Forward Secure Mixes

George Danezis

University of Cambridge Computer Laboratory
William Gates Building, JJ Thomson Avenue
Cambridge CB3 0FD, United Kingdom
George.Danezis@cl.cam.ac.uk

Abstract. New threats such as compulsion to reveal logs, secret and private keys as well as to decrypt material are studied in the context of the security of mix networks. After a comparison of this new threat model with the traditional one, a new construction is introduced, the *fs-mix*, that minimizes the impact that such powers have on the security of the network, by using forward secure communication channels and key updating operation inside the mixes. A discussion about the forward security of these new proposals and some extensions is included.

Keywords: Anonymity, mixes, forward security, traffic analysis

1 Introduction

Research concerning anonymous communications has for a long time concentrated on protecting against traditional cryptographic threat models. These include passive and active attacks against network links, and a subset of compromised mix nodes [6]. Unfortunately these models fail to adequately capture the subtlety of the attacks that mix networks might be subject to, such as compulsion to reveal keys and logs, as well as being forced to decrypt seized material. The shortcoming of the traditional threat model has led engineers to produce solutions that are weak under these, quite realistic, threats.

In section 2 we analyze the traditional threat models and argue that they are still relevant in order to protect mix networks. In section 3 we present some new threats and attempt to complement the traditional threat model. We then explore how existing mixing technology copes with the new threats and how partial solutions can be implemented to protect against them, in section 4 and 5 respectively. We then present a generic solution protecting against the new attacks and analyze its security and how it could in practice be used (sections 6, 7 and 8).

2 Traditional Threat Models

In the context of anonymous communications the objective of the attacker is to discover the patterns of traffic of users in the network. A secondary goal of the attacker might be to disrupt and discredit the anonymous network.

The traditional threat model for mix networks has been based on the general threat model used in security and cryptology research. In this model attackers are first classified as *passive* or *active*. A passive attacker has the ability to monitor all network links and record their traffic. David Chaum demonstrated the security of mix networks against such an attacker [6]. In other studies the attacker is assumed to be less powerful and is only allowed to monitor some proportion of links [12]. An active attacker not only has the ability to monitor links but can also delay, delete, replay and inject new messages into the network. Such an attacker can perform replay and tagging attacks that some modern proposals for mix networks address [20, 14].

In most threat models against mix networks it is also assumed that the attacker controls a subset of the mix nodes. These compromised nodes can either simply monitor the traffic, therefore revealing the correspondence between their inputs and outputs to the adversary, or can actively misbehave in order to extract information out of the network or decrease its reliability. Distribution of trust by using many mixes is usually used to address the issue of information leakage, while solutions based on publicly verifiable proofs of correctness [1, 8, 18, 17] or reputation systems [9, 10] have been proposed to address the reliability issues. Different mix network topologies such as mix-cascades [4] as implemented in JAP [3], have also been explored to minimize the risk that corrupt nodes break the security of the network.

3 Beyond the Traditional Threat Models

The threats addressed in the traditional threat model are increasingly relevant to the proper functioning of mix networks. The small number of active remailer nodes [19] and the progress in interception technology and legislation, makes it possible for all mix nodes to have their traffic monitored. Common allegations of wide-spread interception of international telecommunications reinforce the importance of that threat [7]. Domestic interception of the content of communications is, at least in theory, regulated and warrants need to be sought before it can take place. Obtaining them takes time and effort, and a clear need has to be demonstrated. Of course this does not apply to monitoring by entities not belonging to the Law Enforcement community.

Active attacks are slightly more difficult to mount but given the current technology deployed [13], where only weak authentication is used between remailer nodes, they are quite possible to perform. There have not been any confirmed reports of compromised mix nodes, but since it would be very difficult to detect them the current state of uncertainty remains. In particular recent court cases in the United States have ruled as legal "hacking" into machines [22] in order to investigate criminal activities or gather evidence. It is not clear if that could extend to uncooperative mix nodes.

In addition to the threats above, that are largely speculative, remailer operators have experienced a series of new threats. The most common one is a subpoena requesting logs of traffic. Such logs do not usually exist in modern systems, although the “anon@penet.fi” [16] remailer node had to stop operating after such an incident compromised the anonymity of one of its users.

Recently introduced U.K. legislation also allows Law Enforcement Agencies to request decryption of material and even private encryption or decryption keys [11]. This introduces the threat that honest mixes will have to collaborate by giving out some partial information to trace particular communications. All currently deployed or theoretical mix architectures are vulnerable to such an attack, particularly if they are requested to decrypt a message that has gone past them, or reveal the route of a single use reply block. Even worse the seizure of their keys would mean that their functioning would be completely transparent to outsiders. Of course such requests for decryption and keys can only be targeted and limited since both their cost is substantial and the authorization procedures are strict and time consuming.

Although interception of all network links is expensive and requires proper warrants in most jurisdictions, the interception of traffic data necessary to perform traffic analysis, does not have the same degree of legal protection. Recent legislative initiatives at international and national levels [5] make interception, access to and blanket retention of traffic data possible. Although the information present in the traffic data is much poorer than in the communication itself, mix network designers needs to keep in mind the ease of access to such data, and engineer their protocols so that they can resist opponents what have access to them.

A major assumption we will make, in trying to protect systems against the threats described above, is that with the exception of blanket traffic data retention, they have to be limited in breadth and in time. For example the attacker can only request the decryption of some material for some period of time. If the degree and the time of the attacks above was to be unlimited it would be extremely difficult to engineer a secure anonymizing protocol.

In addition to the assumed limits in time and breadth of the opponent powers we assume that the opponent does not have any special information in order to perform traffic selection, and therefore has to use information in the network to choose the victims of his targeted actions. As we will see, the techniques used to protect against the above threats rely of the attacker having to intercept or decrypt exponentially many messages relative to the number they are really interested in, or the number of remailers used.

4 Existing Networks and Designs

We will review the effect that the attacks above have on a traditional mix system, that in addition to providing a forward sender-anonymous channel also provides facilities for single use reply blocks (SURBs) as first described in [6] and used in [14]. Most of the constructions using zero-knowledge proofs [1, 8, 18, 17] are concerned with verifying correctness, therefore increasing robustness. They do not include features such as SURBs and are difficult to modify while retaining their useful properties, therefore we shall not examine them in detail, although some of the techniques we present could be adapted and used by such schemes.

The object of our study is to determine how an opponent could use the powers described by the extended threat model in order to trace a sender-anonymous communication or to find out the originator of a reply block.

In the absence of any content interception, it is extremely difficult to use any decryption powers to trace back a sender-anonymous communication. Assuming that the mix network has a majority of honest nodes and that mixing has rendered traffic data useless, the recipient of the sender anonymous message does not have in their possession any bit-string that they could ask any node of the mix network to decrypt. All the routing information has been stripped and deleted by the time the message arrives. However an opponent interested in tracing communications sent by a particular user, only needs to put this specific user under surveillance and request all intermediate nodes used by messages to reveal the destinations of the messages.

When some content interception takes place the communication can at best be traced back, from its final recipient, to the point where the message was intercepted. Often there seems to be some confusion about the exact process that an opponent would need to follow in order to trace the communication. It is not the case that the opponent can start from the last hop and work his way backwards following the path of the message backwards, by compelling mixes to decrypt the messages. Such an exercise would require the opponent to provide mixes with the material to decrypt which by default he does not have. In order to trace back a communication it is required to work forward starting at the interception point, by requiring mixes to decrypt the material intercepted hoping that the communication traced ends up being the one that had to be traced. This process means that the opponent will need to acquire exponentially many decryption in the number of hops to potentially trace a message. Good anonymizing protocols would force this effort to be as large as to require all messages present in the mix network to be decrypted.

If one can assume that near ubiquitous content surveillance is in place, the procedure above becomes much more efficient. It is only necessary for the opponent to require the last hop to decrypt all communications that were intercepted until the message to be traced is decrypted. Then the procedure is repeated on the previous hops recursively. Different mixing strategies [21] may make such an attack slightly harder but it should in general be linear on the number of mixes used by the communication (although all content has to be intercepted). There is a trade off between the number of messages that have to be intercepted and stored versus the number of decryption requests that might be requested when tracing a message. Dynamic traffic selection could be employed to select traffic according to the likelihood that it corresponds to a message sent by, or to, a subject under surveillance. Without any additional information, performing such a task is equivalent to performing traffic analysis, and will require the interception of all material. In section 5.1 we will examine how super-encrypting communications between mixes with ephemeral keys renders content surveillance of the links useless.

As stated, the above methods for tracing communications concern sender-anonymous channels. In the case that a single use reply block has to be traced back to the destination it points, much more information is available to the attacker than in the sender-anonymous case. Reply blocks contain all the routing information needed to deliver the message encrypted under the private keys of intermediate mixes. Therefore the attacker can simply request the reply block to be decrypted iteratively by all the servers. This will take some time proportional to the number of mixes that the reply block contains. Tracing reply blocks does not directly affect the anonymity of other users and is therefore likely to be considered quite proportionate if a warrant is necessary. Note that there is no need for any content or traffic data interception to trace back SURBs.

5 Partial Solutions

While they do not address the full range of attacks described above some designs propose measures against some of them. In this section we will study some of these countermeasures and assess the degree to which they protect the network. In the next section we will integrate them along with additional countermeasures to provide generic protection against attacks based on compulsion to reveal keys, logs or decryption of material.

It is worth mentioning that the proposals below address the extended threat model but no countermeasure can be effective against an adversary that can force mix nodes to keep a detailed record of their internal workings. Indeed the solutions presented rely on the honest mix nodes being able to effectively forget key material or not even being able to access it, through the use of tamper resistant cryptographic hardware. An attacker that through legislative means or hacking techniques manages to record this material will therefore be able to defeat all countermeasures.

5.1 Forward Secure Link Encryption

A cheap way to render link level surveillance and recording of content fruitless for an opponent is to use a forward secure encrypted channel between mix nodes. Technically this invloves encrypted channels established using key exchanges with ephemeral public keys, signed with long term signing keys. The ephemeral keys are discarded immediately after a session key has been established. After each message is processed the session key is updated using some hash function, some exchanged fresh nonces and possibly the message content. The old keys and the nonces are systematically deleted after the update has taken place. This makes it impossible for nodes to decrypt past messages that are presented to them, since the keys used have been deleted. It is essential that fresh nonces are used during the key updating. This forces an adversary that at some point in time seizes the keys used by the mixes to observe indefinitely the channel to keep his knowledge of the current keys up to date. Standard secure communication protocols such as SSL [2] support such forward secure modes of operation, and special purpose protocols such as JFK [23] are also available.

This technique renders interception at the link level useless, assuming that mixes have good ways of authenticating each other. If only a small number of mixes operate, this should not be a practical problem. The task of authenticating mixes could also be performed by the creator of the anonymous message, by including the hash of their verification keys in the anonymous routing data. An honest node should check that this hash matches the verification key during the key exchange protocol before establishing the encrypted channel. It also makes it difficult to perform active attacks such as inserting, delaying, deleting or modifying messages on the network links, that can be used to flood nodes to decrease the amount of anonymity they provide as described in [21].

Messages used to compel mixes into decrypting them can still be intercepted by malicious nodes and presented to the next honest mix in the chain. In order to detect malicious nodes the name of the previous mix could be contained in the headers of messages. In that way it is impossible to hide which node performed the interception or the previous decryption.

5.2 Public Key Cycling

It is best practice to change the encryption/decryption key pairs of the mixes often, and systematically destroy the old ones. This makes the decryption of messages encrypted under the old keys impossible.

The main problem arising with frequent changes of the public encryption keys is the short life of reply blocks. When a key changes, and is destroyed, it is impossible to route the SURBs encrypted under that key any more. Advertising future public keys in advance open mixes to the risk of having their future keys seized as well.

5.3 Secure Cryptographic Hardware

In order to minimize the risk of being compelled to surrender key material or decrypt intercepted material, one could implement the core functions of the mix inside a tamper proof hardware box. The sensitive functions including the decryption of messages and the replay prevention, would need to be performed by the secure co-processor in order to avoid compelled decryption. Assuming that the cryptographic co-processor is secure it should be extremely difficult to extract the secret key material inside it.

This construction protects against compulsion to reveal keys, but does not protect against corrupt mix owners, that want to trace the correspondence between inputs and outputs in the cryptographic module. In addition to the decryption and replay prevention, the secret permutation needs to be performed inside the cryptographic module in order to protect against such attacks.

6 Generic Solution: The *fs-mix*

Forward secure link encryption makes the product of interception useless, since if intermediate nodes forget the old keys, the material transmitted on the lines cannot be decrypted, but still allows corrupt mixes to intercept messages that could be decrypted to trace the communication. A more generic approach would be to make the decryption of the message impossible by honest mixes after a certain amount of time or after a certain event. We shall call this property *forward secure anonymity* and a mix that implements it a *forward secure mix* (*fs-mix*).

6.1 Presentation of the scheme

We can achieve forward secure anonymity by introducing state into the mix nodes. This is not a radically new requirement since most techniques implementing replay prevention already force mixes to keep hashes of the packets, or parts thereof, that have been processed in the past. The difference, as we will see, is that the state we require the mixes to keep in order to implement forward security needs to be kept secret since it will be part of the keying information.

In traditional mix systems [20, 15, 14] the address of the next mix ($A_{M_{n+1}}$) and the key used to decrypt the payload to be sent (K_{message}) are included in the asymmetrically encrypted header, under the public key of the mix (Pk_n).

$$M_{n-1} \rightarrow M_n : \{K_{\text{message}}, A_{M_{n+1}}\}_{Pk_n}, \{\text{Message}\}_{K_{\text{message}}}$$

Traditional mix systems, such as mixmaster, store a packet ID that is used along with some integrity checking to avoid messages being processed more than once by the node [20]. In the case of [14] a special public hash N_{replay} of the symmetric key K, otherwise used to decrypt parts of the message, is kept in order to avoid replays of the same message. In case another message's secret hashes to the same value it is dropped.

$$N_{replay} := H_1(K_{\text{message}})$$

We propose keeping a second secret hash of the symmetric secret (K_i) in a table indexed by a public value N_i:

$$K_i := H_2(K_{\text{message}}) \text{ and } N_i := H_3(K_i)$$

When a message goes though the mixing network it leaves behind it a sequence of keys and their indexes on each of the nodes it went past. Future packets can use these keys, in conjunction with secrets they carry, in order to decrypt both their addressing information and their payload. So in practice a node would first decrypt the headers of a message using its private decryption keys and then read the index of the key to be used N_j, and retrieve the appropriate secret K_j. It would then compute a secret shared key K_{final} based on both the key K_j and the secret K_{message} contained in the decrypted header:

$$K_{final} := H_4(K_{\text{message}}, K_j), K_l := H_2(K_{\text{final}}) \text{ and } N_l := H_3(K_l)$$

$$M_{n-1} \rightarrow M_n : \{S_{\text{message}}, N_j\}_{Pk_n}, \{A_{M_{n+1}}, \text{Message}\}_{K_{\text{final}}}$$

In order to make it impossible for an attacker to force decryption of this message or to gain any information by accessing the list of secrets K some key updating operations need to be performed, to replace the old keys by new values:

$$K_j := H_5(K_{\text{final}}) \text{ and } N_j := H_3(K_j)$$

As soon as the old values (N_j, K_j) are replaced by the newly generated (N_l, K_l) they must be permanently deleted.

To summarize, the H_1 function is used to extract a public tag of the message to avoid replays and is always applied to the secret contained in the message. The function H_2 is applied to the final symmetric secret in order to generate keys that can be used in future communications, and the function H_3 is applied to these keys to generate their public indexes. The function H_4 is used on the secrets stored in the mix and the packet to create final shared secrets while the function H_5 is used to update secrets. All functions $H_x()$ are secure hash functions, in particular because they have to be pre-image resistant, to avoid giving any information about the key K_i associated with an index $N_i := H_3(K_i)$ or the relation between new and old keys $K_j := H_5(K_{\text{final}})$.

There must be a special index N_0 indicating that the message does not depend on any existing key present on the server, and the message should in that case be processed in the traditional way. If messages refer to a key index that does not exist they should be dropped.

6.2 The cost of an *fs-mix*

The properties provided by *fs-mixes*, as described above, are not achieved for free. The next section will explore the security advantages of this new schemes but first we will discuss the additional expenses both in terms of storage and processing.

A traditional mix only needs to maintain a public record of the ID of all the messages it has ever processed under its current private key. Therefore the storage required after processing n messages is $\mathcal{O}(n)$. For each new message processed a lookup is performed on the public record. Assuming that the lookup table is implemented using a hash table one can expect a cost of $\mathcal{O}(\log n)$ to perform each lookup.

An fs-mix stores more state than a traditional mix. It will need to store m pairs of (N, K) values. Unlike n, the number of messages processed by the current private key of the server, m is proportional to the number of messages that have ever been processed by the mix. In particular it will be maximum if the forward secure functionality is never used by messages, since no pairs are ever deleted. As above the cost of finding a particular element should be proportional to $\mathcal{O}(\log m)$. In addition to the lookups as many as four hash operation might need to be performed per message.

In order to minimize the state needed by an fs-mix entries in the index and key table can be made to expire, either automatically or as requested by the sender. More details about this scheme will be presented in section 8.3.

7 Security Analysis

We shall argue that the modifications presented do not make the mix any weaker while providing additional security features. It is clear that if the keys K contained in every mix were public, an adversary would be exactly in the same position as in a traditional mix architecture. He would have to compel the mix to perform decryption using its private keys or surrender it, in order to trace any material that is in his possession. Therefore in that case the new scheme is equivalent to the traditional one.

7.1 Protection Against Compulsion

An attacker that tries to trace back a message that has already gone past an fs-mix cannot decrypt it unless all the messages upon which this communication's key depends were also intercepted and decrypted. Such an exercise would require all traffic to all the mixes to be logged and all of it to be decrypted in order to work, since the attacker does not have any a-priory knowledge of the message dependencies. The above is true for both sender-anonymous communications and single use reply blocks.

In case keys K are seized the first messages referring to them can be intercepted. The attacker then needs to be intercepting all subsequent messages in order to update his knowledge of keys to maintain his decryption capabilities. For each of these messages there must be a decryption request made to the mix (unless the private keys are seized).

Since not only the address is impossible to decrypt but the whole message, there is no way an opponent could try to use the body of the message in order to find out its destination. That means that even a single honest mix in a chain that implements a *forward secure anonymous* channel, that has genuinely deleted the used keys, is sufficient to order to provide forward security for the whole communication.

7.2 Traffic Analysis

Although the cryptographic security of the messages is stronger than in the traditional mixes, there is a new requirement imposed on the selection of routes that packets need to travel through. If there is a need for state to be present on the mix from previous packets, that means that the same mixes are chosen repetitively in a non random manner. That selection process might provide enough information for the adversary to be able to link messages between them.

Since the number of nodes applying the proposed scheme does not need to be very large, and provided that there is a small number of mixes, the traffic analysis of the route should be hard. This is the case because a particular node has a large probability of being present in two routes anyway. In the case of a large peer-to-peer mix scheme, the security of such protocols against traffic analysis has to be re-evaluated.

Additionally the intermediate mixes are aware of the link between messages, since the only party that knows the keys stored is the party that has sent the previous messages. They are also aware of two other nodes on the path that are seeded by the same messages (the one preceding them and the one after them).

It is worth noting that the messages used to distribute keys are identical to otherwise normal messages, as far as any outside observer or compromised mix is concerned. This is an essential feature, that makes the traffic selection task of extracting messages that contain key material extremely hard for the adversary. In many cases the key trail left behind normal messages could be used as key material for further messages.

7.3 Robustness

If the anonymity of a network cannot be broken, an attacker might chose to degrade its performance, hopping that it will put people off form using it. Therefore a large body of work concentrates on proving the correctness of mix networks [1, 8, 18, 17]. Unfortunately, the requirement upon fs-mix nodes to store additional state is making the transport of messages more fragile and the nodes more prone to denial of service attacks.

Given that messages can get dropped at random, due to traffic congestion or network failures, making future messages reliant on past ones could potentially make the network even less reliable overall. Fortunately one could think of strategies in order to ensure the proper delivery of special, key distribution messages, before making further messages dependent on the keys they distributed. One way of doing so could be to create a message finally addressed to oneself and check for its delivery before using the secrets distributed to route further messages. Unfortunately there are security implications, namely the risk of having the key distribution message intercepted and traced, as well as all further messages linked to it.

8 Additional Features

In addition to the mechanisms described above, having facilities to make messages dependent on keys on the nodes could be used to implement *interdependent reply blocks* and *path burning messages*.

8.1 Interdependent Reply Blocks

Without modifications to the mechanisms described above it is possible to make many reply blocks depend on each other, in such a way that once one of then has been used the other ones cannot be traced back. To do this a common mix is used in the path of all the reply blocks, that has been given a particular secret by a previous message. The SURBs are constructed in such a way that they all can only be decrypted by a single secret entry on the shared mix node. As soon as the first of the messages routed using one of the interdependent SURBs uses the key, the key updating operation takes place, and the messages using the other SURBs automatically get dropped. Furthermore it is impossible to trace any of them back.

8.2 Path Burning Messages

Much in the same way as above, it is possible to make reply blocks valid for only a limited amount of time. After a determined time period a message is constructed that uses the same intermediate secrets as the SURBs, and is fired into the mix network. This message updates the keys, and any subsequent messages depending on them will get dropped. The same techniques can be used to make reply blocks valid only after a particular time period by only providing the necessary keys at some future time.

8.3 Automatic Key Expiry Date

By adding a time stamp to each of the keys one can make sure that the intermediate nodes automatically delete the keying material. This would provide the same functionality as the Path Burning Messages without requiring the principals willing to maintain their anonymity to actively delete keys. One could consider the encryption/decryption key pair rotation that mixes should perform periodically to be providing some automatic key expiry anyway, and requiring the stored keys to expire at the same time.

Alternatively the expiry time could be specified by the user along with the key. This has the disadvantage that it could be used for traffic analysis if any logic is used to calculate this expiry date, that takes into account the current time, or other information local to the user.

9 Conclusions

We have shown in this paper that the traditional threat model of active and passive attackers and corrupt nodes does not fully capture the variety of attacks that todays mixes could come under. In particular compulsion to reveal keys or decrypt material could be used to trace communications going through them.

By adding some additional state in the nodes we manage to limit in many cases the value that an opponent can extract using her compulsion powers, and allow the network to regain its security in the absence of continuous and ubiquitous surveillance.

Acknowledgments I would like to thank my supervisor Dr R. Anderson and my colleagues Richard Clayton and Andrei Serjantov for reading the manuscript and providing valuable feedback. This paper has also been improved by the comments of the anonymous referees.

References

1. Masayuki Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Advances in Cryptology - EUROCRYPT 1998, LNCS Vol. 1403*. Springer-Verlag, 1998.
2. Paul C. Kocher Alan O. Freier, Philip Karlton. The SSL Protocol Version 3.0. `http://home.netscape.com/eng/ssl3/draft302.txt`.
3. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Designing Privacy Enhancing Technologies, LNCS Vol. 2009*, pages 115–129. Springer-Verlag, 2000.
4. Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In *Designing Privacy Enhancing Technologies, LNCS Vol. 2009*, pages 30–45. Springer-Verlag, 2000. `http://www.tik.ee.ethz.ch/~weiler/lehre/netsec/Unterlagen/anon/disadvantages_berthold.pdf`.
5. Electronic Privacy Information Center. Data retention. Internet Web Resource. `http://www.epic.org/privacy/intl/data_retention.html`.
6. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1982. `http://www.eskimo.com/~weidai/mix-net.txt`.
7. Temporary committee on the ECHELON interception system. Report on the echelon interception system. European Parliament Report A5-0264/2001 PAR1. `http://www.europarl.eu.int/committees/echelon_home.htm`.
8. Yvo Desmedt and Kaoru Kurosawa. How to break a practical MIX and design a new one. In *Advances in Cryptology - EUROCRYPT 2000, LNCS Vol. 1803*. Springer-Verlag, 2000. `http://citeseer.nj.nec.com/447709.html`.
9. Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. Proceedings of the Information Hiding Workshop 2001. `http://www.freehaven.net/papers.html`.
10. Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. Proceedings of Financial Cryptography 2002. `http://www.freehaven.net/papers.html`.
11. Foundation for Information Policy Research. Regulation of Investigatory Powers Information Centre. `http://www.fipr.org/rip/`.
12. Michael J. Freedman, Emil Sit, Josh Cates, and Robert Morris. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, Cambridge, MA, March 2002.
13. Electronic Frontiers Georgia (EFGA). Anonymous remailer information. `http://anon.efga.org/Remailers/`.
14. Nick Mathewson George Danezis, Roger Dingledine and David Hopwood. Mixminion: Design of a Type III Anonymous Remailer Protocol. Manuscript. `http://seul.org/~arma/minion-design.ps`.

15. C. Gulcu and G. Tsudik. Mixing E-mail with Babel. In *Network and Distributed Security Symposium - NDSS '96*. IEEE, 1996. `http://citeseer.nj.nec.com/2254.html`.
16. Johan Helsingius. Anon.penet.fi is closed! `http://www.penet.fi/`.
17. Markus Jakobsson. Flash Mixing. In *Principles of Distributed Computing - PODC '99*. ACM, 1999. `http://citeseer.nj.nec.com/jakobsson99flash.html`.
18. M. Mitomo and K. Kurosawa. Attack for Flash MIX. In *Advances in Cryptology - ASIACRYPT 2000, LNCS Vol. 1976*. Springer-Verlag, 2000. `http://citeseer.nj.nec.com/450148.html`.
19. Christian Mock. Mixmaster stats (Austria). `http://www.tahina.priv.at/~cm/stats/mlist2.html`.
20. Ulf Möller and Lance Cottrell. Mixmaster Protocol — Version 2. Unfinished draft, January 2000. `http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-mixmaster2-protocol-00.txt`.
21. Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. Information Hiding Workshop 2002.
22. Bob Sullivan. FBI software cracks encryption wall. MSNBC. `http://www.msnbc.com/news/660096.asp?cp1=1`.
23. M. Blaze R. Canetti J. Ioannidis A.D. Keromytis W. Aiello, S.M. Bellovin and O. Reingold. Just Fast Keying (JFK). Network Working Group, Internet Draft, draft-ietf-ipsec-jfk-04.txt.