlichen Schlüssel mitteilt, einerseits dem Schlüsselregister eine „normale“ schriftliche Erklärung unterschreiben, welches sein öffentlicher Schlüssel ist. Andererseits sollte er vom Schlüsselregister eine Unterschrift erhalten, welchen Schlüssel er ihm

141 Hier drücke ich mich – hoffentlich – leicht verständlich, aber nicht präzise aus: Ausführlich müßte es heißen: Jeder Teilnehmer in der Rolle, die in der bisherigen Beschreibung durch Teilnehmer *A* wahrgenommen wurde, ... Entsprechendes gilt für: Jeder Teilnehmer *B* ...

genannt hat. Diese zweite Unterschrift kann, da der Testschlüssel des Schlüsselregisters als bekannt vorausgesetzt werden kann, auch eine digitale Signatur sein. Nach Austausch dieser beiden unterschriebenen Erklärungen sind Dispute zwischen Teilnehmer und Schlüsselregister immerhin so gut auflösbar, wie das schwächere der beiden Signatursysteme sicher ist.

Es gibt (mindestens) zwei sinnvolle Variationen der Parallelschaltung asymmetrischer kryptographischer Systeme:

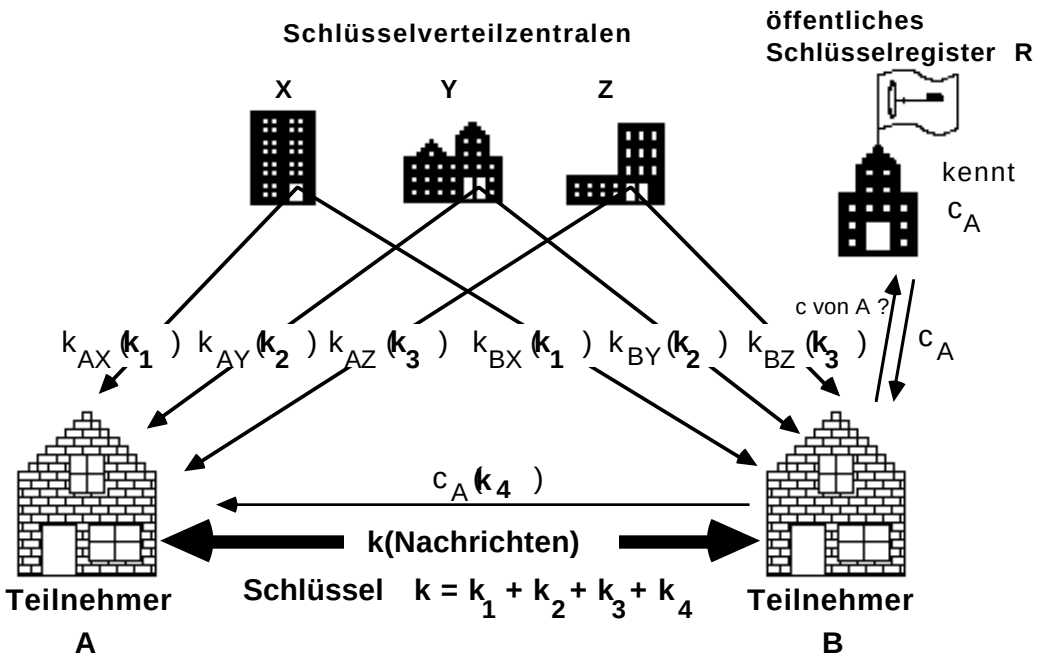
1. Bleiben öffentliche Schlüssel unbegrenzt gültig, muß also auf ihre Aktualität nicht geachtet werden, so kann jedes Schlüsselregister auch die Signaturen der anderen Schlüsselregister unter jeden der Schlüssel speichern. Dann kann ein Teilnehmer den Schlüssel eines anderen und alle Signaturen unter diesen Schlüssel von *einem* Schlüsselregister anfordern. Dies spart Kommunikationsaufwand. An der Prüfung der Signaturen ändert sich nichts.



2. Hat man Zweifel an der Sicherheit des asymmetrischen kryptographischen Systems, kann man statt des einen Systems mehrere verwenden (vgl. Aufgabe 3-3) und statt des einen, mehrfach authentizierten Schlüssels entsprechend der Systemanzahl mehrfache Schlüssel verwenden. Bei Konzelationssystemen KS_i wird dann der Schlüsseltext von KS_j mit KS_{j+1} verschlüsselt. Bei digitalen Signatursystemen wird jeweils der Klartext signiert und die Signaturen werden einfach hintereinandergehängt.

d) Ja. Wie im folgenden Bild gezeigt, führt man das für symmetrische und das für asymmetrische Konzelationssysteme bekannte Schlüsselverteilungsprotokoll *parallel* durch:[142]

Ein Teilnehmer *B* sendet dann dem anderen noch einen Schlüssel k_4 des symmetrischen Konzelationssystems, den er mit dem ihm bekannten öffentlichen Schlüssel c_A des Partners *A* konzeliert. Beide Partner verwenden als Schlüssel die Summe *k* aller ihnen im symmetrischen Schlüsselverteilprotokoll mitgeteilten Schlüssel (k_1,k_2,k_3) und des zusätzlichen Schlüssels k_4. Um diese Summe *k* zu erhalten, muß ein Angreifer alle Summanden erfahren, d.h. die passive Mithilfe aller Schlüsselverteilzentralen haben *und* das asymmetrische Konzelationssystem brechen.



Man muß nach dem Schlüsselaustausch eine *Vernam-Chiffre* (one-time pad) für Konzelation und *informationstheoretisch sichere Authentikationscodes* für Authentikation verwenden.
Rechnet man auch mit *aktiven* Angriffen von Schlüsselregistern, sollte das eine Schlüsselregister um weitere ergänzt werden, vgl. Aufgabenteil c).
Anmerkung: Folgende Variation der Fragestellung kann *praktisch* bedeutsam sein: Wir nehmen nicht an, daß der Angreifer, sofern er keine passive Mithilfe der Schlüsselverteilzentralen erhält, komplexitätstheoretisch unbeschränkt ist, sondern „nur" die asymmetrischen kryptographischen Systeme brechen kann. Dann braucht man als symmetrisches Konzelationssystem nicht unbedingt eine Vernam-Chiffre und für Authentikation nicht unbedingt informationstheoretisch sichere Authentikationscodes zu verwenden.

e) Wird der Schlüssel in einem symmetrischen Konzelationssystem verwendet, kann der Partner jeweils nicht mehr richtig entschlüsseln. Wird der Schlüssel in einem symmetrischen Authentikationssystem verwendet, werden auch die vom Partner „richtig" authentisierten Nachrichten vom andern Partner als „gefälscht" eingestuft.
Um unbemerktes Verändern der Schlüssel auf den Leitungen zu verhindern, sollten die Schlüssel nicht nur konzeliert, sondern auch authentisiert werden (in welcher Reihenfolge dies geschickterweise erfolgen sollte, wird in Aufgabe 3-13 behandelt). Stimmt die Authentisierung

142 Hierbei kann und, wenn auch verändernde Angriffe des öffentlichen Schlüsselregisters toleriert werden sollten, sollte die aus Übersichtlichkeitsgründen im Bild nicht gezeigte Parallelschaltung auch bezüglich der öffentlichen Schlüsselregister angewandt werden, vgl. Aufgabenteil c).

der zur Schlüsselverteilung verwendeten Nachrichten, dann ist der Kreis der jeweils möglichen Täter auf Schlüsselverteilzentralen und die beiden Teilnehmer beschränkt.
Die Teilnehmer sollten sich direkt nach dem Schlüsselaustausch durch ein *Testprotokoll für Schlüsselgleichheit* Gewißheit darüber verschaffen, ob sie den gleichen Schlüssel haben. Dies kann geschehen, indem sie sich gegenseitig konzelierte Zufallszahlen zuschicken und vom Partner die Zufallszahl jeweils im Klartext zurückerwarten. (Dieses kleine Kryptoprotokoll ist natürlich nur sicher, wenn das verwendete symmetrische Konzelationssystem erstens mindestens einem gewählten Schlüsseltext-Klartext-Angriff, vgl. §3.1.3.2, widersteht und zweitens ein verändernder Angreifer mit Kenntnis der inkonsistenten Schlüssel zwischen den beiden Teilnehmern ausgeschlossen wird. Zusätzlich muß noch der in 3-18a erwähnte Spiegelangriff verhindert werden, etwa indem jeweils nur einer der Teilnehmer Zufallszahlen schickt und erst alle Antworten abwartet, bevor er selbst Antworten generiert. Zu guter Letzt kann in diesem kleinen Kryptoprotokoll die Vernam-Chiffre nicht verwendet werden! Denn bei ihr erführe der Angreifer immer genau den Bereich des Schlüssels, der auf Schlüsselgleichheit getestet wird. Kurzum: Wir brauchen ein besseres Testprotokoll oder eine sehr sehr gute Blockchiffre. Ersteres wird nachfolgend vorgestellt.)

Als Hinführung zunächst eine *falsche* Lösung, die aber in die richtige Richtung weist:
Im Netz sei bekannt, wie eine Prüfsumme über Zeichenketten gebildet wird (Sie kennen sowas sicher unter den Namen Paritätsbit, Cyclic Redundancy Check (CRC), o.ä.). Die Teilnehmer bilden nach einem Verfahren solch eine Prüfsumme und einer schickt sein Ergebnis an den andern. Stimmen die Prüfsummen-Werte überein, so meinen unsere beiden Teilnehmer, dann haben sie mit großer Wahrscheinlichkeit den gleichen Schlüssel. Was ist ihr Denkfehler? Nun, Prüfsummen sind für dumme Fehler, nicht aber für intelligente Angreifer entworfen. Bei üblichen Prüfsummen ist es leicht, viele unterschiedliche Werte zu finden, die die gleiche Prüfsumme ergeben. Also könnte dies der Angreifer tun und solche inkonsistenten Schlüssel verteilen. Dann würden dies die beiden Teilnehmer erst dann bemerken, wenn sie den Schlüssel tatsächlich brauchen. Geschickt wäre also eine Prüfsumme, deren Bildung der Angreifer nicht vorhersehen kann!

In [BeBE_92 Seite 30] findet sich unter Verweis auf [BeBR_88] folgende nette Idee für ein *informationstheoretisch sicheres Testprotokoll für Schlüsselgleichheit*: Die Teilnehmer wählen mehrmals hintereinander, jeweils unabhängig zufällig die Hälfte aller Schlüsselbits aus und teilen sich deren bitweise Summe mod 2 mit. (Stimmen die Schlüssel nicht überein, so wird dies in jedem Schritt mit der Wahrscheinlichkeit 1/2 entdeckt.) Damit ein Angreifer, der dies mithört, durch solch einen Schritt nichts über den zukünftig zu verwendenden Schlüssel erfährt, wird eins der Bits, über die die Summe mod 2 öffentlich kommuniziert wurde, aus dem Schlüssel entfernt, beispielsweise immer das letzte Bit. Der Schlüssel wird bei jedem Schritt also um ein Bit kürzer. Dies ist nicht schlimm, denn man braucht nur wenige Schritte: Nach n Schritten ist die Wahrscheinlichkeit, daß das Testergebnis stimmt, $1-2^{-n}$, also spätestens für $n = 50$ genügend groß genug. (Auch bei diesem Testprotokoll für Schlüsselgleichheit muß ein verändernder Angreifer mit Kenntnis der inkonsistenten Schlüssel zwischen den beiden Teilnehmern ausgeschlossen werden: Er könnte die von den Teilnehmern geschickten Summen so ändern, daß aus Sicht der Teilnehmer alles stimmt.)

Eine Schlüsselverteilzentrale: Wird beim Schlüsselaustausch nur symmetrische Authentikation verwendet (also weder digitale Signaturen noch unbedingt sichere Pseudosignaturen, vgl. §3.9.3), könnten aus Sicht jedes der beiden Teilnehmer jeweils der Partner oder die Schlüsselverteilzentrale lügen (dies ist im Kontext des sogenannten Byzantinischen Übereinstimmungs-

problems seit langem bewiesen, vgl. [LaSP_82]). Entweder geben die Teilnehmer auf oder sie wechseln, sofern möglich, die Schlüsselverteilzentrale.

Mehrere Schlüsselverteilzentralen in Serie: Werden zwischen den Teilnehmern Inkonsistenzen ihrer Schlüssel festgestellt, so sollten auch die Schlüsselverteilzentralen untereinander die Konsistenz der Schlüssel auf genau die gleiche Weise testen und, wenn möglich, verändernd angreifende Schlüsselverteilzentralen umgehen. Dies entspricht dem Wechseln der Schlüsselverteilzentrale bei „einer Schlüsselverteilzentrale".

Mehrere Schlüsselverteilzentralen parallel: Stellen die Teilnehmer eine Inkonsistenz der Summe ihrer Schlüssel fest, so wiederholen sie das Testprotokoll für Schlüsselgleichheit für jeden einzelnen der addierten Schlüssel und lassen alle inkonsistenten in der Summe weg. Bleiben nicht mehr genug zu addierende Schlüssel übrig, sollten die Teilnehmer entweder zusätzliche Schlüsselverteilzentralen verwenden oder ggf. auf ihre Kommunikation unter diesen Umständen verzichten. Sonst kann ein verändernder Angreifer zwischen den Teilnehmern die Aufnahme aller ihm unbekannten Schlüssel in die Summe verhindern.

f) Lösungsideen (möglicherweise nicht vollständig, weitere sind erwünscht):

1. Schlüssel haben von Anfang an einen Teil, der ihren Verfallszeitpunkt angibt. Dann brauchen die Teilnehmer zu ihrer Sicherheit nur noch richtig gehende Uhren. Nachteil: Wird ein Schlüssel vor seinem Verfallszeitpunkt kompromittiert, dann ist seine Verwendung so nicht zu stoppen.

   (Eine verwandte Lösung wäre, die maximale Anzahl von Benutzungen des Schlüssels festzulegen. Dies macht allerdings nur bei symmetrischen kryptographischen Systemen Sinn, wo alle – beiden – Betroffenen jeweils mitzählen können. Sie müssen sich dann allerdings auch abgelaufene Schlüssel auf Dauer merken.)
2. Expliziter Rückruf von Schlüsseln – sei es durch den Schlüsselinhaber selbst oder die seinen öffentlichen Schlüssel zertifizierende Stelle(n). Realisiert werden kann dies mittels einer digital signierten Nachricht, die neben dem zurückgerufenen öffentlichen Schlüssel bzw. Zertifikat möglichst auch den genauen Zeitpunkt des Rückrufs und natürlich den Rückrufer enthalten sollte. (Bei Rückruf durch den Schlüsselinhaber selbst ist die Authentisierung dieses Rückrufs die letzte Aktion mit dem zugehörigen geheimen Signierschlüssel.)

   Expliziter Rückruf von Schlüsseln geht natürlich nur bezüglich der Teilnehmer, die ein verändernder Angreifer vom Rückrufer nicht abschneiden kann. Diese zusätzliche Einschränkung gegenüber 1. ist der Preis für die größere Flexibilität.

   (Allerdings gibt es Protokolle, mit deren Hilfe die Teilnehmer ihre Isolation schnell erkennen können: Sie senden sich spätestens nach einer jeweils zu vereinbarenden Anzahl Zeiteinheiten eine Nachricht, wobei alle Nachrichten durchnumeriert und authentisiert sind (inkl. ihrer Durchnumerierung!). Kommt dann zu lange keine Nachricht oder fehlen zwischendrin welche, dann wissen die Teilnehmer, daß sie jemand zumindest temporär zu isolieren trachtet(e).)
3. Wir kombinieren 1. und 2., so daß wir die Vorteile von beidem haben.
4. Vor Verwendung eines Schlüssels wird beim Schlüsselinhaber nachgefragt, ob er noch aktuell ist. Nachteil: Wenn Teilnehmer(geräte) nicht immer online sind, entstehen möglicherweise lange zusätzliche Verzögerungszeiten, falls auf die Antwort ohne Timeout gewartet wird.
5. Variante von 4., indem nur eine relativ kurze Zeit auf eine Antwort gewartet wird. Da ein verändernder Angreifer eine Antwort unbemerkt für eine kurze Zeit unterdrücken kann, sollte dies mit 3. kombiniert werden.

g) Worauf muß beim Austausch der öffentlichen Schlüssel geachtet werden?

Die öffentlichen Schlüssel müssen *authentisch* sein, d.h. der Empfänger muß ihre Authentizität prüfen (können).

Auf wen könnte die Funktion des vertrauenswürdigen öffentlichen Schlüsselregisters verteilt werden?

Die Funktion des vertrauenswürdigen öffentlichen Schlüsselregisters könnte auf manche der anderen *Teilnehmer* verteilt werden.

Wie erfolgt dann der Schlüsselaustausch?

Statt sich den Zusammenhang zwischen seiner Identität und seinem öffentlichen Schlüssel von einem (oder auch mehreren, vgl. Aufgabenteil c)) öffentlichen Schlüsselregister(n) durch dessen (deren) digitale Signatur(en) zertifizieren zu lassen, läßt sich der Teilnehmer diesen Zusammenhang von allen Teilnehmern zertifizieren, die ihn kennen und mit denen er gerade auf sichere Weise kommunizieren kann, z.B. außerhalb des zu sichernden Systems. Diese Zertifikate werden zusammen mit dem öffentlichen Schlüssel weitergegeben. Der Empfänger eines solchermaßen zertifizierten öffentlichen Schlüssels muß nun entscheiden, ob er dessen Authentizität trauen will. Hierzu können ihm in direkter Weise die Zertifikate dienen, deren Testschlüssel er bereits kennt und für authentisch hält. Damit ihm die anderen Zertifikate nützen, benötigt er wiederum für deren Testschlüssel Zertifikate, und so fort. Ein öffentlicher Schlüssel *t* wird einem Teilnehmer *T* in der Regel desto vertrauenswürdiger sein, von je mehr Teilnehmern (in [Zimm_93] *Vorsteller* (*introducer*) genannt) Schlüssel *t* in für Teilnehmer *T* überprüfbarer Weise zertifiziert wurde. Das einzelne Zertifikat wiederum wird *T* desto vertrauenswürdiger sein, je weniger andere Zertifikate *T* zu seiner Authentizitätsprüfung heranziehen muß.

Was ist zu tun, wenn nicht jeder jedem gleich viel vertraut?

Statt alle Teilnehmer als Vorsteller für gleich vertrauenswürdig zu halten (ihnen also gleiche Ehrlichkeit, Sorgfalt, sowie eine gleich vertrauenswürdige lokale Rechenumgebung zu unterstellen, zumindest im Ergebnis), können ihnen vom empfangenden Teilnehmer individuelle Wahrscheinlichkeiten für Vertrauenswürdigkeit zugeordnet werden. Für jede Zertifikatskette kann dann wiederum eine Wahrscheinlichkeit für Vertrauenswürdigkeit berechnet werden, ebenso wie für jeden mehrfach zertifizierten öffentlichen Schlüssel. Der benutzende Teilnehmer kann dann jeweils entscheiden, ob ihm für die gerade beabsichtigte Anwendung die errechnete Wahrscheinlichkeit für Vertrauenswürdigkeit ausreicht.

Was ist zu tun, wenn ein Teilnehmer entdeckt, daß sein geheimer Schlüssel anderen bekannt wurde?

Gibt es keine zentral-hierarchische Schlüsselverteilung, dann gibt es wohl auch keine zentral-hierarchische Liste aller ungültigen Schlüssel. Also muß unser armer Teilnehmer seine signierte Nachricht, „mein Schlüssel ist ab sofort ungültig“ an alle Netzteilnehmer schicken und sich, wenn es um Rechtsverbindliches gehen soll, den Empfang dieser *Schlüssel-Widerrufs-Nachricht* auch noch von allen mit Datum und Uhrzeit quittieren lassen. (Eine naheliegende, aber falsche Lösung wäre: Es müßte jeder vor Benutzung eines Schlüssels dessen Inhaber fragen, ob sein Schlüssel noch gültig ist. Leider kann jetzt aber jeder, der den geheimen Schlüssel kennt, die Antwort erteilen und signieren: „Aber natürlich ist mein öffentlicher Schlüssel noch gültig, ich passe doch auf meinen geheimen Schlüssel auf!“)

Was sollte ein Teilnehmer tun, wenn sein geheimer Schlüssel zerstört wurde?

Neben dem Generieren eines Schlüsselpaares und dem Bekanntmachen des öffentlichen sollte er allen mitteilen, daß sie seinen alten Schlüssel für neue Nachrichten nicht mehr benutzen. Damit nicht jeder auf diese Art fremde Schlüssel für ungültig erklären kann,

sollte diese Nachricht signiert sein – am besten mit dem gerade als zerstört angenommenen Schlüssel. Also sollte nach Generierung eines Schlüsselpaares als allererstes diese Nachricht „Bitte meinen Schlüssel *t* nicht mehr benutzen." signiert werden. Sie ist nicht so geheimhaltungsbedürftig wie der geheime Schlüssel des Schlüsselpaares, für den man in vielen Anwendungen wohl eher keine Kopie halten wird. Diese Nachricht kann also in mehreren Kopien gespeichert werden. Oft dürfte es auch sinnvoll sein, sie mittels eines Schwellwertschemas (vgl. §3.9.6) so auf mehrere Bekannte zu verteilen, daß nur einige zusammen sie rekonstruieren können. Dies (und ihre anschließende Veröffentlichung) geschieht auf Bitten des Teilnehmers, wenn sein geheimer Schlüssel zerstört wurde.

Was ist zu tun, wenn ein Teilnehmer entdeckt, daß er eine falsche Zuordnung Teilnehmer <—> öffentlicher Schlüssel zertifiziert hat?

Der Teilnehmer erklärt sein Zertifikat mittels einer signierten Nachricht für ungültig. Bezüglich der Verteilung dieser *Zertifikat-Widerrufs-Nachricht* gilt das oben bzgl. des Bekanntwerdens des geheimen Schlüssels Gesagte.

Kann man das in dieser Teilaufgabe entwickelte Verfahren mit dem aus Teilaufgabe c) kombinieren?

Das in dieser Teilaufgabe entwickelte Schlüsselaustauschverfahren kann problemlos mit dem in Teilaufgabe c) und d) entwickelten Verfahren kombiniert werden. Zentralhierarchische Schlüsselregister und anarchische Schlüsselverteilung à la PGP ergänzen sich also. Die erreichbare Sicherheit ist so höher als mit einem der Verfahren allein.

3-4a Welche Sorte von Trust ist für welche Funktion eines TrustCenters nötig?

Blindes Vertrauen in eine Instanz ist immer dann nötig, wenn sie beobachtend angreifen kann. Kann eine Instanz nur verändernd angreifen, dann kann man die Abläufe oftmals so gestalten, daß der betroffene Teilnehmer dieses verändernde Angreifen bemerkt und im Umkehrschluß, wenn er keinen verändernden Angriff feststellt, sicher sein kann, daß sein Vertrauen bisher gerechtfertig war.

Schlüsselgenerierung erfordert blindes Vertrauen, denn der Generier kann lokal immer noch eine Kopie des Schlüssel(paare)s speichern und ggf. heimlich weitergeben.

Schlüsselzertifizierung erfordert nur überprüfendes Vertrauen, denn bei einer falschen Schlüsselzertifizierung bekäme der betroffene Teilnehmer das falsche Zertifikat beim ersten Streit vorgelegt. Bei geeignet gestalteten Abläufen, vgl. §3.1.1.2, kann er diesen verändernden Angriff sogar beweisen, erleidet also keinen nachhaltigen Schaden.

Verzeichnisdienst erfordert nur überprüfendes Vertrauen, denn das Bereithalten falscher Schlüssel ist beweisbar, wenn die Antwort auf Abfragen vom Verzeichnisdienst digital signiert wird.

Sperrdienst erfordert nur überprüfendes Vertrauen, vgl. Verzeichnisdienst.

Zeitstempeldienst erfordert nur überprüfendes Vertrauen, wenn eine geeignet verteilte Logdatei geführt wird, da dann der Zeitstempeldienst nicht rückdatieren kann, vgl. [BaHS_92, HaSt_91].

Auch diese Überlegungen legen also nahe, wenn schon Sicherheitsfunktionen in einer Instanz gebündelt werden, möglicherweise Schlüsselzertifizierung, Verzeichnisdienst, Sperrdienst und Zeitstempeldienstauf zu bündeln, jedoch auf keinen Fall die Schlüsselgenerierung, vgl. auch die Aufgaben 3-1 und 3-2a.

3-5 Hybride Verschlüsselung

Fall 1:

a) Generierung eines Schlüssels *k* eines schnellen symmetrischen Konzelationssystems, z.B. DES. Verschlüsselung von *k* mit dem öffentlichen Chiffrierschlüssel c_E des Empfängers *E*. Verschlüsseln der Datei mit *k*. Übertragen wird also: $c_E(k)$, *k*(Datei).

   Empfänger entschlüsselt ersten Teil mit seinem Dechiffrierschlüssel d_E, erhält so *k* und benutzt *k*, um die Datei zu entschlüsseln.

b) Generierung eines Schlüssels *k* eines schnellen symmetrischen kryptographischen Systems, das Konzelation *und* Authentikation bewirkt, z.B. DES in der Betriebsart PCBC, vgl. §3.8.2.5 (wenn Sie lieber zwei reinrassige kryptographische Systeme nehmen, müssen Sie im folgenden *k* durch k_1,k_2 ersetzen und wie in Aufgabe 3-13 verfahren). Signieren von *k* (evtl. zusätzlich: Uhrzeit, etc.) mit dem eigenen geheimen Signierschlüssel s_S (S wie Sender) und danach Verschlüsseln des signierten *k* mit dem öffentlichen Chiffrierschlüssel c_E des Empfängers *E*. (Erst konzelieren und danach signieren ginge auch.) Verschlüsselung und Authentikation der Datei mit *k*. Übertragen wird also: $c_E(k,s_S(k))$, *k*(Datei). (Falls das asymmetrische Konzelationssystem weniger Aufwand macht, wenn die Signatur $s_S(k)$ nicht mit ihm konzeliert wird, kann das Nachrichtenformat $c_E(k)$, $k(s_S(k)$,Datei) günstiger sein.)

   Empfänger *E* entschlüsselt ersten Teil mit seinem Dechiffrierschlüssel d_E, erhält so das signierte *k*, prüft die Signatur mit dem ihm bekannten Testschlüssel t_S. Ist die Signatur in Ordnung, benutzt *E* den Schlüssel *k*, um die Datei zu entschlüsseln und ihre Authentizität zu prüfen.

   (Für die reinrassigen kryptographischen Systeme: Sei k_1 Schlüssel eines symmetrischen Konzelationssystems, k_2 Schlüssel eines symmetrischen Authentikationssystems. Dann wird bei der oben erwähnten Ersetzung von *k* durch k_1,k_2 übertragen: $c_E(k_1,k_2,s_S(k_1,k_2))$, k_1(Datei,k_2(Datei)). Mensch sieht, daß die Authentikation von k_1 nicht nötig ist. $c_E(k_1,k_2,s_S(k_2))$, k_1(Datei,k_2(Datei)) tut es also auch.)

c) Wie b), nur daß das symmetrische kryptographische System hier lediglich Authentikation bewirken muß.

Fall 2:

Hier kann *S* natürlich genauso verfahren wie bei Fall 1, d.h. für jede Datei einen neuen Schlüssel *k* erzeugen, ihn asymmetrisch verschlüsseln, usw. *E* muß dann auch genauso wie bei Fall 1 verfahren. Alternativ kann *S* den bei der ersten Dateiübertragung generierten Schlüssel *k* wiederverwenden – und dies *E* mitteilen. So kann erheblicher Aufwand gespart werden: Keine zusätzliche Schlüsselgenerierung, Schlüsselverschlüsselung, Schlüsselübertragung und Schlüsselentschlüsselung – und bei b) und c) auch kein Schlüsselsignieren, keine Signaturübertragung und auch kein Signaturtest.

Fall 3:

Hier kann *E* genauso verfahren wie *S* in Fall 1, d.h. einen neuen Schlüssel *k* erzeugen, ihn für *S* asymmetrisch verschlüsseln, usw. Möchte *E* wie *S* in Fall 2 den bei der allerersten Dateiübertragung verwendeten symmetrischen Schlüssel *k* wiederverwenden, ist Vorsicht geboten: *E* hat *k* nicht selbst erzeugt und muß sich also vergewissern, daß *k* nur dem intendierten Empfänger seiner Datei bekannt ist. Bei b) und c) ist dies *E* durch die digitale Signatur von *S* klar. Bei a) kann *E* nicht sicher sein, wer *k* generiert hat, denn hier ist *k* in keiner Form authentisiert, insbesondere nicht durch *S*. Wenn bei a) *E* nun nicht einen neuen symmetrischen Schlüssel *k*' erzeugt, sondern den alten Schlüssel *k* nimmt, dann könnte ein Angreifer *A* , der ursprünglich *k* generierte und die allererste Datei an *E* schickte, die an *S* adressierte Nachricht *k*(Datei von *E*) abfangen und mit dem Schlüssel *k* entschlüsseln.

Was lernen wir daraus? Ein Protokoll, in dem ein Schlüssel zufällig generiert wird, und das dann sicher ist, kann nicht ohne weitere Überlegung mit einem *vorhandenen* Schlüssel verwendet werden. Alles getreu dem Merksatz: Alle Schlüssel, Startwerte etc. in der Kryptographie sollten zufällig und unabhängig voneinander gewählt werden – es sei denn, sie müssen abhängig gewählt werden oder die abhängige Wahl ist genauestens analysiert und in den Sicherheitsnachweis einbezogen.

3-6 Situationen aus der Praxis (zur Systematik der kryptographischen Systeme)

a) Ein *symmetrisches Authentikationssystem* reicht, um Fälschungen bei sich vertrauenden Partnern zu erkennen. Zwischen zwei alten Bekannten ist direkter Schlüsselaustausch problemlos. Das System ist weder sehr sicherheitskritisch noch zeitkritisch.
Um die Mietoptionen und Präferenzen der Pfitzleute vor neugierigen Vermietern, die selbst oder deren Freunde als Fernmeldetechniker beschäftigt sind, geheim zu halten, kann zusätzlich auch symmetrische Konzelation verwendet werden. Bei der Auswahl der kryptographischen Systeme hat Familie Pfitzmann also die Qual der Wahl.

b) *Symmetrisches Konzelationssystem,* direkter Schlüsselaustausch, Sicherheit kritisch (Kryptographen!), Zeit unkritisch. Verwenden wird David ein one-time-pad oder Pseudo-one-time-pad – wenn's nur um Ideen klauen geht. Hat David auch ein Interesse daran, daß seine Ideen unverändert bei den ihm vertrauenswürdigen Kryptographen ankommt, dann ist zusätzlich noch Authentikation nötig.
Da David die ihm vertrauenswürdigen Kryptographen vermutlich alle schon einmal getroffen hat, dürfte der direkte Schlüsselaustausch möglich sein. Es stellt sich allerdings die Frage, ob David seine Idee für jeden vertrauenswürdigen Kryptographen extra verschlüsseln will – was bei kanonischer Schlüsselverteilung (mit jedem Kryptographen einen anderen Schlüssel) nötig ist. Hat David vor, öfter Ideen vertraulich an eine bestimmte Gruppe zu schicken, dann kann er allen in dieser Gruppe denselben Schlüssel geben – das spart ihm Arbeit beim Verschlüsseln. Bzgl. Vertraulichkeit hat dies keine Nachteile, denn den Klartext erfährt sowieso jeder in dieser Gruppe. Problematisch wird es bei solch einem Gruppenschlüssel, wenn David jemand aus der Gruppe ausschließen will – dann muß er allen, die in der Gruppe bleiben sollen, einen neuen Schlüssel zukommen lassen. Ebenfalls problematisch ist solch ein Gruppenschlüssel bzgl. Authentikation. Hier könnte dann jede(r) in der Gruppe mit dem symmetrischen Authentikationssystem die neue Idee authentisieren und so an Davids gutem Ruf, daß seine Ideen gut, schön und neu sind, kratzen. Das ist bei paarweiser Schlüsselkenntnis nicht möglich.

c) Es reicht ein *symmetrisches Authentikationssystem,* wobei hier gar kein Schlüsselaustausch nötig ist; dafür ist Zeit sehr kritisch, und Platz auch; es bietet sich an, DES zu verwenden (wobei sich die Betriebsart CBCauth anbietet, siehe §3.8.2.2).

d) Der Rechnerfan sollte ein *digitales Signatursystem* verwenden. Der Schlüsselaustausch muß mittels Zentrale oder Telefonbuch geschehen, damit der Verkäufer den richtigen Testschlüssel des Käufers kennt und auch sicher ist, daß die Zuordnung Käufer – Testschlüssel juristisch bindend ist. Da das kryptographische System zeitunkritisch, aber sicherheitskritisch ist, sollte der Rechnerfan GMR verwenden.

e) *Konzelationssystem* zur Wahrung von Geschäftsgeheimnissen (evtl. auch schon *Authentikation*, siehe unten, aber *symmetrisch* reicht noch, weil es sich nicht um einen rechtsverbindlichen Vertrag handelt). Die Schlüsselverteilung dürfte über ein öffentliches Schlüsselregister erfolgen. Also wird wegen des Schlüsselaustausches ein asymmetrisches Konzelationssystem benötigt, beispielsweise RSA (oder ein *hybrides,* vgl. §3.1.4, etwa RSA kombiniert mit DES). Findet die Anfrage über Telefax statt, was heutzutage maximal mit 9600 bit/s, im ISDN mit 64 kbit/s arbeitet, ist die Anwendung nicht zeitkritisch. Verwenden die Firmen Breitbandfax (habe den Begriff noch nie gehört, er wird den Marketingleuten der Telekom aber

sicher gefallen), dann ist der Einsatz eines hybriden Konzelationssystems (vgl. §3.1.4) nötig. Die Sicherheit von RSA und DES dürfte für die nicht sehr sicherheitskritische Anwendung vollkommen ausreichen.

Wird auf Authentikation gänzlich verzichtet, so kann mit folgendem Angriff die Wahrung von Geschäftsgeheimnissen auch bei perfektem Konzelationssystem vereitelt werden: Um die Preise von Konkurrenten in Erfahrung zu bringen, sendet ein Angreifer ihnen einfach eine entsprechende Anfrage (inkl. Schlüssel des Konzelationssystems). Werden Netzadressen allein als „Absender" und damit Authentikation verwendet, so muß der Angreifer nur die Antwortnachricht abfangen, damit der vorgebliche Empfänger keinen Verdacht schöpft und der Angreifer selbst den Schlüsseltext erhält. Wird der „Absender" vom Netz generiert, dann muß der Angreifer ihn halt passend auf die Leitung des Angegriffenen aufmodulieren.

Bei der Antwort ist zusätzlich zur Konzelation asymmetrische Authentikation nötig, damit das Angebot rechtsverbindlich ist, vgl. d).

3-7 Vernam-Chiffre

a) Ausgetauscht sei der Schlüssel 0 0 1 1 1 0 0 1. Zu verschlüsseln sei der Klartext 1 0 0 0 1 1 1 0. Durch Addition modulo 2 berechnet der Verschlüsseler hieraus den Schlüsseltext: 1 0 1 1 0 1 1 1. Durch Addition von Schlüssel und Schlüsseltext modulo 2 erhält der Entschlüsseler wieder den Klartext. Als Rechnung geschrieben lautet dies:

Verschlüsseler:

	Klartext	1 0 0 0 1 1 1 0
⊕	Schlüssel	0 0 1 1 1 0 0 1
	Schlüsseltext	1 0 1 1 0 1 1 1

Entschlüsseler:

	Schlüsseltext	1 0 1 1 0 1 1 1
⊕	Schlüssel	0 0 1 1 1 0 0 1
	Klartext	1 0 0 0 1 1 1 0

b) Seien Klartext x, Schlüsseltext S und Schlüssel k Elemente der Gruppe. Nach Definition der Vernam-Chiffre gilt: Der Verschlüsseler berechnet $x+k =: S$. Der Entschlüsseler berechnet $S–k =: x$ (hierbei werde die Addition des inversen Elementes als Subtraktion geschrieben). Erfährt ein Angreifer S, so gibt es für jeden Klartext x' genau einen passenden Schlüssel k', da in jeder Gruppe die Gleichung $x'+k' = S$ nach k' aufgelöst werden kann: $k' = –x'+S$. Da die Schlüssel aus Sicht des Angreifers alle gleich wahrscheinlich sind, erfährt er durch den Schlüsseltext S also nichts über den Klartext x.

Anmerkungen:

1. Der Beweis verwendet keine Kommutativität, gilt also für *beliebige* Gruppen. Prüfen Sie, ob Ihrer evtl. nur für *abelsche* Gruppen gilt.
2. Der Beweis ignoriert, daß je nach Kommunikationsprotokoll der Angreifer durchaus etwas über den Klartext erfährt: Üblicherweise seine Länge, zuallermindest seine maximale Länge. Ersteres kann vermieden werden, indem Nachrichten auf Standardlängen aufgefüllt und/oder durch bedeutungslose Nachrichten ergänzt werden, vgl. §5.4.3, am besten wird ein *gleichmäßiger Zeichenstrom* übertragen, damit ein Abhörer nicht beobachten kann, wann keine Nachrichten übertragen werden, vgl. §5.2.1.1.

c) Das Ergebnis ist nur dann eindeutig, wenn die Verschlüsselungsoperation als vereinbart unterstellt wird. (Diese Vereinbarung kann als Teil der Festlegung des Konzelationssystems oder des Schlüssels betrachtet werden.) Ist als Operation Addition modulo 2 unterstellt, ergibt sich als Klartext

	Schlüsseltext	0 1 0 1 1 1 0 1
⊕	Schlüssel	1 0 0 1 1 0 1 1
	Klartext	1 1 0 0 0 1 1 0

Ist hingegen als Operation Addition modulo 16 unterstellt, ergibt sich ein anderer Klartext

	Schlüsseltext	0101 1101
−(16)	Schlüssel	1001 1011
	Klartext	1100 0010

falls zeichenweise (4-Bit-weise) von links, sonst ergibt sich

	Schlüsseltext	1011 1010
−(16)	Schlüssel	1101 1001
	Klartext	1110 0001

nach unserer Konvention bitweise geschrieben 1000 0111.

d) Rechnung modulo 2: Da jedes Bit einzeln ver- bzw. entschlüsselt wird, gilt: Invertiere drittes Bit von rechts, d.h. übertrage 01011101 01011101 01011**0**01.

(Modulo jeder Zahl bewirkt eine Änderung des Schlüsseltextes von S zu S' eine Änderung des Klartextes um die gleiche Differenz $d := S' - S$, denn S' wird entschlüsselt als $x' = S'-k = (S+d)-k = (S-k)+d = x+d$.)

Die folgenden Rechnungen gehen davon aus, daß zeichenweise von links nach rechts geschrieben wird.

Rechnung modulo 4: Da immer Zweiergruppen von Bits ver- bzw. entschlüsselt werden, ist zu klären, was mit dem dritt- und viertletzten Bit zu geschehen hat. Wird zu ihnen 01 modulo 4 addiert, so wird das drittletzte Bit sicher invertiert, aber auch das viertletzte, sofern ein Übertrag auftritt. Kennt der Angreifer das drittletzte Klartextbit nicht, kann er die Änderung des viertletzten Klartextbits nicht deterministisch vermeiden. Da in der Aufgabenstellung über das viertletzte Bit nichts gesagt wurde, kann man folgendes als Lösung akzeptieren: Der Angreifer überträgt 01011101 01011101 0101**00**01. Soll bei unbekanntem drittletzten Klartextbit die Änderung des viertletzten deterministisch vermieden werden, ist die Aufgabe modulo 4 unlösbar.

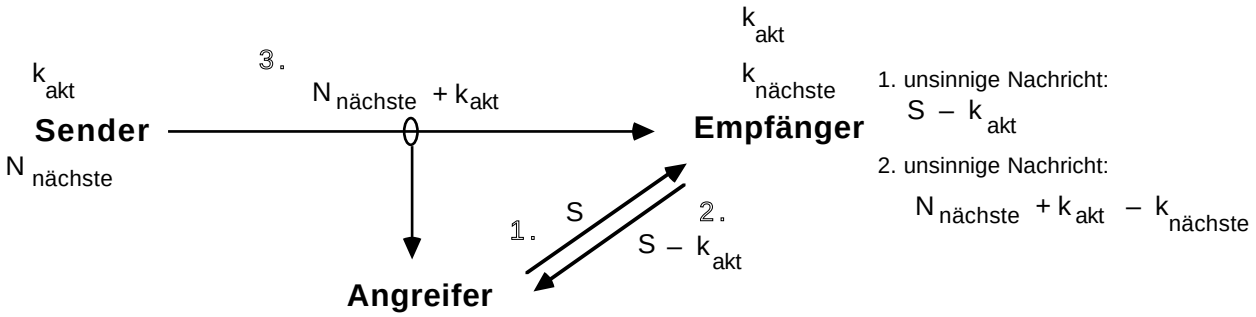
Rechnung modulo 8: Da immer Dreiergruppen von Bits ver- bzw. entschlüsselt werden, ist zu klären, was mit den drei letzten Bits zu geschehen hat. Wird zu ihnen 100 modulo 8 addiert, so wird das drittletzte Bit sicher invertiert. Der Angreifer überträgt also 01011101 01011101 01011**001**.

Rechnung modulo 16: Da immer Vierergruppen von Bits ver- bzw. entschlüsselt werden, ist zu klären, was mit den vier letzten Bits zu geschehen hat. Wird zu ihnen 0100 modulo 16 addiert, so wird das drittletzte Bit sicher invertiert, aber auch das viertletzte, sofern ein Übertrag auftritt. Kennt der Angreifer das drittletzte Klartextbit nicht, kann er die Änderung des viertletzten Klartextbits nicht deterministisch vermeiden. Da in der Aufgabenstellung über das viertletzte Bit nichts gesagt wurde, kann man folgendes als Lösung akzeptieren: Der Angreifer überträgt 01011101 01011101 0101**0001**. Soll bei unbekanntem drittletzten Klartextbit die Änderung des viertletzten deterministisch vermieden werden, ist die Aufgabe modulo 16 unlösbar.

e) Da jeder Abschnitt des Schlüssels vom Angegriffenen nur einmal verwendet wird und alle Abschnitte stochastisch unabhängig voneinander generiert werden, kann ein Angreifer aus früheren Reaktionen nichts über zukünftige lernen. (Im Prinzip gibt's natürlich den Angriff, nur nützt er nichts.) Gleiches gilt bzgl. (nicht adaptiven) aktiven Angriffen.

Es sei allerdings angemerkt, daß beide Sorten aktiver Angriffe bezogen auf *Nachrichten* erfolgreich sein können: Durch einen aktiven Angriff (gewählter Schlüsseltext-Klartext-Angriff) auf den Empfänger ermittelt der Angreifer den Wert des für den Sender gerade aktuellen Schlüsselabschnitts k_{akt}. Dann kann er die nächste vom Sender verschlüsselte Nachricht $N_{nächste}$ entschlüsseln. Die Lehre hieraus ist: Der Partner, der entschlüsselt, darf entschlüsselte Nachrichten (Klartexte) nicht einfach ausgeben und hierdurch einen aktiven Angriff ermöglichen. Von dieser Einschränkung kann abgewichen werden, wenn die empfangene Nachricht

richtig authentisiert ist (sinnvollerweise mit einem informationstheoretisch sicheren Authentikationscode).



$$k_{akt} := S - (S - k_{akt})$$

$$N_{nächste} := N_{nächste} + k_{akt} - k_{akt}$$

Eine zusätzliche Anmerkung: Bei der Vernam-Chiffre müssen Schlüssel (oder zumindest Schlüsselabschnitte) vom einen Partner nur zum Ver- und vom andern nur zum Entschlüsseln verwendet werden. Keinesfalls genügt es, daß zwei Partner einen gemeinsamen Schlüssel haben und immer der, der dem anderen gerade eine Nachricht senden will, verwendet hierfür als „Schlüssel" den nächsten Schlüsselabschnitt. Sonst kann es passieren, daß beide Partner *gleichzeitig* eine Nachricht senden, so daß ein Angreifer die Differenz ihrer Klartexte erfährt.

f) Da es hier um Eigenschaften der Ver-/Entschlüsselungsfunktion an sich, d.h. unabhängig von einzelnen Schlüsseln, geht, ist die in §3.1.1.1 eingeführte abgekürzte Schreibweise ungeschickt (in ihr kommt die Ver-/Entschlüsselungsfunktion an sich ja gerade nicht vor), so daß wir zur ebenfalls dort erläuterten ausführlicheren Notation übergehen. Zunächst sei die Definition (1) in der ausführlicheren Notation hingeschrieben:

Definition für informationstheoretische Sicherheit für den Fall, daß alle Schlüssel mit gleicher Wahrscheinlichkeit gewählt werden:

$$\forall S \in \mathcal{S}\ \exists const \in \mathbb{N}\ \forall x \in \mathcal{X}:\ \left|\{k \in \mathcal{K} \mid ver(k,x) = S\}\right| = const. \qquad (1)$$

Wir zeigen: Zumindest wenn, wie bei *allen* Vernam-Chiffren, Schlüsselraum = Klartextraum = Schlüsseltextraum ist[143], ist *notwendig* und *hinreichend*: Die Funktion *ver* muß bei Festhalten eines der beiden Eingabeparameter und Variation des anderen den gesamten Wertebereich durchlaufen.

1 (*k* festgehalten) Bei festem *k* muß jeder Schlüsseltext eindeutig entschlüsselt werden können, also nur bei einem Klartext vorkommen (d.h. *ver*(*k*, •) injektiv). Da es genausoviele Schlüsseltexte wie Klartexte gibt, ist äquivalent, daß jeder Schlüsseltext mindestens einmal vorkommt (d.h. *ver*(*k*, •) surjektiv).

2a (*x* festgehalten, Durchlaufen ist *notwendig*) Aus Definition (1) folgt, daß hinter jedem überhaupt je vorkommenden Schlüsseltext *S* jeder Klartext stecken kann. Aus 1. folgt, daß jeder Wert *S* des Schlüsseltextraums überhaupt je vorkommt. Zusammen lautet dies formal geschrieben: $\forall S\ \forall x\ \exists k$: $ver(k, x) = S$. Dies ist äquivalent zu $\forall x\ \forall S\ \exists k$: $ver(k, x) = S$, d.h. daß bei festgehaltenem *x* alle *S* vorkommen müssen.

2b (*x* festgehalten, Durchlaufen ist *hinreichend*) Kommen umgekehrt bei festgehaltenem *x* alle *S* vor, so folgt die Sicherheit: Ist *S* gegeben, so gibt es also zu jedem *x* ein *k*, so daß *ver*(*k*,

[143] Klartextzeichen, Schlüsselzeichen und Schlüsseltextzeichen sind Elemente der *gleichen* Gruppe, und zusätzlich sind Klartext, Schlüssel und Schlüsseltext jeweils *gleich lang*.

x) = S gilt. Um zu zeigen, daß auch alle x gleichwahrscheinlich sind, brauchen wir wieder, daß wegen der gleichen Größe der Räume $ver(k, x) = S$ nur für ein k gelten kann.

Für den binären Fall der Vernam-Chiffre suchen wir also 2x2-Funktionstafeln, in denen in jeder Zeile und Spalte eine 0 und eine 1 steht. Es gibt genau zwei, die die Funktionen XOR und NOT XOR darstellen. (Daß es genau zwei sind, sieht man z.B. so: Legt man für eine Wertekombination der beiden Eingabeparameter den Funktionswert fest, ist die Funktion klar, da durch Änderung eines Parameterwertes sich das Funktionsergebnis jeweils ändern muß. Da man für den ersten Funktionswert zwei Möglichkeiten hat, gibt es also zwei Funktionen.)

$S = k(x)$	x: 0	x: 1
k: 0	y	$\bar{y}$
k: 1	$\bar{y}$	y

$\bar{y}$ sei das Inverse zu y.

$S = k(x)$	x: 0	x: 1
k: 0	0	1
k: 1	1	0

Für $y = 0$: XOR

$S = k(x)$	x: 0	x: 1
k: 0	1	0
k: 1	0	1

Für $y = 1$: NOT XOR

g) Lieber jüngerer Freund!
Erstens kannst Du, wenn es Dir nicht intuitiv klar ist, in historischen Quellen [Sha1_49] nachlesen, daß natürliche Sprache zu weit mehr als der Hälfte aus *Redundanz* besteht und deshalb kein Ersatz für einen rein zufällig gewählten Schlüssel, d.h. einen maximaler *Entropie*, ist. Deshalb kann man, wenn natürliche Sprache auf diese Art und Weise verschlüsselt wird, den Sinn längerer Schlüsseltexte verstehen, ohne Buch, Seite und Zeile bestimmen zu müssen. Ja man kann sogar den Sinn der zur Verschlüsselung verwendeten Buchpassage verstehen und sie ggf. so ermitteln. (Ein genaues Verfahren zum vollständigen Brechen mittels eines reinen Schlüsseltext-Angriffs ist in [Hors_85 Seite 107 f.] skizziert. Vollständiges Brechen mittels eines Klartext-Schlüsseltext-Angriffs ist noch wesentlich einfacher.)

Zweitens solltest Du, wenn Du es in letzter Zeit nicht getan hast, Dich über die momentane und in Zukunft zu erwartende Rechenleistung und Speicherkapazität von PCs informieren. Dann wirst auch Du befürchten, daß man demnächst die gesamte Weltliteratur auf einigen optischen Speicherplatten speichern und dann durch vollständige Suche in einigen Stunden jeden maschinell erkennbare Redundanz enthaltenden Klartext, der nach Deinem Verfahren verschlüsselt wurde, automatisch entschlüsseln kann. (Sehr defensive Plausibilitätsrechnung: Weltliteratur habe 10^8 Bände zu im Mittel 1 MByte. Also werden bei 1990 üblichen 5,25"-optischen Platten mit je 600 MByte insgesamt $10^8/600$, also ungefähr 160 000 optische Platten benötigt. Deren Durchsuchen ab jeder Bitposition erfordert $10^8 \cdot 2^{20} \cdot 8 \approx 8 \cdot 10^{14}$ Schritte, die je etwa 10^{-6} s dauern dürften. Also dauert das vollständige Durchsuchen längstens $8 \cdot 10^8$ s, also längstens 92593 Tage bei einem Rechner heutiger Leistung. Beginnt man mit den wahrscheinlichen Werken der Weltliteratur, endet die Suche im Mittel weit schneller.)

Zum Schluß noch ein kleiner Trost: Du bist nicht der erste, dem solch ein Fehler unterläuft. Außerdem hattest Du Glück, durch Veröffentlichung Deines Verfahrens einen Freund zu finden, der Dich auf Deinen Fehler hinweist. Viele Deiner Kollegen, die entweder nichts mehr dazulernen wollen oder deren Auftraggeber Urheberrechte sichern wollen, halten Ihre Verfahren geheim. Du kannst Dir denken, was dann oft passiert.

h) Nein, denn man kann sich einen beliebigen anderen Klartext derselben Länge ausdenken und dann den Schlüssel so ausrechnen, daß er zu jenem Klartext und dem vorhandenen Schlüsseltext paßt. Die Strafverfolgungsbehörde erhält also keine sinnvolle Information.

i) Das eine ausgetauschte Speichermedium werde für Zweiwegkommunikation genutzt, so daß für jede Richtung nur die Hälfte der Speicherkapazität genutzt werden kann. Dann ergibt sich

	Textkommunikation (Tippgeschwindigkeit 40 bit/s)	Sprachkommunikation (normale ISDN-Codierung: 64 kbit/s)	Hochauflösende Bewegtbildkommunikation (140 Mbit/s)
3,5"-Diskette (1,4 MByte)	1,62 Tage	87,5 s	0,04 s
optische 5,25"-Platte (600 MByte)	694 Tage	10,4 h	17 s
Video-8-Streamertape (5 GByte)	5787 Tage	3,6 Tage	143 s
3,5"-Diskette (120 MByte)	139 Tage	2,1 h	3,4 s
DVD-RAM einseitig (2,6 GByte)	3009 Tage	1,9 Tage	74 s
Dual-Layer-DVD beidseitig (17 GByte)	19676 Tage	12,3 Tage	486 s

Austausch einer 1,4 MByte Diskette reicht also für Textkommunikation für die meisten Anwendungen bereits „auf Lebenszeit“ (denn nur wenige werden Ihnen so am Herzen liegen, daß Sie insgesamt 1,62 Tage nonstop für sie tippen), entsprechend reicht Austausch einer optischen 600 MByte Platte für viele Sprachkommunikations-Anwendungen (auch wenn viele Menschen ausdauernder reden als tippen). Für unkomprimierte hochauflösende Bewegtbildkommunikation sind die heutigen Massenspeicher noch zu klein, aber für komprimierte Bewegtbildkommunikation niedriger Auflösung wie im ISDN vorgesehen (ein zusätzlicher Kanal von 64 kbit/s für das Bild) durchaus brauchbar: Mit Hilfe eines Video-8-Streamertapes 1,8 Tage informationstheoretisch sicher seine(n) TelefonpartnerIn zu sprechen und zu betrachten, dürfte auch hier „auf Lebenszeit“ reichen. Für flüchtige Bekanntschaften tut's neuerdings eine 120 MByte Diskette – und für intensive Telebeziehungen künftig eine Dual-Layer-DVD.

Eine ganz andere Frage ist, ob der Lesezugriff auf die Medien schnell genug geht, um in Realzeit auf den Schlüssel zugreifen zu können. Für Text- und Sprachkommunikation kein Problem. Für hochauflösende Bewegtbildkommunikation mit 140 Mbit/s für alle Medien bisher nicht möglich – ein weiterer Grund, weshalb Bewegtbildsignale nicht nur verlustfrei, sondern sogar verlustbehaftet komprimiert werden. Ersteres bringt etwa den Faktor 4, bei letzterem kommt es darauf an, welche Qualitätseinbuße man gewillt ist, in Kauf zu nehmen. Wenn Sie Ihren Taschenrechner noch ausgepackt haben, dann können Sie die Zahlen für hochauflösende Bewegtbildkommunikation noch mal mit den typischen Werten von 34 Mbit/s für verlustfreie und 2 Mbit/s für verlustbehaftete Kompression durchrechnen. Was ergibt sich? Video-8-Streamertape und DVD sind für hochauflösende Bewegtbildkommunikation bei verlustbehafteter Kompression durchaus geeignet – sowohl für die Speicherung des One-Time Pads wie auch für die Speicherung des Bewegtbildsignals. Kein Wunder: Schlüssel, Schlüsseltext und Klartext sind gleich lang – und für letzteres wurden die Medien konzipiert.

3-7a Symmetrisch konzelierter Message Digest = MAC ?

a) Das Rezept ist vollkommen unsicher, wie das in der Aufgabenstellung genannte Beispiel zeigt: Aus der abgefangenen Nachricht kann jeder mittels des öffentlich bekannten Verfahrens die Prüfziffer berechnen.

Aus der Prüfziffer und ihrer ebenfalls abgefangenen Verschlüsselung mit der Vernam-Chiffre kann jeder durch Differenzbildung den aktuellen Schlüsselabschnitt bestimmen.

Zu jeder (vom Angreifer frei wählbaren) Nachricht kann mittels des öffentlich bekannten Verfahrens die Prüfziffer berechnet werden und diese dann mittels des aktuellen Schlüsselabschnitts zu einem „authentischen“ MAC verschlüsselt werden.

Erreicht wird durch diesen passiven Angriff ein selektives Brechen, was wie schon in §3.1.3.1 angemerkt, nicht akzeptabel ist.

b) Für die üblichen kollisionsresistenten Hashfunktionen ist dies nach heutigem Wissen eine sichere Konstruktion. Allerdings sind diese nicht nur kollisionsresistent, sondern sie haben noch eine darüber hinausgehende, für die Sicherheit der bei TLS verwendeten Konstruktion wesentliche Eigenschaft:

Ihr Funktionswert verrät *nichts* über die Eingabe.

Um zu verdeutlichen, warum dies wesentlich ist, nehmen wir das Gegenteil an: Die kollisionsresistente Hashfunktion gäbe 80 Bit ihrer Eingabe als Teil ihres Funktionswertes aus. Sind sowohl die Funktionswerte lang genug als auch die Eingaben, so kann solch eine Hashfunktion prima kollisionsresistent sein. Bei der bei TLS verwendeten Konstruktion eingesetzt verkürzt sie allerdings die Schlüssellänge um 80 Bit, wenn der Schlüssel „dummerweise“ an der passenden Stelle der Eingabe steht. Wird mit einer Schlüssellänge von 128 Bit gearbeitet, was normalerweise mehr als ausreichend ist, dann muß ein Angreifer höchstens 2^{48} mal probieren, um den Schlüssel zu bestimmen. Dazu ist heute nahezu jeder relevante Angreifer in der Lage.

c) Ja: Hashe (<Schlüssel>, <Nummer des Bytes>, <Byte>) und übertrage die Funktionswerte (ggf. zusammen mit <Nummer des Bytes>, falls der Empfänger nicht mitzählt oder das Kommunikationsmedium unzuverlässig ist). Der Empfänger muß nun nur noch die Werte des Byte durchprobieren, indem er (<Schlüssel>, <Nummer des Bytes>, <Wert des Byte>) hasht und mit dem übermittelten Funktionswert vergleicht, um den Funktionswert zum Klartextbyte zu entschlüsseln. Außer Sender und Empfänger kann dies niemand, da niemand sonst den Schlüssel kennt. Die Numerierung der Bytes verhindert, daß gleiche Bytes gleiche Funktionswerte ergeben, es wird so eine synchrone Stromchiffre erzeugt, vgl. §3.8.2. Im Beispiel wird angenommen, daß ein Byte einen sinnvollen Kompromiß zwischen der Anzahl der Versuche bei der Entschlüsselung durch den Empfänger und der Nachrichtenexpansion durch die wesentlich längeren Funktionswerte der Hashfunktion (typischerweise 160 Bit) und Nummer des Bytes (typischerweise 64 Bit) ist. Es ist klar, daß statt Bytes vom Bit bis (bei heutiger Technologie etwa) zum Langwort von 4 Bytes alles genommen werden kann.

3-8 Authentikationscodes

a) Ausgetauscht sei der Schlüssel 00 11 10 01. Zu sichern sei der Klartext H T T T.
Beim Authentikationscode wird dann H,0 T,1 T,0 T,1 als „Schlüsseltext“ generiert und übertragen.

b) Fängt ein Angreifer eine Kombination aus H oder T sowie MAC ab, so bleiben für ihn immer zwei (der vier) Schlüsselwerte ununterscheidbar, da H bzw. T in jeder Spalte der Tabelle, in der sie vorkommen, genau zweimal vorkommen. Genau diese zwei Werte unterscheiden sich aber darin, ob für den anderen Wert des „Münzwurfs“ der MAC den Wert 0 oder 1 haben muß. (Behauptung kann in dem einfachen Authentikationscode leicht durch vollständige Betrachtung aller Möglichkeiten verifiziert werden.) Deshalb hat der Angreifer keine bessere Strategie als Raten, was mit der Wahrscheinlichkeit 0,5 dazu führt, daß der Empfänger die Verfälschung des Angreifers erkennt.

Einfachere Darstellung: Wenn $k = k_1k_2$, so ist $k(\mathrm{H}) = k_1$ und $k(\mathrm{T}) = k_2$. (Dies ist im folgenden, der Aufgabenstellung entnommenen Bild veranschaulicht, in dem k_1 fett gezeichnet

ist.) Also kann der Authentikationscode wie rechts im Bild gezeichnet dargestellt werden: Hier ist der Wert des MAC in Abhängigkeit vom Schlüssel und Text dargestellt. Offensichtlich erfährt man also durch den aktuellen MAC für H nichts über den für T und umgekehrt.

Schlüssel \ Text mit MAC	H,**0**	H,**1**	T,0	T,1
00	H	–	T	–
01	H	–	–	T
10	–	H	T	–
11	–	H	–	T

Schlüssel \ Text	H	T
00	**0**	0
01	**0**	1
10	**1**	0
11	**1**	1

c) Nein, weil der Empfänger seine Schlüsselabschnitte ja in einer festen Reihenfolge verwendet.

d) *Beispiel:* Ausgetauscht sei der Schlüssel 00 11 10 01. Zu sichern sei der Klartext H T T T. Dann wird 00 11 01 00 als Schlüsseltext generiert und übertragen.
Bzgl. Authentikation gilt die obige Argumentation sinngemäß: Für jeden der nach einer Beobachtung des Angreifers möglichen beiden Schüsselwerte muß er bei der angestrebten Änderung des Klartextes unterschiedliche Schlüsseltexte nehmen. Er hat also wieder keine bessere Erfolgsmöglichkeit, als mit der Wahrscheinlichkeit 0,5 richtig zu raten.
Führt ein Angreifer einen verändernden Angriff durch, kann er allerdings in jedem Fall an der Reaktion des Angegriffenen (akzeptiert gefälschte Nachricht oder nicht) erkennen, welcher der beiden nach seiner ersten Beobachtung noch möglichen Schlüsselwerte vorliegt, und dadurch den Schlüsselwert genau bestimmen. Aus ihm und dem beobachteten Schlüsseltext ergibt sich der Klartext. Das sei an einem Beispiel erläutert: Der Angreifer fange den Schlüsseltext 00 ab und tippe von den möglichen zwei Schlüsselwerten auf den Wert 00, also auf den Klartext H. Er ändere den Klartext in T ab, indem er den Schlüsseltext 10 weiterschickt. Akzeptiert der Empfänger, war der Tipp des Angreifers richtig, der ursprünglich gesendete Klartext also in der Tat H. Akzeptiert der Empfänger nicht, war der Wert des Schlüssels 01 und damit der ursprünglich gesendete Klartext T.

e) Es wird ein zusätzliches Schlüsselbit verwendet. Ist es 0, bleibt alles wie bisher. Ist es 1, werden in der Tabelle genau alle H durch T ersetzt und umgekehrt. Selbst nach dem oben beschriebenen, bzgl. Informationsgewinn über den Schlüssel optimalen Angriff weiß der Angreifer den Wert des zusätzlichen Schlüsselbits nicht. Dieses Schlüsselbit wirkt als Vernam-Chiffre, so daß die Konzelation auch nach dem Angriff noch perfekt ist.
Leider ist das kombinierte System nun nicht mehr effizienter als Authentikationscode und Vernam-Chiffre, sofern erst das Nutzbit konzeliert und danach authentisiert wird. Macht man es andersherum, braucht man statt 3 Schlüsselbits 4.

f) Zu zeigen ist, daß es zu jedem möglichen Wert *MAC'* (d.h. $MAC' \in K$) genau einen mit der Beobachtung (m, *MAC*) verträglichen Wert (a, b) gibt. Das Gleichungssystem

$$MAC = a + b \bullet m$$
$$MAC' = a + b \bullet m'$$

mit den Unbekannten a, b hat genau dann genau eine Lösung, wenn $m \neq m'$ ist. $m \neq m'$ aber ist genau das Ziel des Angreifers.

Für diejenigen, die sich nicht mehr so genau an die Lineare Algebra erinnern, eine etwas ausführlichere Begründung: Ein Gleichungssystem C = M•X mit der Matrix M und den

Vektoren C und X ist bei vorgegebenem C und M genau dann eindeutig lösbar, wenn M regulär ist. M ist dann und nur dann regulär, wenn sich durch Zeilen und Spaltenoperationen aus M eine Dreiecksmatrix machen läßt (d.h. alle Diagonalelemente sind $\neq 0$ und entweder alle Werte unter der Diagonale oder alle Werte oberhalb der Diagonale sind 0). Dies wird für unsere Matrix

$$\begin{pmatrix} 1 & m \\ 1 & m' \end{pmatrix}$$

kurz vorgeführt: Subtraktion der ersten Zeile von der zweiten ergibt:

$$\begin{pmatrix} 1 & m \\ 0 & m'-m \end{pmatrix}$$

Dies ist genau dann eine Diagonalmatrix, wenn $m \neq m'$ ist.

Um einen subtilen Beweisfehler zu zeigen, wird nachfolgend noch ein anderer Beweis angegeben:

$$MAC := a + b \bullet m \quad \Leftrightarrow \quad a := MAC - b \bullet m$$

Also kann b beliebig gewählt werden, da durch a die Gleichung erfüllt werden kann. Also erfährt der Angreifer nichts über b. Bei Authentikation einer anderen Nachricht m' müßte er aber b kennen, um das MAC zu MAC' korrigieren zu können {, da $b \bullet m$ bei Variation von m seinen Wert beliebig und gleichverteilt ändert}.

Ohne den Zusatz der geschweiften Klammer im „Beweis“ wäre der Authentikationscode $MAC := a + b \bullet m^2$ ein nettes Gegenbeispiel: Denn der „Beweis“ trifft auch auf ihn zu, bei ihm kann aber jede Nachricht m durch $-m$ ersetzt werden, ohne daß das MAC geändert werden muß.

3-9 Mathematische Geheimnisse

a) Ganz klar ist es nicht, da man die Primzahlen nun nicht mehr mit gleicher Wahrscheinlichkeit wählt, sondern solche, die nach langen Lücken in der Primzahlfolge kommen, mit größerer, solche, die nur zwei größer sind als eine andere Primzahl, nur, wenn sie am Anfang als Zufallszahl gewählt wurden.

Für Spezialisten: Zumindest unter der Cramerschen Vermutung ist das Verfahren aber trotzdem sicher: Diese Vermutung (die ich aus [GoKi_86] kenne), besagt $\pi(x + \log^2 x) - \pi(x) > 0$. Danach erhält man die Primzahlen nach den längsten Lücken nur bis zu einem Faktor von $\log^2 x \approx l^2$ häufiger als die oberen aus Primzahlzwillingen. Für die Produkte n ergeben sich also Wahrscheinlichkeitsunterschiede bis zu l^4. Wenn es für diese Verteilung einen erfolgreichen Faktorisierungsalgorithmus gäbe, so wäre er bei der korrekten Verteilung höchstens um den Faktor l^4 weniger erfolgreich. Damit sinkt auch dessen Erfolgswahrscheinlichkeit nicht schneller als jedes Polynom, im Widerspruch zur echten Faktorisierungsannahme. (Ohne Cramersche Vermutung käme man künstlich wieder zu demselben Ergebnis, wenn man das Suchen in 2-er-Schritten nicht beliebig lange macht, sondern ein Abbruchkriterium verwendet, etwa nach l^2 2-er-Schritten.)

b) Man kann erwarten, daß die beiden Primzahlen sehr nah beieinanderliegen, also beide etwa $\sqrt{n}$ groß sind. Deswegen kann man die Zahlen in der Umgebung von $\sqrt{n}$ durchsuchen (d.h. n durch jede von ihnen teilen.)

Genauer: Es gilt in den meisten Fällen z.B. $q \leq p + 16l$ (Primzahlsatz mit etwas Spielraum, da im Mittel der Abstand zur nächsten Primzahl $l \bullet \ln(2)$ ist. Die besonders schöne Zahl 16 wurde gewählt, um das folgende Rechnen zu erleichtern).

In diesen Fällen ist also $p^2 < n < p(p + 16l) < (p + 8l)^2$, d.h. $p < \sqrt{n} < p + 8l$, also $\sqrt{n} - 8l < p < \sqrt{n}$. Man muß also nur diese $8l$ Zahlen durchsuchen und hat mit signifikanter, sogar großer Wahrscheinlichkeit faktorisiert.

c) $19 = (10011)_2$. Mod 101 gilt:

$3^{(1)_2} = 3$; $3^{(10)_2} = 3^2 = 9$; $3^{(100)_2} = 9^2 = 81 \equiv -20$; $3^{(1001)_2} = (-20)^2 \cdot 3 = 1200 \equiv -12$;

$3^{(10011)_2} = (-12)^2 \cdot 3 = 144 \cdot 3 \equiv 43 \cdot 3 \equiv 129 \equiv 28$.

d) $77 = 7 \cdot 11$; $\phi(77) = 6 \cdot 10 = 60$ und $3 \in \mathbb{Z}_{77}^*$, da ggT(3,77) = 1. Also sagt der kleine Fermatsche Satz: $3^{60} \equiv 1 \bmod 77$.
Also gilt: $3^{123} \equiv (3^{60})^2 \cdot 3^3 \equiv 1^2 \cdot 27 \equiv 27$.

e) Euklidscher Algorithmus:

$101 = 4 \cdot 24 + 5$;
$24 = 4 \cdot 5 + 4$;
$5 = 1 \cdot 4 + 1$.

Rückwärts:

$1 = 5 - 1 \cdot 4$ (Jetzt 4 durch 5 und 24 ersetzen)
$= 5 - 1 \cdot (24 - 4 \cdot 5) = -24 + 5 \cdot 5$ (Jetzt 5 durch 24 und 101 ersetzen)
$= -24 + 5 \cdot (101 - 4 \cdot 24) = 5 \cdot 101 - 21 \cdot 24$. (Probe: $5 \cdot 101 = 505$, $-21 \cdot 24 = -504$)

Also: $(24)^{-1} \equiv -21 \equiv 80$.

f) Ein Inverses von 21 mod 77 gibt's nicht, weil ggT(21, 77) = $7 \neq 1$.

g) Zunächst wenden wir wieder den erweiterten Euklidschen Algorithmus an, um u und v mit $u \cdot 7 + v \cdot 11 = 1$ zu suchen. (Auch wenn man es evtl. sofort sieht: –21+22=1):

$11 = 1 \cdot 7 + 4$;
$7 = 1 \cdot 4 + 3$;
$4 = 1 \cdot 3 + 1$.

Rückwärts:

$1 = 4 - 1 \cdot 3$
$= 4 - 1 \cdot (7 - 1 \cdot 4) = -7 + 2 \cdot 4$
$= -7 + 2 \cdot (11 - 1 \cdot 7) = 2 \cdot 11 - 3 \cdot 7$.

Damit sind für $p = 7$ und $q = 11$ die „Basisvektoren“: $up = -21 \equiv 56$, $vq = 22$.
Mit der Formel $y = up\, y_q + vq\, y_p$ ergibt sich: $y = -21 \cdot 3 + 22 \cdot 2 = (22-21) \cdot 2 - 21 = -19 \equiv 58$.
(Probe: $58 \equiv 2 \bmod 7 \wedge 58 \equiv 3 \bmod 11$.)

Für z können wir in die gleiche Formel einsetzen (up und vq muß man ja nur einmal berechnen). Wir vereinfachen vorher: $6 \equiv -1 \bmod 7$ und $10 \equiv -1 \bmod 11$. Also:
$z = -21 \cdot (-1) + 22 \cdot (-1) = -1 \equiv 76 \bmod 77$.
Wir hätten das auch noch einfacher haben können: Nach Formel (✻) aus §3.4.1.3 ist
$z \equiv -1 \bmod 7 \wedge z \equiv -1 \bmod 11 \Leftrightarrow z \equiv -1 \bmod 77$.

h) Da alles modulo Primzahlen ist, kann man das Euler-Kriterium anwenden:

$\left(\frac{13}{19}\right) \equiv 13^{\frac{19-1}{2}} \equiv (-6)^9 \bmod 19$; $(-6)^2 \equiv 36 \equiv -2 \Rightarrow (-6)^4 \equiv 4$; $(-6)^8 \equiv 16 \equiv -3$;

$(-6)^9 \equiv -3 \cdot (-6) \equiv 18 \equiv -1$: kein Quadrat.

$\left(\frac{2}{23}\right) \equiv 2^{11} \equiv 2048 \equiv 1 \bmod 23$: Quadrat.

$\left(\frac{-1}{89}\right) \equiv (-1)^{44} \equiv 1 \bmod 89$: Quadrat.

Der einfache Fall $p \equiv 3 \bmod 4$ liegt von den beiden Quadraten nur bei $p = 23$ vor. Also kann man hier selbst mit unseren Algorithmenkenntnissen Wurzelziehen: Eine Wurzel ist

$w = 2^{\frac{23+1}{4}} = 2^6 = 64 \equiv -5 \equiv 18 \bmod 23$. (Probe: $(-5)^2 \equiv 25 \equiv 2$)

(Die andere Wurzel ist also +5.)

i) Es ist $58 \equiv 2 \bmod 7 \wedge 58 \equiv 3 \bmod 11$. Für $p = 7, 11$ gilt $p \equiv 3 \bmod 4$, und es ist

$$2^{\frac{7-1}{2}} \equiv 8 \equiv 1 \bmod 7 \quad \text{und} \quad 3^{\frac{11-1}{2}} \equiv 9 \bullet 9 \bullet 3 \equiv (-2) \bullet (-2) \bullet 3 \equiv 1 \bmod 11,$$

2 und 3 sind also quadratische Reste mod 7 bzw. 11. Also können wir den einfachen Wurzelziehalgorithmus anwenden, um Wurzeln w_7 mod 7 und w_{11} mod 11 zu erhalten:

$$w_7 = 2^{\frac{7+1}{4}} = 2^2 = 4 \bmod 7.$$

$$w_{11} = 3^{\frac{11+1}{4}} = 3^3 = 27 \equiv 5 \bmod 11.$$

Mit dem chinesischen Restsatz und den schon in Teil g) berechneten „Basisvektoren“ erhalten wir: $w = -21 \bullet w_{11} + 22 \bullet w_7 = -21 \bullet 5 + 22 \bullet 4 = (22-21) \bullet 4 - 21 = -17 \equiv 60 \bmod 77$.
(Probe: $(-17)^2 = 289 = 3 \bullet 77 + 58$.)
Setzen wir jeweils die „andere“ Wurzel mod 7 oder mod 11 ein, erhalten wir die anderen 3 Wurzeln:
z.B. $w' = -21 \bullet 5 + 22 \bullet (-4) = -105 - 88 = -193 \equiv 38 \bmod 77$.
(Probe: $38^2 = 1444 = 18 \bullet 77 + 58$.)
Statt so weiterzurechnen, nehmen wir jetzt jeweils das Inverse der beiden schon bekannten Wurzeln:
$w'' = 17$
$w''' = 39$.

j) Man kennt noch eine weitere, wesentlich verschiedene Wurzel von 4, nämlich 2.
Also kann man einen Faktor p als ggT(2035+2, 10379) erhalten. Euklidscher Algorithmus:

$$10379 = 5 \bullet 2037 + 194$$
$$2037 = 10 \bullet 194 + 97$$
$$194 = 2 \bullet 97 + 0$$

Also ist $p = 97$. Die vollständige Faktorisierung erhält man als $q = 10379 \text{ div } 97 = 107$; und p und q sind prim.
(Für Liebhaber: Man beginnt mit den Faktoren, hier p=97 und q=107. Die Wurzeln von 4 mod p und q sind jeweils ±2. Um eine von 2 wesentlich verschiedene Wurzel mod $p \bullet q$ zu erhalten, für die also +2 mod p und –2 mod q oder aber –2 mod p und +2 mod q gilt, muß man also CRA(+2, –2) oder CRA(–2, +2) bilden.)

k) Sei $w_p := x^{\frac{p+1}{4}}$ und $w_q := x^{\frac{q+1}{4}}$.
Nach §3.4.1.8 ist klar, daß x mod n vier Wurzeln w, w', w'', w''' hat, die aus w_p und w_q von x folgendermaßen errechnet werden können:

$w = \text{CRA}(w_p, w_q)$
$w' = \text{CRA}(w_p, -w_q)$
$w'' = \text{CRA}(-w_p, w_q)$
$w''' = \text{CRA}(-w_p, -w_q)$.

Nach §3.4.1.6 ist $w_p \in \text{QR}_p$ und $w_q \in \text{QR}_q$, aber $-w_p \notin \text{QR}_p$ und $-w_q \notin \text{QR}_q$.
Nach §3.4.1.7 gilt

$$x \in \text{QR}_n \Leftrightarrow x \in \text{QR}_p \wedge x \in \text{QR}_q.$$

Also ist $w \in \text{QR}_n$ und $w', w'', w''' \notin \text{QR}_n$.

l) Ist die Faktorisierungsannahme falsch, kann ein Angreifer einen nichtvernachlässigbaren Anteil der Moduli $n = p \bullet q$ faktorisieren. Für solche Moduli kann er dann das Euler-Kriterium anwenden und für x, dessen Quadratheit zu prüfen ist,

$$\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \bmod p \quad \text{und} \quad \left(\frac{x}{q}\right) \equiv x^{\frac{q-1}{2}} \bmod q$$

berechnen. x ist ein Quadrat gdw. $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = +1$. Da das Auswerten der Formeln höchstens kubischen Aufwand (in der Länge von n) verursacht, insbesondere also polynomial in der Länge von n ist, kann der Angreifer somit für einen nichtvernachlässigbaren Anteil aller Moduli die Quadratheit beliebiger x entscheiden. Dies widerspricht der Quadratische-Reste-Annahme.

Um den Umgang mit den formalen Definitionen der Faktorisierungs- und der Quadratische-Reste-Annahme zu üben, wird jetzt der „nichtvernachlässigbare Anteil" formal hingeschrieben: Es gibt einen (probabilistischen) polynomialen Algorithmus F, so daß für ein Polynom Q gilt: Für jedes L existiert ein $l \geq L$, so daß gilt: Wenn p, q als unabhängig zufällige Primzahlen der Länge l gewählt werden und $n := p \bullet q$

$$W(F(n) = (p; q)) > \frac{1}{Q(l)}.$$

Aus diesem Algorithmus F konstruieren wir nun einen ebenfalls (probabilistischen) polynomialen Algorithmus R. R ruft F auf. Faktorisiert F, so berechnet R, wie gerade beschrieben, $\left(\frac{x}{p}\right)$ und $\left(\frac{x}{q}\right)$. Sind beide +1, so gibt R „true" aus, andernfalls „false". R ist genau dann erfolgreich, wenn es F ist.

Also gilt: Es gibt einen (probabilistischen) polynomialen Algorithmus R, so daß für das obige Polynom Q gilt: Für jedes L existiert ein $l \geq L$, so daß gilt: Wenn p, q als zufällige Primzahlen der Länge l gewählt werden, $n := p \bullet q$, und x zufällig unter den Restklassen mit Jacobi-Symbol +1, so gilt:

$$W(R(n, x) = qr(n, x)) > \frac{1}{2} + \frac{1}{Q(l)}.$$

Dies widerspricht der Quadratische-Reste-Annahme.

3-10 Welche Angriffe sind auf welche kryptographischen Systeme zu erwarten?

a) Der Notar muß ein *digitales Signatursystem* verwenden. Da er bis auf den Zeitstempel (z.B. Anhängen von Datum und Uhrzeit an die Nachricht) beliebige ihm vorgelegte Nachrichten damit signiert, muß das digitale Signatursystem *adaptiven aktiven Angriffen* widerstehen. Das Nachrichtenformat könnte folgendermaßen sein:
<vorgelegte Nachricht>, Datum, Uhrzeit, s_{Notar}(<vorgelegte Nachricht>, Datum, Uhrzeit).

b) Es können symmetrische oder asymmetrische *Konzelationssysteme* verwendet werden.

Erstere müssen mindestens einem Klartext-Schlüsseltext-Angriff (known-plaintext attack) standhalten, da der Angreifer den Klartext zum beobachteten Artikel demnächst in der Zeitung nachlesen kann. Wenn der Angreifer sensationelle Dinge tut, kann er zumindest teilweise einen gewählten Klartext-Schlüsseltext-Angriff erreichen, z.B. kann er den Text, den er gerne verschlüsselt sähe, ans Rathaus sprühen. (Diese beiden Angriffe sind beim asymmetrischen System irrelevant, weil der Angreifer da ja sowieso beliebige Texte selbst verschlüsseln kann.)

Würde die Zeitung auch beliebig unsinnige Artikel drucken (vielleicht weil der Korrespondent auch Chiffreanzeigen entgegennimmt), muß das Konzelationssystem (symmetrisch oder asymmetrisch) sogar einem gewählten Schlüsseltext-Klartext-Angriff (chosen-ciphertext attack) standhalten, da der aktive Angreifer den Klartext zum von ihm geschickten Schlüsseltext demnächst in der Zeitung nachlesen kann.

3-11 Wie lange sind Daten zu schützen? Dimensionierung kryptographischer Systeme; Reaktion auf falsche Dimensionierung; Öffentlichen Schlüssel eines Signatursystems oder asymmetrischen Konzelationssystems veröffentlichen?

a) Die Wahrscheinlichkeit, daß das kryptographische System innerhalb dieser Zeit gebrochen wird, muß hinreichend klein sein. Wenn man ein informationstheoretisch sicheres System verwendet (also beim One-time-pad oder geeigneten Authentikationscodes), sind die folgenden 3 Punkte irrelevant. Andernfalls ist zu beachten:

- Trivialerweise wächst mit der Länge der Zeit, die der Schutz wirksam sein soll, auch die Zeit, die dem Angreifer für eine Kryptoanalyse zur Verfügung steht.
- Vermutlich steigt die Rechenleistung/DM weiterhin exponentiell mit einer Verdoppelungszeit etwa jedes Jahr, bis irgendwann technologische Grenzen erreicht werden. Diese steigende Rechenleistung wird dem Angreifer zur Verfügung stehen.
- Außerdem können in dieser Zeit effizientere Algorithmen entdeckt werden. (Denn bei all den nicht informationstheoretisch sicheren Systemen gibt es ja noch keine brauchbaren unteren Schranken für die Komplexität des Brechens.)

In den beiden Beispielen muß man die Systeme so überdimensionieren, daß sie hoffentlich mindestens 80 Jahre sicher sind.

Um ein Gefühl zu bekommen, was das bedeutet, ein Beispiel: Nehmen wir an, das Wachsen der Rechenleistung hielte solange an. (Das dürfte allerdings mit physikalischen Grenzen schon knapp werden, vgl. [Paul_78 Seite 236f., DaPr_89 Seite 41].) Dann ist es so, als ob der Angreifer bei heutiger Rechenleistung etwa 2^{81} Jahre zur Verfügung hätte. Könnte man das System nicht schneller als durch vollständige Suche im Schlüsselraum brechen, so müßte dieser um den Faktor 2^{81} zu groß gewählt werden, also die Schlüssel um 81 bit zu lang, falls sie optimal codiert sind. Das wäre nicht einmal sehr viel.

Anders ist es schon bei auf Faktorisierung beruhenden Systemen. Um einerseits die Fortschritte bei den Faktorisierungsalgorithmen und andererseits die bei der Rechenleistung zu verdeutlichen, werden zwei Beispielrechnungen durchgeführt:

Nehmen wir an, die Komplexität sei und bliebe genau

$$L_{1990}(n) = \exp(\sqrt{\ln(n) \ln\ln(n)}),$$

$$L_{1995}(n) = \exp(1{,}923 \cdot \sqrt[3]{\ln(n) (\ln\ln(n))^2}),$$

dies ist die Komplexität des besten 1990 bzw. 1995 bekannten Faktorisierungsalgorithmus, vgl. [Schn_96 Seite 256].

Im Jahre 1995 verwendeten wir n der Länge 512 bit und das wurde etwa damit begründet, daß diese Zahlen in einem Jahr nicht zu faktorisieren waren (was heutzutage anders ist, vgl. [Riel_99]). Also sollten wir aus der Perspektive von 1995 für die Zukunft n^* vorsehen mit

$$L(n^*) = 2^{81} \cdot L(n)$$

Aus der algorithmischen Sicht von 1990 bedeutet dies:

$$L_{1990}(n^*) = 2^{81} \cdot L_{1990}(n)$$

$$\Leftrightarrow \exp(\sqrt{\ln(n^*) \ln\ln(n^*)}) = 2^{81} \cdot \exp(\sqrt{\ln(n) \ln\ln(n)})$$

$$\Leftrightarrow \sqrt{\ln(n^*) \ln\ln(n^*)} = 81 \ln(2) + \sqrt{\ln(n) \ln\ln(n)}.$$

Außerdem wissen wir $\ln(n) = 512 \cdot \ln(2) \approx 360$ und $\ln\ln(n) \approx 6$, sowie $\sqrt{360 \cdot 6} \approx 44$, also etwa

$$\Leftrightarrow \sqrt{\ln(n^*) \ln\ln(n^*)} = 56 + 44$$

$$\Leftrightarrow \ln(n^*) \ln\ln(n^*) = 100^2$$

Ein bißchen Taschenrechnerakrobatik ergibt, daß diese Ungleichung für $\ln(n^*) \approx 1400$ erfüllt ist. Also $\log_2(n^*) \approx 1400 / \ln(2) \approx 2020$.

Man müßte also die Zahlen etwa 3,95 mal so lang wählen wie bisher. Der Berechnungsaufwand erhöht sich dadurch um den Faktor $3{,}95^3 \approx 61$.

Aus der algorithmischen Sicht von 1995 bedeutet dies:

$$L_{1995}(n^*) = 2^{81} \cdot L_{1995}(n)$$

$$\Leftrightarrow \exp(1{,}923 \cdot \sqrt[3]{\ln(n^*)\,(\ln\ln(n^*))^2}) = 2^{81} \cdot \exp(1{,}923 \cdot \sqrt[3]{\ln(n)\,(\ln\ln(n))^2})$$

$$\Leftrightarrow 1{,}923 \cdot \sqrt[3]{\ln(n^*)\,(\ln\ln(n^*))^2} = 81\ln(2) + 1{,}923 \cdot \sqrt[3]{\ln(n)\,(\ln\ln(n))^2}$$

$$\Leftrightarrow \sqrt[3]{\ln(n^*)\,(\ln\ln(n^*))^2} = 42{,}12\ln(2) + \sqrt[3]{\ln(n)\,(\ln\ln(n))^2}$$

$$\Leftrightarrow \sqrt[3]{\ln(n^*)\,(\ln\ln(n^*))^2} = 29 + \sqrt[3]{\ln(n)\,(\ln\ln(n))^2}.$$

Außerdem wissen wir $\ln(n) = 512 \cdot \ln(2) \approx 360$ und $(\ln\ln(n))^2 \approx 36$, sowie $\sqrt[3]{360 \cdot 36} \approx 23$, also etwa

$$\Leftrightarrow \sqrt[3]{\ln(n^*)\,(\ln\ln(n^*))^2} = 52$$

$$\Leftrightarrow \ln(n^*)\,(\ln\ln(n^*))^2 = 52^3$$

Ein bißchen Taschenrechnerakrobatik ergibt, daß diese Ungleichung für $\ln(n^*) \approx 2400$ erfüllt ist. Also $\log_2(n^*) \approx 2400 / \ln(2) \approx 3462$.

Man müßte also die Zahlen etwa 6,76 mal so lang wählen wie bisher. Der Berechnungsaufwand erhöht sich dadurch um den Faktor $6{,}76^3 \approx 309$.

Wir sehen: Der algorithmische Fortschritt innerhalb von 5 Jahren hat „genausoviel“ Einfluß auf die notwendige Länge der Zahlen wie die innerhalb von 80 Jahren erwartete Erhöhung der Rechenleistung. Beides erzwingt eine Verlängerung um je etwa 1500 Bit für den betrachteten Zeitraum.

(Zum Glück gibt es aber viele Anwendungen, wo Signaturen nur kurz gelten müssen, z.B. in einem digitalen Zahlungssystem, wenn man jedes Vierteljahr einen Kontoabschluß erhält und gegen diesen noch wenige Monate Widerspruchsrecht hat.)

b) Klar ist, daß man bei beiden Systemtypen hinfort für alle neu zu verschlüsselnden bzw. zu signierenden Daten nur noch stärker dimensionierte Systeme verwendet, dazu neue Schlüssel austauschen muß, etc. Vertraut man darauf, daß das demnächst brechbare digitale Signatursystem noch nicht gebrochen ist, kann man es zum Austausch von öffentlichen Schlüsseln der stärkeren asymmetrischen kryptographischen Systeme benutzen. Andernfalls ist – wie auch bei Verwendung symmetrischer kryptographischer Systeme für den Schlüsselaustausch – ein neuer Ur-Schlüsselaustausch außerhalb des zu schützenden Rechnernetzes notwendig. Möchte man mit dem digitalen Signatursystem rechtlich Verbindliches regeln, sollte der öffentliche Testschlüssel neu registriert werden. In jedem Fall ist es günstig, sich zumindest den Wechsel *öffentlicher* Schlüssel von den Schlüsselregistern bestätigen zu lassen. Damit Schlüsselregister nicht willkürlich von sich aus Schlüssel für ungültig erklären können, sollten sie hierzu eine digitale Signatur (ggf. im alten kryptographischen System) des Inhabers unter eine diesbezügliche Nachricht (Schlüssel ungültig ab: Datum, Uhrzeit) vorweisen.

Unterschiedlich ist das Vorgehen bzgl. der Daten, die mit dem alten, demnächst zu schwachen System konzeliert/signiert wurden:

Konzelierte Daten müssen mit dem stärkeren Konzelationssystem verschlüsselt werden. Dies kann je nach Anwendung, d.h. gibt es für diese neue Verschlüsselung eine sichere Umgebung mit Kenntnis der zur Entschlüsselung der vorliegenden Schlüsseltexte nötigen Schlüssel oder nicht, durch Entschlüsseln der vorliegenden Schlüsseltexte mit dem demnächst zu schwachen Konzelationssystem und dem anschließenden Verschlüsseln des Klartextes mit dem stärker dimensionierten Konzelationssystem geschehen oder durch eine zusätzliche Verschlüsselung der vorliegenden Schlüsseltexte. Bei asymmetrischen Konzelationssystemen dürfte der letztere Fall der häufigere sein. Unabhängig von der Verwendung symmetrischer oder asymmetrischer Konzelationssysteme bedeutet das Vorgehen nach dem letzteren Fall, daß die Implementierung des demnächst zu schwachen Konzelationssystems und die verwendeten Schlüssel weiterhin verfügbar bleiben müssen. Das Vermeiden des Risikos des Umschlüsselns (es liegt für kurze Zeit der Klartext vor) hat seinen Preis. Leider ist man nie sicher, ob alle Kopien des Schlüsseltextes, wie beschrieben, stärker verschlüsselt werden. Das skizzierte Vorgehen hilft also nur gegen Angreifer, die nach der stärkeren Verschlüsselung auf den Plan treten und ggf. auch Schwierigkeiten haben, eine weitere Kopie des nur schwach verschlüsselten Schlüsseltextes zu finden. Das skizzierte Vorgehen ist also kein Ersatz für richtige Dimensionierung von Anfang an, aber natürlich besser, als einfach nur abzuwarten und die entstehende Unsicherheit des zu schwach dimensionierten Konzelationssystems geheim halten zu wollen.

Signierte Daten müssen mit einer zusätzlichen Signatur im stärker dimensionierten Signatursystem versehen werden, am besten vom ursprünglichen Signierer. Hierzu könnte man in einer Übergangszeit jeden verpflichten: Jeder publiziert stärker dimensionierte Testschlüssel (zur zumindest zeitweisen Authentikation jeweils mit dem noch nicht gebrochenen Signatursystem signiert) und ist bei Vorlage einer von ihm im schwächeren alten System signierten Nachricht verpflichtet, diese Nachricht im stärkeren neuen System zu signieren. Diese Übergangszeit muß natürlich enden, bevor das Brechen des alten Signatursystems preiswert genug wird. Ist der ursprüngliche Signierer zum zusätzlichen Signieren nicht in der Lage oder nicht willens, so kann ein *Zeitstempeldienst* in Anspruch genommen werden: Jeder, der eine von einem anderen signierte Nachricht hat und den Verfall dieses seines Beweismittels aufhalten will, legt die Nachricht samt Signatur (im alten Signatursystem) einem Zeitstempeldienst vor. Dieser signiert im neuen, stärker dimensionierten Signatursystem "Nachricht, Signatur im alten Signatursystem, Datum, Zeit". Wird dies später vorgelegt, so ist diese zeitgestempelte Signatur so authentisch, wie sie zu "Datum, Zeit" im alten Signatursystem war, wie der Zeitstempeldienst gegen Korrumpierung geschützt ist und wie kryptographisch sicher die neue Signatur zum Vorlagezeitpunkt ist. Der Schutz des Zeitstempeldienstes gegen Korrumpierung kann durch die übliche Technik erhöht werden: Statt einer Signatur im neuen Signatursystem wird von unabhängig implementierten und betriebenen Rechnern jeweils eine (von allen anderen unabhängige) Signatur im neuen Signatursystem angebracht. Eine Nachricht gilt nur dann als korrekt zeitgestempelt, wenn genügend viele (unabhängige) Signaturen im neuen Signatursystem vorgelegt werden. Da beim Nachsignieren jeder autonom seine eigenen Interessen wahren kann, ist das hierfür skizzierte Vorgehen weitaus zufriedenstellender, als das bzgl. Konzelation. Dies gilt insbesondere, wenn durch die Verwendung von Fail-Stop-Signaturen das Ende der kryptographischen Sicherheit von digitalen Signaturen (und damit auch das Ende jeder Übergangszeit) sicher ermittelt werden kann.

c) Ich gebe den Testschlüssel eines asymmetrischen Signatursystems weiter. Denn dann kann mein Partner später mit seiner Hilfe prüfen, ob mein ihm dann übermittelter öffentlicher Schlüssel eines asymmetrischen Konzelationssystems authentisch ist, indem ich diesen hierfür signiere.

d) Ja, denn ein „Upgrade“ von Signatursystemen ist leichter sicher durchführbar als eins von Konzelationssystemen. Außerdem muß für rechtliche Verbindlichkeit der Testschlüssel sowieso registriert sein.

3-12 Kryptopolitik durch Wahl einer geeigneten Schlüssellänge?

Dieses Vorgehen ist prinzipiell aussichtslos:

Selbst wenn Staaten deutlich mehr Geld als große Konzerne haben (was alles andere als klar ist), so können Konzerne (wie auch neugierige NachbarInnen) ihre kryptanalytischen Anstrengungen sehr stark auf einzelne Sender und Empfänger fokussieren, während Verbrechensbekämpfung (und auch Spionage) Kryptoanalyse in einer gewissen Breite, d.h. von vielen Sendern und Empfängern, wünscht. Pro zu brechendem Schlüssel dürften Staaten eher weniger als mehr Rechenleistung als große Konzerne haben.

Neben diesem aus meiner Sicht grundlegenden Argument gibt es natürlich eine Reihe weiterer Argumente, warum ein solcher „Schlüssellängenkompromiß“ unsinnig ist:

Mit dem Zuwachs an allgemein verfügbarer Rechenleistung müßte die Schlüssellänge fortwährend angepaßt werden (vgl. Aufgabe 3-11 a)), was aber bzgl. Konzelation für zurückliegende Nachrichten nicht sinnvoll zu bewerkstelligen ist (vgl. Aufgabe 3-11 b)).

Verbrecher werden nicht allein auf solch schwache kryptographische Konzelationssysteme setzen, sondern ggf. zusätzlich noch mit stärkeren verschlüsseln und ggf. noch zusätzlich steganographische Konzelationssysteme einsetzen, vgl. §4 und §9.

3-13 Authentikation und Konzelation in welcher Reihenfolge?

Sei x die Nachricht.

Wenn *asymmetrische Authentikation* gewünscht ist, also eine digitale Signatur, sollte man zuerst signieren und dann verschlüsseln, so daß der Empfänger nach dem Entschlüsseln eine signierte Klartextnachricht zum Aufbewahren hat. Die Signatur muß mitverschlüsselt werden, weil sie i.allg. Information über die Nachricht enthält. Also: Sender bildet $k(x, s(x))$ bzw. $c(x, s(x))$, je nachdem ob das Konzelationssystem symmetrisch oder asymmetrisch ist.

Wenn *symmetrische Authentikation* genügt, ist die Reihenfolge bzgl. Nützlichkeit egal: Man kann es machen wie oben, also $k(x, MAC)$ bilden. (In diesem Fall wird man wohl kein asymmetrisches Konzelationssystem haben.) Oder man kann die verschlüsselte Nachricht authentisieren, also einen MAC zu $k(x)$ bilden. Letzteres hat bzgl. Aufwand und Leistung 3 Vorteile:

- Es „verbraucht“ bei Verwendung eines One-time-pads zur Konzelation weniger Schlüssel sowie immer weniger Rechenaufwand, da Konzelation die Länge der Nachricht nicht erhöhen muß, während Authentikation dies immer tut – und der Aufwand der zweiten Operation steigt mit der Länge ihrer Eingabe, d.h. der Ausgabe der ersten Operation.
- Prüft man erst die Authentikation und schlägt dieser Test fehl, kann mensch sich das Entschlüsseln (und somit Aufwand) sparen.
- Authentikationsprüfung und Entschlüsselung können parallel erfolgen, so daß die Antwortzeit verkürzt werden kann.

3-14 s^2-mod-n-Generator

a) Für $n = 7 \cdot 11 = 77$, $s = 2$ ergibt sich als

Folge s_i:	4	16	25	9	4	…
Folge b_i:	0	0	1	1	0	…

und damit für den Klartext 0 1 0 1 bei Addition modulo 2 der Schlüsseltext 0 1 1 0.

Der Entschlüsseler bildet auf genau die gleiche Art die Folge b_i und errechnet durch Addition modulo 2 aus dem Schlüsseltext 0 1 1 0 den Klartext 0 1 0 1.

Und weil es so schön war, ein zweites Beispiel:
Für $n = 7 \cdot 11 = 77$, $s = 13$ ergibt sich als

Folge s_i:	15	71	36	64	15	…
Folge b_i:	1	1	0	0	1	…

und damit für den Klartext 0 1 0 1 der Schlüsseltext 1 0 0 1.

b) Zunächst müssen wir aus s_4=25 die Folge der anderen s_i zurückberechnen. Wir brauchen den CRA, also berechnen wir zuerst die „Basisvektoren“ dafür:
Euklidscher Algorithmus:

$$11 = 3 \cdot 3 + 2; \quad 3 = 1 \cdot 2 + 1;$$

Rückwärts:

$$1 = 3 - 1 \cdot 2 = 3 - 1 \cdot (11 - 3 \cdot 3) = -11 + 4 \cdot 3$$

Damit sind $up = 12$, $vq = -11$, und wir haben die Formel $s = 12s_q - 11s_p = 11(s_q - s_p) + s_q$.
Außerdem können wir $(p+1)/4 = 1$ und $(q+1)/4 = 3$ vorwegberechnen.

s_3: (Merke: Hier können wir nicht einfach 5 (die vom Rechnen mit den ganzen Zahlen bekannte Wurzel) nehmen, weil wir von den 4 Wurzeln aus 25 mod 33 ja diejenige brauchen, die selbst quadratischer Rest ist.)
Modulo 3: $s_4 \equiv 1 \Rightarrow s_3 \equiv 1^1 \equiv 1$.
Modulo 11: $s_4 \equiv 3 \Rightarrow s_3 \equiv 3^3 \equiv 5$.
$\Rightarrow s_3 = 11(5-1) + 5 = 49 \equiv 16 \bmod 33$.

s_2: Modulo 3: $s_3 \equiv 1 \Rightarrow s_2 \equiv 1^1 \equiv 1$. (Man sieht, daß das immer so bleibt.)
Modulo 11: $s_3 \equiv 5 \Rightarrow s_2 \equiv 5^3 \equiv 25 \cdot 5 \equiv 4$.
$\Rightarrow s_2 = 11(4-1) + 4 \equiv 4 \bmod 33$.

s_1: Modulo 11: $s_2 \equiv 4 \Rightarrow s_1 \equiv 4^3 \equiv 16 \cdot 4 \equiv 9$.
$\Rightarrow s_1 = 11(9-1) + 9 \equiv 31 \bmod 33$.

s_0: Modulo 11: $s_1 \equiv 9 \equiv -2 \Rightarrow s_0 \equiv (-2)^3 \equiv -8 \equiv 3$.
$\Rightarrow s_0 = 11(3-1) + 3 = 25$.

Die letzten Bits davon sind $b_0 = 1$, $b_1 = 1$, $b_2 = 0$, $b_3 = 0$.
Der Klartext ist also $1001 \oplus 1100 = 0101$.

c) Was hier passiert, hängt gar nicht speziell vom s^2-mod-n-Generator ab, sondern passiert immer, wenn man ein One-time-pad (vgl. Aufgabe 3-7 d)) oder ein mehr oder weniger sicheres Pseudo-One-time-pad bitweise zur Nachricht addiert (Stromchiffre im engeren Sinn, vgl. §3.8.1):

Wenn ein Bit des Schlüsseltextes kippt, ist nach dem Entschlüsseln mittels Addition modulo 2 auch genau dieses eine Bit des Klartextes falsch. Wenn hingegen ein Bit verlorengeht, ist ab dieser Stelle der gesamte Klartext unleserlich:

Sei das i-te Bit des Klartexts x_i. Das i-te Schlüsseltextbit ist dann $S_i := x_i \oplus b_i$. Kommt statt S_i beim Empfänger $S_i^* = S_i \oplus 1$ an, so entschlüsselt er es als $S_i^* \oplus b_i = x_i \oplus 1$.

Fällt dagegen S_i aus, so entschlüsselt er ab da jedes Bit mit dem falschen b_i: Er hält S_{i+1} für S_i und entschlüsselt es mit b_i usw.

d) Der s^2-mod-n-Generator garantiert in seiner normalen Verwendung als symmetrisches (und erst recht als asymmetrisches) Konzelationssystem natürlich genausowenig Authentizität wie die Vernam Chiffre (vgl. Teil c)).

Trotzdem lautet die Antwort auf die Frage: Ja. Man nehme einen informationstheoretisch sicheren Authentikationscode und verwende die durch den s^2-mod-n-Generator generierte Folge als Schlüssel des Authentikationscodes. (Der wirklich auszutauschende Schlüssel ist also nur noch der Startwert für den s^2-mod-n-Generator.)

Das resultierende Authentikationssystem könnte gebrochen werden, wenn der Angreifer entweder den Authentikationscode oder den s^2-mod-n-Generator brechen kann. Der Authentikationscode ist aber informationstheoretisch sicher, also bleibt der s^2-mod-n-Generator.

Für Spezialisten: Man müßte beweisen können, daß der Angreifer, um das System zu brechen, wirklich die QRA (Quadratische-Reste-Annahme) brechen muß, d.h. daß das resultierende System kryptographisch sicher ist. Skizze: Wenn man den Authentikationscode brechen kann, wenn die Schlüssel vom s^2-mod-n-Generator gewählt werden, während er mit zufälligen Schlüsseln unbrechbar ist, ist das ein meßbarer Unterschied zwischen den Pseudozufallsfolgen und den echten Zufallsfolgen. Es ist aber bewiesen, daß unter der QRA beim s^2-mod-n-Generator die Folgen durch keinen statistischen Test von echten Zufallsfolgen zu unterscheiden sind.

e) Der Schlüsselaustausch bleibt gleich. Allerdings muß sich jeder der beiden nach der ersten Nachricht nicht mehr den Startwert s, sondern jeweils das aktuelle s_i merken. {Dies setzt voraus, daß der Empfänger die Schlüsseltexte jeweils genau einmal und in der gleichen Reihenfolge entschlüsselt, wie sie der Verschlüsseler verschlüsselt hat. Andernfalls muß sich der Entschlüsseler mehr merken. Vergleiche auch §3.8.2.)

f) Nach §3.4.1.7 und §3.4.1.8 sowie Aufgabe 3-9 k) hat mod n (mit $n=p\cdot q$, p, $q \equiv 3 \bmod 4$, $p \neq q$) jedes Quadrat genau vier Wurzeln, von denen genau eine selbst ein Quadrat ist. Die Wurzel aus 1, die selbst wieder ein Quadrat ist, ist 1. Also gilt: Ist $s_0 \neq 1$, so sind auch alle folgenden inneren Zustände $\neq 1$, denn 1 entsteht eben nur aus dem Quadrat 1. Also ist die Wahrscheinlichkeit, nach i Schritten den inneren Zustand 1 erreicht zu haben, gleich der Wahrscheinlichkeit, als Startwert s zufällig eine der vier Wurzeln der 1 zu wählen, also $4 / |\mathbb{Z}_n^*|$.

3-15 GMR

a) $R = 28$ liegt nicht im Definitionsbereich des Permutationenpaares, denn $ggT(R, n) = ggT(28, 77) = 7$. Also kann man daran auch nichts signieren. Deswegen folgt jetzt das Beispiel mit $R = 17$. (Dabei ist $17 \equiv 28 \bmod 11$, also kann jemand, der mit 28 gerechnet hat, seine Rechnung mod 11 hier wiederfinden.)

Da $ggT(17, 77) = 1$ und $\left(\frac{17}{77}\right) = \left(\frac{17}{11}\right) \cdot \left(\frac{17}{7}\right) = \left(\frac{6}{11}\right) \cdot \left(\frac{3}{7}\right) = (6^{(11-1)/2} \bmod 11) \cdot (3^{(7-1)/2} \bmod 7) = (-1) \cdot (-1) = 1$, liegt $R = 17$ im Definitionsbereich des Permutationenpaares.

Da die Nachrichten immer 2 bit lang sind, braucht man keine präfixfreie Abbildung. Es ist also direkt $S := f_m^{-1}(R)$ zu bilden. Dies ist (vgl. §3.5.3)

$$S = f_{01}^{-1}(17) = f_1^{-1}(f_0^{-1}(17)).$$

Schritt 1: Zunächst wird $x := f_0^{-1}(17)$ benötigt. Nach §3.5.2 ist zunächst zu testen, ob 17 oder –17 das Quadrat ist. (Vgl. §3.4.1.7)

Modulo $p = 11$: $17 \equiv 6$; Euler-Kriterium: $6^{(11-1)/2} = 6^5$;

$6^2 \equiv 3$; $6^4 \equiv 3^2 \equiv 9$; $6^5 \equiv 9 \cdot 6 \equiv -1 \Rightarrow 17 \notin QR_p \Rightarrow 17 \notin QR_n$.

Also ist die Wurzel w aus $-17 \equiv 60$ zu ziehen.

Modulo $p = 11$: $60 \equiv 5$; $5^{(11+1)/4} = 5^3 \equiv 4$;

Modulo $q = 7$: $60 \equiv 4$; $4^{(7+1)/4} = 4^2 \equiv 2$;

(Man kann das folgende abkürzen, aber der Vollständigkeit halber sei es zunächst vorgeführt: Mit dem QRA und den „Basisvektoren“ aus Aufg. 3-9 g) ergibt sich $w = -21 \cdot 4 + 22 \cdot 2 = -40 \equiv 37$. Dies w ist diejenige der 4 Wurzeln, die selbst Quadrat ist. x soll die mit Jacobi-Symbol +1 sein (also $\pm w$), die $< 77/2$ ist. Das trifft auf w zu, also $x = 37$.)

Schritt 2: Jetzt wird $S = f_1^{-1}(x) = f_1^{-1}(37)$ berechnet, also $S \in D_n$ mit $(2S)^2 \equiv \pm 37$.

(Als erstes müßten wir wieder testen, ob x oder $-x$ das Quadrat ist. Das wissen wir aber noch aus dem ersten Schritt: Wir hatten ja sowieso erst die Wurzel w berechnet, die das Quadrat war. Man kann also gleich mit w weitermachen. Außerdem müssen wir jetzt zum Wurzel-

ziehen aus w wieder mod p und q einzeln rechnen. Wir hätten also auch w nicht mit dem QRA zusammensetzen müssen, d.h. die ganzen eingeklammerten Teile auslassen können.)

Also: Wurzel w^* aus dem Ergebnis von Schritt 1:

Modulo 11: $4^{(11+1)/4} = 4^3 \equiv 9$;

Modulo 7: $2^{(7+1)/4} = 2^2 \equiv 4$;

Als nächstes muß durch 2 dividiert werden. Auch das geht mod p, q einzeln: (Erweiterter Euklidscher Algorithmus zur Bestimmung des multiplikativen Inversen von 2 und anschließende Multiplikation hiermit, oder Halbieren in $\mathbb{Z}$ von w^* oder w^*+p bzw. w^*+q.)

Modulo 11: $w^*/2 \equiv 10$;

Modulo 7: $w^*/2 \equiv 2$;

Die gesuchte Signatur S ist also eine der 4 Zahlen CRA(±10, ±2). Vor dem QRA müssen wir wieder für das Jacobi-Symbol +1 sorgen. Statt des Euler-Kriteriums können wir verwenden, daß (§3.5.1)

$$\left(\frac{2}{p}\right) = -1 \text{ und } \left(\frac{2}{q}\right) \equiv +1.$$

Danach ist, da die Jacobi-Symbole von w^* mod p und q jeweils 1 waren,

$$\left(\frac{w^*/2}{p}\right) = -1 \text{ und } \left(\frac{w^*/2}{q}\right) \equiv +1.$$

Also bilden wir CRA(–10, 2). (Man könnte auch CRA(10, –2) nehmen.)

CRA(–10, 2) = CRA(1, 2) = –21•1 + 22•2 = 23.

Wieder haben wir zufällig bereits diejenige der Zahlen ±S erwischt, die < 77/2 ist ⇒ S = 23.

Anmerkungen:

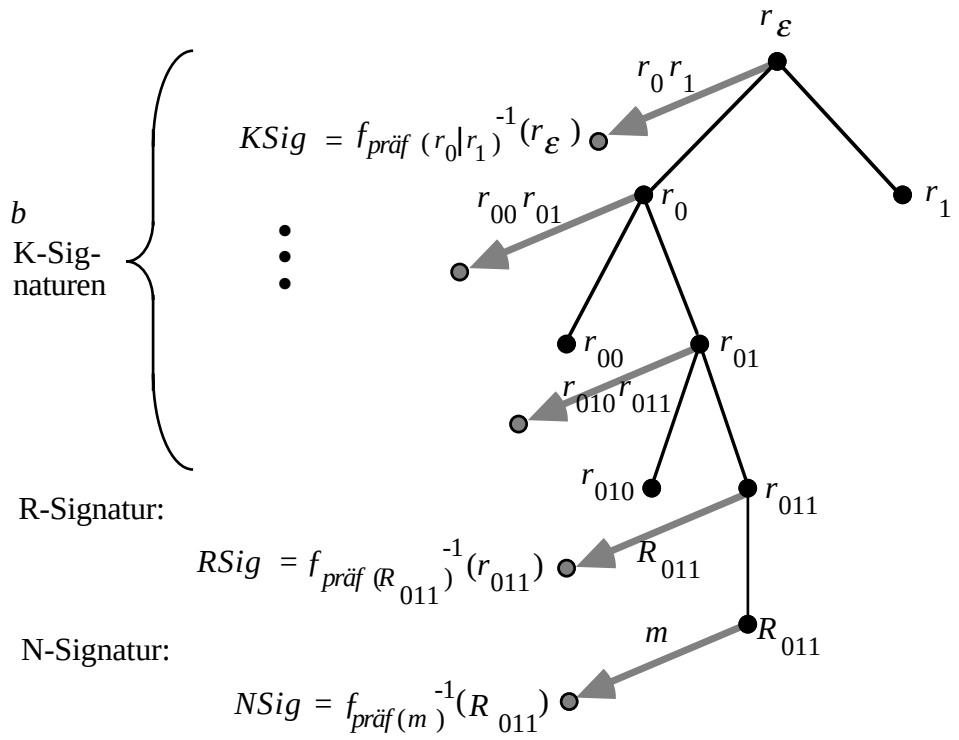
1. Es ist egal, ob erst durch 2 dividiert und dann der CRA angewandt wird oder umgekehrt.
2. Wenn Sie als Signatur 12 erhalten, dann haben Sie vergessen, die Jacobi-Symbole zu beachten. 12 hat das Jacobi-Symbol

$$\left(\frac{12}{77}\right) = \left(\frac{12}{11}\right) \cdot \left(\frac{12}{7}\right) = 12^5 \bmod 11 \cdot 12^3 \bmod 7 = 1^5 \bmod 11 \cdot (-2)^3 \bmod 7 = -1.$$

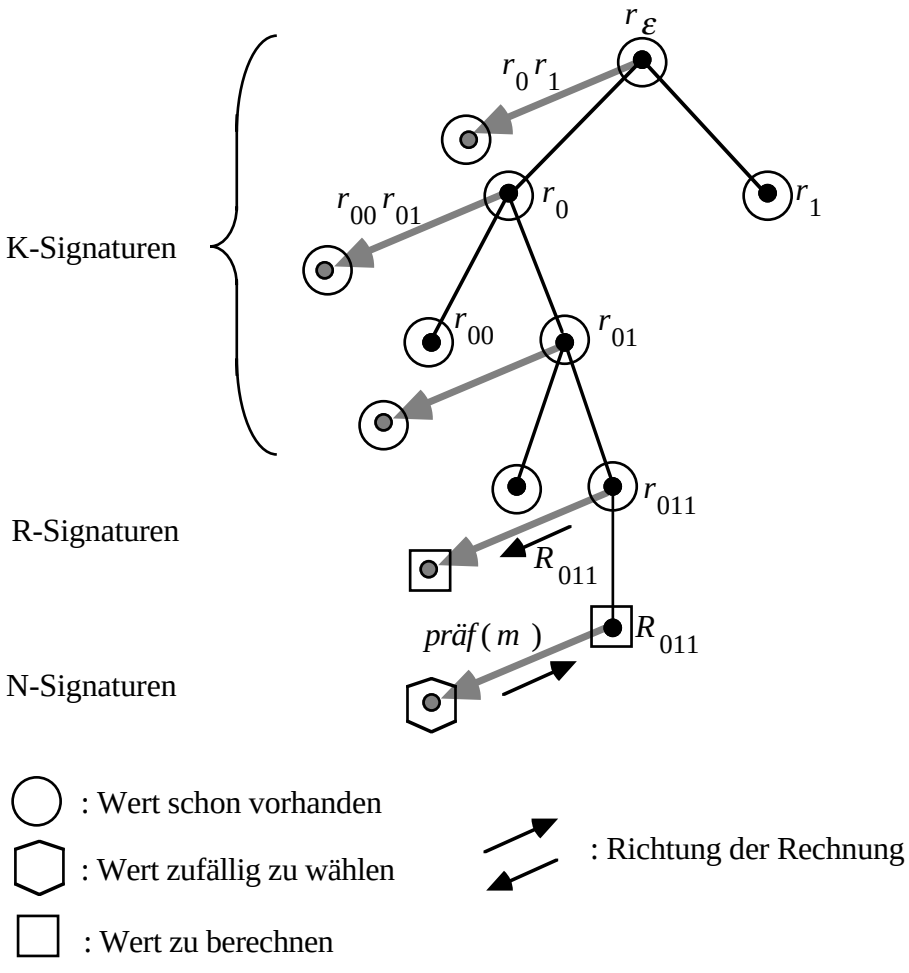
   12 liegt also nicht im Definitionsbereich der Permutationen. Das Vertrackte ist, daß der Test der Signatur scheinbar trotzdem klappt: $f_0(f_1(12)) = 17$, denn $f_1(12) = 4 \cdot 12^2 \bmod 77 = 4 \cdot 144 \bmod 77 = 4 \cdot -10 \bmod 77 = 37$. $f_0(37) = -(37^2) \bmod 77 = 17$. Also muß die Prüfung, ob die Signatur im Definitionsbereich der Permutationen liegt, Teil des Tests der Signatur sein.

b) Da 8 Nachrichten signiert werden sollen, hat der Referenzenbaum die Tiefe b = 3, wie in den Bildern in §3.5.4.

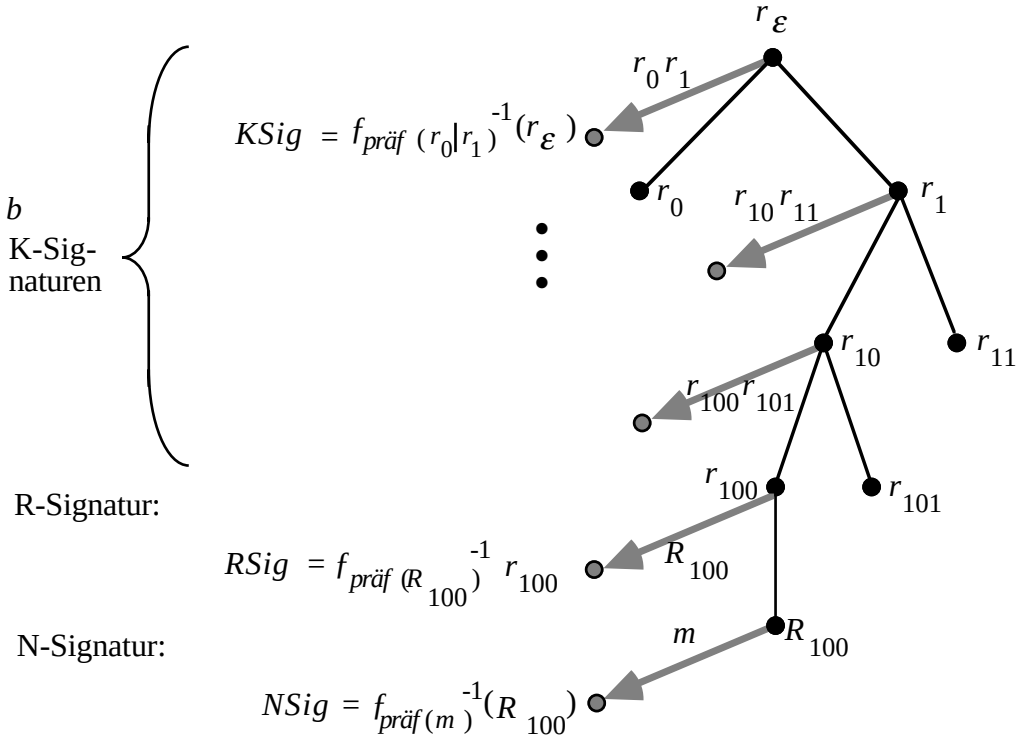
Analog zu Bild 3-26 besteht die 4. Signatur, d.h. die an R_{011} hängende, aus folgenden Teilen: („Teile“ sind dabei genau die Punkte: Die schwarzen Referenzen und die grauen Signaturen.)



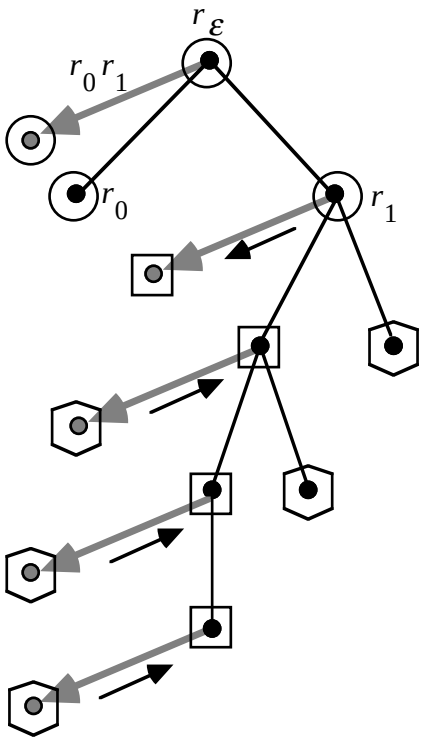
Durch Vergleich mit Bild 3-26 sieht man, was neu zu wählen bzw. zu berechnen ist: Die einzige neue Referenz ist R_{011}. Neue Signaturen sind die von R_{011} bzgl. r_{011} und (natürlich) die der Nachricht m bzgl. R_{011}. Betrachtet man auch noch wie in Bild 3-27, in welcher Reihenfolge man am besten rechnet, so ergibt sich:



Analog hat die fünfte Signatur folgende Teile:



Durch Vergleich mit der obigen vierten Signatur sieht man, daß nur die obersten drei Referenzen (r_ε, r_0, r_1) schon da sind sowie die oberste K-Signatur. (Dies ist gerade der Wechsel von der linken in die rechte Baumhälfte, also nach der ersten Signatur die zweitaufwendigste Signatur überhaupt.) Für die Rechenreihenfolge ergibt sich:



Verschicken kann man natürlich jeweils alles Gezeichnete, was aber nur für paralleles Testen nötig ist. Wird das Testen sequentiell durchgeführt, so braucht man die inneren Knoten der Signaturen nicht zu verschicken: Der Empfänger der Signatur muß alle grauen Pfeile entgegen

ihrer Richtung nachrechnen und nur vergleichen, ob sich die ihm bekannte Wurzel des Baumes r_ε ergibt.

c) Zu zeigen ist: Wenn es einen Algorithmus gäbe, der effizient Kollisionen von Funktionen $f_{präf(m)}$ findet, gäbe es auch einen, der Kollisionen für Paare (f_0, f_1) findet.

Dazu wird gezeigt, wie man aus jeder Kollision von $f_{präf(m)}$ direkt eine von (f_0, f_1) berechnet. Sei die gegebene Kollision $m; m^*, S, S^*$ mit $m \neq m^*$ und $f_{präf(m)}(S) = f_{präf(m^*)}(S^*)$. Setze $v := präf(m)$ und $w := präf(m^*)$.

Nennen wir die Bits von $v = v_0v_1...v_k$ bzw. $w = w_0w_1...w_{k'}$. Seien v_i und w_i die ersten Bits von links, in denen sich v und w unterscheiden. Da v und w präfixfreie Codierungen unterschiedlicher Nachrichten m, m^* sind, gilt $i \leq \min\{k,k'\}$, und natürlich gilt $0 \leq i$. Dann ist

$$f_v(S) = \boxed{f_{v_0} \circ f_{v_1} \circ ... \circ f_{v_i} \circ ... \circ f_{v_k}(S)} = \underline{\boxed{f_{w_0} \circ f_{w_1} \circ ... \circ f_{w_i} \circ ... \circ f_{w_{k'}}(S^*)}} = f_w(S^*)$$

Da im Kasten in beiden Zeilen gleiche Permutationen stehen (für alle j mit $0 \leq j < i$ gilt $v_j=w_j$), folgt aus der Gleichheit der Bilder die Gleichheit der Urbilder. Also ist

$$f_{v_i} \circ ... \circ f_{v_k}(S) = f_{w_i} \circ ... \circ f_{w_{k'}}(S^*).$$

Somit ist

$$f_{v_i}(f_{v_{i+1}} \circ ... \circ f_{v_k}(S)) = f_{w_i}(f_{w_{i+1}} \circ ... \circ f_{w_{k'}}(S^*))$$

eine Kollision von f_0 und f_1.

Die präfixfreie Codierung wird benötigt, damit sich v und w in einer Bitposition unterscheiden, die bei beiden *vorhanden* ist (andernfalls gäbe es entweder v_i oder w_i nicht).

3-16 RSA

a) *Schlüsselgenerierung:* Sei $p = 3$, $q = 11$, also $n = 33$.
Also ist $\phi(n) = (p\text{-}1) \bullet (q\text{-}1) = 20$.
Sei $c = 3$; prüfe ob ggT(3, 20) = 1. (Euklidscher Algorithmus) Dies ist der Fall.

Für den Entschlüsselungsexponenten muß man unterscheiden, ob man die Standardversion oder mod p und q einzeln rechnen will:
Standardversion: Gesucht $d = c^{-1} \bmod \phi(n)$, also $3^{-1} \bmod 20$. Mit dem erweiterten Euklidschen Algorithmus ergibt sich $d = 7$.
Effizientere Variante:
$d_p := c^{-1} \bmod p–1$, also $d_p := 3^{-1} \equiv 1 \bmod 2$.
$d_q := c^{-1} \bmod q–1$, also $d_q := 3^{-1} \bmod 10$. Mit erw. Eukl. Alg.: $d_q = 7$.

Verschlüsselung: Sei die Klartextnachricht $m = 31$. Dann ist der zugehörige Schlüsseltext $S = 31^3 \equiv (–2)^3 \equiv –8 \equiv 25 \bmod 33$.

Entschlüsselung: Standardversion: $S^d = 25^7 \equiv (–8)^7 \equiv 64^3 \bullet (–8) \equiv (–2)^3 \bullet (–8) \equiv 64 \equiv 31 \bmod 33$, was tatsächlich der Klartext ist.

Effizientere Variante: Mod 3: $S^{d_p} \equiv 1^1 = 1$. Mod 11: $S^{d_q} \equiv 3^7 = 9^3 \bullet 3 \equiv (–2)^3 \bullet 3 \equiv 3 \bullet 3 = 9$. Mit dem CRA (wie üblich) ergibt sich wieder $m = 31$.

b) Um uns das Leben bequem zu machen, können wir das Beispiel aus a) abwandeln: Die Schlüsselgenerierung sei identisch (nur daß c jetzt Testschlüssel, d Signierschlüssel heißt).

Wenn die Klartextnachricht zufällig $m = 25$ ist, so ist die Signatur die 3-te Wurzel daraus, was wie oben $Sig = 31$ ergibt, und beim Testen ergibt sich $Sig^3 = 31^3 = 25$.

Merke aber: 1. Wenn jemand *sowohl* Konzelation *als auch* für möglicherweise andere Nachrichten Authentikation will, sollte er *zwei verschiedene Schlüsselpaare* wählen. Möchte jemand den Klartext zu einem konzelierten RSA-Block erfahren, so braucht er andernfalls nur den Schlüsselpaarbesitzer dazu zu bringen, ihm diesen Block zu signieren. Wird vor dem Signieren auf die Nachricht eine Einweg-Hashfunktion angewendet oder ist das umfassende kryptographische Protokoll so gebaut, daß keine beliebigen Nachrichten signiert werden, so braucht dieses Risiko nicht zum Tragen zu kommen. Wozu aber soll man es überhaupt eingehen, um die Hälfte des nicht sehr großen Schlüsselgenerier- und Speicheraufwandes bei RSA einzusparen?

2. Im Beispiel wurden weder Redundanz noch Hashfunktionen verwendet; das System war also noch anfällig gegen Davida-Angriffe und sollte nicht so verwendet werden.

c) Unabhängig von der Wahl des Schlüsselpaares jeweils als 0 und 1. Wir lernen, daß diese Klartexte durch passende Codierung der Eingabe an RSA vermieden werden sollten.

Wenn Leerzeichen (das Zeichen, das mit weitem Abstand am häufigsten als lange Folge vorkommt) nicht als 0...0 codiert werden, dürfte dies mit hinreichender Wahrscheinlichkeit automatisch der Fall sein. Man kann es auch durch die sowieso benötigte Redundanz bzw. Hashfunktion erreichen.

d) Beispielsweise könnte man 100 Bits an festen Positionen in jedem RSA-Block zufällig wählen.

Man braucht so viele zufällige Bits, daß ein Angreifer, der einen Schlüsseltext S sieht und einen wahrscheinlichen Klartext m rät, wenigstens nicht durch vollständige Suche ausprobieren kann, ob tatsächlich m in S steckt. Dazu würde er selbst m mit dem öffentlichen Schlüssel c verschlüsseln, wobei er der Reihe nach die Möglichkeiten für die zufälligen Bits probiert. Bei 8 Bits bräuchte er nur 256 Versuche.

Die Uhrzeit, selbst wenn sie mehr als 8 Bits enthält, ist nicht zufällig genug: Der Angreifer braucht ja nur die in Frage kommenden Uhrzeiten durchzuprobieren.

e) Wenn die zufällig gewählten Bits zur Ausgabe des asymmetrischen Konzelationssystems gehören, ist die vorgeschlagene indeterministische Variante von RSA nicht sicher gegen aktive Angriffe: Der aktive Angriff geht genauso wie bei deterministischem RSA.

Wenn die zufällig gewählten Bits *nicht* zur Ausgabe des asymmetrischen Konzelationssystems gehören, ist die Antwort schwieriger. Wie in §5.4.6.8, ausführlicher in [PfPf_90] begründet, lautet sie für obigen Vorschlag auch nein, zumindest wenn es die ersten oder letzten hundert Bits sind.

Für eine praktische Anwendung von RSA sollte man also z.B. 100 zufällig gewählte Bits mit einem Redundanzprädikat kombinieren. Das Nachrichtenformat könnte $m^* = (m, z, h(m,z))$ sein (die Reihenfolge ist unerheblich), wobei m die eigentliche Nachricht ist, z die Zufallsbits und h eine Einweg-Hashfunktion. Die Verschlüsselung ist m^{*c}.

Hat h einen Bildbereich von 160 Bit (vgl. §3.8.3), müssen also von der Brutto-RSA-Blocklänge 260 Bit abgezogen werden, um die Netto-RSA-Blocklänge zu erhalten. Bei einer Moduluslänge von 1000 Bit bleiben 740 Bit, also 74% übrig.

f) Zufallsbits können entweder in der zu signierenden Nachricht untergebracht werden oder im mit RSA zu exponenzierenden Block (oder an beiden Stellen).

Werden Zufallsbits in der zu signierenden Nachricht untergebracht, dann werden sie mit gehasht. Für die Sicherheit der digitalen Signatur wichtig ist, daß die Hashfunktion auch dann noch kollisionsresistent ist, wenn die Zufallsbits beliebig gewählt werden können. Bei einer

guten kollisionsresistenten Hashfunktion sollte das der Fall sein – aber wozu dieses zusätzliche Risiko eingehen, wozu diesen zusätzlichen Aufwand treiben, mehr Bits zu hashen?

Werden Zufallsbits in dem zu exponenzierenden Block untergebracht, dann steigt die Gefahr multiplikativer Angriffe. Wozu dieses Risiko eingehen?

Kurzum: Ich würde mit RSA nur deterministisch signieren.

g) Gemäß §3.4.1.1 ist der Rechenaufwand zur Verschlüsselung eines RSA-Blocks der Länge $2l$ bei Länge des Exponenten $|c|$ proportional zu $(2l)^2 \cdot |c|$. Bei extrem kurzen Nachrichten wächst der Rechenaufwand pro Bit also genau so, da nur ein Block verschlüsselt werden muß. Bei extrem langen Nachrichten ist zu berücksichtigen, daß die Anzahl Bits, die in einem RSA-Block untergebracht werden können, mit l wächst. Ignorieren wir Zufallszahl und Hashwert, die ja Platz beanspruchen, dann können in jedem RSA-Block $2l$ Bits verschlüsselt werden. Damit ist der RSA-Rechenaufwand für extrem lange Nachrichten proportional zu $(2l)^2 \cdot |c| / (2l) = 2l \cdot |c|$. Im günstigsten Fall wächst der Rechenaufwand also nur linear mit l. (In der Realität wächst er zumindest für sehr große l noch etwas weniger, da sich dann die in §3.4.1.1 erwähnten schnelleren Algorithmen lohnen. Andererseits wird für extrem lange Nachrichten üblicherweise hybride Verschlüsselung (§3.1.4) und nicht reinrassiges RSA verwendet.)

h) Sei l der Sicherheitsparameter, d.h. die Länge von p, q.

Mod n: Zu berechnen ist $x^d \bmod n$. ($d = c^{-1} \bmod \phi(n)$ wurde schon bei der Schlüsselgenerierung berechnet.) Die Längen von x und d sind jeweils etwa $2l$.

Die Exponentiation mit „square-and-multiply“ besteht also aus etwa $3/2 \cdot 2l = 3l$ modularen Multiplikationen von Zahlen der Länge $2l$. (Hierbei wurde Quadrieren als Multiplikation gezählt.)

Mod p, q: Zu berechnen sind $x^{d_p} \bmod p$, $x^{d_q} \bmod q$ und der CRA dieser beiden Werte. (d_p, d_q und die „Basisvektoren“ für den CRA wurden schon bei der Schlüsselgenerierung berechnet.) Damit bleiben zwei Exponentiationen, wo beide Parameter nur die Länge l haben (x, weil es mod p bzw. q reduziert werden kann, und d_p bzw. d_q, weil sie mod p–1 bzw. q–1 berechnet wurden), und der CRA. Letzterer besteht nur aus zwei Multiplikationen und kann gegen die Exponentiationen vernachlässigt werden.

Es sind also etwa $2 \cdot 3/2 \cdot l = 3l$ modulare Multiplikationen von Zahlen der Länge l.

Der Faktor ist also genau der zwischen einer modularen Multiplikation von Zahlen der Länge l bzw. $2l$. Die modulare Multiplikation besteht aus Multiplikation und Division (jedenfalls in der Standardversion), und der Aufwand beider ist $O(l^2)$, also bei doppelt so langen Zahlen viermal so hoch.

i) Näherungsweise hat ein zufälliger Exponent die Länge $2l$-1. Statt bei einem zufälligen Exponenten $2l$-2 Quadrierungen und $(2l\text{-}2)/2$ Multiplikationen auszuführen, werden 16 Quadrierungen und eine Multiplikation gebraucht. Insgesamt werden also statt $3l$-3 Operationen nur 17 gebraucht. Also wird die öffentliche Operation um den Faktor $(3l\text{-}3)/17$ schneller. Für einen realistischen Wert von 512 für l also um den Faktor $(3 \cdot 512\text{-}3)/17 \approx 90$.

j) Eine prima Idee: Da $\text{kgV}(p–1, q–1) < \phi(n) = (p–1) \cdot (q–1)$ ist, denn p–1 und q–1 enthalten zumindest 2 als gemeinsamen Faktor, ergibt sich bei direktem Ersetzen von $\phi(n)$ durch $\text{kgV}(p–1, q–1)$:

c wird so in Schritt 3 aus einem etwas kleineren Intervall gewählt, innerhalb des Intervalls bleiben aber dieselben Zahlen wählbar. In Schritt 4 wird für c als multiplikatives Inverses d ein im Mittel kleinerer Wert berechnet. Es gilt dann $c \cdot d \equiv 1 \bmod \text{kgV}(p–1, q–1)$.

Hieraus folgt sowohl $c \cdot d \equiv 1 \mod p–1$ als auch $c \cdot d \equiv 1 \mod q–1$. Der Beweis in §3.6.1 für das Funktionieren der Ver- und Entschlüsselung gilt also weiterhin.

Ändert sich durch diese Modifikation die Sicherheit von RSA? Ich halte es für sehr sehr unwahrscheinlich, da sich lediglich die Wahl des sowieso zu veröffentlichenden *c* etwas ändert – und oftmals sowieso ein kleines *c* gewählt wird, um den Verschlüsselungsaufwand gering zu halten, vgl. §3.6.5.1 und auch Aufgabenteil i). Für absolute Puristen bietet sich folgende Hybridlösung an:

Schritt 3 wie in §3.6.1 mit $\phi(n)$, Schritt 4 einmal mit $\phi(n)$ und einmal mit kgV(p–1, q–1). Genommen wird in Schritt 4 das kleinere *d*.

Auf diese Art ändert sich nach außen nichts – die Sicherheit von RSA bleibt also absolut unverändert. Gespart wird etwas beim Entschlüsseln, denn *d* ist etwas kleiner. Allerdings nicht sehr viel, denn bei zufälliger und stochastisch unabhängiger Wahl von *p* und *q* haben *p*–1 und *q*–1 nicht allzuviele gemeinsame Primfaktoren.

k) Wir verwenden das Beispiel aus a) also $n_1 = 33$.
Wir wählen: $n_2 = 5•17 = 85$, $n_3 = 23•29 = 667$.
Also ist $\phi(n_2) = 4•16 = 64$ und $\phi(n_3) = 22•28 = 616$. ggT(3, 64) = 1 und ggT(3, 616) = 1. (Wenn Sie als 7 oder 13 als Primfaktoren von *n* gewählt haben, dann haben sie vermutlich gemerkt, daß $\phi(n)$ dann jeweils durch 3 teilbar ist.)
Für die Klartextnachricht $m = 31$ ergeben sich als zugehörige Schlüsseltexte
$S_1 = 31^3 \equiv (–2)^3 \equiv –8 \equiv 25 \mod 33$
$S_2 = 31^3 \equiv 29791 \equiv 41 \mod 85$
$S_3 = 31^3 \equiv 29791 \equiv 443 \mod 667$
Erweiterter Euklidscher Algorithmus ergibt -18•33 + 7•85 = 1.
CRA ergibt -18•33 • 41+ 7•85 • 25 =: -9479. Mod 33•85 ergibt 1741.
(Probe: 1741 mod 33 = 25 und 1741 mod 85 = 41)
Nun noch mal mit den Zahlen 33•85 = 2805 und 667:
Erweiterter Euklidscher Algorithmus[144] ergibt 778•667 - 185•2805 = 1.
CRA ergibt 778•667 • 1741 - 185•2805 • 443 =: -673566391. Mod 667•2805 ergibt 29791.
(Probe: 29791 mod 667 = 443 und 29791 mod 2805 = 1741)
Die dritte Wurzel aus 29791 ist 31, also genau die Klartextnachricht *m*.

---

[144] Da die Zahlen das Kopfrechnen schon etwas anstrengend machen, gebe ich die Rechnung an.
Euklidscher Algorithmus:
2805 = 4•667 + 137
667 = 4•137 + 119
137 = 1•119 + 18
119 = 6•18 + 11
18 = 1•11 + 7
11 = 1•7 + 4
7 = 1•4 + 3
4 = 1•3 + 1
Rückwärts:
1 = 4 - 3
= 4 - (7-4)
= -7 + 2(11-7)
= 2•11 -3(18-11)
= -3•18 + 5(119-6•18)
= 5•119 - 33(137-119)
= -33•137 + 38(667-4•137)
= 38•667 - 185(2805-4•667)
= 778•667 - 185•2805

l) *Vollständiges* Brechen von RSA bedeutet, zu gegebenem *n* und *c* das zugehörige *d* bestimmen zu können. Im in der Aufgabenstellung zitierten Beweis wird nun gezeigt: Mit Kenntnis von *n*, *c* und *d* kann *n* faktorisiert werden – also ist vollständiges Brechen genauso schwer wie Faktorisieren. Leider schließt dieser Beweis nicht einmal aus, daß *universelles* Brechen, d.h. das Finden eines zum Schlüssel äquivalenten Verfahrens gelingt, ohne *n* faktorisieren zu können.

3-17 DES

a) Verschlüsseln: $K_1 = 011, \quad K_2 = 010.$

$L_0 = 000, \quad R_0 = 101.$

$L_1 = R_0 = 101, \quad R_1 = L_0 \oplus f(R_0, K_1) = 000 \oplus (R_0 \bullet K_1 \bmod 8) = 101 \bullet 011 \bmod 8$
$= 5 \bullet 3 \bmod 8 = 111.$

$L_2 = R_1 = 111, \quad R_2 = L_1 \oplus f(R_1, K_2) = 101 \oplus (111 \bullet 010 \bmod 8) = 101 \oplus (7 \bullet 2 \bmod 8)$
$= 101 \oplus 110 = 011.$

Vertauschen der Hälften ergibt den Schlüsseltext 011111.

Entschlüsseln: Verwendet werden die Bezeichner in der rechten Hälfte des Bildes aus §3.7.3, vgl. die Erklärung dort.

$R_2 = 011, \quad L_2 = 111$ sind gegeben.

$R_1 = L_2 = 111, \quad L_1 = R_2 \oplus f(R_1, K_2) = 011 \oplus (111 \bullet 010 \bmod 8) = 011 \oplus (7 \bullet 2 \bmod 8)$
$= 011 \oplus 110 = 101.$

$R_0 = L_1 = 101, \quad L_0 = R_1 \oplus f(R_0, K_1) = 111 \oplus (R_0 \bullet K_1 \bmod 8) = 111 \oplus (101 \bullet 011 \bmod 8)$
$= 111 \oplus (5 \bullet 3 \bmod 8) = 111 \oplus 111 = 000.$

Wegen der in der Bezeichnerwahl implizit enthaltenen Vertauschung von links und rechts brauchen die beiden Hälften nicht explizit vertauscht zu werden. Also lautet der Klartext: 000101, was mit dem ursprünglich verschlüsselten übereinstimmt.

b) Man legt explizit Tabellen an, so daß man jedes Ergebnis sofort auslesen kann. Möglichst faßt man noch mehrere S-Boxen zu einer großen zusammen (bei DES wenigstens jeweils 2: Array von 2^{12} Einträgen).

c) Der Witz an schnellen Softwareimplementierungen solcher Systeme ist, daß es auch gelingt, die Permutationen im wesentlichen zu tabellieren, die sonst viele Bitoperationen benötigen würden. Eine Tabelle mit 2^{32} Einträgen wäre aber zu groß. Als Beispiel zeigen wir, wie man mit 4 Tabellen von 2^8 Einträgen auskommt:

Die Einträge werden 32 bit lang, und die *i*-te Tabelle beschreibt, wohin die *i*-ten 8 Bits permutiert werden.

Beispiel: 2-te Tabelle, also Bits 9 bis 16. Sei W(9), …, W(16) = 28,13,1,5,30,14,2,19.

In jedem Eintrag steht an Position 28 Bit 9, an Position 13 Bit 10 , … , an Position 19 Bit 16, und alle anderen Bits sind immer 0.

Das Gesamtergebnis der Permutation erhält man, indem man die *i*-te Tabelle mit den *i*-ten 8 Bits der Eingabe indiziert und das Oder oder XOR der 4 Ergebnisse bildet.

d) Für ein Klartextblock-Schlüsseltextblock-Paar werden nacheinander verschiedene Schlüssel probiert, bis einer paßt. Dies ist im Mittel nach 2^{55} Versuchen der Fall. Danach wird probiert, ob dieser Schlüssel auch für die anderen Klartextblock-Schlüsseltextblock-Paare paßt. Dies ist mit sehr großer Wahrscheinlichkeit der Fall. Der mittlere Aufwand ist also 2^{55} DES-Verschlüsselungen.

e) Wie in §3.7.6 beschrieben, wird zu einem der gegebenen Klartextblock-Schlüsseltextblock-Paare ein komplementäres gewählt, d.h. der komplementäre Klartextblock gebildet und der

entsprechende Schlüsseltextblock erfragt. Damit kann dann im Mittel nach 2^{54} Versuchen, also halb so vielen DES-Verschlüsselungen, der Schlüssel ermittelt werden.

Formal hingeschrieben:

Gegeben sei das Klartextblock-Schlüsseltextblock-Paar (x, DES(k, x)).

Gebildet wird $\overline{x}$. Erfragt wird DES(k, $\overline{x}$) .

Nach der Komplementaritätseigenschaft gilt: DES(k, $\overline{x}$) = $\overline{\text{DES}(\overline{k}, x)}$. Durch Komplementieren wird DES($\overline{k}$, x) berechnet. Damit liegt für den Angriff die Verschlüsselung von x sowohl unter dem Schlüssel k wie auch unter dem komplementären Schlüssel $\overline{k}$ vor.

Um den Schlüssel k zu ermitteln, werden solange Schlüssel k' durchprobiert, d.h. DES(k', x) berechnet, bis dies entweder gleich DES(k, x) oder gleich DES($\overline{k}$, x) ist. Im ersten Fall gilt $k = k'$, im zweiten Fall gilt $k = \overline{k'}$.

Wird beim Durchprobieren des Schlüsselraumes darauf geachtet, daß jeder der 2^{56} möglichen Werte von k' und $\overline{k'}$ höchstens einmal angenommen wird, dann endet die Suche nach spätestens 2^{55}, im Mittel nach 2^{54} Schritten, wobei jeder Schritt aus einer DES-Verschlüsselung und zwei Vergleichen des entstehenden Schlüsseltextes mit zwei vorhandenen Schlüsseltexten besteht. Da ein Vergleich wesentlich weniger aufwendig als eine DES-Verschlüsselung ist, ist der hier beschriebene *gewählter Klartext-Schlüsseltext-Angriff* nur halb so aufwendig wie der vorher beschriebene *Klartext-Schlüsseltext-Angriff*.

Auch hier muß ggf. noch an weiteren Klartext-Schlüsseltext-Paaren geprüft werden, ob der gefundene Schlüssel tatsächlich der richtige ist.

f) Für jedes der 64 Schlüsseltextbits gibt es eine lineare Gleichung, in die entsprechend den Permutationen entsprechende Klartextbits und entsprechend den Permutationen und der Teilschlüsselerzeugung entsprechende Schlüsselbits eingehen – wobei die Gleichungen konstante Koeffizienten haben, denn die Permutationen und auch die Teilschlüsselerzeugung sind bekannt und fest. Da Schlüsseltext und Klartext bekannt sind, haben wir 64 lineare Gleichungen mit 56 Unbekannten. Die löst jedes Softwarepaket zur Lösung von Gleichungen spielend. Das Ergebnis ist der gesuchte DES-Schlüssel.

Ob die Blocklänge 64 oder z.B. 128 Bit ist, ob der Schlüssel 56 oder z.B. 256 Bit lang ist, darauf kommt es offensichtlich nicht an. Der Algorithmus bricht offensichtlich jede Feistelchiffre mit bekannten festen Permutationen. Vermutlich läßt er sich auf beliebige lineare Blockchiffren übertragen.

3-18 Betriebsarten

a) Man hat keinerlei Redundanz. Genauer: Jeder Schlüsseltext läßt sich zu irgendeinem Klartext entschlüsseln. Wenn man dann wieder verschlüsselt, ergibt sich natürlich genau wieder dieser Schlüsseltext und insbesondere sein letzter Block. Wenn die Nachrichten also keine interne Redundanz enthalten, etwa bei Files von Zahlen, kann der Angreifer beliebige Schlüsseltexte schicken.

b) (Dieses Schema ist etwas besser: Bei einem x-beliebigen Schlüsseltext wird i.allg. der sich ergebende letzte Klartextblock nicht aus lauter Nullen bestehen, also wird der Angriff erkannt. Aber:)

Wenn der Schlüsseltext aus mindestens drei Blöcken besteht, kann der Angreifer alle außer den letzten zweien beliebig ändern. Da die Betriebsart CBC sich innerhalb von 2 Blöcken

selbst synchronisiert, wird der letzte Block trotzdem richtig entschlüsselt, also als lauter Nullen, und der Angriff wird nicht automatisch erkannt.

c) Wenn hier ein Schlüsseltextblock geändert wird, so wird die Entschlüsselung dieses Blocks vollkommmen anders. Somit steht beim nächsten Block im Klartextblockspeicher ein ganz falscher Block. Dieser Fehler wird i.allg. nicht wegsynchronisiert, da bei jedem Block nach der Entschlüsselung eine Funktion des alten falschen Klartexts dazuaddiert wird, so daß sich wieder ein falscher Klartext ergibt und somit auch der Klartextblockspeicher wieder falsch geladen wird.
{Die Begründung, PCBC sei eine synchrone Chiffre, und dies sei hinreichend, daß der Angriff nicht funktioniert, ist falsch. Hierfür ist „synchron“ in §3.8.1 zu schwach definiert, da hierfür selbst die Abhängigkeit von der Position innerhalb der Nachricht reicht. Ein Beispiel möge dies erläutern: OFB ist eine synchrone Stromchiffre, aber solange man keine Schlüsseltextbits des Authentikators ändert oder Schlüsseltextbits wegläßt, kann man im Schlüsseltext beliebig ändern.}

d) ECB: Der Angreifer kann nicht viel erreichen, da die Blockchiffre als sicher angenommen und in ECB eins-zu-eins als Blockchiffre verwendet wird. Somit verbleiben nur die bei der Definition der Blockchiffre erwähnten Nachteile und die ihnen entsprechenden Angriffe: Bei *deterministischen* Blockchiffren kann der Angreifer seine Möglichkeit zum aktiven Angriff nutzen und wahrscheinliche Klartexte verschlüsseln lassen. Die zugehörigen Schlüsseltexte kann er dann mit beobachteten Schlüsseltexten vergleichen. Mit Glück kann er so manche der beobachteten Schlüsseltexte „entschlüsseln“, selbst bei Pech kann er so manche Klartexte ausschließen.

CBC zur Konzelation: Der Angreifer kann durch aktive Angriffe die Werte in den Speichergliedern setzen. Werden diese Werte dann einfach weiterverwendet, d.h. beginnt nicht jede Nachricht mit einem vom Sender gewählten Wert der Speicherglieder, so kann nach einem aktiven Angriff auf den ersten Block ein Angriff wie für ECB und mit *gleicher* Erfolgswahrscheinlichkeit durchgeführt werden. Wird der Wert der Speicherglieder nach den aktiven Angriffen jeweils wieder neu gesetzt, so kann der Angreifer zwar im Schlüsseltext nach Blöcken suchen, deren Klartext er kennt (beispielsweise von einem aktiven Angriff in der Betriebsart ECB oder CBC). Zu diesen Blöcken kann er dann den ihnen in der Betriebsart CBC entsprechenden Klartext errechnen: Er braucht nur den vorherigen Schlüsseltext zu subtrahieren. Aber dieses Suchen ist mühseliger, da der Angreifer die verschlüsselten Werte nicht so gut vorhersehen (und damit während seines aktiven Angriffs verschlüsseln lassen) kann. Seine Erfolgswahrscheinlichkeit ist also bei geeignetem Neusetzen der Werte der Speicherglieder *geringer* als bei dem entsprechenden Angriff auf ECB.

CBC zur Authentikation: Werden einzelne Blöcke authentiziert, so gilt alles für Konzelation mit CBC Gesagte entsprechend. Bei der Authentikation mehrerer Blöcke zusammen muß der Angreifer selbst im günstigen Fall, daß die Werte der Speicherglieder nicht immer wieder neu initialisiert werden, die genauen Werte der Blöcke und ihre Folge richtig raten, eine bei sinnvoller Blocklänge fast aussichtslose Aufgabe. Vorausgesetzt wird hierbei, daß die zu authentisierenden Werte vom Angreifer nicht schon während seines aktiven Angriffes „abgefragt“ werden können, da hiergegen kein kryptographisches Verfahren Sicherheit bieten kann. Dieses „Nicht-rechtzeitig-vorher-Wissen“ sicherzustellen, ist Aufgabe des umgebenden Protokolls. Im einfachsten Fall könnte jeweils der Partner einen Teil der zu authentisierenden Nachricht erzeugen.

CFB zur Konzelation: Der Angreifer ist immer dann bezüglich der folgenden Ausgabeeinheit erfolgreich, wenn er die Verschlüsselung des im Schieberegister befindlichen Wertes kennt.

Werden die Werte im Schieberegister nicht vor jeder Nachricht neu initialisiert, so kann der Angreifer zu einem von ihm gewählten Klartext den Schlüsseltext mittels eines aktiven Angriffs erfragen (geht sowohl bei der Betriebsart ECB, CBC, als auch CFB) und danach diesen Klartext im Schieberegister hinterlassen und geduldig auf die nächste Ausgabeeinheit warten, den ihm bekannten Wert subtrahieren, sich freuen ...

CFB zur Authentikation: Es gilt das bzgl. „CBC zur Authentikation“ Gesagte.

OFB: Der Angreifer muß den Wert im Schieberegister vorhersehen oder herbeiführen. Letzteres kann er nicht durch die Wahl eines geeigneten Klartextstromes, da dieser nicht in die Rückführung eingeht. Es hängt also allein von der Initialisierung des Schieberegisters ab, ob ein aktiver Angriff erfolgreich ist.

PCBC: Um die bei CBC beschriebenen Vorteile, d.h. *gleiche* Erfolgswahrscheinlichkeit wie bei ECB, zu erreichen, kann ein Angreifer auch bei PCBC versuchen, die Werte in den beiden Speichern aktiv zu setzen, um einen ihm genehmen Wert von h einzustellen. Außer bei Initialisierungen hat er aber hierbei gegenüber CBC zusätzliche Schwierigkeiten, was der folgende Versuch, bei PCBC die Werte in den Speichern auf gewünschte Werte zu setzen, verdeutlicht:
Sende Klartextblock $K1$. Danach ist der Wert im Speicher für den Klartextblock bekannt ($K1$) und durch Beobachtung des ausgegebenen Schlüsseltextblocks $S1$ auch der Wert im Speicher für den Schlüsseltextblock. Hieraus kann $h(K1,S1)$ berechnet werden. Trotzdem ist es nun schwierig, beide Werte in den Speichern auf gewünschte Werte zu setzen. Zwar kann man den Klartextblock $K2$ so wählen, daß $K2+h(K1,S1)$ einen gewünschten Wert annimmt und dadurch der diesem Wert zugeordnete Schlüsseltextblock $S2$ entsteht. Aber hierzu muß man den Klartextblock $K2$ variieren und hierdurch ändert sich der Wert im Speicher für den Klartextblock. Entsprechendes gilt, wenn man den Wert des Klartextblocks $K2$ wählt. Dann bewirkt die Addition von $h(K1,S1)$ einen unerwünschten Wert $S2$.
Für geeignete Funktionen h ist das Einstellen eines Wertes aus einer kleinen Menge von Werten also schwierig. Deshalb kann selbst eine dem Angreifer bekannte ungleichmäßige Verteilung von Klartexten vermutlich nicht zur Erzielung von Vorteilen bei aktiven Angriffen genutzt werden, wenn die Initialisierung der Speicher geeignet erfolgt, vgl. das unter „CBC zur Authentikation“ Gesagte.

OCFB: Im Gegensatz zu PCBC kann bei OCFB der Wert im Schieberegister vom Angreifer auch ohne Initialisierung eingestellt werden, sofern

- h keine Einwegfunktion ist und
- er das Ergebnis der Verschlüsselung *Erg* erfährt, bevor er sich auf die zugehörige Einheit des Klartextstromes festlegen muß:

Mit *Erg* kennt der Angreifer den Wert des einen Eingabeparameters von h und kann, sofern h keine Einwegfunktion ist, zu vielen gewünschten Ergebnissen von h einen passenden Wert des zweiten Eingabeparameters ermitteln. Von diesem Wert subtrahiert der Angreifer „Wähle aus“(*Erg*) und erhält so die Einheit des Klartextstromes, die er eingeben muß, um das von ihm gewünschte Ergebnis von h zu erhalten. Wiederholt der Angreifer dies $\lceil b/r \rceil$ mal, so steht im Schieberegister ein von ihm gewünschter Wert. Damit gilt das unter „CFB zur Konzelation“ Gesagte entsprechend.

e) Die Codierung erfolgt folgendermaßen: Hänge an den Klartext ein Bit eines festen Wertes, z.B. 1. Entsteht hierdurch nicht ein Vielfaches der Blocklänge, so fülle danach mit Bits des anderen Wertes, z.B. 0, bis zum nächsten Vielfachen der Blocklänge auf.
1. gilt, da bis zum nächsten Vielfachen der Blocklänge aufgefüllt wird.

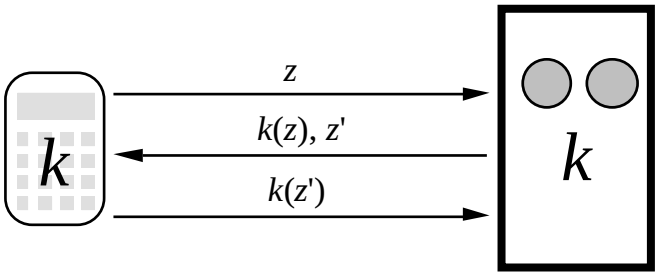
2. gilt, da von hinten bis zum ersten Bit des festen Wertes, z.B. 1, zurückgegangen werden kann. Alle diese Bits sind angehängt – ihr Abhängen ergibt den ursprünglichen Klartext.
3. erfordert zunächst, daß definiert wird, was "minimale Längenexpansion" bedeutet. Sinnvolle Definitionen könnten lauten:
   - Die Expansion über alle möglichen Klartexte ist minimal.
   - Die Expansion über alle möglichen Klartexte, gewichtet mit der Wahrscheinlichkeit ihres Auftretens, ist minimal. Bei Gleichverteilung ist diese Definition gleich der vorherigen.
   - Im ungünstigsten Fall, d.h. für den ungünstigsten Klartext, ist die Expansion minimal.

   Für die letzte Definition etwa gilt die Behauptung für die angegebene Codierung, da jede solche Codierung bei zumindest einem Klartext immer mindestens ein Bit hinzufügen muß (um zwischen "aufgefüllt" und "nicht aufgefüllt" zu unterscheiden) und die angegebene Codierung, wenn möglich, nur genau ein Bit hinzufügt. Vermutlich erfüllt die angegebene Codierung auch die erste Definition (Sie können es ja mal versuchen zu beweisen). Bei der zweiten Definition kann man für jede feste Codierung die Verteilung des Auftretens der Klartexte so wählen, daß die Codierung nicht minimal längenexpandierend ist.

f) Das Schieberegister und die Rückkopplung wird durch einen Zähler ersetzt, der ausgehend von einem Initialisierungswert für jede Ausgabeeinheit hochzählt. Um Ausgabeeinheit *i* zu bearbeiten, wird also zum Initialisierungswert *i*-1 addiert – dann kann's losgehen. Naheliegenderweise heißt diese Betriebsart **Counter Mode** [Schn_96 Seite 205].

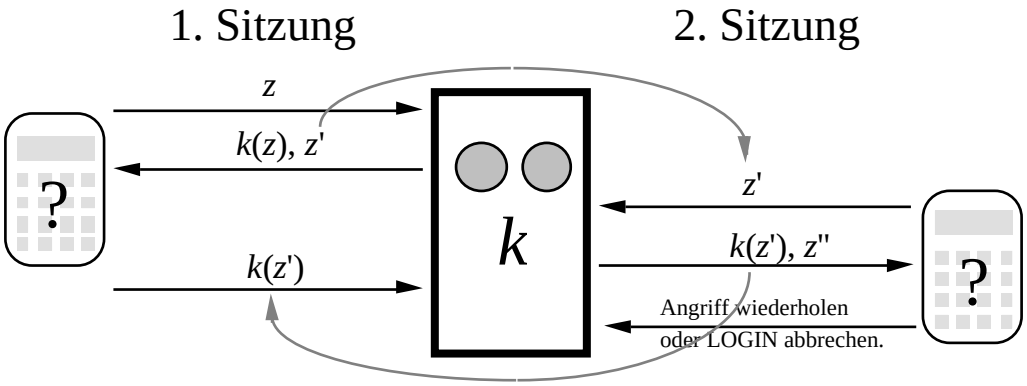
3-18a Spiegelangriff

Der Großrechner implementiere (neben anderen nützlichen Funktionen) dieselbe sichere Blockchiffre. Zwischen Großrechner und jedem Teilnehmer (genauer: „Taschenrechner“) sei jeweils ein unabhängig gewählter Schlüssel ausgetauscht. Beim LOGIN führen sie jeweils folgendes Identifikationsprotokoll durch: Sende Zufallszahl, erwarte Verschlüsselung, entschlüssele, vergleiche. All dies in beiden Richtungen, d.h. Großrechner identifiziert Teilnehmer und Teilnehmer identifiziert Großrechner. (Der Einfachheit halber ist im folgenden Bild die mit zu übermittelnde Identität des Teilnehmers, damit der Großrechner unter vielen ihm bekannten den richtigen Schlüssel auswählen kann, nicht gezeichnet.)



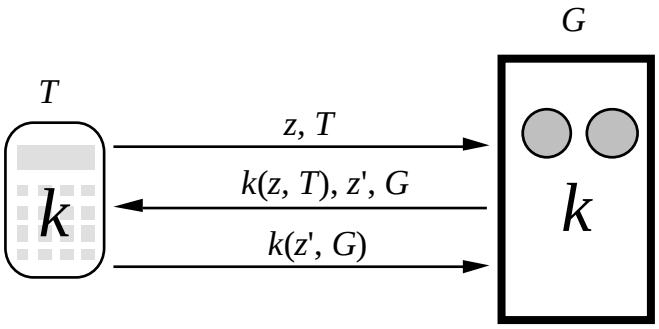
Verhalten sich „Taschenrechner“ und Großrechner absolut gleich (wie in der bisherigen Beschreibung und im Bild unterstellt), so ist dies Protokoll unabhängig von der Stärke der Blockchiffre noch unsicher: Erlaubt der Großrechner mehrere Sitzungen gleichzeitig, so kann die von ihm generierte, zweite Zufallszahl in der ersten Sitzung als erste Zufallszahl in der zweiten Sitzung verwendet werden. Wenn der Großrechner dies nicht merkt (etwa durch die restriktive Maßnahme: nur ein LOGIN pro Teilnehmer gleichzeitig) und in Sitzung zwei richtig antwortet, kann ihm die in Sitzung zwei gegebene richtige Antwort auf seine eigene Frage in Sitzung eins als richtige Antwort gegeben werden. Mit diesem *Spiegelangriff* erkennt der Großrechner also nicht den symmetrisch gleichen „Taschenrechner“, sondern im Spiegel nur sich selbst. Pech gehabt!

Kleiner Denkfehler – große Unsicherheit. (Entsprechendes gilt für den „Taschenrechner“, wenn er auch als Zweiter im Protokoll agiert, das Protokoll also auch bzgl. der Rollen in ihm symmetrisch ist. Vorstellbar wäre dies etwa, wenn Teilnehmer sich gleichzeitig in mehreren Großrechnern einloggen können.)

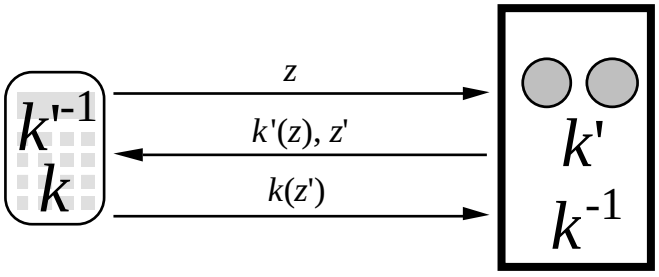


Richtige Protokolle kann man erhalten, indem entweder die beteiligten Instanzen ein Gedächtnis bekommen und an die Zufallszahlen Bedingungen stellen (was der Idee Zufallszahl eigentlich widerspricht) oder sich beide nicht symmetrisch verhalten. Dies können sie, indem entweder

- vor dem Verschlüsseln z.B. die eigene Identität (etwa *T* für Taschenrechner und *G* für Großrechner) mit der Zufallszahl konkateniert wird oder



- der eine nur ver- und der andere nur entschlüsselt oder
- statt einem Schlüssel je Paarbeziehung zwei verwendet werden. Mit dem einen verschlüsselt der Großrechner und der „Taschenrechner“ entschlüsselt mit ihm nur, und mit dem anderen verschlüsselt nur der „Taschenrechner“ und der Großrechner entschlüsselt mit ihm nur.



Wenig geschickt, aber auch sicher, wäre es, wenn – wie in 3-4 e) – jeder selbst erst dann auf Zufallszahlen reagiert, wenn er auf alle von ihm selbst ausgesendeten schon Antwort erhalten hat. Damit könnte der Großrechner dann keine LOGINs mehr parallel abwickeln.

Haben Großrechner und „Taschenrechner“ eine genügend genaue Uhr, so kann man sich einmal Abtippen vom Terminal ersparen, indem statt Zufallszahlen Datum und Uhrzeit verwendet werden. (Hier sei an Aufgabe 2-3 erinnert und die dortige Bemerkung, möglichst schwache Annahmen über die Zufallszahlenerzeugung zu machen.)

Nicht toleriert wird, daß ein Angreifer z.B. nach dem LOGIN „einsteigt“. Um diesen Angriff abzuwehren, ist kontinuierliche Authentikationsprüfung nötig.

(Man beachte aber, daß in dieser Aufgabe eine Blockchiffre mit ziemlich starken Sicherheitseigenschaften benötigt wird. Sie wird ja hier eher als Authentikations- als als Konzelationssystem verwendet, und nicht jedes Konzelationssystem eignet sich auch zur Authentikation, s. Aufg. 3-7 d). Es muß z.B. gelten, wenn man die Uhrzeit verwendet, daß ein Angreifer nicht aus der Verschlüsselung einer Nachricht die Verschlüsselung anderer, ähnlicher Nachrichten erraten kann.)

3-19 Ende-zu-Ende-Verschlüsselung

Egal ob mit symmetrischer oder asymmetrischer Kryptographie gearbeitet wird: Jeder Benutzer benötigt zumindest einen geheimen Schlüssel. Das Hauptproblem ist bei einem von mehreren Personen genutzten Rechner also, wo jeder seine geheimen Schlüssel so aufheben kann, daß die anderen legitimen Benutzer des Rechners sie nicht unbefugt nutzen können.

Traut man dem Betriebssystem, daß es verschiedene legitime Benutzer bezüglich des Zugriffs auf vertrauliche Daten voneinander isoliert, und dem Hardwareschutz (Gehäuse etc.) des Rechners bzgl. der Vereitelung von Eindringversuchen physischer Art, so ist das Problem per Definition gelöst.

Vertraut man dem Rechner gar nicht, so muß nicht nur die Schlüsselspeicherung, sondern auch die Verschlüsselung, Entschlüsselung, das Testen von Signaturen und vor allem das Signieren in einem separaten, persönlichen Rechner geschehen. Dieser sollte, damit er mit seinem Benutzer direkt kommunizieren kann, zumindest über eine kleine Tastatur und ein kleines Display verfügen. Physische Ausprägungen reichen von sogenannter „Super smart card“ (Chipkarte mit Display und Tatstatur), über sogenannte Smart Diskettes (Rechner in Diskettenform, die mit anderen über deren Laufwerk kommunizieren, so daß kein Chipkartenleser benötigt wird und mehr Platz als in einer Chipkarte mit den genormten Magnetstreifenkartenabmessungen zur Verfügung steht) bis Notebookcomputer.

Ein Kompromiß bzgl. Aufwand und Sicherheit besteht darin, daß jeder Benutzer beim LOGIN ein Paßwort solcher Länge (genauer: Entropie) eingibt, daß aus ihm ein Schlüssel eines symmetrischen Konzelationssystems gebildet werden kann. Mit diesem Schlüssel werden dann die eigentlich für die Ende-zu-Ende-Verschlüsselung benötigten Schlüssel zu Sitzungsbeginn ent- und am Sitzungsende wieder verschlüsselt. Dieses Verfahren ist natürlich zu brechen, wenn entweder die Paßworte doch zu wenig Entropie enthalten oder das symmetrische Konzelationssystem gebrochen werden kann oder andere Benutzer durch ein Anwendungs-Programm, was das LOGIN simuliert (so daß der nächste Benutzer glaubt, er kommuniziere mit dem Betriebssystem), an eine Kopie des Paßwortes gelangen können. Letzteres kann etwa dadurch verhindert werden, daß vor dem LOGIN der Rechner grundsätzlich auf Hardwaremanipulationen untersucht und mit dem authentischen Betriebssystem gebootet wird. Damit sind „nur“ noch Eindringversuche erkennende Gehäuse, vgl. §2.1, und authentisches Booten nötig, vgl. [Cles_88, ClPf_91, Groß_91].

3-20 Ende-zu-Ende-Verschlüsselung ohne vorher ausgetauschten Schlüssel

a) Wer dem andern eine Nachricht senden will, generiert *viele One-time pads*. Er sendet dann die pads einzeln über *verschiedene Wege* (z.B. via Umweg über einem Bekannten und möglichst von unterschiedlichen Orten aus, z.B. Wohnung, Büro, Telefonzellen) und *Medien* (z.B. Telefon, Telex, Datex) und dies möglichst noch zu unterschiedlichen *Zeiten* (sofern der Dienst dies zuläßt, d.h. keine Realzeitforderungen stellt). Die Summe aller pads wird dann zum Ver-

bzw. Entschlüsseln und/oder Authentisieren der Nachricht genommen. Ein Angreifer muß dann zu vielen verschiedenen Zeiten an vielen verschiedenen Orten präsent sein, um alle Pads und die verschlüsselte Nachricht abzuhören bzw. die authentisierte Nachricht zu verfälschen. Die Maßnahme erschwert zwar Angriffe auf die Vertraulichkeit und Integrität, erleichtert aber aktive Angriffe auf die Verfügbarkeit: Es genügt, daß der Angreifer eines der pads (oder natürlich auch die eigentliche Nachricht) abfängt oder verfälscht.

Als kryptographische Systeme können das One-time pad und informationstheoretisch sichere Authentikationscodes verwendet werden. Ersteres geht nun wirklich einfach und auch letzteres mit vertretbarem Aufwand ohne Rechnerhilfe.

b) Nachdem wieder beide je ein Kryptogerät haben, können sie auch *asymmetrische* kryptographische Systeme verwenden. Hierzu müssen alle Kryptogeräte, die ihre geheimen Schlüssel gelöscht haben, neue Schlüsselpaare erzeugen. *Zusätzlich* zu a) sagt einer dem andern z.B. am Telefon noch die öffentlichen Schlüssel – denn Alice Kryptogerät hat auch Bobs öffentliche Schlüssel gelöscht, da es beim besten Willen nicht vorhersehen konnte, ob nur die geheimen Schlüssel ausgeforscht oder auch die öffentlichen unbefugt geändert werden sollten. (Heutzutage wird durch die Erkennung des Sprechers beim Telefonieren der Inhalt authentifiziert. Klappt demnächst Sprachsynthese auch in der Variante Sprecherimitation gut, fällt dieses Argument weg. Man sollte sich also nicht *allein* auf diese Authentikation verlassen.) Danach werden mit den öffentlichen Schlüsseln noch ein oder mehrere zusätzliche pads ausgetauscht oder signierte Nachrichten direkt geprüft.

Haben Alice und Bob Vorsorge für den Fall eines Schlüsselverlustes getroffen und ein gemeinsames Geheimnis vereinbart, das beide in ihrem Kopf (und folglich immer dabei) haben, können sie natürlich auch dieses Geheimnis als Startwert für die Generierung eines gemeinsamen Schlüssels verwenden. Insbesondere interessant ist dies für Fall b). Aus einem von beiden leicht zu merkenden Passphrase (einer relativ langen Zeichenkette mit wenig Entropie pro Zeichen) kann durch Anwenden einer kryptographischen Hashfunktion (die allgemein bekannt sein kann) ein kürzerer Funktionswert mit mehr Entropie pro Zeichen gewonnen werden. Dieser Funktionswert kann dann als Startwert für die Schlüsselgenerierung oder auch direkt als symmetrischer Schlüssel verwendet werden.

3-21 Identifikation und Authentikation mittels asymmetrischem Konzelationssystem

a) *U* generiert eine Zufallszahl *z* und verschlüsselt diese so, daß nur *T* sie entschlüsseln kann: $c_T(z)$. *U* sendet $c_T(z)$ an die Instanz, die *T* sein soll, und erwartet, daß sie ihm – je nach Protokoll – entweder *z* zurückschickt oder *z* in einer weiteren kryptographischen Operation verwendet und das Operationsergebnis zurückschickt. Tut sie das, so war (unter der Annahme, daß c_T authentisch und das asymmetrische Konzelationssystem sicher ist und *z* wirklich zufällig gewählt wurde) das „richtige" *T* (mit durch die Länge von *z* bestimmter Wahrscheinlichkeit) involviert. Hiermit hat *U* den Teilnehmer *T* identifiziert. Aber Vorsicht: Wird so naiv vorgegangen, funktioniert der in Aufgabe 3-18a beschriebene verändernde Angriff auch hier. Will sich ein Angreifer *A* als Teilnehmer *T* ausgeben, schickt er $c_T(z)$ einfach an *T* weiter und das, was er von *T* erhält, danach an *U*. Kurzum: Das Protokoll identifiziert nicht den direkten Kommunikationspartner, sondern stellt nur sicher, daß *T* auch involviert ist, insbesondere *z* erfährt. Codiert *U* die *Botschaft*, die er *T* zukommen lassen möchte, also in diese erste Nachricht, etwa $z := c_T(Botschaft)$, so erfüllt das Protokoll seinen Zweck doch. *U* weiß nach Erhalt von *z*, daß *T* die Botschaft *Botschaft* erhalten hat.

(Warnung: Wenn sie eine wesentlich andere Lösung haben, dann prüfen Sie, ob sie *T* seinen geheimen Schlüssel d_T haben auf etwas anwenden lassen, das vorher nicht mit c_T verschlüsselt wurde. Solch eine Operation, d.h. erst d_T (zum Signieren) und danach c_T (zum Prüfen der

Signatur) ist zwar bei manchen Konzelationssystemen, z.B. RSA, möglich – aber eben *nicht* bei allen. Ist es möglich, so können Sie das Konzelationssystem auch als Signatursystem verwenden, wodurch die Aufgabe natürlich trivial lösbar ist.)

b) Das asymmetrische Konzelationssystem muß gegen adaptive Angriffe mit gewählten Schlüsseltexten sicher sein. Sonst können die Antworten des sich Identifizierenden zum Brechen seines asymmetrischen Konzelationssystems verwendet werden.

c) Konstruktion eines symmetrischen Authentikationsprotokolls aus einem asymmetrischen Konzelationssystem:
   1. T schickt Nachricht N_i an U.
   2. U generiert Zufallszahl z, verschlüsselt N_i und z und schickt $c_T(N_i,z)$ an T.
   3. T entschlüsselt mit d_T und prüft, ob sein N_i aus Schritt 1 dem in Schritt 2 erhaltenen entspricht. Wenn ja, schickt er z an U, ansonsten gibt er z nie aus.
   4. Erhält U die von ihm generierte Zufallszahl z, so weiß er, daß N_i von T und authentisch ist.

   Das Authentikationssystem ist nur symmetrisch, da die Zufallszahl z für andere keinerlei Aussagekraft, insbesondere also keinerlei Beweiswert hat – schließlich wurde z von U und nicht etwa T generiert. Also kann U sich z beliebig selbst zuschicken.
   Der Nachteil gegenüber normalen Authentikationssystemen ist, daß das Protokoll *drei* Nachrichten statt *einer* braucht. Es auszuführen dauert also einerseits länger und belastet andererseits das Kommunikationsnetz mehr.

   Tabellarische Darstellung: Symmetrische Authentikation mittels asymm. Konzelationssystem

   Vorwissen: U kennt den öffentlichen Chiffrierschlüssel c_T von T.

T	—— N_i ——>	U
T	<—— $c_T(N_i, z)$ ——	U
T	—— z ——>	U

   Nachwissen: Wenn das asymmetrische Konzelationssystem die Eigenschaft hat, daß jede vom Angreifer effektiv durchführbare Änderung von $c_T(N_i, z)$ auch z ändert, was bei einer Blockchiffre im allgemeinen gegeben sein sollte, dann gilt nach der dritten Nachricht: U weiß, daß Nachricht N_i von T stammt.

   Dies entspricht: Symmetrische Authentikation mittels symm. Authentikationssystem

   Vorwissen: U und T kennen k, Schlüssel eines symm. Authentikationssystems.

T	—— $N_i, k(N_i)$ ——>	U

d) Da hier T die Nachricht N_i bildet und U nur einen Anteil, nämlich z liefert und T nicht reagiert, wenn N_i nicht enthalten ist, muß hier das asymmetrische Konzelationssystem keinem adaptiven aktiven Angriff in seiner schärfsten Form standhalten.

3-22 Konzelation mittels symmetrischem Authentikationssystem

Statt jedes Bits der zu konzelierenden Nachricht wird jeweils nur ein passender MAC übertragen. Der Empfänger probiert dann jeweils aus, welches Bit zum MAC paßt.

Sender *A*		**Empfänger *B***
Kennt k_{AB}		Kennt k_{AB}
Zu übertragen sei Nachricht $b_1, \ldots b_n$ mit $b_i \in \{0, 1\}$		
Berechnet $MAC_1 := code(k_{AB},b_1) \ldots MAC_n := code(k_{AB},b_n)$		
Überträgt $MAC_1 \ldots MAC_n$	——————————>	Probiert, ob $MAC_1 = code(k_{AB},0)$ oder $MAC_1 = code(k_{AB},1)$ und empfängt den passenden Wert b_1 … probiert, ob $MAC_n = code(k_{AB},0)$ oder $MAC_n = code(k_{AB},1)$ und empfängt den passenden Wert b_n

Damit der Empfänger ausprobieren kann, welches Bit zum MAC paßt, müssen die MACs lang genug sein. Der kleine Authentikationscode aus Bild 3-15 beispielsweise ist ungeeignet: Bei Schlüssel 00 kann mit dem MAC 0 sowohl H wie auch T authentisiert werden – der Empfänger kann also nicht entscheiden, ob H oder T übermittelt wird. Entsprechendes gilt für den Schlüssel 11 für MAC 1. Ist der MAC genügend lang, so tritt solch ein Fall nur mit vernachlässigbarer Wahrscheinlichkeit auf.

Die Nachricht ist mit dem beschriebenen Verfahren perfekt konzeliert, da außer Sender und Empfänger niemand prüfen kann, welche Bits zu den MACs passen und welche nicht. Denn könnte das jemand mit einer besseren Wahrscheinlichkeit als 0,5, dann wäre dies ein Ansatz zum Brechen des Authentikationssystems.

Sind die MACs genügend lang, dann ist die konzeliert übertragene Nachricht sogar noch authentisiert: Ändert ein Angreifer einen MAC, so paßt der geänderte MAC mit überwältigender Wahrscheinlichkeit entweder weder zu 0 noch zu 1 oder aber weiterhin zum gleichen Bitwert. Wäre es anders, so könnte das Authentikationssystem mit nichtvernachlässigbarer Wahrscheinlichkeit gebrochen werden: Der Angreifer würde den MAC in gleicher Weise ändern und das bei normaler Verwendung des Authentikationssystems explizit übertragene Nachrichtenbit invertieren.

Statt einzelne Bits zu authentisieren, kann es effizienter sein, dies jeweils mit einem MAC für mehrere Bits gleichzeitig zu tun. Dies spart Übertragungsbandbreite (mehr Bits pro MAC), erfordert aber bei l gleichzeitig authentisierten Bits pro MAC bei vollständigem Durchprobieren 2^l und bei Abbruch nach Erfolg im Mittel 2^{l-1} Probierschritte des Empfängers. Werden l Bits einzeln übermittelt, so ist der Aufwand bei vollständigem Durchprobieren $l \bullet 2$, wird bei Fehlschlagen des ersten Probierens einfach der andere Bitwert genommen, so ist der Aufwand l. Vergleichen wir den Aufwand zwischen l Bits zusammen und l Bits einzeln übermittelt: Da für l=2 sowohl $2^l = l \bullet 2$ wie auch $2^{l-1} = l$ ist, hat l Bits zusammen nur Vorteile, während danach der Rechenaufwand pro Bit empfängerseitig stark steigt. Vielleicht ist l=8 für viele Anwendungen ein sinnvoller Kompromiß.

3-23 Diffie-Hellman Schlüsselaustausch

a) *Kernidee:*

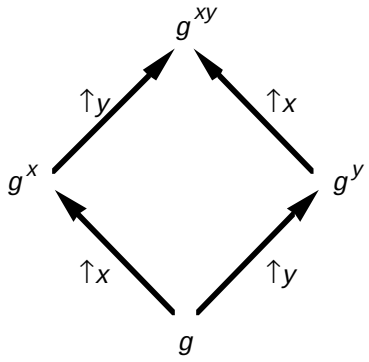
Die Kernidee ist, daß die *Exponentiation kommutativ* ist:

Die Reihenfolge, in der exponenziert wird, ist egal.

Als Formel geschrieben:

Für beliebige Elemente g, x, y gilt: $(g^x)^y = g^{xy} = g^{yx} = (g^y)^x$.

Bildlich dargestellt:



b) *Anonymität:*

Organisatorische Variante: Es ist keineswegs notwendig, daß den öffentlichen Schlüsseln Personen erkennbar zugeordnet sein müssen. Es kann sich auch um Pseudonyme handeln.

Protokollvariante: Statt seinen publizierten öffentlichen Schlüssel zu nehmen, bildet der Sender für jede anonym zu übermittelnde und zu verschlüsselnde Nachricht jeweils ein neues Schlüsselpaar. Ausgehend von dem neuen geheimen Schlüssel bestimmt er den gemeinsamen geheimen Schlüssel mit seinem Kommunikationspartner und verschlüsselt die Nachricht entsprechend. Danach sendet er diese verschlüsselte Nachricht zusammen mit dem eigens generierten neuen öffentlichen Schlüssel[145] an seinen Partner. Der verwendet diesen zugesandten öffentlichen Schlüssel zusammen mit seinem geheimen zur Berechnung des gemeinsamen geheimen Schlüssels und entschlüsselt damit die Nachricht. (Falls Sie mal den Namen asymmetrische Verschlüsselung mit dem **ElGamal-Konzelationssystem** hören: Es ist ein Spezialfall des Beschriebenen. ElGamal verwendet als symmetrische Verschlüsselung im beschriebenen hybriden Konzelationssystem die modulare Multiplikation [ElGa_85, Schn_96 Seite 478].)

Wenn der Sender eine anonyme Antwortnachricht will? Nichts leichter als das: Der gemeinsame geheime Schlüssel kann einfach weiterverwendet werden.

Damit sind asymmetrische Konzelationssysteme und Diffie-Hellman Schlüsselaustausch auch bzgl. Anonymität gleichwertig.

c) *Individuelle Parameterwahl:*

Jeder Teilnehmer wählt für sich sein p und sein g, die er als Teil seines öffentlichen Schlüssels veröffentlicht.

Jeder kann mit einem anderen Teilnehmer einen geheimen Schlüssel austauschen, indem er dessen p und g nimmt und damit wie in §3.9.1 beschrieben verfährt. Danach teilt er dem anderen Teilnehmer seinen gerade berechneten, paarspezifischen öffentlichen Schlüssel mit.

Wegen des letzten Schrittes ist diese Modifikation des Diffie-Hellman Schlüsselaustauschs nicht geeignet für Steganographie mit öffentlichen Schlüsseln, vgl. §4.1.2. Sofern nicht „Faster key generation“ eingestellt ist, wird diese Modifikation des Diffie-Hellman Schlüsselaustauschs bei PGP ab Version 5 verwendet. Sie ist praktisch die Protokollvariante

---

145 Deshalb ist diese Modifikation des Diffie-Hellman Schlüsselaustauschs nicht geeignet für Steganographie mit öffentlichen Schlüsseln, vgl. §4.1.2.

der Lösung von Aufgabenteil b), nur daß zusätzlich jede Instanz ihr eigenes *p* und *g* wählt und veröffentlicht.

Auch wenn Teilnehmer paarspezifisch öffentliche Schlüssel erzeugen, darf natürlich nicht vergessen werden, daß die öffentlichen Schlüssel authentisch sein müssen. Dies kann beispielsweise durch ihre Zertifizierung, vgl. §3.1.1.2, und Prüfung der Zertifikate durch den Empfänger sichergestellt werden. Kann der Sender digitale Signaturen leisten, die der Empfänger prüfen kann, so wird es oftmals zweckmäßig sein, daß der Sender seine paarspezifischen öffentlichen Schlüssel selbst zertifiziert, vgl. auch §9.2.

d) *DSA (DSS) als Basis:*

Diffie-Hellman Schlüsselaustausch funktioniert auf dieser Basis tadellos, da alle Werte hierfür mit passendem Geheimhaltungsgrad vorhanden sind. Je nachdem, ob *p* und *g* für alle Teilnehmer gleich sind oder nicht, ist im zweiten Fall gemäß b) zu verfahren. DSA (DSS) stellt also entgegen den erklärten Zielen auch eine Basis für kryptographische Konzelation, im Falle *p* und *g* für alle Teilnehmer gleich sogar für Steganographie mit öffentlichen Schlüsseln dar, vgl. §4.1.2.

Nicht einfach zu beantworten ist die Frage, ob der so realisierte Diffie-Hellman Schlüsselaustausch genauso sicher ist wie die Verfahren in §3.9.1 bzw. Aufgabenteil b), denn die Verteilung von *g* ist subtil anders und die Länge von *x* deutlich anders. Soweit mir bekannt, kann das Verfahren trotzdem als sicher betrachtet werden.

3-24 Blind geleistete Signaturen mit RSA

Der Text (inkl. des Ergebnisses der kollisionsresistenten Hashfunktion, vgl. §3.6.4.2), der blind signiert werden soll, sei 2, der Blendungsfaktor 17. Damit ergibt sich aufbauend auf Aufgabe 3-16 b) als geblendeter Text $2 \cdot 17^t \bmod 33 = 2 \cdot 17^3 = 2 \cdot 29 \bmod 33 = 58 \bmod 33 = 25$. (Beim Ausdenken solch eines Beispiels geht mensch natürlich umgekehrt vor: 25, die zu signierende Nachricht aus Aufgabe 3-16 b), ist vorgegeben. Also bestimme 2 Zahlen, deren Produkt (mod 33) 25 ergibt. Nach Wahl von 2 und 29 muß die 3-te Wurzel von 29 mod 33 bestimmt werden, wozu der inverse Exponent benutzt werden kann: $29^7 \bmod 33 = 29^2 \cdot 29^2 \cdot 29^2 \cdot 29 \bmod 33 = (-4)^2 \cdot (-4)^2 \cdot (-4)^2 \cdot (-4) \bmod 33 = 16 \cdot 16 \cdot 16 \cdot (-4) \bmod 33 = 256 \cdot (-64) \bmod 33 = 25 \cdot 2 \bmod 33 = 17$.)

Aus Aufgabe 3-16 b) kann nun der geblendete Text mit Signatur übernommen werden: 25, 31 (denn $31^3 \bmod 33 = (-2)^3 \bmod 33 = -8 \bmod 33 = 25 \bmod 33$).

Der Empfänger muß nun die zweite Komponente, die blind geleistete Signatur, durch den Blendungsfaktor 17 teilen.

Euklidscher Algorithmus zur Bestimmung von $17^{-1} \bmod 33$:

$33 = 1 \cdot 17 + 16$;

$17 = 1 \cdot 16 + 1$.

Rückwärts:

$1 = 17 - 1 \cdot 16$

$= 17 - 1 \cdot (33 - 1 \cdot 17)$

$= 2 \cdot 17 - 33$.

Also ist $17^{-1} \bmod 33 = 2$.

Als Signatur ergibt sich also:

$31 \cdot 17^{-1} \bmod 33 = 31 \cdot 2 \bmod 33 = 29$.

Nun überprüfen wir die Richtigkeit unserer Rechnung, indem wir nachrechnen, wie der Empfänger der blind geleisteten Signatur prüft, ob ihn der Aussteller übers Ohr hauen will:

$\text{Signatur}^t \bmod 33 = 29^3 \bmod 33 = (-4)^3 \bmod 33 = -64 \bmod 33 = 2 =?= \text{Text}$.

Wir stellen (erleichtert) fest, daß dies der Fall und unsere Rechnung deshalb höchstwahrscheinlich richtig ist.

3-25 Schwellwertschema

Die kleinstmögliche Primzahl p ist 17, denn die Zahlen zwischen 13 und 17 sind keine Primzahlen.

Das Polynom q(x) lautet $q(x) = 2x^2 + 10x + 13 \mod 17$.

Durch Einsetzen erhalten wir die Teile (i, q(i))

q(1) = 2+10+13 mod 17 = 25 mod 17 = 8
q(2) = 8+20+13 mod 17 = 41 mod 17 = 7
q(3) = 18+30+13 mod 17 = 61 mod 17 = 10
q(4) = 32+40+13 = 85 mod 17 = 0
q(5) = 50+50+13 = 113 mod 17 = 11

Nach der Formel in §3.9.6 ergibt sich:

q(x) =
8 (x–3)/(1–3) (x–5)/(1–5) + 10 (x–1)/(3–1) (x–5)/(3–5) + 11 (x–1)/(5–1) (x–3)/(5–3) mod 17 =
8/8 (x–3) (x–5) + 10/(-4) (x–1) (x–5) + 11/8 (x–1) (x–3) mod 17 =
(x–3) (x–5) + 10/(-4) (x–1) (x–5) + 11/8 (x–1) (x–3) mod 17 =
(x–3) (x–5) + 10•4 (x–1) (x–5) + 11•15 (x–1) (x–3) mod 17 =
(x–3) (x–5) + 6 (x–1) (x–5) + 12 (x–1) (x–3) mod 17 =
$2x^2 + 10x + 13$

q(0) hat dann natürlich den Wert des konstanten Terms, also hier 13.

Alternativ kann es günstiger sein, nicht erst allgemein das Polynom q(x) auszurechnen, sondern gleich den Wert 0 für x einzusetzen:

8 (–3)/(1–3) (–5)/(1–5) + 10 (–1)/(3–1) (–5)/(3–5) + 11 (–1)/(5–1) (–3)/(5–3) mod 17 =
120/8 + 50/(–4) + 33/8 mod 17 =
15 + 50 •4 + 33•15 mod 17 =
15 + (–1)•4 + (–1)•15 mod 17 =
13

3-26 Beweisbar sicherer Gebrauch mehrerer Konzelationssysteme?

Bei n zu verwendenden Konzelationssystemen wird der Klartext mittels eines informationstheoretisch sicheren Schwellwertschemas so in n Teile zerlegt, daß alle n nötig sind, um den Klartext zu rekonstruieren. Jedes der n Teile wird mit einem anderen der n Konzelationssysteme verschlüsselt.

Der Mehraufwand besteht neben der Verwendung des Schwellwertschemas darin, daß alle n Teile (und nicht nur das Ergebnis der Produktchiffre) zum Empfänger übermittelt werden müssen, so daß näherungsweise der n-fache Übertragungsaufwand entsteht.

Keine der beiden Kombinationen ist der anderen bzgl. Sicherheit überlegen:

- Zwar kann man bei der Produktchiffre über den Nutzen der Konzelationssysteme 2 bis n überhaupt nichts beweisen. Da die Zwischenergebnisse dem Angreifer aber nicht bekannt werden, ist eine Produktchiffre üblicherweise sehr, sehr viel sicherer als das beste verwendete Konzelationssystem, ja sogar sehr, sehr viel sicherer als die Summe aller verwendeten Konzelationssysteme. Letzteres bedeutet, daß der Aufwand, die Produktchiffre zu brechen, sehr, sehr viel höher ist als die Summe der Aufwände, die einzelnen Konzelationssysteme zu brechen.
- Damit der Beweis, daß die Kombination mindestens so schwierig zu brechen ist, wie das sicherste Konzelationssystem, durchgeht (und dann auch noch gleich liefert, daß die Kombination mindestens so sicher ist wie die Summe aller verwendeten Konzelationssysteme), braucht man die in der Aufgabenstellung genannten starken Voraussetzungen. Zwar dürften die üblicherweise gelten, aber jetzt haben wir genau wie bei der

Produktchiffre dieses „üblicherweise“. Deshalb überwiegen aus meiner Sicht die Vorteile der Produktchiffre nicht nur bzgl. Effizienz, sondern auch bzgl. Sicherheit deutlich.

Als besonders einfache Implementierung eines informationstheoretisch sicheren Schwellwertschemas, bei dem *alle n* Teile für die Rekonstruktion des Geheimnisses gebraucht werden, bietet sich die *n*-1-fache Anwendung der Vernam-Chiffre an: Die *n* Teile sind die *n*-1 Schlüsseln und der Schlüsseltext.

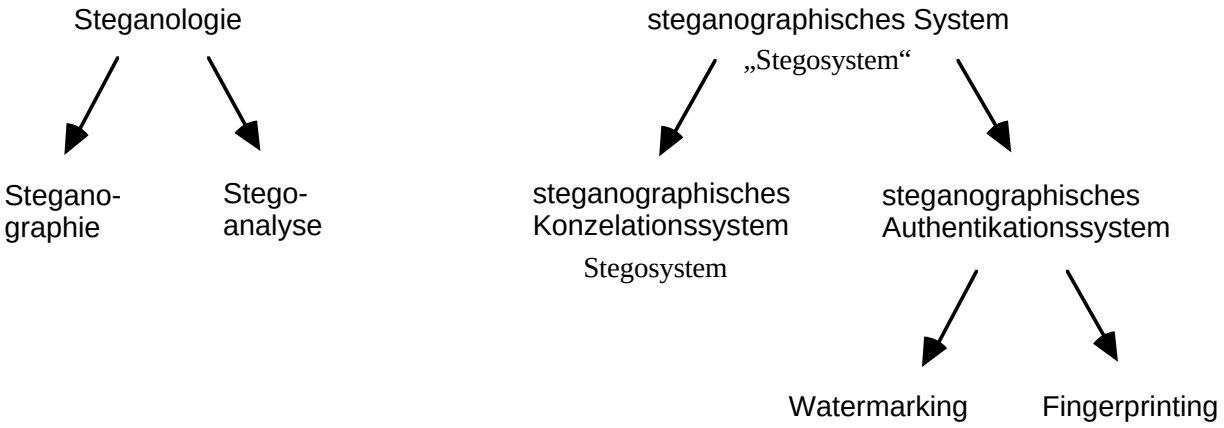
Lösungen zu Steganographische Grundlagen

4-1 Begriffe

Steganologie umfaßt sowohl Steganographie (die Wissenschaft von den Algorithmen der „gutartigen“ Benutzer) wie auch Stegoanalyse (die Wissenschaft von den Algorithmen der Angreifer auf steganographische Systeme). Kenntnisse im Bereich Stegoanalyse sind notwendig, um die Sicherheit steganographischer Systeme und ihrer Anwendung beurteilen zu können.

Steganographische Systeme sind entweder Konzelationssysteme (Schutzziel: Vertraulichkeit der Vertraulichkeit) oder Authentikationssysteme (Schutzziel: robuste Einbettung von Urheberinformation). Unterschieden wird danach, ob lediglich urheberbezogene Information eingebettet wird (Watermarking) oder auch nutzerbezogene (Fingerprinting).

Der Vorteil, den Begriff *Stegosystem* oben rechts im Begriffsbaum anzusiedeln, ist, einen kurzen üblichen Begriff allgemein verwenden zu können. Der Nachteil hiervon ist, daß die Unterscheidung zwischen Konzelation und Authentikation begrifflich in den Hintergrund rückt, so daß ich diesen Begriff nur für steganographische Konzelationssysteme benutze – allgemeiner verwende ich diesen Begriff nur mit Anführungszeichen. Mein Begriffsbaum sieht also folgendermaßen aus:



4-2 Eigenschaften der Ein- und Ausgaben kryptographischer und steganographischer systeme

a) Nachfolgend finden Sie Kopien der Bilder aus §3.1.1.1, die genau die Blockschaltbilder kryptographischer Konzelation sind.

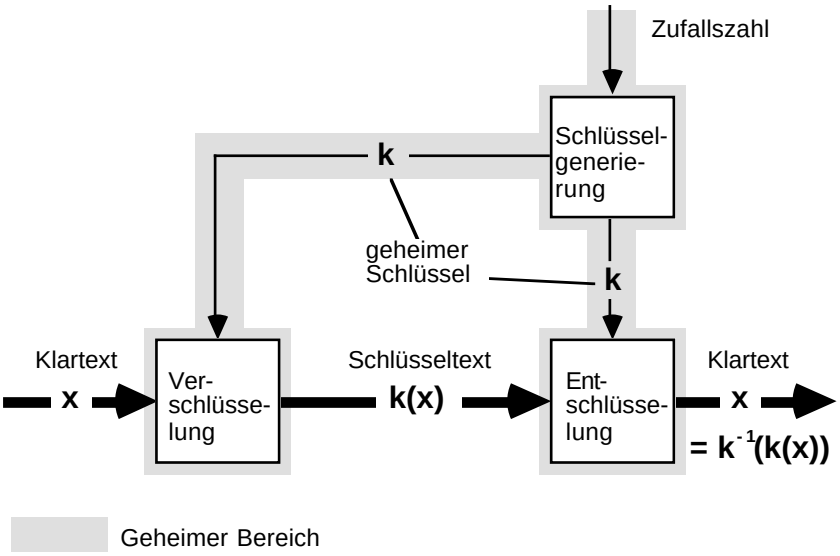


Bild 3-2: Symmetrisches kryptographisches Konzelationssystem

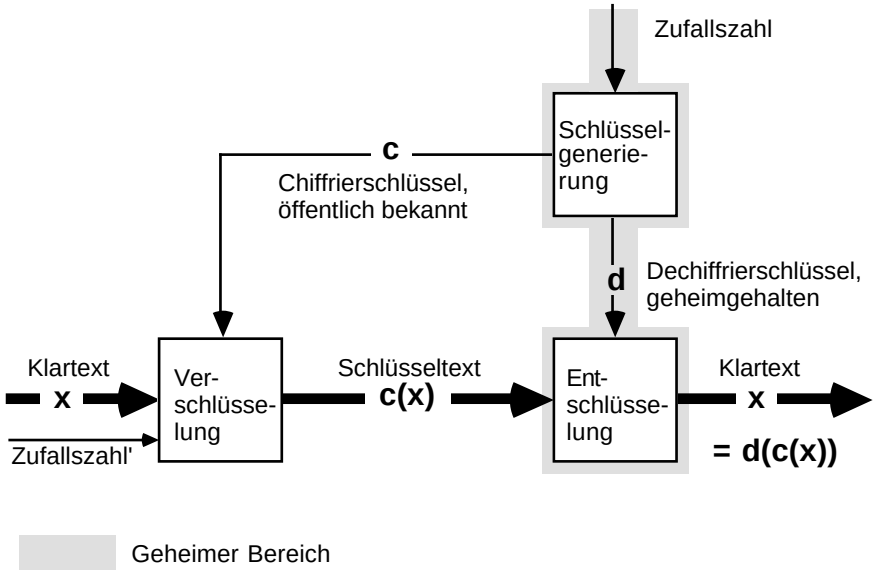


Bild 3-4: Asymmetrisches kryptographisches Konzelationssystem

Betrachtet man jeweils alle 3 Algorithmen als Gesamtsystem, unterscheiden sich die Ein- und Ausgaben der symmetrischen und asymmetrischen kryptographischen Konzelation nur darin, ob die Verschlüsselung als Eingabe auch eine Zufallszahl erhält. Allerdings variiert die Anforderung an den Kanal für die Schlüsselverteilung. Betrachtet man die Schnittstellen zwischen den 3 Algorithmen, dann besteht der Unterschied zwischen symmetrischer und asymmetrischer Kryptographie hauptsächlich darin, ob zum Ver- und Entschlüsseln der gleiche Schlüssel (der folglich geheim gehalten werden muß) verwendet wird oder unterschiedliche Schlüssel (so daß der Verschlüsselungsschlüssel veröffentlicht werden kann).

Ein kryptographisches System muß (und darf) die Annahme machen, daß die eingegebenen Zufallszahlen wirklich zufällig gewählt sind – ggf. sollte die Zufallszahlenerzeugung als Teil des Gerätes implementiert werden, das den jeweiligen Algorithmus ausführt. Ein gutes kryptographisches Konzelationssystem sollte keinerlei Annahmen über die Eigenschaften des Klartextes machen – denn der ist eine Eingabe (und damit Vorgabe) von außen. Damit ergibt sich bei Veröffentlichung des Verschlüsselungsschlüssels die Notwendigkeit, die Verschlüsselung indeterministisch zu machen, damit nicht „einfache“ Klartexte geraten und durch Ausprobieren der Verschlüsselung die Richtigkeit des Ratens überprüft werden kann. Auf der

anderen Seite brauchen kryptographische Konzelationssysteme keinerlei Vorgaben für die Eigenschaften der Schlüsseltexte zu erfüllen – vielleicht mit der Ausnahme, daß Schlüsseltexte nicht unnötig lang sein sollten.

b) Nachfolgend finden Sie eine Kopie des Bildes aus §4.1.1.1, das genau das Blockschaltbild steganographischer Konzelation gemäß Einbettungsmodell ist.

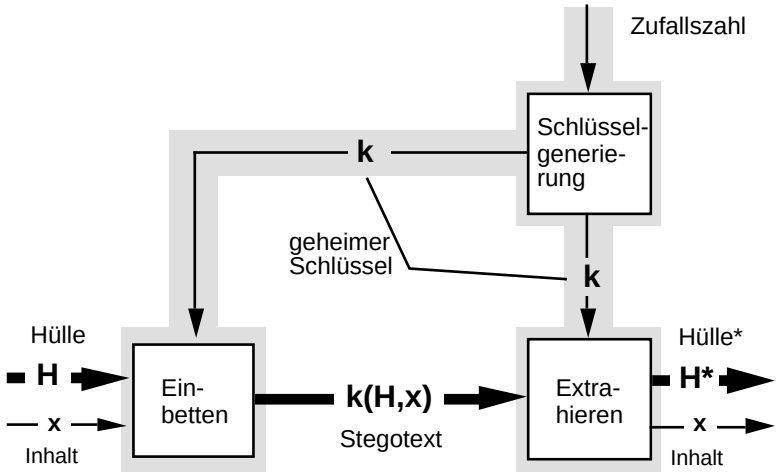


Bild 4-4: Steganographisches Konzelationssystem gemäß Einbettungsmodell

Da bisher keine asymmetrischen steganographischen Konzelationssysteme bekannt sind, gibt es bisher nur ein Blockschaltbild.

Im Idealfall bräuchte ein steganographisches Konzelationssystem keinerlei Annahmen über die Eigenschaften der verwendeten Hülle zu machen, denn diese ist beim Einbettungsmodell eine Eingabe (und damit Vorgabe) von außen. (Würden wir die Hülle intern durch das Stegosystem generieren – synthetische steganographische Konzelation – müßten wir uns als Steganographen Gedanken über die Plausibilität der generierten Hülle im jeweiligen Anwendungskontext machen.) Leider wird kein Stegosystem mit beliebigen Hüllen arbeiten können, gewisse Annahmen sind also unumgänglich und damit jedes steganographische Konzelationssystem abhängig von der Einhaltung genau dieser Annahmen. (In gewissem Umfang kann und sollte das steganographische System die Einhaltung dieser Annahmen prüfen und bei Verletzung der Annahmen nicht einbetten.)

Für alle anderen Eingaben gilt das Gleiche wie bei Kryptographie. Für die Ausgabe *Stegotext* gilt die Forderung, daß er genauso plausibel wie die eingegebene *Hülle* sein soll.

c) Bei kryptographischen Konzelationssystemen gibt es keine Annahmen, deren Erfüllung nicht durch die Implementierung sichergestellt werden kann. Bei steganographischer Konzelation (nach dem *Einbettungsmodell* im Gegensatz zum *Synthesemodell*) ist dies anders: Die Richtigkeit der Annahmen über Hülle kann durch die Implementierung nicht sichergestellt werden.

4-3 Macht gute Steganographie Verschlüsselung überflüssig?

Wenn ein Angreifer nicht merken kann, ob (und damit auch nicht daß) eine geheime Nachricht eingebettet ist, dann kann er sie auch nicht verstehen.

Einerseits würde ich in der Praxis zusätzlich verschlüsseln, damit wenigstens der Inhalt der Nachricht vertraulich bleibt, falls das steganographische Konzelationssystem doch nicht so gut ist wie gehofft. Mindestens so wichtig ist aber, daß durch gute Verschlüsselung der einzubettende Inhalt von weißem Rauschen nicht zu unterscheiden ist. Dies erlaubt dem steganographischen Konzelationssystem, genauer: seiner Einbettungsfunktion, sehr starke (und trotzdem realistische) Annahmen über die Eingabe *Inhalt*.

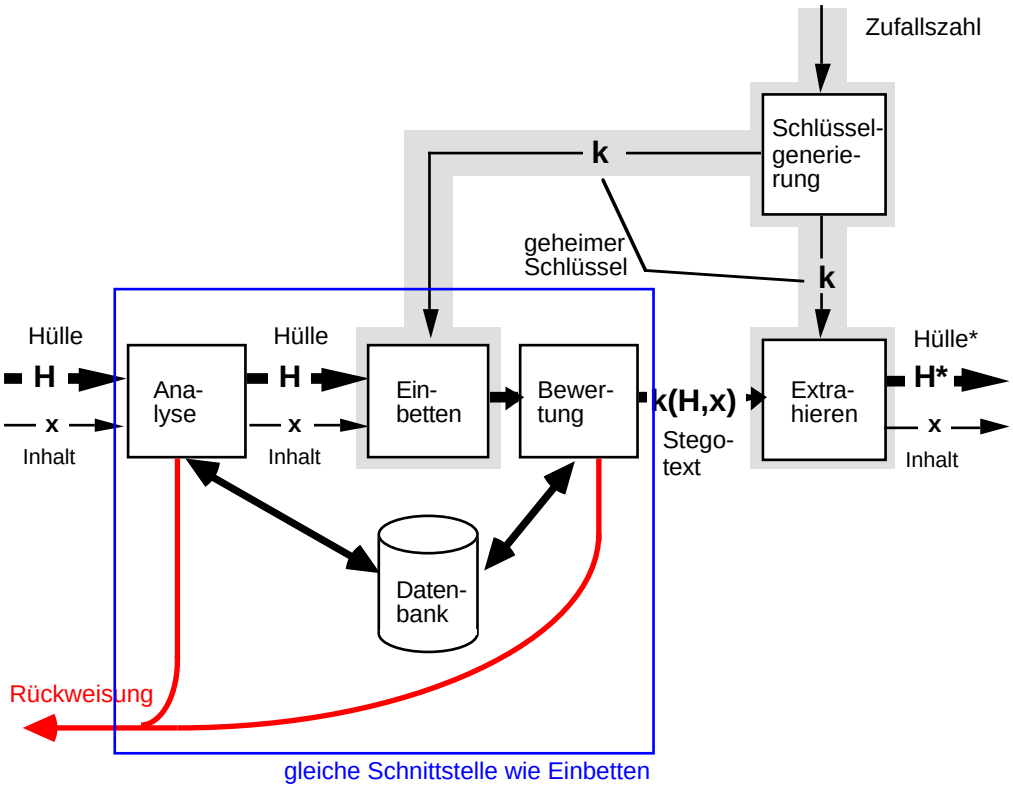
4-4 Beispiele steganographischer Systeme

1. Aus meiner Sammlung von ca. 90 aktuellen Urlaubsphotos wähle ich das Photo aus, was „zufälligerweise“ die 4 Bits enthält, die ich geheim übertragen will. Die Sicherheit dürfte sehr gut sein – wenn es ein solches Photo gibt, was ziemlich wahrscheinlich ist.
2. Das Urlaubsphoto, das ich meinem Freund schicken will, digitalisiere ich auf dem Scanner so oft, bis es genau die 4 Bits enthält, die ich übertragen will. Die Sicherheit dürfte sehr gut sein.
3. Um 40 Bits zu übertragen, sende ich 10 Urlaubsphotos, die ich wie in 1. passend ausgewählt habe (klappt immer noch mit guter Wahrscheinlichkeit). Hin und wieder muß ich halt ein Photo doppelt schicken. Die Sicherheit dürfte nur akzeptabel gut sein, wenn der Sender als ziemlich zerstreut gilt. Es handelt sich also um ein Stegosystem vorzugsweise für zerstreute Professoren.
4. Um 40 Bits zu übertragen, sende ich meinem Freund die 10 schönsten Urlaubsphotos in chronologischer Reihenfolge ihrer Aufnahme, die ich wie in 2. passend digitalisiert habe. Die Sicherheit dürfte sehr gut sein, die notwendige Scannsession etwas Ausdauer erfordern.

4-4a Modellierung steganographischer Systeme

Modellbildung ist mehr eine Kunst als ein deterministischer Prozeß – es gibt also eher brauchbare (im Gegensatz zu weniger brauchbaren) Modellen als richtige (und falsche) Modelle. Ein schlauer Mensch hat einmal gesagt: Eine Maschine ist nicht dann zu Ende konstruiert, wenn man nichts mehr hinzufügen kann, sondern dann, wenn man nichts mehr weglassen kann. Dies ist sicherlich eine gute Maxime für die Kunst der Modellbildung.

Für die Modellierung steganographischer Systeme bedeutet dies: Anja, Berta und Caecilie machen sehr richtige und wertvolle Bemerkungen über die *Implementierung mancher* steganographischer Systeme. Denn wenn etwa klar ist, daß sich Hüllen nie exakt wiederholen, dann brauche ich keine Datenbank, etc. Da also einerseits Analyse, Datenbank und Bewertung nicht Teil *aller* steganographischer Systeme sein müssen und andererseits sie ohne Probleme als Implementierung des in Bild 4-4 gezeigten Modells aufgefaßt werden können (vgl. nachfolgendes Bild), gebe ich Dörte recht. Aber auch nicht ganz: Denn neu ist die Erkenntnis von Anja, daß zumindest bei manchen steganographischen Systemen ein Fehlerausgang Rückweisung vorgesehen werden muß. Ob dies im Modell explizit getan werden muß oder ob sich das sowieso von selbst versteht, weil jede Operation fehlschlagen kann und dann im umfassenden System darauf reagiert werden muß, also alle Operationen Fehlerausgänge benötigen, darüber kann eine Herrenrunde bei Gelegenheit diskutieren. Die Erkenntnis der Damenrunde ist: Im Modell steganographischer Systeme ist ein Fehlerausgang vorzusehen.



4-5 Steganographische Konzelation mit öffentlichen Schlüsseln?

Da Einbettung und Extraktion in keiner Weise mittels unterschiedlicher Information erfolgen, würde ich dies nicht Steganographie mit öffentlichen Schlüsseln nennen. Aus meiner Sicht handelt es sich um einen *Bootstrap von einen Stegoschlüssel verwendender symmetrischer Steganographie mittels keinen Schlüssel verwendender schlüsselloser Steganographie*. Dies ist in vielen Situationen sinnvoll, etwa wenn die schlüssellose Steganographie eine geringere Nutzbandbreite zur Verfügung stellt, mehr Berechnungsaufwand erfordert oder unsicherer ist.

4-6 Steigerung der Sicherheit durch Gebrauch von mehreren steganographischen Systemen

Mehrere steganographische Konzelationssysteme zu kombinieren macht wenig Sinn, da die Existenz der vertraulichen Nachricht auch bei Anwendung in Serie (vgl. Aufgabe 3-3) nur so gut geschützt wird wie das äußerste steganographische Konzelationssystem gut ist. Außerdem würde ein enormer Datenumfang entstehen.

Bzgl. steganographischen Authentikationssystemen ist die Antwort schwieriger: Ein paar wenige Systeme in Serie anzuwenden dürfte sinnvoll sein, da sie so alle gebrochen werden müssen. Andererseits bedeutet die Anwendung jedes Systems eine – wenn auch minimale – zusätzliche Störung des Werkes, weswegen nicht zu viele verwendet werden sollten.

4-7 Krypto- und steganographische Konzelation kombiniert?

Eine Kombination ist sinnvoll, da so wenigstens die Vertraulichkeit des Nachrichteninhalts mindestens so gut geschützt werden kann, wie das schwächere der beiden Verfahren (sofern Stego- und Kryptoschlüssel unabhängig voneinander gewählt werden!). Damit die Existenz der vertraulichen Nachricht nicht schlechter geschützt wird als bei Verwendung allein des Stegosystems, wird sie zuerst mit dem kryptographischen Konzelationssystem (oder nach Belieben auch mit mehreren kryptographischen Konzelationssystemen, vgl. Aufgabe 3-3) verschlüsselt und danach der Schlüsseltext steganographisch eingebettet. Der Empfänger muß natürlich zuerst extrahieren und danach entschlüsseln.

4-8 Krypto- und steganographische Authentikation in welcher Reihenfolge?

Erst steganographische Authentikation (also Watermarking oder Fingerprinting), danach das steganographisch veränderte Werk digital signieren. (Bei umgekehrter Reihenfolge macht die steganographische Authentiktion die digitale Signatur ungültig.)

4-9 Stego-Kapazität von komprimierten und nichtkomprimierten Signalen

Bei verlustfreier Kompression bleibt die gesamte Entropie des Signals (und damit seine absolute Stego-Kapazität) erhalten. Das Signal wird, wenn es Redundanz enthielt, lediglich kürzer codiert. Folglich steigt seine prozentuale Stego-Kapazität bei Anwendung von Utilities wie Gzip, StuffIt etc.

Bei verlustbehafteter Kompression bleibt nicht die gesamte Entropie des Signals erhalten, die absolute Stego-Kapazität sinkt also. Wie es sich mit der prozentualen Stego-Kapazität verhält, hängt vom Verhältnis Redundanzreduktion zu Entropieverlust ab.

Lösungen zu Sicherheit in Kommunikationsnetzen

5-0 Quantifizierung von Unbeobachtbarkeit, Anonymität und Unverkettbarkeit

a) Die einfachste Kenngröße ist die *Anzahl* der Ereignisse oder Instanzen, die für den Angreifer ununterscheidbar sind. Allerdings sagt diese Kenngröße allein nicht sehr viel aus, wenn es für ihn sehr viele sehr sehr unwahrscheinliche und ein oder ganz wenige sehr wahrscheinliche Ereignisse oder Instanzen gibt. Deshalb dürfte es sinnvoll sein, die Anzahl mit der Wahrscheinlichkeitsverteilung zu wichten, wie dies in der Informationstheorie von Claude Shannon definiert wurde [KlPS_96]: Dort ist die Entropie einer Informationsquelle in bit die Summe über alle möglichen Ereignisse x_i, wobei jeweils das Produkt aus Auftretenswahrscheinlichkeit des Ereignisses $W(x_i)$ und Logarithmus zur Basis 2 des Kehrwertes der Auftretenswahrscheinlichkeit des Ereignisses $W(x_i)$ aufsummiert wird:

$$\sum_i W(x_i)\ \mathrm{ld}\ \frac{1}{W(x_i)}$$

Bei Gleichverteilung der Auftretenswahrscheinlichkeiten der Ereignisse ist der Kehrwert der Auftretenswahrscheinlichkeit gerade die Anzahl der Ereignisse. Der Logarithmus wird gebildet, damit Entropien additiv sind und die Basis 2 für den Logarithmus wird gewählt, damit als Einheit bit, also eine Binärentscheidung, herauskommt.

Entsprechend der Entropie einer Informationsquelle kann die *Entropie* von Unbeobachtbarkeit, Anonymität und Unverkettbarkeit definiert werden.

b) Sinnvolle Abstufungen könnten sein, vgl. [ReRu_98, ReRu_99]:
   - *Perfekt*: Der Angreifer erfährt/weiß *nichts*. Teilnehmer sind damit perfekt geschützt.
   - *Unverdächtig*: Der Angreifer erfährt/weiß *fast nichts*. Beispielsweise können es bzgl. Anonymität alle Teilnehmer, die nicht mit dem Angreifer kooperieren, mit nahezu unveränderter/mit jeweils nahezu gleicher Wahrscheinlichkeit sein. Damit bleiben alle unverdächtig.
   - *Wahrscheinlich nicht*: Der Angreifer erfährt/weiß *nicht sehr viel*: Beispielsweise ist bzgl. Anonymität für jeden die Wahrscheinlichkeit, daß er es nicht war, größer als daß er es war.
   - *Möglicherweise nicht*: Der Angreifer erfährt/weiß *fast alles*: Beispielsweise ist bzgl. Anonymität für einen die Wahrscheinlichkeit, daß er es war, größer als 0,5 aber nichtvernachlässigbar kleiner als 1.
   - *Identifiziert*: Der Angreifer erfährt/weiß *alles*: Beispielsweise ist bzgl. Anonymität für einen die Wahrscheinlichkeit, daß er es war, 1.

5-1 Ende-zu-Ende- oder Verbindungs-Verschlüsselung

Streng nach §5.2.1 lautet die Antwort: Da die Konzernleitung das Fernmeldenetz nicht betreibt und deshalb nicht ändern kann, kann sie lediglich Ende-zu-Ende-Verschlüsselung anwenden, wodurch die Vermittlungsdaten natürlich vor niemand geschützt werden.

Gibt der Konzern sich mit diesem Defizit nicht zufrieden und beginnt er, auf dem realen Fernmeldenetz ein eigenes, virtuelles Fernmeldenetz (oft virtuelles privates Netz, VPN genannt) zu implementieren, so sind die folgenden Überlegungen angebracht:

Will sich die Konzernleitung mittels Verschlüsselung „nur“ gegen das Abhören in ihrem Fernmeldenetz schützen und nicht auch gegen das Unterlaufen des Zugriffsschutzes ihrer Rechner, so genügt *virtuelle Verbindungs-Verschlüsselung.* Bei ihr werden alle Daten einer ggf. variabel konfigurierbaren Verbindung (Fernmeldenetz schaltet Standleitungen, virtuelle Verbindungen, vermittelt Kanäle oder Pakete, etc.) verschlüsselt [Pfit_89].

Sind bei Standleitungen die Kosten unabhängig vom übertragenen Datenvolumen, sollte Verbindungs-Verschlüsselung mittels eines (Pseudo-) One-time-pads verwendet werden. Auch wenn keine Nutzinformation zu senden ist, werden Leerzeichen, Nullen o.ä. verschlüsselt und der resultierende Schlüsseltextstrom übertragen. Dann kann bei sicherer Verschlüsselung im Fernmeldenetz nicht einmal beobachtet werden, ob, geschweige denn welche, Daten zwischen den Geschäftsstellen übertragen werden.

Sind die Kosten abhängig vom übertragenen Datenvolumen, so ist zu unterscheiden, ob die Mehrkosten für obige Maßnahme durch das Schutzziel „Unbeobachtbarkeit der Kommunikation zwischen Geschäftsstellen“ gerechtfertigt ist. Wenn nein, sollte nur verschlüsselt und übertragen werden, wenn Nutzinformation zu übertragen ist. Dies ist dann die zu Beginn erwähnte „Ende-zu-Ende-Verschlüsselung“, wobei lediglich die „End(e)“punkte weiter vom eigentlichen Benutzer entfernt sind (beispielsweise vor den Modems bzw. Netzabschlüssen), als dies für Ende-zu-Ende-Verschlüsselung anzustreben wäre.

5-2 Erst verschlüsseln und dann fehlertolerant codieren oder umgekehrt?

Es sollte erst verschlüsselt und danach fehlertolerant codiert werden, da anderenfalls die Fehlercharakteristik des Kommunikationskanals durch die Verschlüsselung für die darüber liegende fehlertolerante Codierung geändert wird.

Selbst Einzelbitfehler des Kommunikationskanals werden bei manchen Verschlüsselungsverfahren viele oder gar alle folgenden Bits mit der Wahrscheinlichkeit 1/2 verfälschen: bei Blockchiffren im ECB-Mode etwa alle Bits des betroffenen Blocks, beim PCBC-Mode gar alle Bits des betroffenen sowie aller folgenden Blöcke. Beides bewirkt, daß sinnvolle Annahmen über das dumme Fehlerverhalten des Kommunikationskanals für die fehlertolerante Codierung nicht mehr stimmen müssen, so daß sie im allgemeinen nicht mehr gut angepaßt sein wird.

Ist Ver- und Entschlüsseln viel aufwendiger als fehlertolerant Codieren und Decodieren, so spart erst verschlüsseln und danach fehlertolerant codieren nicht nur Aufwand bezüglich der *Informationsübertragung,* sondern auch Aufwand bezüglich der *Informationsverarbeitung*: Bei Fehlern wird normalerweise nicht vergeblich entschlüsselt. Zusätzlich wird selbst dann Aufwand gespart, wenn gar keine Fehler auftreten: Jede fehlertolerante Codierung muß Redundanz hinzufügen und damit die Nachricht verlängern. Bei erst fehlertolerant Codieren und dann Verschlüsseln müßte das aufwendige Ver- und Entschlüsseln mit der verlängerten Nachricht erfolgen.

Last but not least kann das Verschlüsseln redundanter Nachrichten die Sicherheit eines nur komplexitätstheoretisch sicheren Konzelationssystems entscheidend schwächen. Denn genau die Redundanz hilft einem Angreifer ja zu erkennen, wenn ein Entschlüsselungsversuch erfolgreich war.

5-3 Behandlung von Adreß- und Fehlererkennungsfeldern bei Verschlüsselung

a) Die Adresse bleibt bei Ende-zu-Ende-Verschlüsselung im Klartext erhalten. Als *verschlüsselte Nutznachricht* ergibt sich

00110110 (Klartext-Nutznachricht)
⊕ 01100111 (verwendetes One-time-pad)
01010001 (verschlüsselte Nutznachricht)

Als Prüfzeichen ergibt sich 01 + 11 + 01 + 01 + 00 + 01 = 11. Also lautet die Ende-zu-Ende-verschlüsselte Nachricht 0111 01010001 11.

Anmerkungen:
Das Prüfzeichen darf man natürlich nicht von der Adresse und Klartext-Nutznachricht bilden: Erstens würde dies einem Angreifer manche Information über die Klartext-Nutznachricht verraten, da das Prüfzeichen im Klartext übertragen wird und der Algorithmus zu seiner Bildung öffentlich bekannt ist. Zweitens würde die Bildung des Prüfzeichens von der Adresse und Klartext-Nutznachricht die Fehlererkennung im Kommunikationsnetz genauso unmöglich machen wie dies das Mitverschlüsseln des Prüfzeichens täte.

Braucht man keine Fehlererkennung im Kommunikationsnetz, sondern nur Ende-zu-Ende, dann kann das Prüfzeichen mitverschlüsselt werden, so daß die obigen Probleme wegfallen. Ist man dazu bereit, was bei One-time-pad-Verschlüsselung natürlich etwas von der knappen Ressource Schlüssel kostet, dann sollte man eigentlich noch einen Schritt weiter gehen und einen Authentikationscode verwenden, so daß nicht nur dumme Fehler, sondern auch intelligente Angreifer Nachrichten nur mit sehr geringer Wahrscheinlichkeit unentdeckt verfälschen oder unterschlagen können.

Hätte man gerne Fehlererkennung im Kommunikationsnetz und Ende-zu-Ende-Authentikation, muß im gegebenen Nachrichtenformat die Authentikation als Teil der Nutznachricht erfolgen. Das Prüfzeichen wird dann wie zu Beginn beschrieben von der Adresse und der sowohl verschlüsselten als auch authentizierten Nutznachricht gebildet. Natürlich kann man jetzt auch zwei Felder für Prüfzeichen einführen, diskutieren, ob die Authentikation die Adresse mit einschließen soll, ob es gar zwei, möglicherweise unterschiedliche Adressen gibt, etc. Wir sehen also, daß beim Einsatz von Verschlüsselung in Kommunikationsnetzen mit mehreren Protokollschichten viele Möglichkeiten bestehen, von denen längst nicht alle sinnvoll sind.

b) Die *gesamte* Nachricht – allerdings ohne Prüfzeichen, vgl. Aufgabe 5-2 – wird verschlüsselt:

0111 00110110 (Klartextnachricht ohne Prüfzeichen)
⊕ 0011 10100111 (verwendetes One-time-pad)
0100 10010001 (Verbindungs-verschlüsselte Nachricht ohne Prüfzeichen)

Als Prüfzeichen ergibt sich 01 + 00 + 10 + 01 + 00 + 01 = 01. Also lautet die Verbindungs-verschlüsselte Nachricht 0100 10010001 01.

(Wie in Teil a) darf man das Prüfzeichen nicht von der Adresse und Klartext-Nutznachricht bilden. Aus den in der vorherigen Aufgabe erläuterten Gründen ist es auch nicht sinnvoll, die gesamte Nachricht mit Prüfzeichen zu verschlüsseln.)

c) Die Ende-zu-Ende-verschlüsselte Nachricht (ohne Prüfzeichen) 0111 01010001 wird Verbindungs-verschlüsselt:

0111 01010001 (Ende-zu-Ende-verschlüsselte Nachricht ohne Prüfzeichen)
⊕ 0011 10100111 (für Verbindungs-Verschlüsselung verwendetes One-time-pad)
0100 11110110 (Ende-zu-Ende- und Verbindungs-verschlüsselte Nachricht ohne Prüfz.)

Als Prüfzeichen ergibt sich 01 + 00 + 11 + 11 + 01 + 10 = 10. Also lautet die Ende-zu-Ende- und Verbindungs-verschlüsselte Nachricht 0100 11110110 10.

5-4 Verschlüsselung bei verbindungsorientierten und verbindungslosen Kommunikationsdiensten

Bei *verbindungsorientierten* Kommunikationsdiensten können beliebige Konzelationssysteme in beliebigen Betriebsarten verwendet werden.

Bei *verbindungslosen* Kommunikationsdiensten müssen die Übertragungseinheiten (z.B. Nachrichten, Pakete, Frames) unabhängig voneinander entschlüsselt werden können. Also darf die (Ver- und) Entschlüsselung nicht von der Vorgeschichte, d.h. früheren Übertragungseinheiten abhängen. Also dürfen keine synchronen Stromchiffren, vgl. §3.8.1, und selbstsynchronisierende Stromchiffren auch nur, wenn auf die Übertragungseinheiten abgestimmt, verwendet werden.

5-4a Abfragen und Überlagern

a) Gegeben seien die 6 Nachrichten

Nachricht 1: 00111010 01111100
Nachricht 2: 10110111 11100001
Nachricht 3: 01110000 00111111
Nachricht 4: 11100111 01011110
Nachricht 5: 01101011 11001000
Nachricht 6: 11000101 10101000

die in den 7 Servern S1, S2, S3, S4, S5, S6 und S7 in gleicher Reihenfolge vorliegen. Der Teilnehmer entschließe sich, 4 Server an seiner Abfrage zu beteiligen. Er bildet folglich 3 Abfragevektoren von je 6 Bit Länge zufällig:

?w = 111011
?x = 011011
?y = 101110

berechnet deren Summe modulo 2 als

z' = 001110

Der Teilnehmer interessiere sich für Nachricht 2, er invertiert also das 2. Bit:

?z = 011110

Der Teilnehmer sendet die 4 Abfragevektoren beispielsweise an die Server S1, S2, S5 und S6, konkret ?w an S1 und erhält !w

Nachricht 1: 00111010 01111100
Nachricht 2: 10110111 11100001
Nachricht 3: 01110000 00111111
Nachricht 5: 01101011 11001000
⊕ Nachricht 6: 11000101 10101000
!w = 01010011 11000010

?x an S2 und erhält !x

Nachricht 2: 10110111 11100001
Nachricht 3: 01110000 00111111
Nachricht 5: 01101011 11001000
⊕ Nachricht 6: 11000101 10101000
!x = 01101001 10111110

?y an S5 und erhält !y

Nachricht 1: 00111010 01111100
Nachricht 3: 01110000 00111111
Nachricht 4: 11100111 01011110
⊕ Nachricht 5: 01101011 11001000
!y = 11000110 11010101

?z an S6 und erhält !z
Nachricht 2: 10110111 11100001
Nachricht 3: 01110000 00111111
Nachricht 4: 11100111 01011110
⊕ Nachricht 5: 01101011 11001000
!z = 01001011 01001000

Der Teilnehmer rechnet
!w = 01010011 11000010
!x = 01101001 10111110
!y = 11000110 11010101
⊕ !z = 01001011 01001000
Nachricht 2: 10110111 11100001

b) Zwischen Teilnehmer und Server muß jeweils verschlüsselt werden, da sonst ein Angreifer, der alle diese Nachrichten abhört, sie nur modulo 2 addieren müßte, um als Ergebnis eine 1 genau an der Stelle in der Summe der Abfragevektoren zu erhalten, die der Speicherzelle entspricht, für deren Inhalt sich der Teilnehmer interessiert. In obigem Beispiel würde der Angreifer

1001
1100
⊕ 0111
0010

rechnen und frohlocken: 1 an Stelle 3, siehe da, der Teilnehmer interessiert sich also für Nachricht 3.

Auch die Antwort der Server an den Teilnehmer muß verschlüsselt werden. Andernfalls addiert ein Angreifer, der alle diese Nachrichten abhört, sie genau wie der Teilnehmer und erhält genau wie der die Nachricht, für die sich der Teilnehmer interessiert.

c) Um die nötige Übertragungsbandbreite vom Teilnehmer zu den Servern zu minimieren, werden statt s-1 lange Abfragevektoren zufällig zu erzeugen, diese pseudozufällig erzeugt. Dann brauchen an diese s-1 Server nicht lange Abfragevektoren, sondern nur kurze Parameter für Pseudozufallsgeneratoren verschlüsselt übertragen zu werden. Der s-te Abfragevektor wird wie üblich berechnet und übertragen. Leider sehe ich nicht, wie auch er pseudozufällig generiert werden könnte. Der Preis für das Einsparen von Übertragungsbandbreite ist, daß nun keine informationstheoretische, sondern nur noch komplexitätstheoretische Sicherheit möglich ist. Der in §5.4.2 angegebene Beweis für die Sicherheit des Verfahrens „Abfragen und Überlagern“ gilt weiterhin, da er nur die Symmetrie der s Abfragevektoren bzgl. aller algebraischen Gesetze verwendet, nicht aber, daß die Server nicht wissen, wer welchen wie generierten Abfragevektor erhält.

Teilnehmer sendet statt s langen Abfragevektoren einen langen und s-1 kurze.

Um die nötige Übertragungsbandbreite von den Servern zum Teilnehmer zu minimieren, wird die Verschlüsselung zwischen Server und Teilnehmer so gewählt, daß die für den Teilnehmer

verschlüsselten Antworten der Server modulo 2 addiert werden können und der Teilnehmer trotzdem danach die Summer der Klartexte erhalten kann. Dies ist möglich, wenn die Verschlüsselung zwischen Server und Teilnehmer mit einer „Vernam-Chiffre“ (one-time pad), vgl. §3.2, oder einem Pseudo-one-time-pad, vgl. §3.4.2, modulo 2 erfolgt. Dann braucht der Teilnehmer nur auf die modulo-2-Summe der verschlüsselten Nachrichten alle (pseudo-)one-time pads modulo 2 zu addieren, um die modulo-2-Summe aller Klartextnachrichten zu erhalten. Der Preis für das Einsparen von Übertragungsbandbreite ist lediglich, daß nur eine bestimmte Klasse von Konzelationssystemen zwischen Servern und Teilnehmer verwendet werden kann.

Teilnehmer empfängt statt *s* Antworten nur eine.

Wer die Summe der verschlüsselten Antworten der Server bildet, ist für den Schutz der Information, welche Speicherzelle der Teilnehmer abfragt, unerheblich. Beispielsweise könnte also einer der Server die Antworten seiner Kollegen sammeln, zusammen mit seiner modulo 2 addieren und an den Teilnehmer senden. Dies eröffnet die Möglichkeit, die oben erwähnten *s*-1 kurzen Abfragenachrichten einzusparen: Der Teilnehmer habe mit allen Servern, die er jemals zu benutzen gedenkt, jeweils alle notwendigen Parameter für einen individuellen Pseudozufallsgenerator und Schlüssel für die Verschlüsselung ihrer Antworten ausgetauscht. Dann sendet er nur noch den langen Abfragevektor (und, sofern es nicht fest vereinbart ist, eine Liste der zu verwendenden Server) an einen der Server. Dieser bildet seine verschlüsselte Antwort, und sendet sie (ggf. mit der Liste) und dem Namen des Teilnehmers an den nächsten Server. Der bildet seine verschlüsselte Antwort, indem er den Pseudzufallsgenerator mit diesem Teilnehmer den Abfragevektor generieren läßt, entsprechend die Speicherzellen modulo 2 addiert, das Ergebnis verschlüsselt und danach zur erhaltenen Nachricht modulo 2 addiert, und an den nächsten Server der Liste schickt. Der letzte Server der Liste schickt das Ergebnis an den Teilnehmer.

Teilnehmer sendet und empfängt statt *s* Nachrichten jeweils nur eine.

d) Der Vorschlag ist unsinnig, d.h. er stellt eine Verschlechterung gegen die normale Stärkung des Verfahrens durch Verwendung eines zusätzlichen Servers dar, der nicht ignoriert wird. Begründung: Der Vorschlag verkraftet nicht einmal einen Angriff von beliebigen *s* Servern, was das normale Verfahren bei *s*+1 verwendeten Servern ja tut: Nehmen wir an, die angreifenden *s* Server seien die, deren Antworten vom Teilnehmer wirklich verwendet werden, d.h. modulo 2 addiert den Inhalt einer der Speicherzellen ergeben. Da üblicherweise bei weitem nicht jeder mögliche Inhalt einer Speicherzelle auch tatsächlich vorkommt, können die *s* Server bequem testen, ob sie den ignorierten Server unter sich haben: Ergibt die Summe modulo 2 ihrer Antworten den Inhalt einer der Speicherzellen, so ist dies mit überwältigender Wahrscheinlichkeit nicht der Fall und die Summe genau die Nachricht, für die sich der Teilnehmer interessiert.

5-4b Vergleich „Verteilung“ und „Abfragen und Überlagern“

a) Eine offene implizite Adresse bei „Verteilung“ sollte einer Speicherzelle bei „Abfragen und Überlagern“ entsprechen. In diese Speicherzelle wird immer die letzte an diese offene implizite Adresse gesendete Nachricht geschrieben – die Inhalte von Speicherzellen des Nachrichten-Service sind damit nicht mehr WORM- (Write once, Read many), sondern wirklich normale RAM-Speicherzellen. Wenn dafür Sorge getragen wird, daß alle Server weiterhin zu jedem Zeitpunkt die gleichen Inhalte in ihren Speicherzellen haben, daß also zeitliche Änderungen synchron erfolgen, dann funktioniert der Nachrichten-Service auch so – und er ist weitaus flexibler und effizienter einsetzbar.

Teilnehmer können ihre offenen impliziten Adressen effizient abfragen, indem sie in einer Abfrage die Bits mehrerer oder gar aller Speicherzellen, die ihre impliziten Adressen sind, invertieren. Sie erhalten dann die aktuelle Summe modulo 2 der Inhalte dieser Speicherzellen. Durch Differenzbildung mit der vorherigen Abfrage der gleichen Speicherzellen kann die Summe aller neuen Nachrichten gebildet werden. Ist dies mehr als eine, so werden solange Teilsummen abgefragt, bis durch Differenzbildung alle neuen Nachrichten ermittelt sind.

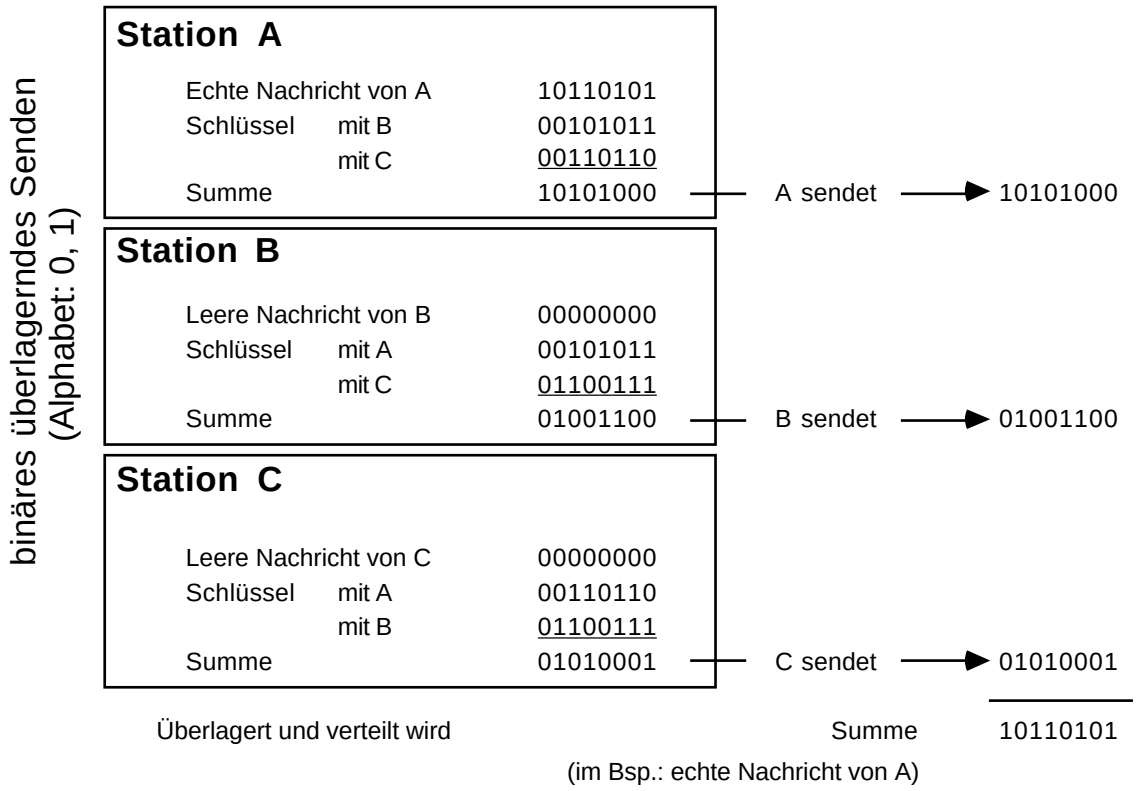
Hat ein Teilnehmer einen festen Satz offener impliziter Adressen, dann kann es effizient sein, daß er mit den Servern vereinbart, daß sie eine Abfrage nach diesen Speicherzellen in zyklischen Abständen immer wieder beantworten sollen. Da hierzu keine neuen Abfragevektoren generiert werden müssen, sondern die alten immer wieder verwendet werden können, spart dies Bandbreite vom Teilnehmer zu den Servern.

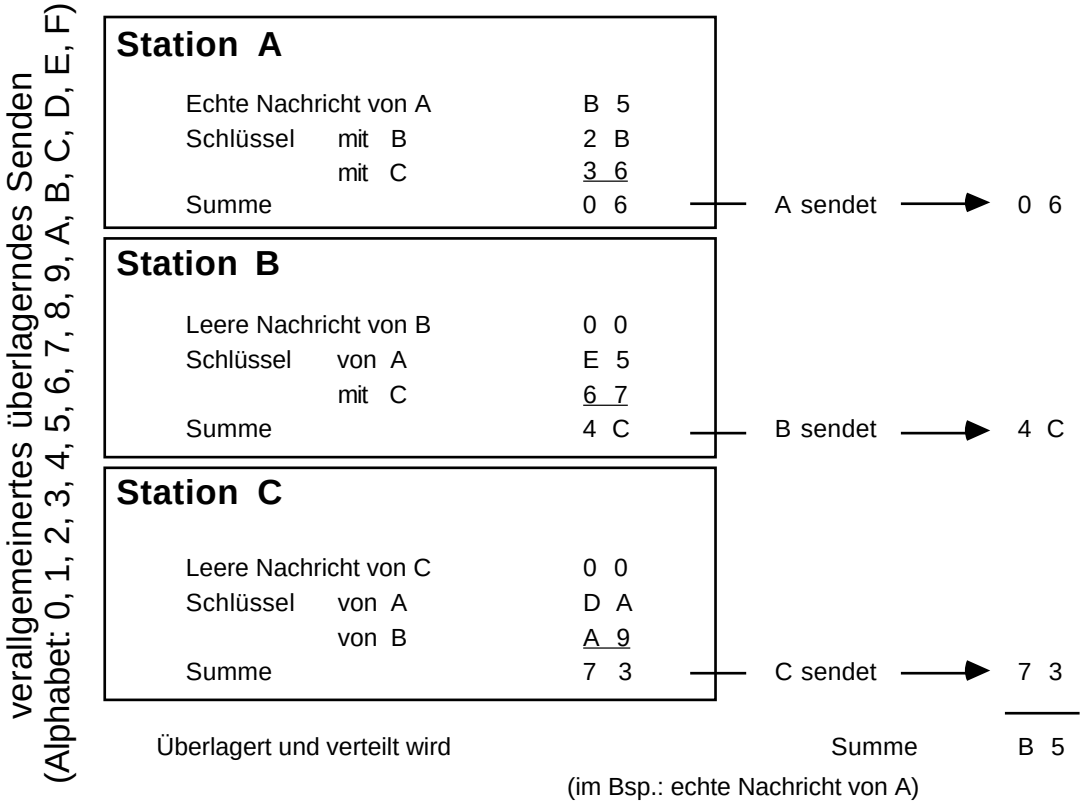
Mit dieser Art, offene implizite Adressen mittels des Nachrichten-Service zu implementieren, können auch bequem Kanäle geschaltet werden: Der Kanalaufbau ist die Vereinbarung einer Speicherzelle und der Auftrag an die Server, eine Abfrage, deren Ergebnis der Inhalt dieser Speicherzelle ist, zyklisch auszuführen.

Eine offene Frage ist, ob und ggf. wie verdeckte implizite Adressen mit Hilfe des Nachrichten-Service effizient realisiert werden können.

b) Unabhängig von allen Performance-Betrachtungen gilt dies immer dann, wenn Teilnehmerstationen nicht ständig eingeschaltet oder nicht ständig erreichbar sind. Beispiele sind Laptops und Mobilfunknetze.

5-5 Überlagerndes Senden: Ein Beispiel





„Abschreiben“ muß man die echten bzw. leeren Nachrichten und jeweils einen Schlüssel jedes Schlüsselpaares, da dadurch das Beispiel festgelegt ist. Als das additive Inverse muß man jeweils

den anderen Schlüssel jedes Schlüsselpaares berechnen, da modulo 16 zwischen Addition und Subtraktion unterschieden werden muß. Nicht abschreiben darf man die lokalen Summen und die globale Summe, da ihre Werte im allgemeinen vom Modulus abhängen. (Lediglich im obigen Beispiel, wo nur eine Station eine echte Nachricht sendet, dürfte man auch die globale Summe abschreiben.)

5-6 Überlagerndes Senden: Bestimmen einer passenden Schlüsselkombination zu einer alternativen Nachrichtenkombination

Station C muß die echte Nachricht 10110101 – 01000101 = 11110000 bzw. B5 – 45 = 70 gesendet haben, denn die Summe aller echten Nachrichten muß ja konstant bleiben.

Die Schlüsselkombination ist nicht eindeutig, da der Schlüsselgraph nicht minimal zusammenhängt.

Es gilt allgemein:

wirkliche Nachricht + wirklicher Schlüssel + alle anderen Schlüssel =
alternative Nachricht + alternativer Schlüssel + alle anderen Schlüssel

⇔ wirkliche Nachricht + wirklicher Schlüssel – alternative Nachricht = alternativer Schlüssel

Läßt man den Schlüssel zwischen A und B unverändert, errechnet sich der Schlüssel zwischen A und C als wirkliche Summe von Nachricht und Schlüssel minus alternative Nachricht, also 10110101 + 00110110 – 00000000 = 10000011 bzw. B5 + 36 – 00 = EB für Station A und 25 für Station C.

Der zweite C bekannte Schlüssel (mit B) errechnet sich als wirkliche Summe von Nachricht und Schlüsseln minus alternative Nachricht minus alternativer Schlüssel, also 01010001 – 11110000 – 10000011 = 00100010 bzw. 73 – 70 – 25 = EE. Also muß B das additive Inverse 22 verwenden.

Probe, d.h. stimmt alles bei B? Gelten muß: Wirkliche Summe = alternative Summe, also 01001100 = 01000101 – 00101011 + 00100010 bzw. 4C = 45 + E5 + 22. Beides ist der Fall, also stimmt die Rechnung vermutlich.

Im folgenden Bild ist die Gesamtsituation gezeigt. Alle Änderungen gegenüber dem Bild von Aufgabe 5-5 sind kursiv.

binäres überlagerndes Senden
(Alphabet: 0, 1)

Station A

Echte Nachricht von A		*00000000*
Schlüssel	mit B	00101011
	mit C	*10000011*
Summe		10101000

A sendet → 10101000

Station B

Leere Nachricht von B		*01000101*
Schlüssel	mit A	00101011
	mit C	*00100010*
Summe		01001100

B sendet → 01001100

Station C

Leere Nachricht von C		*11110000*
Schlüssel	mit A	*10000011*
	mit B	*00100010*
Summe		01010001

C sendet → 01010001

Überlagert und verteilt wird — Summe 10110101

verallgemeinertes überlagerndes Senden
(Alphabet: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

Station A

Echte Nachricht von A		*0 0*
Schlüssel	mit B	2 B
	mit C	*E B*
Summe		0 6

A sendet → 0 6

Station B

Leere Nachricht von B		*4 5*
Schlüssel	von A	E 5
	mit C	*2 2*
Summe		4 C

B sendet → 4 C

Station C

Leere Nachricht von C		*7 0*
Schlüssel	von A	*2 5*
	von B	*E E*
Summe		7 3

C sendet → 7 3

Überlagert und verteilt wird — Summe B 5

5-7 Mehrfachzugriffsverfahren für additive Kanäle, beispielsweise überlagerndes Senden

a) Reservierungsverfahren

Die Summe der zweiten Spalte des Reservierungsrahmens wäre 1. Deshalb würden die Teilnehmerstationen 1, 2 und 4 jeweils vermuten, daß sie als einzige den ersten Nachrichtenrahmen (wenn alle *möglichen* Nachrichtenrahmen gezählt werden: den zweiten) reserviert hätten, so daß in ihm eine Kollision von 3 Nachrichten stattfände.

b) Paarweises überlagerndes Empfangen

In Erfahrung bringen müssen sie als erstes, in welcher Gruppe das DC-Netz addiert. (Dies müssen „normale“ Benutzer nicht wissen. Selbst die Generierung der Schlüssel für ein One-time-pad ist unabhängig davon.)

Arbeitet das DC-Netz modulo 2, so gilt aus Sicht von Sparsi:

	1010 0001	(Summe)
$\oplus$	0110 1100	(von Sparsi selbst gesendet)
	1100 1101	(vom Partner Knausri empfangen)

Also hat bei DC-Netz modulo 2 Knausri 1100 1101 an Sparsi gesendet.

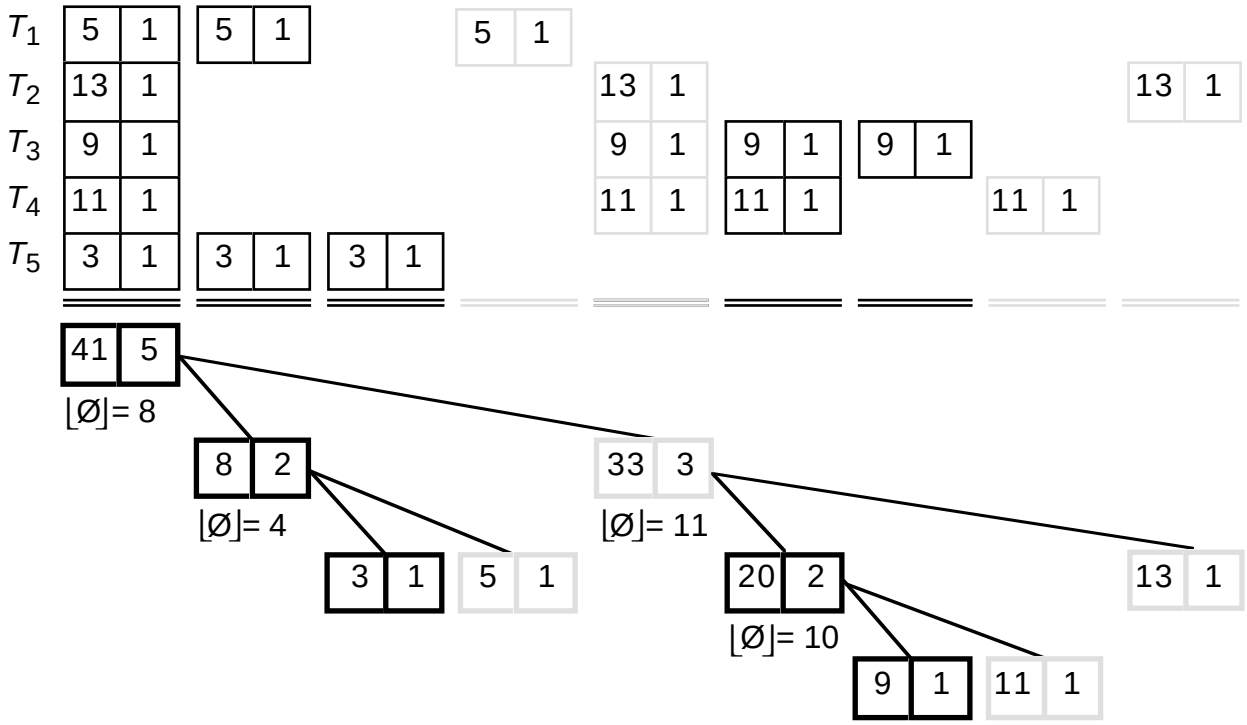
Arbeitet es modulo 256, so gilt aus Sicht von Sparsi:

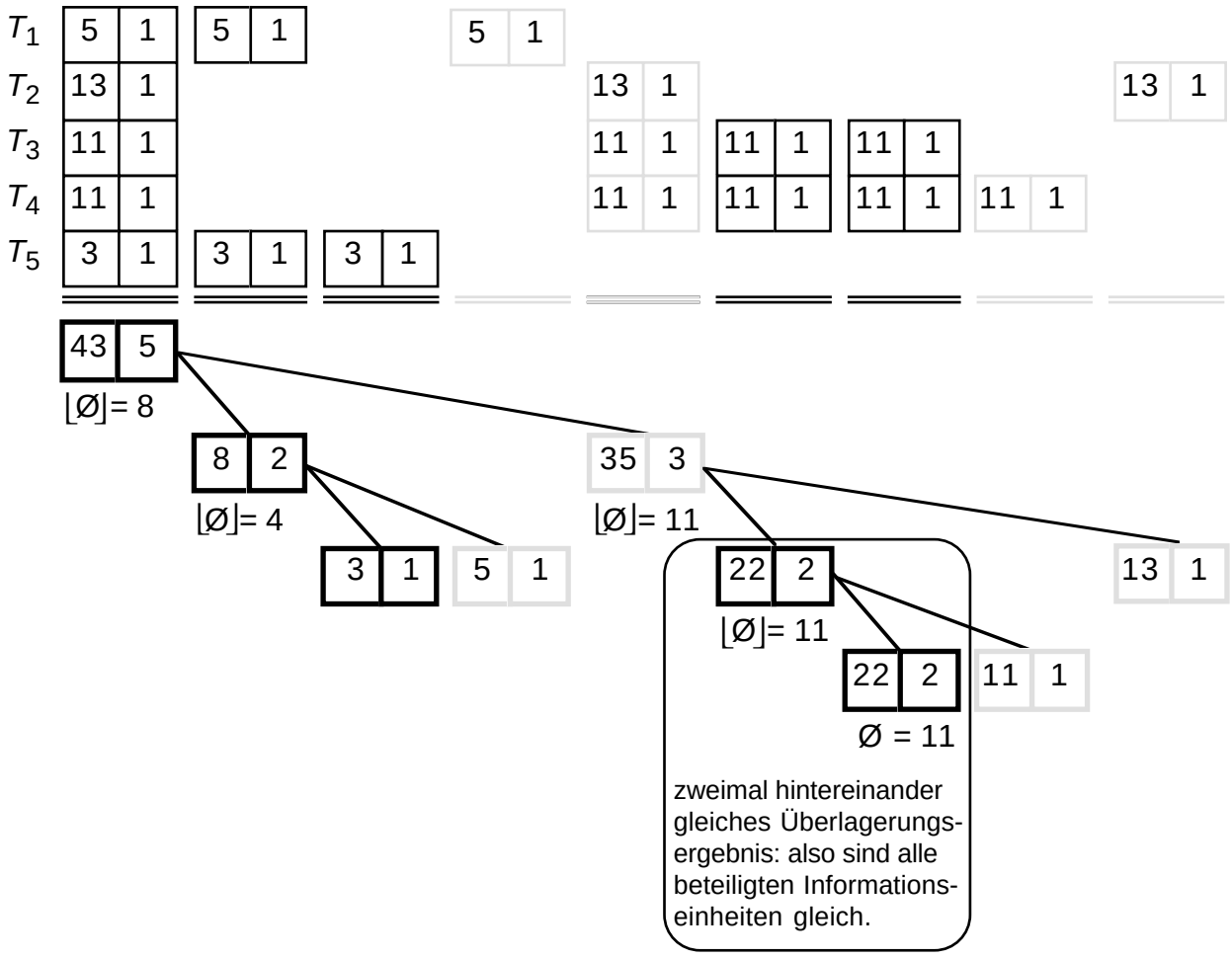
	1010 0001	(Summe)
$-_{(256)}$	0110 1100	(von Sparsi selbst gesendet)
	0011 0101	(vom Partner Knausri empfangen)

Also hat bei DC-Netz modulo 256 Knausri 0011 0101 an Sparsi gesendet.

(Probe im Dezimalsystem für letzteres: $161 -_{(256)} 108 = 53$)

c) Globales überlagerndes Empfangen: Kollisionsauflösungsalgorithmus mit Mittelwertbildung





Anmerkung: Durch folgende Erweiterung des Protokolls hätte im letzten Beispiel noch ein physischer Sendeschritt eingespart werden können: Nach Empfang des globalen Überlage-

rungsergebnisses (22,2) in Schritt 6 (= physischer Sendeschritt 4) kann das Ergebnis von Schritt 7 (= physischer Sendeschritt 5) bereits ausgerechnet werden. Zwei Summanden, die nach dem Ergebnis von Schritt 5 jeweils höchstens 11 groß sein dürfen, haben die Summe 22. Also muß jeder der beiden Summanden den Wert 11 haben. Also wird sich als globales Überlagerungsergebnis von Schritt 7 abermals der Wert (22,2) ergeben. Folglich kann Schritt 7 virtualisiert und damit der physische Sendeschritt eingespart werden. Ob sich der Einbau des zusätzlichen Tests „Prüfe, ob vorherige Obergrenze mal Anzahl kollidierter Nachrichten = Summe der kollidierten Nachrichten“ lohnt, ist unklar. Denn diese Situation wird üblicherweise nur sehr selten auftreten.

5-8 Schlüssel-, Überlagerungs- und Übertragungstopologie beim Überlagernden Senden

a) Da in jeder abelschen Gruppe *Kommutativ-* und *Assoziativgesetz* gelten, ist Reihenfolge und Teilsummenbildung bei der Überlagerung egal. Deshalb hat die Überlagerungs- und erst recht die Übertragungstopologie keinen Einfluß darauf, ob zwei Stationen einen Schlüssel austauschen können oder nicht. Die Schlüsseltopologie kann also frei gewählt werden.

b) Um Übertragungsaufwand zu sparen, sollten Teilsummen möglichst „früh“ gebildet werden, d.h. anstatt zwei Werte nacheinander über denselben Übertragungsweg zu schicken, sollten sie vorher addiert werden.

5-9 Abstimmung der Überlagerungs- und Übertragungsalphabete beim Überlagernden Senden

Die Größe des Überlagerungsalphabets wurde als Potenz der Größe des minimalen Übertragungsalphabets gewählt. Dann kann übertragungsalphabetstellenweise addiert und jedes entstehende Summenzeichen gleich übertragen werden, vgl. Bild 5-16.

5-10 Vergleich „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ und „Überlagerndes Senden“

a) *Stärke der berücksichtigten Angreifer*: Da beim Überlagernden Senden ein Angreifer eine Station auch dann nicht als Sender identifizieren kann, wenn er sie (bis auf den Schlüsselaustausch) physisch vollständig umzingelt hat, sind hier wesentlich stärkere Angreifer berücksichtigt.

Aufwand: Normalerweise ist bei jedem Netz das Verlegen der Leitungen (Graben der Kabelkanäle etc.) das Aufwendigste. Wenn also für „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ keine passende Leitungstopologie schon besteht bzw. sowieso erst noch auf der grünen Wiese frei festgelegt werden muß, dann ist „Überlagerndes Senden“ weniger aufwendig. Ansonsten ist Überlagerndes Senden aufwendiger, da hier zusätzlich zu Übertragung (und Überlagerung) auch Schlüssel erzeugt und ausgetauscht werden müssen.

Flexibilität: Da das Überlagernde Senden von der physischen Netztopologie weitgehend unabhängig ist, ist es das flexiblere Konzept.

Überlagerndes Senden ist das elegantere und schönere Konzept, da seine Senderanonymität nicht von physischen Gegebenheiten wie Netztopologie abhängt. Kurzum: Das Angreifermodell des Überlagernden Sendens ist kürzer beschreibbar, sprich: kanonischer.

b) Das Konzept „Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung“ deckt Empfängeranonymität üblicherweise mit ab, da ein Angreifer, der an bestimmten Stellen nicht abhören kann, an ihnen plausiblerweise auch nicht die Konsistenz der Verteilung störend verändernd angreifen kann.

Beim Überlagernden Senden kann durch Einbeziehung der verteilten Zeichen in die Schlüsselgenerierung Empfängeranonymität erreicht werden.

Bei interaktiven Diensten ist Sender- und Empfängeranonymität jeweils nur so stark gegeben wie das *Schwächere* von beiden, da jede Station abwechselnd als Sender und Empfänger fungiert.

5-11 Vergleich „Überlagerndes Senden“ und „umcodierende MIXe“

Schutzziel: Überlagerndes Senden schützt Sender und Empfänger. Dies ist mehr als nur die Beziehung zwischen Sender und Empfänger zu schützen, was die umcodierenden MIXe tun.

Stärke der berücksichtigten Angreifer: Im Gegensatz zum Überlagernden Senden kann mit umcodierenden MIXen keine informationstheoretische Anonymität erreicht werden – außer der Sender traut dem ersten und der Empfänger dem letzten MIX bedingungslos.

Aufwand: Wird kein allzu großer Anteil aller Stationen als MIX verwendet, dann erfordern MIXe wesentlich weniger Übertragungsbandbreite als das Überlagernde Senden.

Flexibilität: Beide Konzepte sind flexibel, wobei die umcodierenden MIXe durch die von der Teilnehmerzahl unabhängige Wahlmöglichkeit der Zahl der MIXe noch etwas flexibler sind.

Bezüglich Übertragungsbandbreite (insbesondere im Teilnehmeranschlußbereich, wo ihr Erhöhen besonders aufwendig ist) sind MIXe praktikabler. Bezüglich organisatorischer Strukturen werfen MIXe größere Probleme auf: Für sie werden zumindest in ihrer Summe vertrauenswürdige Betreiber benötigt.

5-12 Reihenfolge der Grundfunktionen von MIXen

Wiederholungen sollten, um Verwaltungsaufwand zu minimieren, möglichst früh erkannt und ignoriert werden. Erfolgt dies vor dem Puffern, so ist es selbst weniger zeitkritisch. In jedem Fall muß es vor dem Test „Genügend viele Nachrichten von genügend vielen Absendern?“ erfolgen. Sonst bräuchte ein Angreifer nur Nachrichtenkopien an MIXe zu schicken, um eine deutlich kleinere Schubgröße zu erreichen.

Umcodieren kann zwar parallel zu Puffern beginnen, die letzte Umcodierung kann aber erst erfolgen, nachdem der Eingabeschub vollständig ist.

Entsprechend kann die Umsortierung erst nach der letzten Umcodierung abgeschlossen werden.

5-13 Reicht Ausgabenachrichten von MIXen gleichlang?

Zuallererst ist zu klären, wie denn die gleiche Länge der Ausgabenachrichten entsteht:

Werden alle Nachrichten, die länger als die kürzeste Eingabenachricht sind, an dieser Stelle abgeschnitten? Dann wäre es erstens günstiger, alle Sender einigten sich auf diese Länge – so würde Aufwand gespart. Zweitens kann es sein, daß der Empfänger der Nachricht den MIX überbrücken kann: Ist seine Nachricht abgeschnitten und gab es nur eine Eingabenachricht an den MIX, die länger als die kürzeste Eingabenachricht war, entspricht diese seiner.

Werden alle Nachrichten an einer – sei es vom MIX vorgegebenen oder sich aus den Eingabenachrichten ergebenden – Stelle abgeschnitten bzw. falls sie zu kurz sind, auf diese Länge ergänzt, so kann auch hier der gerade beschriebene Fall eintreten. Auch diese Variante ist also unsicher.

Werden alle Nachrichten, die nicht sowieso die Maximallänge haben, vom MIX auf diese Länge ergänzt, so ist diese Ergänzung dem Empfänger der Nachricht ersichtlich. Er kann die Länge der zugehörigen Eingabenachricht also rückrechnen und so den MIX überbrücken.

5-14 Keine Zufallszahlen --> MIXe sind überbrückbar

Der Angreifer zeichnet alle Nachrichten im Netz auf, d.h. sowohl die Nachrichten zwischen Sendern und MIXen, zwischen MIXen und auch die zwischen MIXen und Empfängern. Danach verschlüsselt er die Ausgabenachrichten der MIXe jeweils mit deren öffentlichen Schlüssel und erhält als Ergebnis jeweils die zugehörige Eingabenachricht. Damit liegen dem Angreifer alle Kommunikationsbeziehungen offen vor Augen.

Insgesamt handelt es sich um einen passiven beobachtenden Angriff.

5-15 Individuelle Wählbarkeit des symmetrischen Konzelationssystems bei MIXen?

Die Definition symmetrischer Konzelationssysteme sagt nichts darüber, wie sie jeweils Klar- und Schlüsseltexte codieren, ob also insbesondere Klartext- und Schlüsseltextraum zwischen unterschiedlichen Systemen gleich sind. Sind im Extremfall die Räume zwischen unterschiedlichen Systemen disjunkt und wählen die Teilnehmer jeweils unterschiedliche symmetrische Konzelationssysteme, aber jeder Teilnehmer immer dasselbe, dann nützen die MIXe nicht einmal gegenüber äußeren Beobachtern irgend etwas. Selbst wenn die Räume aller Systeme gleich sind, die Teilnehmer die Systeme aber wie beschrieben verwenden, können der erste und letzte MIX der von einem Teilnehmer spezifizierten Umcodierungen die Nachrichten verketten. Folglich sollte jeder MIX nicht nur das asymmetrische, sondern auch das symmetrische Konzelationssystem vorgeben.

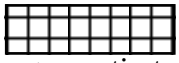
5-15a Weniger Speicheraufwand beim Generierer von anonymen Rückadressen

Der Generierer wählt den eindeutigen Namen *e* der Rückadresse zufällig und bildet dann alle symmetrischen Schlüssel hieraus unter Benutzung eines einzigen Geheimnisses pseudozufällig, d.h. deterministisch und für ihn reproduzierbar. Dann muß er sich nur dieses eine Geheimnis merken und kann, wenn immer ihn ein eindeutiger Name einer seiner Rückadressen erreicht, dieselben symmetrischen Schlüssel nochmals erzeugen.

Für die Sicherheit des Verfahrens ist essentiell, daß selbst aus *e* und m-1 symmetrischen Schlüsseln zusammen nichts über den m-ten symmetrischen Schlüssel errechnet werden kann.

Eine mögliche Implementierung ist, *e* als Startwert und das Geheimnis als Schlüssel für einen Pseudozufallsbitfolgengenerator zu nehmen, vgl. §3.4.2.

5-16 Warum längentreue Umcodierung?

Der Angreifer hört die Ein- und Ausgaben der MIXe ab. Falls er nicht sowieso weiß, um wie viel Bit jede Nachricht durch das Umcodieren kürzer wird, wird er dies leicht erraten: Bildet er Differenzen der Längen von Ein- und Ausgabenachrichten, treten manche Werte besonders häufig auf – oder er berechnet einfach die durchschnittliche Nachrichtenverkürzung, falls zu vermuten ist, daß alle Nachrichten um gleich viele Bits kürzer werden. Damit ist dann in Bild 5-26 unten alles klar: Zur Eingabenachricht  paßt längenmäßig nur  und entsprechend paßt zu  nur  . Bei MIX 2 und 3 kann genauso argumentiert werden.

5-17 Minimal längenexpandierendes längentreues Umcodierungsschema für MIXe

a) Der Kommunikations- und Verschlüsselungsaufwand pro Informationseinheit ist direkt proportional zum Produkt aus der jeweiligen Länge der Informationseinheit und der Zahl der pro Kommunikationsbeziehung benutzten MIXe, da die Informationseinheit natürlich zwischen MIXen jeweils übertragen und von ihnen jeweils umcodiert werden muß.

   Deshalb ist ein bei vorgegebenem asymmetrischem Konzelationssystem minimal längenexpandierendes (und für allgemeine Anwendungen zusätzlich noch längentreues) Umcodierungsschema von großer Bedeutung.

b) Mittels OFB kann man aus der symmetrischen Blockchiffre eine symmetrische Stromchiffre für Einheiten beliebiger Länge erhalten, die die Eigenschaften $k^{-1}(k(x)) = x$ und $k(k^{-1}(x)) = x$ besitzt.

   „Überflüssige“ Bits im mit dem asymmetrischen Konzelationssystem verschlüsselten ersten Block werden vom Verschlüsseler mit Nutzinformation belegt und vom entschlüsselnden MIX *vor* den mit einer symmetrischen Stromchiffre entschlüsselten Rest der Nachricht

gehängt. *An* den Rest der Nachricht wird dann entsprechend viel (genauer: wenig) „zufälliger Inhalt“ angehängt, wodurch die Umcodierung längentreu wird.

Jede Nachricht N_j besteht aus b Bits und wird von MIX_{j-1} gebildet.

Wie im Bild gezeigt, entschlüsselt jeder MIX_j die ersten b_j Bits der Nachricht N_j mit seinem geheimgehaltenen Dechiffrierschlüssel d_j und findet als Ergebnis dieser Entschlüsselung

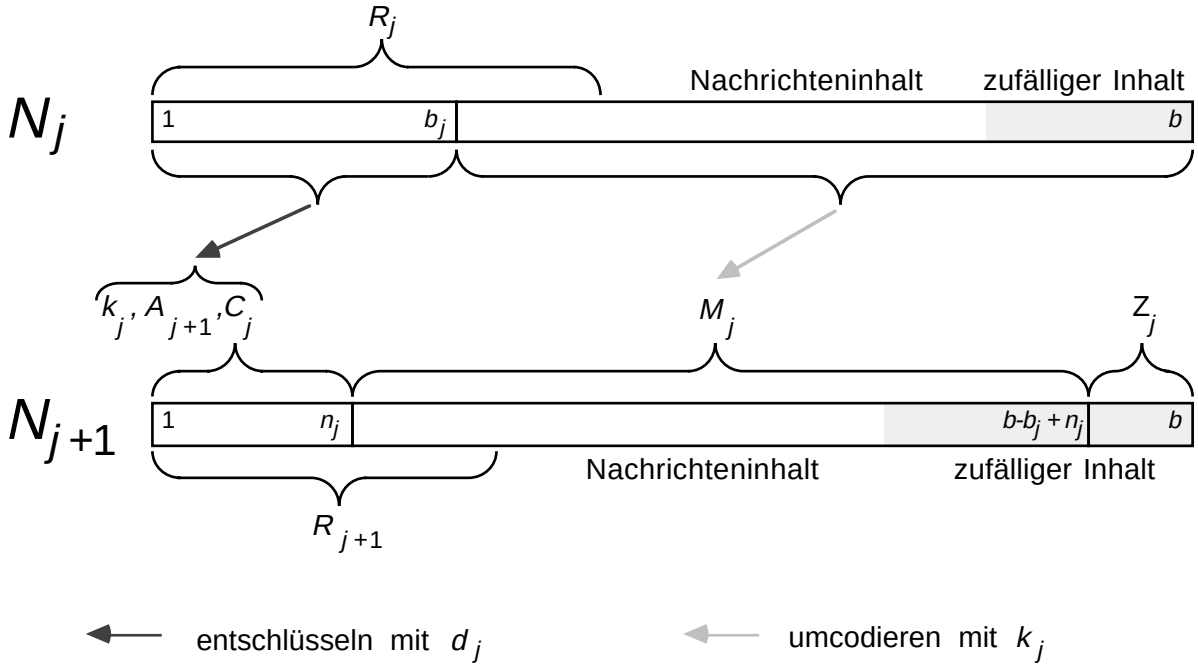
1. einen Schlüssel k_j einer symmetrischen, beliebig lange Informationseinheiten verschlüsselnden Stromchiffre (zum Umcodieren der restlichen b-b_j Bits der Nachricht),
2. die Adresse A_{j+1} des nächsten MIXes (oder Empfängers) und
3. n_j mit chiffrierter Nutzinformation belegte Bits C_j.

Mit k_j verschlüsselt MIX_j die restlichen b-b_j Bits der Nachricht und erhält so den Mittelteil M_j der auszugebenden Nachricht N_{j+1}. Danach hängt er vor M_j die n_j mit Nutzinformation belegten Bits C_j, hinter M_j hängt er, um die Länge der Nachricht nicht zu ändern, b_j-n_j Bits zufälligen Inhalts Z_j. Danach sendet MIX_j die Nachricht $N_{j+1} = C_j,M_j,Z_j$ an die Station mit Adresse A_{j+1}.

Mit den in §5.4.6.5 verwendeten Bezeichnungen besteht jede Nachricht N_j aus dem (Rück-)Adreßteil R_j und dem Nachrichteninhaltsteil I_j. N_1 wird vom Sender (auch MIX_0 genannt) entsprechend den in §5.4.6.5 angegebenen rekursiven Schemata gebildet. Um zu verdeutlichen, wie dies geschieht, wird hier das Bildungsschema für den (Rück-)Adreßteil R_1 explizit angegeben. Im folgenden bedeute $[R_j]_{\leq y}$ die Bits an den Positionen $\leq y$ von R_j, $[R_j]_{>y}$ die Bits an den Positionen $>y$.

$$\begin{aligned} R_{m+1} &= e \\ R_j &= c_j(k_j,A_{j+1},[R_{j+1}]_{\leq n_j}),k_j^{-1}([R_{j+1}]_{>n_j}) \qquad \text{für } j= m,...,1 \end{aligned}$$

Da die Instanz, die k_j generiert, nämlich Sender bzw. Empfänger, jeweils $k_j^{-1}([R_{j+1}]_{>n_j})$ kennt, kann sie den, wie in §5.4.6.5 gebildeten, Nachrichteninhaltsteil I_j jeweils mit der synchronen Stromchiffre passend vor dem Senden bzw. nach dem Empfangen entschlüsseln.



Minimal längenexpandierendes längentreues Umcodierungsschema

5-18 Brechen der direkten RSA-Implementierung von MIXen

RSA ist multiplikativ und dies überträgt sich zumindest näherungsweise auch auf die direkte RSA-Implementierung von MIXen. Deshalb kann ein Angreifer Vielfache einer beobachteten Eingabe-Nachricht in den MIX eingeben und als Vielfache der entsprechenden Ausgabe-Nachricht erkennen.

5-19 Wozu MIX-Kanäle?

a) Um die Verzögerungszeit und den Aufwand des Umcodierens in den MIXen zu senken: Während des Kanals kann ein symmetrisches Konzelationssystem verwendet werden, so daß nur für den Kanalaufbau ein asymmetrisches Konzelationssystem verwendet werden muß. Symmetrische Konzelationssysteme sind bekanntlich wesentlich weniger aufwendig bzw. wesentlich schneller als asymmetrische.

b) Kanäle, die zusammen gemixt werden und sich bzgl. den Kommunikationsbeziehungen gegenseitig schützen sollen, müssen zusammen beginnen *und* enden. Bereits ersteres allein erfordert spürbare Wartezeiten. Beides zusammen würde unerträgliche Wartezeiten erzwingen.

5-19a Freie MIX-Reihenfolge bei fester Anzahl benutzter MIXe?

Der von Ronny entdeckte Angriff geht so: Der Angreifer weiß für jede einzelne Nachricht vor dem einzigen vertrauenswürdigen MIX, wie viele seiner MIXe diese Nachricht wie durchlaufen hat. Ebenso weiß er für jede einzelne Nachricht nach dem vertrauenswürdigen MIX, wie viele seiner MIXe diese Nachricht noch durchläuft. Also gehört genau die Nachricht vor dem vertrauenswürdigen MIX zu der Nachricht nach dem vertrauenswürdigen MIX, wo die Summe der durchlaufenen MIXe paßt. Scheibenkleister. Also entweder freie Routenwahl und zwangsweise zufällige Routenlängen – oder doch Kaskaden.

5-20 Wie sichern, daß MIXe Nachrichten von genügend vielen unterschiedlichen Sendern erhalten?

a) Da der erste MIX wissen darf, welche Informationseinheiten von welchem Sender kommt, funktioniert bei ihm die kanonische Lösung: Sender lassen sich registrieren, veröffentlichen Testschlüssel und signieren die Informationseinheiten, die sie an den ersten MIX senden. Dann kann der erste MIX – sowie jeder, der sich dafür interessiert – prüfen, von wie vielen verschiedenen Sendern Informationseinheiten im Schub sind.

b) Hintere, d.h. nicht erste, MIXe dürfen nicht wissen, wer ihre Eingabe- oder Ausgabe-Informationseinheiten gesendet hat – sonst bewirken die vorderen MIXe den hinteren gegenüber keinen Schutz, was sie aber sollen! Deshalb darf die Lösung von a) hier auf gar keinen Fall angewendet werden. (Hier erscheint Ihnen die Aufgabe nun möglicherweise unlösbar – aber nicht zu früh aufgeben, es geht. Vielleicht probieren Sie es noch mal, bevor Sie weiterlesen.)

Die Lösungsidee besteht darin: Zwar soll im hinteren Teil der MIX-Kaskade niemand mehr testen können, von wem eine Informationseinheit stammt. Aber wenn wir mit a) sicherstellen, daß in den ersten MIX „Informationseinheiten von genügend vielen Sendern“ hineingingen und sich alle MIXe korrekt verhalten, dann braucht dies auch niemand! Hinten sind dann noch Informationseinheiten von genauso vielen protokolltreuen Teilnehmern vorhanden wie vorne! (Der Zusatz protokolltreu ist wichtig, wie wir gleich sehen werden.)

Um sicherzustellen, daß sich alle MIXe korrekt verhalten, vereinbaren wir folgendes Protokoll: Alle Ein- und Ausgaben der MIXe sind öffentlich und von den MIXen signiert. Jeder, der eine Informationseinheit zur Schubfolge beigesteuert hat, überprüft direkt nach jedem Schub jedes einzelnen MIXes, ob seine Informationseinheit richtig gemixt wurde. Wenn nein, erhebt er Einspruch, bevor der nächste MIX seine Ausgabe bekanntgibt. Wie er ihn beweisbar erheben kann, steht in §5.4.7 in der Fußnote – und implizit damit auch, wann Einsprüche als unberechtigt verworfen und derjenige, der den Betrieb nur aufgehalten hat, bestraft werden

sollte. Erhebt niemand Einspruch, dann sind alle Informationseinheiten von protokolltreuen Teilnehmerstationen noch vorhanden: Zu jeder Eingabe gibt es die entsprechende Ausgabe, sonst hätte die protokolltreue Teilnehmerstation Einspruch erhoben. Protokolltreue Teilnehmerstationen schicken keine Kopien von Nachrichten, die sie selbst oder andere schon mal geschickt haben - wenn dies Angreiferstationen tun, verhindert dies nicht, daß eine der Informationseinheiten durchkommt. Da aber Angreiferstationen sowieso nichts zur Unbeobachtbarkeit der Kommunikationsbeziehung beitragen, ist es nicht schädlich, wenn ihre Informationseinheiten-Wiederholungen von MIXen legitimerweise ignoriert werden – selbst wenn ihre originalen Informationseinheiten illegitimerweise unterdrückt oder ersetzt werden und sie nicht protestieren, ist dies nicht schlimm.

Das beschriebene Protokoll, bei dem jeder bzgl. seiner eigenen Nachrichten fortlaufend jeden MIX überprüft, *verhindert* den Erfolg verändernder Angriffe, kann aber zu aufwendig sein: Wenn jeder jede Ausgabe jedes MIXes erhalten soll, um unbeobachtbar prüfen zu können, dann erfordert die naive Implementierung (Verteilung) mehr Aufwand als normale Verteilung zum Schutz des Empfängers, die pro Nachricht ja nur einmal – und nicht wie hier nach jedem MIX – erfolgen muß.

Eine Alternative, die verändernde Angriffe im allgemeinen Fall jedoch lediglich *erkennt*, ist stichprobenartige Überprüfung.

Eine andere Alternative wäre, daß jede Nachricht für jeden MIX (oder aus Effizienzgründen nur für den letzten) eine anonyme Rückadresse enthält, die mit einer digital signierten Quittung über die Auslieferung der Nachricht an den Sender zurückgeschickt wird. Dann kann er überprüfen, ob seine Nachricht den Empfänger ohne Verzögerung erreicht hat, und falls nein ggf. sogar, wo sie unterschlagen wurde.

Senden Teilnehmer bedeutungslose Nachrichten, so können sie diese an sich selbst senden und dabei überprüfen, daß ihre bedeutungslosen Nachrichten sie auch so schnell erreichen, wie dies zu erwarten war. So können Teilnehmer fast ohne Zusatzaufwand überprüfen, daß MIXe Nachrichten der Teilnehmer nicht durch ihre eigenen ersetzen. Aber auch dies verhindert den Erfolg von verändernden Angriffen nicht, sondern erkennt sie nur.

5-21 Koordinations-Protokoll bei MIXen: Wozu und wo?

Ein Koordinations-Protokoll zwischen MIXen ist immer dann notwendig, wenn sich MIXe gegenseitig ersetzen können, wie dies bei *MIXe mit Reserve-MIXen* und *Auslassen von MIXen* der Fall ist. Die Koordination zwischen diesen MIXen muß jeweils sicherstellen, daß keine Umcodierung mehrfach erfolgt – sonst wären Ein- und Ausgangsnachrichten dieser MIXe über ihre Häufigkeiten verkettbar.

5-22 Grenzziehung der Verantwortungsbereiche – Funktionsumfang der Netzabschlüsse

Funktionen, denen sowohl der Teilnehmer für seine persönlichen Sicherheitsinteressen (vor allem Vertraulichkeit und Integrität) als auch der Netzbetreiber für seine anbietermäßigen Sicherheitsinteressen (vor allem Verfügbarkeit der Kommunikation) vertrauen müssen, bedürfen einer Implementierung, deren Korrektheit des Entwurfes, der Produktion, des Betriebes und der Wartung dann *mehrere* Parteien vertrauen müssen. Gibt es keinen beiden vertrauenswürdigen Betreiber, werden ausforschungssichere Geräte benötigt – es gelten alle kritischen Bemerkungen aus §2.1.

Ende-zu-Ende-Verschlüsselung und implizite Adressierung (übrigens genauso wie digitale Signaturen) betreffen jeweils nur die direkt an dieser Kommunikation Beteiligten. Diese Funktionen kann also jeder Teilnehmer so gut oder schlecht realisieren, wie es ihm behagt.

Beim RING-Netz sind Übertragungs- und Zugriffsverfahren sowohl für Sender- und Empfängeranonymität einerseits wie auch Verfügbarkeit der Kommunikation andererseits kritisch. Sie müssen folglich für alle vertrauenswürdig realisiert werden.

Beim Überlagernden Senden sind Schlüsselgenerierung und -überlagerung sowie das Zugriffsverfahren kritisch sowohl für die Senderanonymität der Teilnehmer als auch für die Verfügbarkeit der Kommunikation. Dies muß also für alle vertrauenswürdig realisiert werden. Hingegen betrifft Übertragungsverfahren und Übertragungsmedium nur die Verfügbarkeit der Kommunikation und kann folglich vom Netzbetreiber ganz nach seinen Wünschen realisiert werden.

Bei MIXen muß jeder Teilnehmer bzgl. der Vertraulichkeit seiner Kommunikationsbeziehungen mindestens *einem* der von ihm verwendeten MIXe vertrauen. Hingegen muß der Netzbetreiber, sofern keine Fehlertoleranzmaßnahmen auf MIX-Ebene ergriffen werden, für die Verfügbarkeit der Kommunikation *allen* MIXen vertrauen.

5-26 Mehrseitig sicherer Web-Zugriff

Als Schutzziele können die in §5.1.2 genannten übernommen werden, indem Nachrichteninhalte übermittelte Web-Inhalte bedeuten. Als Schutzmechanismen bieten sich an:

Schutzmechanismen für Vertraulichkeit

c1 Verwendung eines oder mehrerer Konzelationssysteme zwischen Browser und Web-Servern: Ende-zu-Ende-Verschlüsselung, vgl. §5.2.1.2. Eine Beispielimplementierung auf Grundlage des Internet-Standards **Secure Socket Layer (SSL)**, aber mit starker (d.h. nicht durch US-Exportrestriktionen auf 40 Bit Schlüssellänge beschränkter) Verschlüsselung mittels einer außerhalb Nordamerikas entwickelten SSL-Bibliothek (SSLeay) ist in [FeMa_98, FeMa1_98] beschrieben.

c2 Verfahren innerhalb des Kommunikationsnetzes zum Schutz der Verkehrs- und Interessensdaten. Hier ist gegenüber §5.3 bis §5.6.1 eine erhebliche Anpassung nötig: Einerseits ist Web-Zugriff eine Realzeitanwendung – wie die Interpretation von WWW als World Wide Wait unterstreicht, so daß Konzepte, die für e-mail gut passen, nicht einfach übernommen werden können. Andererseits kann, da in vielen Bereichen bereits im Teilnehmerbereich ein shared medium vorliegt, nicht so einfach mit dummy traffic gearbeitet werden, wie dies in §5.5 am Beispiel Telefon-MIXe für das ISDN beschrieben wurde.

Adaptiert man das Konzept Unbeobachtbarkeit angrenzender Leitungen und Stationen sowie digitale Signalregenerierung aus §5.4.4, so kommt man für den anonymen Web-Zugriff auf die Lösung **Crowds** [ReRu_98, ReRu_99].

Adaptiert man das Konzept MIXe aus §5.4.6, so kommt man auf die Lösung **Onion Routing** [GoRS_99] oder **Web-MIXe** [FeMa_98, FeMa1_98].

Adaptiert man das Konzept Pseudonyme aus §6.1 und schreibt ein nettes Tool für das Management der eigenen Identitäten (sprich Pseudonyme) im Web, dann erhält man die Lösung **Lucent Personalized Web Assistant (LPWA)** [GGKM_99].

Möchte man das anonyme Anbieten von Information im WWW unterstützen, dann kommt man vielleicht auf die Lösung **Janus** [RiDe_99].

c3 Momentan für Web-Zugriff noch nicht wichtig. Deshalb wurden bisher keine speziellen Verfahren entwickelt oder gar implementiert.

Schutzmechanismen für Integrität

i1 Verwendung eines oder mehrerer Authentikationssysteme. Eine Beispielimplementierung auf Grundlage von SSL ist in [FeMa_98, FeMa1_98] beschrieben.

i2 Nachrichten werden vom Urheber digital signiert. Testschlüssel sind öffentlich bekannt. Empfänger prüft Signaturen, bevor er Nachrichten akzeptiert. Eine Beispielimplementierung auf Grundlage von SSL ist in [FeMa_98, FeMa1_98] beschrieben.

i3 Momentan für Web-Zugriff nicht wichtig.

i4 Mittels eines digitalen Zahlungssystems (§6) werden Entgelte während der Diensterbringung bezahlt. Momentan für Web-Zugriff nicht wichtig.

Schutzmechanismen für Verfügbarkeit

a1 Diversitäre Netze, d.h. unterschiedliche Übertragungs- und Vermittlungssysteme; faire Aufteilung von Betriebsmitteln; Spiegelung von Web-Inhalten auf unabhängige Server, auffindbar etwa mittels unabhängiger Suchmaschinen.

5-27 Firewalls

Firewall können allenfalls dann wirkungsvoll sein, wenn sie vom Angreifer nicht umgangen werden können. Nur wenn es zwischen den öffentlichen Netzen und dem internen Netz nur wenige, bekannte Übergänge gibt, ist das Konzept Firewall sinnvoll umsetzbar. Allerdings untergräbt der Einsatz von Modems bzw. ISDN-Nebenstellenanlagen zum direkten Anschluß von an das interne Netz angeschlossenen Arbeitsplatzrechnern an öffentliche Netze dies zunehmend. Ganz extrem wird es, wenn tragbare Rechner (Laptops etc.), die über eine Funkanbindung an öffentliche Netze verfügen (etwa für AußendienstmitarbeiterInnen), bei Anwesenheit in der Firma bzw. Behörde direkt an interne Netz angeschlossen werden.

Firewalls können allenfalls auf den Kommunikationsebenen qualifizierte Entscheidungen fällen, auf denen sie die Daten „verstehen“. Das werden einerseits aus Schutzgründen (wer wird zugunsten des „Verständnisses“ der Firewall auf Ende-zu-Ende-Verschlüsselung verzichten wollen und so zentrale Angriffspunkte auf Inhaltsdaten schaffen?), andererseits aus Leistungsgründen (die Interpretation jeder Ebene kostet Rechenleistung und selbst bei vorhandener Rechenleistung Verzögerungszeit) nur die ganz unteren Kommunikationsebenen sein, geschweige denn Ebenen der Anwendungssoftware.

Firewalls sind aus Firmen- bzw. Behördensicht deutlich leichter administrierbar als Arbeitsplatzrechner – es gibt deutlich weniger und sie können einer zentralen Administration unterliegen.

Möchte eine Firma bzw. Behörde ihren MitarbeiterInnen eine Nutzung öffentlicher Netze erlauben, so sind Firewalls weitgehend wirkungslos, da auf unteren Kommunikationsebenen nahezu alles durchgelassen werden muß, da qualifizierte Entscheidungen hier unmöglich sind. Sollen qualifizierte Entscheidungen gefällt werden, so muß dies im wesentlichen auf den Arbeitsplatzrechnern der Mitarbeiter geschehen. Sind diese Arbeitsplatzrechner nicht mit einem sicheren Betriebssystem und komfortablen Tools zur Rechteverwaltung ausgestattet und die Mitarbeiter nicht für Sicherheitsbelange sensibilisiert, sind Firewalls eine teure, aber weitgehend wirkungslose Ersatzhandlung.

Lösungen zu Werteaustausch und Zahlungssysteme

6-1 Electronic Banking – Komfortversion

a) Dies ist für den Kunden zwar ultrabequem, solange er seiner Bank bedingungslos vertraut – denn es gibt bei Streitfällen keine als Beweismittel geeigneten Dokumente. Umgekehrt muß auch die Bank dem Kunden bedingungslos vertrauen – oder darauf, daß sie vor Gericht stets Recht bekommt.

b) Leider nein; bei den heutigen Electronic Banking Systemen erhält die Bank (genauer: ihre Geräte) vom Kunden nichts (genauer: keine digitalen Nachrichten), was sie nicht auch selbst bilden könnte.

c) Es ändert sich nicht, denn auch hier kann die Bank – sofern sie sich eine Kopie der Schlüssel hat machen lassen – alle Nachrichten selbst bilden, die sie von Kunden erhalten kann. Dies ist bei symmetrischen Authentikationssystemen prinzipiell so, wie in §3.1 erläutert wurde. Bei

digitalen Signatursystemen wäre es anders, wenn die Kunden ihre Schlüsselpaare selbst erzeugen würden.

6-2 Personenbezug und Verkettbarkeit von Pseudonymen

Rollenpseudonyme bieten mehr Anonymität als Personenpseudonyme, da bei Rollenpseudonymen die in unterschiedlichen Rollen ausgeübten Aktionen eines Teilnehmers über die Pseudonyme nicht miteinander verkettet werden können, sich also nicht unter einem Pseudonym Information aus unterschiedlichen Rollen ansammelt, was leicht zur Deanonymisierung führen kann. In gleicher Weise bieten Transaktionspseudonyme mehr Anonymität als Geschäftsbeziehungspseudonyme, da bei Transaktionspseudonymen selbst die in derselben Geschäftsbeziehung ausgeübten Aktionen eines Teilnehmers über die Pseudonyme nicht miteinander verkettet werden können. Es sammelt sich also unter den Pseudonymen keinerlei Information an, die über das in den einzelnen Transaktionen Notwendige hinausgeht.

6-3 Eigenauthentikation vs. Fremdauthentikation

Eigenauthentikation ist dann gegeben, wenn der die Erklärung Abgebende sich selbst autorisiert, beispielsweise indem er sich auf eine von ihm selbst früher abgegebene Erklärung bezieht. Die Eigenauthentikation besteht dann darin nachzuweisen, derselbe zu sein, der damals ein Recht erworben hat. Dies kann geschehen, indem zum damals verwendeten Pseudonym (= Testschlüssel eines digitalen Signatursystems) passend eine Nachricht signiert wird. Ein typisches Beispiel ist das Ausgeben von digitalem Geld (= Transfer eines Besitzrechtes), das der die Erklärung Abgebende früher unter diesem Pseudonym erhalten hat.

Fremdauthentikation ist dann gegeben, wenn der die Erklärung Abgebende die Autorisierung hierzu von einer anderen Instanz verliehen bekommt. Dies kann geschehen, indem das verwendete Pseudonym (= Testschlüssel eines digitalen Signatursystems) von dieser anderen Instanz eingebettet in eine passende Autorisierungs-Erklärung signiert wird. Typische Beispiele für Autorisierungs-Erklärungen sind Zeugnisse, Führerscheine, Ausweise, Kreditwürdigkeitsbescheinigungen, Scheck- und Kreditkarten.

Die Unterscheidung ist grundlegend, da oftmals der die Autorität Verleihende bei Mißbrauch der Autorisierung mit haftet, bei Fremdauthentikation haften oftmals also zwei Instanzen, bei Eigenauthentikation immer nur eine. Dies ist insbesondere bei der Gestaltung von Protokollen wichtig, bei denen manche Rollenträger anonym (genauer: pseudonym) agieren können.

6-4 Diskussion der Sicherheitseigenschaften von digitalen Zahlungssystemen

Notwendig ist aus meiner Sicht, daß Banken und Händler *keine Konsumprofile* ihrer Kunden erstellen können, erstrebenswert ist, daß sie dies auch gemeinsam nicht können.

Nett wäre, wenn etwa bei Geräteverlust oder Plattencrash verlorene Zahlungsmittel zumindest nach einer gewissen Frist dem Eigentümer wieder verfügbar gemacht werden könnten (sogenannte *Verlusttoleranz*). Diese Eigenschafft kann auch als Teil von „er ein Recht nur dann verliert, wenn er hierzu den Willen hat“ gesehen werden.

Mehrseitig sicheres digitales Zahlungssystem

Schutzziel Vertraulichkeit

- Zahlungsinhalte sollen vor allen Instanzen außer den Transaktionspartnern (Zahlender, Zahlungsempfänger, ggf. Bank, Zeuge) vertraulich bleiben.
- Zahlender und/oder Zahlungsempfänger von Nachrichten sollen voreinander anonym bleiben und durch Unbeteiligte (inkl. Zahlungssystembetreiber, Bank, Zeuge etc.) nicht beobachtet werden können.

Schutzziel Integrität

- Der Eigentümer verliert ein Recht nur dann, wenn er hierzu den Willen hat.

- Sofern ein zahlungswilliger Benutzer einen anderen Benutzer als Zahlungsempfänger eindeutig bestimmt, erhält auch nur dieser Zahlungsempfänger das Recht.
- Der Zahlende kann, falls notwendig, einen vollzogenen Transfer einem Dritten gegenüber nachweisen (Quittungsproblem).
- Die Benutzer können auch bei Zusammenarbeit ihre Rechte an Geld nicht vermehren.

Schutzziel Verfügbarkeit

- Jeder Benutzer kann erhaltene Rechte transferieren.

6-5 Wie weit erfüllen konkrete digitale Zahlungssysteme Ihre Schutzziele?

Telefonbanking: Die Schutzziele im Bereich Vertraulichkeit sind in keinster Weise erfüllt. Die Schutzziele im Bereich Integrität und Verfügbarkeit sind nur erfüllbar, wenn die Bank nicht als potentieller Angreifer gesehen wird, also die Beweisproblematik ihr gegenüber nicht auftritt. Zusätzlich muß angenommen werden, daß das verwendete Telefonsystem von Angreifern nicht abgehört werden kann, sonst können diese unter Verwendung abgehörter Paßwörter, TANs, PINs oder was auch immer zur Identifikation der Bankkunden und zur Prüfung ihrer Autorisierung verwendet wird, die Integrität mittels Transfer von fremden Rechten verletzen und so den legitimen Benutzern die Verfügbarkeit über ihre Rechte zumindest temporär entziehen.

HBCI: Durch die mögliche Verschlüsselung kann Vertraulichkeit gegenüber Außenstehenden erreicht werden, natürlich nicht gegenüber Bank oder Zahlender/Zahlungsempfänger. Anonymisierungsmaßnahmen sind nicht vorgesehen. Bei Verwendung der Zwischenlösung triple-DES als Hardware-gestützer Signaturersatz ist Integrität (und damit auch Verfügbarkeit) nur gegeben, wenn die Bank nicht als potentieller Angreifer betrachtet wird. Bei Verwendung von RSA inkl. Schlüsselpaarerzeugung beim Teilnehmer sind Integrität und Verfügbarkeit gegeben, soweit der Rechner des Teilnehmers richtig arbeitet.

Lösungen zu Regulierbarkeit von Sicherheitstechnologien

9-1 Digitale Signaturen bei „Symmetrische Authentikation ermöglicht Konzelation“

Die klassische Antwort lautet: Nein. Denn bei digitalen Signaturen kann jeder testen, ob die Authentikation stimmt, und damit kann jeder die richtigen Bits (bzw. Nachrichtenfragmente) erkennen und zusammensetzen.

Für die nichtklassische Antwort „ja“ muß genau dies, daß jeder und damit auch der Angreifer digitale Signaturen testen kann, verhindert werden. Hierzu gibt es folgende Möglichkeiten:

Man verwendet zwar ein digitales Signatursystem, gibt den „öffentlichen“ Testschlüssel aber nur dem Partner bekannt, nutzt also die in Bild 3-12 dargestellte Teilmengenbeziehung digitale Signatursysteme $\subset$ symmetrische Authentikationssysteme. Ob man das noch digitale Signaturen nennen mag, darüber kann man trefflich streiten.

Man verwendet ein digitales Signatursystem mit der Spezialeigenschaft, daß digitale Signaturen nur von ausgewählten Partnern getestet werden können [Chau_95]. Und man wählt nur einen als Partner aus [Rive_98].

9-2 „Symmetrische Authentikation ermöglicht Konzelation“ vs. Steganographie

Das von Rivest vorgeschlagene Verfahren hat eine gewisse Ähnlichkeit zu Steganographie: Die geheimzuhaltende Nachricht wird in einen umfassenden Bitstrom eingebettet und ist vertraulich, da sie in ihm nicht zu erkennen ist. Im Unterschied zu guter Steganographie ist für Abhörer jedoch erkennbar, daß mit dem umfassenden Bitstrom etwas nicht normal ist: In ihm kommen alle möglichen Bitkombinationen vor. Das steganographische Ziel „Verbergen der Existenz der geheimzuhaltenden Nachricht“ wird von Rivests Verfahren also nicht erreicht.

9-3 Kryptoregulierung durch Verbot frei programmierbarer Rechner?

Es geht nicht einmal das:

Zum einen können Menschen, die sich treffen, ein One-Time Pad austauschen und damit per Hand z.B. E-mail verschlüsseln. Wie modular addiert und subtrahiert wird, ist allgemein bekannt und nicht allzu schwierig - zumindest XOR ist deutlich einfacher als Schreiben und sicherlich bereits im Kindergarten zu lernen.

Zum andern können Menschen mittels Key-Escrow-Konzelation ohne rückwirkende Entschlüsselung (vgl. Bild 9-3) genügend lange Schlüssel zu einer Zeit austauschen, in der sie noch nicht überwacht werden dürfen. Einander treffen ist also nicht einmal nötig.

Stichwortverzeichnis (inkl. Abkürzungen)