BEYOND DISCOVERY®

THE PATH FROM RESEARCH TO HUMAN BENEFIT

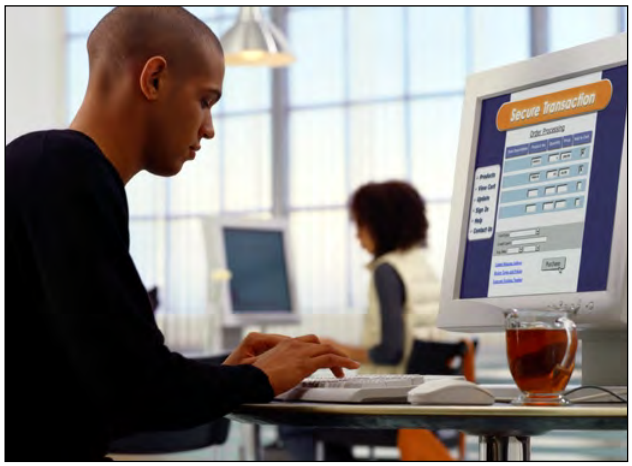
THE CODE WAR

Before the widespread use of computers and the Internet, cryptography—from the Greek kryptos *(hidden) and* graphein *(writing)—was largely the domain of the military or diplomats. Indeed, the earliest recorded instance of encryption dates to about 400 BC, when the Spartans used a device called a* scytale *to send coded messages between military commanders. A strip of parchment or leather was wrapped spirally around a baton or staff of a certain diameter. The sender wrote the message down the length of the staff, and then unwrapped the parchment, effectively scrambling the order of the letters. To decode the message, the recipient had to wrap the parchment around a staff of the same diameter, whereupon the transposed letters returned to their original order.*

In today's information age, we make use of data scrambling whenever we use a password to check e-mail, withdraw money from an automated bank teller machine, make a cellular phone call, or charge a purchase over the Internet. We rely on encryption to ensure the validity of our financial transactions, prove our identity, and safeguard our privacy. Although some people hesitate to conduct business over the Internet, most of us engage in online transactions with confidence that encryption protects our activities.

This faith is generally well founded. Many of the new methods of encrypting and decrypting information involve public-key cryptography, which was invented about 25 years ago. The security of many public-key systems (there are several kinds) is explicitly based on a long-standing challenge in the branch of mathematics known as number theory—the study of the properties and patterns of integers, whole numbers such as -2, -1, 0, 1, 2,…100,… Number theory has for centuries been widely regarded as the purest of the pure sciences, but in recent years it has found many applications.

Number theory has played an essential role in the development of public-key cryptography. Without the basic inquiry carried out by early theorists, today's computer transactions would be easy pickings for would-be thieves and swindlers. The number theorists' challenge to potential intruders is this: Given a (very, very large) number obtained by multiplying two other numbers, find the two (very large) numbers that were multiplied to produce it. These numbers can be thought of as the keys that lock and unlock the encryption code. The task of finding these keys is so difficult that the snoops must either be bizarrely lucky (as lucky, say, as one person winning 20 state lotteries simultaneously) or they must solve a problem that has stumped the smartest people in the world for more than 2000 years. Every time we conduct an encrypted transaction, we're betting that the snoops will lose.



Modern e-commerce is protected by reliable encryption techniques. Photo ©PhotoDisc

Early Ciphers

Encryption's history has been one of unceasing efforts to devise uncrackable codes—and equally unceasing efforts to crack them. Early ciphers were relatively simple systems, easy for both sender and receiver to use. Julius Caesar, for instance, encoded messages with a "substitution cipher" in which each letter is replaced by the third letter after it in the alphabet: A is replaced by D, B by E, etc. At the end of the alphabet, the pattern wraps around to the beginning: X becomes A, Y becomes B, and Z becomes C. Unfortunately, simplicity of use is a double-edged sword: The ciphertext thus encoded is highly susceptible to being decoded. Caesar's cipher can be cracked simply by moving each letter in the encoded message back three spaces in the alphabet. More sophisticated substitution ciphers, in which the alphabet is thoroughly scrambled, are nevertheless easy enough for amateurs to break, as fans of the "Cryptoquote" puzzles in today's magazines and newspapers can attest. In any sufficiently long passage of English text, the most common letter is usually "E," the second most common is "T," and a three-letter word that appears repeatedly is probably "THE." By applying this type of "frequency analysis," an eavesdropper can easily guess which letters in the ciphertext represent "E," "T" and so on.

Over the years, people who wanted greater secrecy came up with more elaborate coding schemes. In the 1500s, Blaise de Vigenère, a French diplomat, invented a method for encrypting different letters in a message with different ciphers. Thus, an "E" in one position might be coded as "M," while an "E" in another position might be coded as "K," thereby foiling anyone attempting to decode the message using frequency analysis.

In the Vigenère cipher, the sender and recipient had to agree on a keyword (or perhaps a literary passage) whose letters told them how far forward or backward to shift the alphabet for every letter in the message. If the keyword "BIG" was used for example, the sender would code the message in sets of three letters. The first letter of the first trio would need to be shifted forward by one (since "B" is one letter after "A"), the second letter would need to be shifted forward by eight ("I" is eight letters after "A"), and the third letter would need to be shifted forward by six ("G" is six letters after "A"). After that, the pattern would repeat itself as in the following example:

Plaintext:	THE BUTCHER THE BAKER AND THE CANDLESTICK MAKER
Key:	BIG BIGBIGB IGB IGBIG BIG BIG BIGBIGBIGBI GBIGB
Ciphertext:	UPK CCZDPKS BNF JGLMX BVJ UPK DITETKTBODS SBSKS

Knowing that "BIG" was the key, the recipient could easily decipher the message by shifting its letters back the corresponding amounts.

For many years Vigenère's cipher was considered unbreakable, but Charles Babbage, an independently wealthy Englishman known mostly for his pioneering work in computer science, showed in the 1850s that it was not so. Babbage hacked the system by looking for repeated strings of letters. Of course, the strength of Vigenère's cipher was supposed to be that it encoded letters differently in different places. The first "THE" in the message above is rendered as "UPK" and the second as "BNF". Also, the two "AKER"s code differently. But the first and third "THE"s both code as "UPK." The "T" in the first "THE" is coded with a "B," and so is the "T" in the third "THE." This happens because the third "THE" begins 21 letters after the first "THE"; hence the 3-letter keyword "BIG" has cycled around 7 times and is back to the beginning again.

In any message that is much longer than the key, some repeats of this sort are bound to occur. How would an eavesdropper exploit this fact? If, say, the ciphertext "UPK" appeared twice, 21 letters apart, then he could deduce that 21 was probably a multiple of the keyword's length. Or to put it another way, the number of letters in the keyword was a divisor of 21. (A *divisor* or *factor* of a number is a number that goes into it with no remainder. The divisors of 21 are 1, 3, 7, and 21.)

Given enough clues of this sort, an eavesdropper could pin down the exact length of the keyword. Once he knew the length, he could do ordinary frequency analysis to decode the message. Notice that the math comes first: The eavesdropper figures out the length of the keyword before even attempting to figure out what its letters are.

Babbage's ingenious technique broke new ground in cryptography by introducing mathematical tools to a subject that previously had seemed to be about words. Even if an encryption system does not use mathematics explicitly, its hidden patterns can often be teased out that way. Mathematics *is*, after all, the science of patterns.

The Enigma Challenge

Although perhaps not fully appreciated, mathematical decryption techniques made a huge contribution to the Allied victory in World War II. In that war, Germany encrypted most of its military transmissions with a machine called "Enigma." Part electrical, part mechanical, it was like a combination lock with more than 10^{23} possible combinations. (For comparison, this is roughly the number of tablespoons of water in all the world's oceans.) Moreover, the Germans changed the combination every day—sometimes several times a day. Recipients of the transmissions needed to possess not only a duplicate Enigma machine, but also to know the correct combination.

If the Allies had had to rely solely on frequency analysis or trial and error, they would still be hunting for that one tablespoon of water in an ocean of possibilities. However, thanks in large part to crucial earlier work by Polish cryptographers and mathematicians, a team of British codebreakers led by mathematician Alan Turing found a shortcut that eliminated almost all of the trial and error for finding the current combination. Now it was more like hunting for one particular tablespoon of water in a small wading pool. Turing's solution exploited both the mathematical structure of the Enigma machine and certain regularities in the German transmissions, such as their punctual release each morning of a weather bulletin containing the word "Wetter" (the German word for "weather").

As this episode shows, *complexity* is no guarantee of security. The most elaborate cryptosystem in the world can be broken if it contains hidden patterns, or if its users unintentionally introduce patterns (such as the weather bulletins). To be truly unbreakable, a cipher would have to be pattern-free. Imagine, for example, a Vigenère cipher whose key is an endless string of randomly generated letters. Such a method has actually been used: It is called a "one-time pad," because the sender and recipient typically store the key on identical pads of paper, use each page once, tear it off, and never use it again.

However, even though the one-time pad offers the ultimate in security, it dismally fails a second important criterion for a successful code: It is not easy to use. For the method to work the sender somehow has to deliver to the recipient—in a secure fashion—a key pad that is longer than the messages to be sent. This might just be feasible for messages to a single spy, but it would never be practical for widespread military or commercial use. The "key distribution problem," as this difficulty is known, would remain an obstacle until the latter part of the twentieth century, when mathematics once again came to the rescue.

In Cryptography We Trust

All encryption systems invented before 1970 had one thing in common: They were *symmetric*. In other words, the keys for encryption and decryption were the same, so a person in possession of the key could either send or receive messages. But in the early 1970s, Whitfield Diffie of Stanford University realized that for some applications this two-way capability was superfluous. If the message traffic is one-way, then the encoding and decoding keys can be different, thus providing an extra level of security.

Together with Martin Hellman of Stanford, Diffie sketched out how such a system would work. The magic ingredient was a "one-way function," a mathematical operation that is easy to do in one direction but virtually impossible to reverse without additional information. First, each message recipient (say, Alice) chooses a "private key" that she will use to decode messages. (The system should offer a huge number of possibilities, so that Alice can pick a key more or less at random.) Then she uses the "one-way function" to work out the corresponding encoding key. This is a "public key," which she can share with the whole world, and anyone can use it to send an encrypted message to her. However, only Alice can decrypt the message. There is only one decrypting key that will work (her private key), and no one else can figure it out because that would require them to reverse the one-way function.

This simple realization, that cryptosystems did not have to be symmetric, led to a new era in cryptography. It took away the cloak and dagger. *Anyone* can use public-key cryptography, not just spies and spy agencies. As Diffie has pointed out, in these systems you don't need to trust or even know the people you are communicating with; you only need to trust the system itself. This makes public-key cryptography perfect for the world of electronic commerce.

Figure 1 shows how public-key cryptography works in a typical commercial situation. Alice, a banking customer, wants to instruct Bob, her banker,

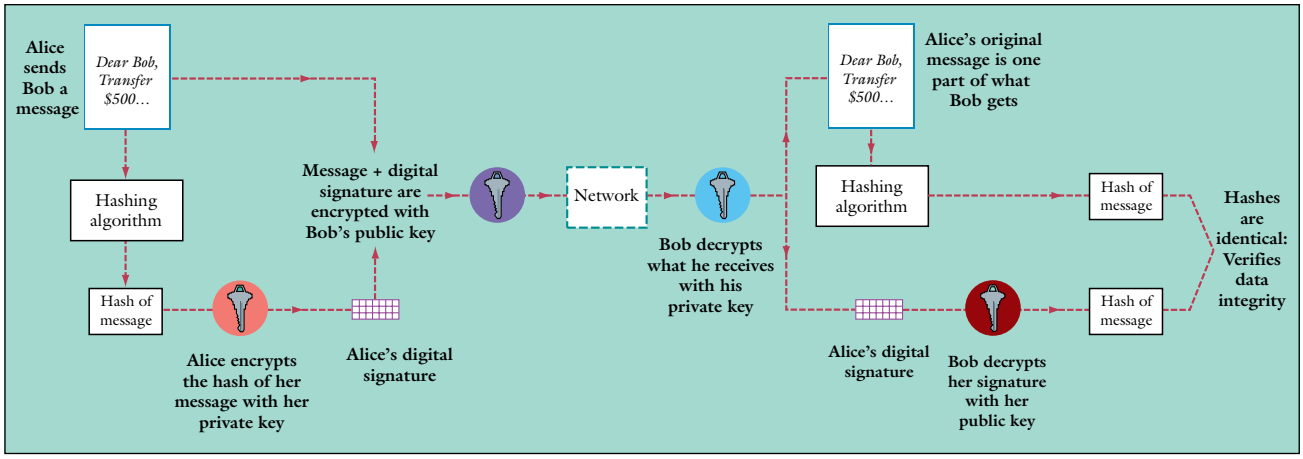


Figure 1: A schematic illustration of public-key encryption, including a digital signature. The figure shows how public-key cryptography might work in a typical commercial situation.

to transfer funds from one of her accounts to another. She scrambles the message using *Bob's* public key before sending it to him. Because Bob is the only person with the private key capable of reversing that function, Alice knows that no eavesdropper can read her message.

But how does Bob know the message really came from Alice? Before Alice encrypts her message to Bob for transmission, she creates an ingenious construct called a "digital signature." To do this she first uses a second one-way mathematical function called a "hash" to scramble and greatly shorten the message she intends to send, including the date. (The hash is different from her private key; it is a cipher without a back door, which no one, even Alice, can unscramble.) She then encrypts this already scrambled "hashed" message, this time using her private key. The final product of this two-step process is Alice's digital signature. She appends what is now a string of gibberish (her digital signature) to her plaintext message. Finally, she encrypts the whole thing with Bob's public key and sends it to him electronically.

Bob decrypts Alice's message using his private key. At the end of the message ("Please transfer $500

"The Code War" — Timeline

As long as there have been secrets, there have been codes designed to preserve them and eavesdroppers interested in breaking those codes. In this century encryption methods have increasingly been based on sophisticated mathematics—particularly number theory, a specialty previously noted for its lack of practical applications. Thousands of lives have been saved and billions of dollars in electronic commerce made possible by advances in the theory of encryption.

Circa 300 BC
Euclid composes *The Elements*. Three of its thirteen books are devoted to number theory, introducing such fundamental concepts as divisibility, prime numbers, and composite numbers.

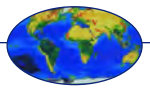
58-51 BC
Julius Caesar conquers Gaul. His book on the Gallic Wars contains the first documented use of encrypted messages.

1586
Blaise de Vigenère, a French diplomat, develops the first polyalphabetic cipher, in which letters may be encoded differently depending on their position in the document.

1640
French mathematician Pierre de Fermat discovers "Fermat's little theorem," which is still used to test large numbers for primality, even though it is not infallible.

1801
German mathematician Carl Friedrich Gauss publishes *Disquisitiones Arithmeticae*, the founding document of modern number theory. He is the first to appreciate the power of modular arithmetic, which greatly clarifies the somewhat mysterious results of Fermat.

1940
Relying on earlier work by Polish mathematicians and cryptographers, British mathematician Alan Turing cracks the Enigma cipher. The ability of Western commanders to decipher secret German messages hastened the Allied victory in World War II.



from my savings account to my checking account."), Bob sees a string of gibberish—Alice's digital signature—and knows that if the message really came from Alice he can reverse the effects of Alice's private key by applying her public key to the signature. What then pops out is the gibberish "hash" of the message. As the hash is a one-way function, for all practical purposes, it is impossible to extract any meaning from the hashed data. But Bob can also hash the message he has already decrypted, using the same hashing function Alice used, and produce his own string of gibberish. If the two strings agree, he knows the message that he decrypted is authentic and came from Alice. The "double gobbledygook" nature of Alice's digital signature prevents Bob from forging Alice's signature in the future, even though he can produce the same gibberish she did. Since her digital signature is inextricably connected to this particular message and the time it was sent, knowing how to produce it will not help Bob at all if he tries to pass himself off as Alice on another occasion.

Of course, Bob and Alice do not need to be mathematical wizards to do all this; it can all be automated in Alice's ATM card and Bob's computer. Similar exchanges of information now go on all the time even in places you would never expect.

It is worth noting that conventional "symmetric" cryptography has not been made obsolete by public-key cryptography. In fact, the two usually work hand in hand. A common application of public-key encryption occurs today on secure websites using just such a combination. When a secure session is initiated between two computers, one of them creates a symmetric key, encodes it via public-key encryption, and sends it to the other. They then use symmetric-key encryption for the rest of the session, because this is faster. After each session is completed the symmetric key is discarded and a new one is generated for the next session.

The Method Behind the Magic

As mentioned above, public-key cryptosystems rely on one magic ingredient: the one-way function. At the time that Diffie and Hellman wrote their first paper they did not have any particular function in mind. Shortly thereafter Ralph Merkle, a student of Hellman's, proposed one, which eventually proved to be unsatisfactory because it was not as hard to reverse as it initially appeared. It was left to another troika of mathematicians to invent a one-way function that was both simple and resistant to attack, and their invention remains the most popular public-key system to this day.

1976
Whitfield Diffie, Martin Hellman and Ralph Merkle propose a new approach to cryptography in which the encryption and decryption keys are different. This launches the era of public-key cryptography. They were unaware that James Ellis of British intelligence had already come up with the same idea but had to keep it secret.

1977
Ronald Rivest, Adi Shamir, and Leonard Adleman invent the RSA encryption algorithm, a public-key system whose security depends on the difficulty of factoring large numbers. They publicly challenge anybody to decode a message encoded with a 129-digit number.

1981
Carl Pomerance develops the "quadratic sieve" method, which allows large factorization problems to be parceled out to many computers working in parallel.

1988
The "number field sieve" method is invented by John Pollard.

1994
Rivest, Shamir, and Adleman's 1977 message is decoded by a team of hundreds of computers using the quadratic sieve method.

1994
Peter Shor of AT&T Research develops a "quick" (i.e. polynomial-time) factoring algorithm that would work on a quantum computer. However, it remains uncertain whether such a computer can ever be built.

1999
A 155 digit RSA challenge number is factorized by a group of researchers using the general number field sieve method.

2002
Manindra Agrawal, Neeraj Kayal, and Nitin Saxena develop a polynomial-time testing algorithm for prime numbers that works on ordinary computers. It relies on an ingenious modification of Fermat's little theorem.

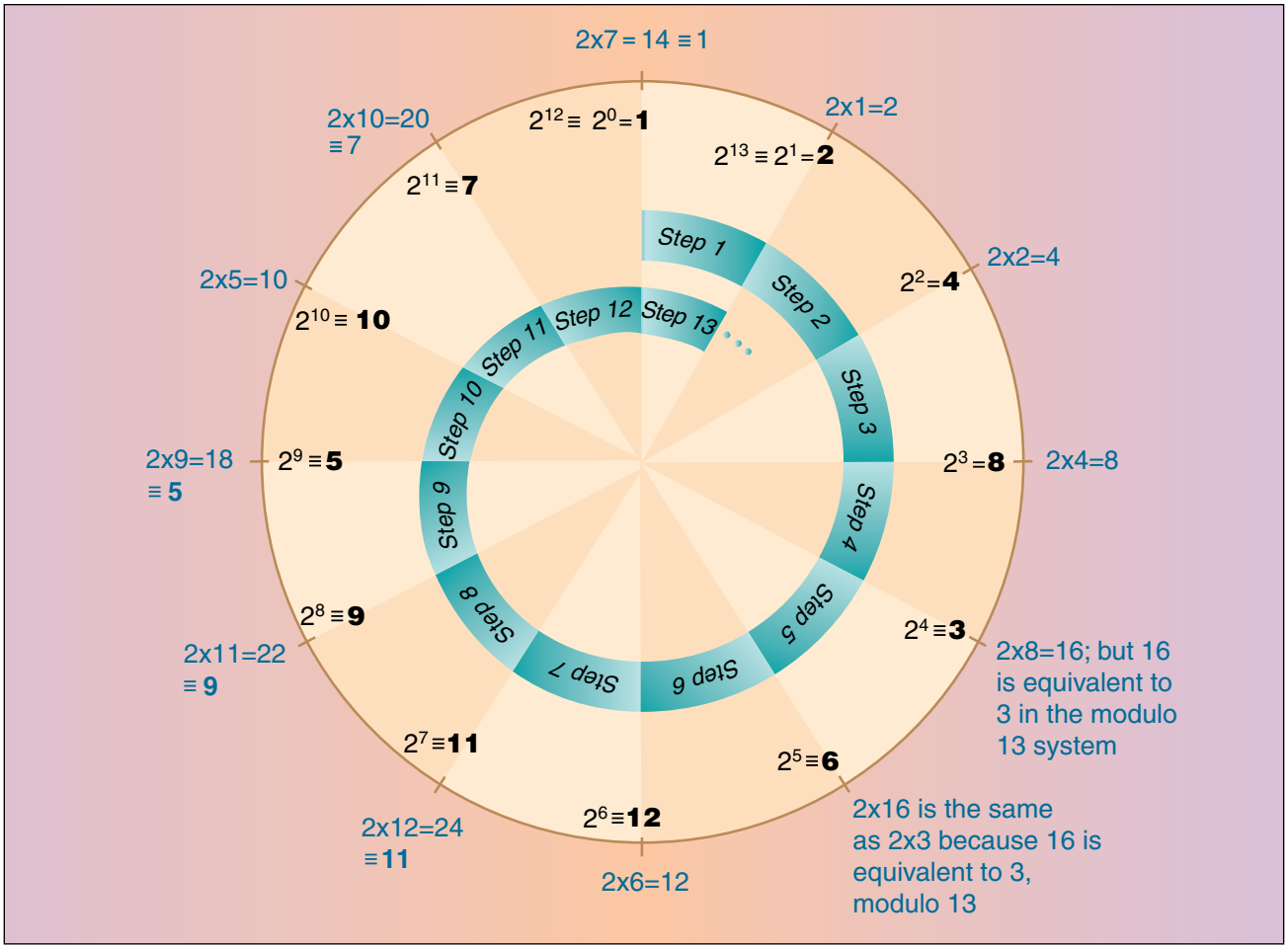


Figure 2: The figure illustrates the "circularity" of multiplication modulo a prime number by showing the computation of 2^{13} modulo 13. The multiples of 2, modulo 13, (2^0, 2^1, 2^2, … 2^{13}) are arranged around a clock face in the following order: 1 (in the 12:00 position), 2 (in the 1:00 position), 4 (in the 2:00 position) and so on. Each number is obtained by doubling the previous number and then reducing it modulo 13. The "≡" symbol denotes equivalence or "congruence" modulo 13. Note that thanks to Fermat's little theorem, each time you take 13 steps around this clock face, you end up in the same place you started.

The RSA cryptosystem, devised in 1977 by Ronald Rivest, Adi Shamir, and Leonard Adleman, rests on the idea that multiplying numbers is easy, but finding their divisors is hard. To make things as hard as possible for the computer, you should make sure your initial numbers are *prime* numbers. These are the numbers, such as 2, 3, 5, 7, and so forth that have no divisors aside from themselves and 1. Any desktop computer can multiply two 150-digit numbers together and print out the 300-digit result in a fraction of a second. But if you feed that 300-digit number to the biggest and fastest computer in the world, it will be unable to discover the two 150-digit numbers that you used to create it.

Thus, the simple act of multiplying two prime numbers together has all the hallmarks of a one-way function: It is easy to do, and hard to undo. But how do you tranform it into a public-key cryptosystem? The answer is far from obvious, and a great tribute to the ingenuity of Rivest, Shamir, and Adleman.

Their system exploits a subtle difference between prime numbers and composite (non-prime) numbers that was first noticed around 1640 by the French mathematician Pierre de Fermat. Suppose you pick any number, n, and any other number a that is smaller than n. (As we shall see later, in the RSA system, n is a public key and a is the plaintext; but for Fermat they were just numbers.) Now multiply a by itself, over and over. To keep the numbers from getting too big, at each step divide by n and take the remainder. (This is called "reduction modulo n.") Figure 2 shows what happens with $n = 13$ and $a = 2$: The sequence starts out 1, 2, 4, 8, 3 (the sequence always starts with 1 and is then multiplied by a repeatedly; here, the sequence goes as 1, 1×2=2, 2×2=4, 4×2=8, 8×2=16—which reduces to

3 modulo 13—and so on), and after 13 steps, it comes back to 2 again.

According to Fermat, this is no accident. When n is prime, the repeated-multiplication trick always cycles back to its starting point after n steps. (Mathematicians say this as follows: a^n is congruent to a modulo n. The symbol a^n refers to a multiplied by itself n times, or a to the n-th power.) But when n is not prime, the number of steps needed to cycle around is usually *not* equal to n. Predicting just how many steps it will take is hard: To do it, you need to know what the divisors of n are. (Remember that whenever n is not prime, it will have divisors.)

Until the 1970s, number theorists had never suspected that this repeated-multiplication trick (called "Fermat's little theorem," to distinguish it from the more famous "Fermat's last theorem") could be used for cryptography. Rivest, Shamir, and Adleman's unprecedented insight was this: If the number a is thought of as a *message,* then multiplying it repeatedly, say e times by itself (i.e., raising it to the e-th power) and then reducing modulo n is a way of *scrambling* the message. To unscramble the ciphertext, you don't have to reverse the process: Instead, you just keep on multiplying a^e by itself! After some additional number of steps (say, d steps), the message will magically pop out again.

Figure 3 shows in detail how Rivest, Shamir, and Adleman incorporated this idea into a cryptosystem. Remember that, in public-key cryptography, message recipients are responsible for generating their own public and private keys. First, Bob chooses two very large prime numbers p and q, say 150 digits long each, for his private key. Aside from the restriction that they are prime, they can be chosen completely arbitrarily. His public key consists of n, which is p times q (and is therefore not prime), and e, the encoding key, which must satisfy a few mild conditions relating to the factors of n. These conditions do not pose any difficulty for Bob, because he knows p and q. Finally he computes the unique decoding key d that will work in the manner described in the last paragraph. That is, when any number a is multiplied by itself e times modulo n, and the result (a^e) is multiplied by itself d more times modulo n, the original number a results. This number d also becomes part of the private key, and can **only** be

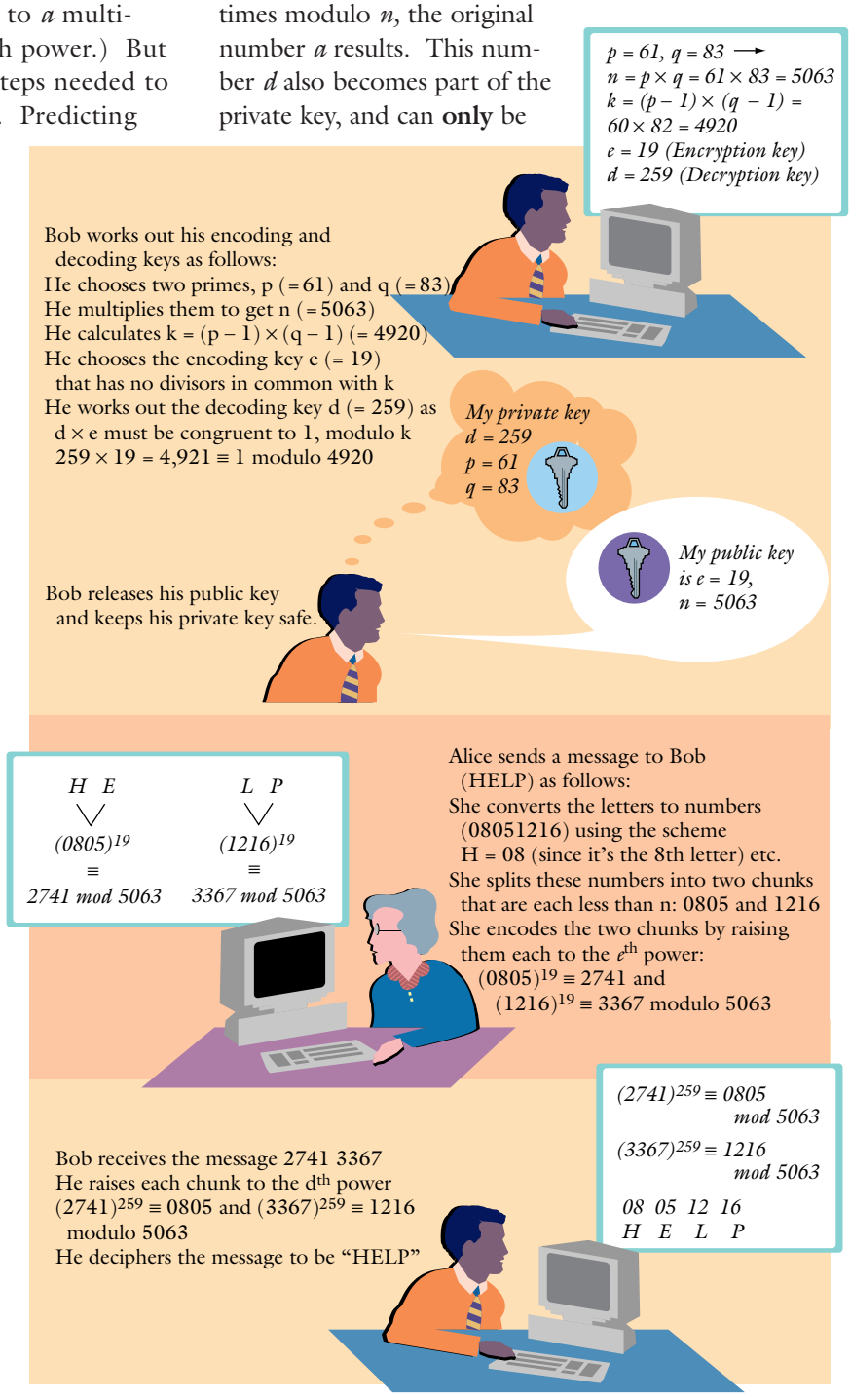


Figure 3: A schematic illustration of the RSA system of encryption, worked out using primes p = 61 *and* q = 83. *The seemingly enormous calculations involved in multiplying numbers by themselves 19 and 259 times can actually be done quite easily, as illustrated in the figure. The "≡" symbol denotes equivalence in the modular system.*

computed by someone who is privy to the secret values of p and q. These calculations are shown in detail in figure 3.

To send Bob the message "HELP" using Bob's public key, Alice first converts the letters into numbers using a scheme such as A = 1, B = 2, etc. (Punctuation and spaces can also be converted, for example by using the standard ASCII codes that are used by computer word processors. This is not part of the RSA cryptosystem *per se*.) If necessary, she splits the complete text of the converted message into chunks that are no larger than n. Then she multiplies each chunk by itself e times and reduces it modulo n. The resulting number is the ciphertext. To recover the original message, Bob multiplies the ciphertext by itself d more times and reduces it modulo n.

Now consider the predicament of an eavesdropper whom we will call Eve. If n were a prime number, Fermat's little theorem would tell her exactly how many multiplications would unscramble Alice's message. But because n is not prime, she needs to know its divisors to figure that out (since the private key d is derived from the divisors of n as shown in figure 3) – and that would force her to reverse a one-way function. Nor can she undo Alice's e multiplications, because undoing multiplications, even modulo n, is difficult. She can't even use trial and error, multiplying the ciphertext by itself until a coherent message pops out, because in practice d is too large for that to work.

Computation of 2^{64} *modulo* 14879

$2^1 = 2$

$2^2 = 4$

$2^4 = 4^2 = 16$

(at each step, we are squaring the result of the previous step)

$2^8 = 16^2 = 256$

$2^{16} = 256^2 = 65536 \equiv 6020$ *modulo* 14879

(The number 65536 is reduced to a smaller number due to the modular arithmetic. Each time we do an operation, we replace the result by the remainder when 14879 divides that result. Here, 6020 replaces 65536 since 14879 goes into 65536 four times, with a remainder of 6020.)

$2^{32} = 6020^2 = 36240400 \equiv 10035$ *modulo* 14879

(14879 divides 36240400 a total of 2435 times, with a remainder of 10035; hence a congruent 5-digit number replaces the 8-digit result.)

$2^{64} = 10035^2 = 100701225 \equiv 153$ *modulo* 14879

(14879 divides 100701225 a total of 6768 times, with a remainder of 153; hence the 9-digit number gets compressed to 3 digits.)

Figure 4: A simple example to demonstrate the power of modular arithmetic. By using only the remainders of numbers when they are divided by n *(in this example,* n *= 14879), seemingly difficult calculations (in this example, multiplying 2 by itself 64 times) become tractable.*

Figure 4 shows how modular computation works. In this example, Bob knows that $d = 64$ and 2^{64} will yield the message. He can take a short cut while Eve has a much more laborious job. By repeated squaring, Bob finishes the calculation after only 6 multiplications as shown in the figure. But Eve, who has to check each power of 2 to see if she gets a message or only gibberish, would need to do one multiplication at a time until she gets to 2^{64} and will hence take 64 steps. And Bob's advantage over Eve grows bigger and bigger as d gets larger. So Eve is stuck, and Alice's secret is safe.

The RSA technique is vulnerable when the sender encrypts precisely the same message (e.g., "SELL") more than once, resulting in the same ciphertext. An eavesdropper could note this and, perhaps, guess the message content even without figuring out the key. Thus, in practice, the message is usually scrambled first by a fast symmetric-key cipher as described in the example of secure websites. This will produce a different ciphertext each time because the symmetric key will change.

How Safe is Safe?

One of the great virtues of the RSA technique is its adaptability. If you want more security, you can simply use larger primes p and q to create the public key n. Prime numbers are not only fairly common but there is a limitless supply of them; so you'll never run out.

Indeed, key length is the parameter that governs the security of the RSA cryptosystem, and all similar mathematical systems. In 1977, when Rivest, Shamir, and Adleman announced their system, they posed a famous challenge in *Scientific American,* offering $100 to anyone who could decode a message that was encoded using a 129-digit key. (Their number was n

= 114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541. Can *you* find its two prime factors?) With the factoring methods and computer technology then available, the three codemakers estimated that it would take someone 40 quadrillion years to break the cipher.

This was like waving a red flag in front of a bull. In the end it took only 17 years, accompanied by tremendous advances in computer technology and factoring algorithms, for persistent mathematicians and computer scientists to decode the message. Led by a group of experts, a team of more than 600 volunteers in two dozen countries, collaborating over the Internet, factored the RSA 129-digit key in 1994. The team used a new factoring algorithm called the "quadratic sieve," invented in 1981 by Carl Pomerance, now at Bell Labs, which has the ability to distribute the computation among many computers.

Even though the message was decoded (it read: "The magic words are squeamish ossifrage."), the three cryptologists had proved their larger point. Despite knowing exactly how RSA worked, the experts needed a huge investment of time, widespread use of the Internet, and the development of new mathematical methods to crack it. RSA-129 was broken, but RSA itself was still secure. RSA factoring challenges are still mounted periodically. For example, one of the RSA 155-digit keys was factored in 1999 using the "number field sieve" invented by John Pollard. This shows that users must use longer keys as technology improves—perhaps 300 digits instead of 129 or 155. There is only one development that would seriously threaten the security of RSA itself: a breakthrough in the time it takes to split a number into its prime factors.

At present, it is much easier to determine whether a number is prime than it is to find the divisors of a composite number. As noted above, Fermat's little theorem can be used as a sort of litmus test for primality. If a number n fails the test—that is, if some smaller number a, when multiplied by itself n times and then reduced modulo n, does *not* yield the original number a back again—then you are guaranteed that n is composite. (Even though you have no clue what its divisors are!)

Unfortunately, this test is not quite so foolproof in the reverse direction. A few composite numbers n do manage to pass Fermat's test for primality. (The smallest such "pseudoprime" is 561, which is $3\times11\times17$.) In recent years, mathematicians have come closer and closer to eliminating this loophole. And in 2002 three computer scientists—two of them undergraduate students—finally finished the job. Manindra Agrawal, Neeraj Kayal, and Nitin Saxena of the Indian Institute of Technology in Kanpur, India, astounded the mathematical world when they discovered an improved version of Fermat's test that has no exceptions. More than that, they demonstrated that their method could be programmed to run quickly on a computer. No previous primality test had met this double "gold standard" of guaranteed correctness and practicability.

Some of the initial publicity over this new primality test suggested that it might weaken the RSA cryptosystem. In fact, precisely the opposite is the case. The RSA system depends on the validity of two assertions: Finding prime numbers must be easy (otherwise Bob would never be able to generate his keys), but finding the prime divisors of composite numbers must be hard (otherwise Eve would be able to read Bob's mail). Thanks to Agrawal, Kayal, and Saxena, the first of these two statements can now be made confidently, without any hemming and hawing about pseudoprimes. But their work has no effect on the second assertion. The only grounds for worry are psychological: If mathematicians missed the Agrawal et al. primality test for so long, perhaps they could also be missing an easy factorization method.

Facing the Future

It scarcely qualifies as easy, but scientists have found a factorization method that may one day spoil the RSA cryptosystem. This method assumes that, one day, physicists will be able to build a quantum computer—a computer that, unlike the computers of today, would work not according to the digital logic we are used to, but would rely on quantum mechanical principles to carry out a huge number of operations simultaneously, that is, in parallel. The laws of quantum physics would make these computers behave very differently from classical ones. A circuit in a classical computer is either on or off, representing a bit of data that is either 1 or 0. But in a quantum computer, the particles can exist in many states at once; or in the language used by some physicists, they exist in many different universes. In effect, all those computers in the parallel universes could be put to work factoring large numbers very quickly—a boon for codebreakers.

The catch is that it will be formidably difficult to make a quantum computer. At present, no one knows how to control large numbers of subatomic

particles with sufficient precision. According to even the most optimistic estimates, quantum computers are still decades away.

But supposing a quantum computer does someday become a reality, does that mean that no secrets will ever be safe again? Hardly. Other public-key algorithms use different "one-way functions" that are not known to be reversible by a quantum computer. Presumably one of these would step into the breach if RSA lost its luster. But more fundamentally, if physicists learn to control quantum states well enough to build a computer, they will also be able to control them well enough to create a new kind of cryptosystem. This "quantum cryptography" has already been demonstrated in principle. Thanks to the Heisenberg Uncertainty Principle, which says that just observing a particle alters its state, any eavesdropper reading a specially quantum-encrypted message would alter the message, thereby alerting both sender and receiver that the message had been tampered with.

Whatever the future may bring, cryptography has moved past the era of clever gadgets, into an era when the security of encoded messages will be protected by the most fundamental principles in science —either the structure of our number system or the subatomic architecture of our universe. The more we can learn to decode nature's secrets, it seems, the better we will be able to guard our own.

This article, which was published in 2003 and has not been updated or revised, was written by science writer Dana Mackenzie—with the assistance of Drs. Ronald Graham, Arjen Lenstra, Barry Mazur, AndrewOdlyzko, Carl Pomerance, Kenneth Ribet and Mr. Moses Liskov—for Beyond Discovery®: The Path from Research to Human Benefit, a project of the National Academy of Sciences.

The Academy, located in Washington, D.C., is a society of distinguished scholars engaged in scientific and engineering research, dedicated to the use of science and technology for the public welfare. For more than a century, it has provided independent, objective scientific advice to the nation.

Funding for this article was provided by the National Academy of Sciences.

 February 2003