CRYPTOGRAPHY

Cryptography is the science and study of the security aspects of communications and data in the presence of a malicious adversary. Cryptanalysis is the study of methods used to break cryptosystems. Cryptographic schemes and protocols are being and have been developed to protect data. Until 1974, only privacy issues were studied, and the main users were diplomats and the military (1). Systems are also being deployed to guarantee integrity of data, as well as different aspects of authenticity and to identify individuals or computers (called entity authenticity). Emerging topics of study include anonymity and traceability, authorized wiretapping (called law enforcement), copyright, digital contracts, freedom of speech, revocation of rights, timestamping, witnessing, etc. Related disciplines are computer security, network security, physical security (including tempest), spread spectrum, and steganography.

Fast computers and advances in telecommunications have made high-speed, global, widespread computer networks possible, in particular the Internet, which is an open network. It has increased the access to databases, such as the open World Wide Web. To decrease communication cost and to be user-friendly, private databases containing medical records, proprietary information, tax information, etc., are often accessible via the Internet by using a low-security password scheme.

The privacy of data is obviously vulnerable during communication, and data in transit can be modified, in particular in open networks. Because of the lack of secure computers, such concerns extend to stored data. Data communicated and/or accessible over such networks include bank and other financial transactions, love letters, medical records, proprietary information, etc., whose privacy must be protected. The authenticity of (the data in) contracts, databases, electronic commerce, etc. must be protected against modifications by an outsider or by one of the parties involved in the transaction. Modern cryptography provides the means to address these issues.

FUNDAMENTALS

To protect data, one needs to know what type of attacks the untrusted party (enemy) can use. These depend on the security needs. The two main goals of modern cryptography are privacy and authenticity. The issue of protecting privacy is discussed now.

Privacy

The threat undermining privacy is eavesdropping. The untrusted party, called the eavesdropper, will have access to the transmitted or stored data, for example, by tapping the line or capturing (even rather minimal) electromagnetic interference from a screen. To protect the data, called the plaintext or cleartext, it is transformed into ciphertext. This transformation is called encryption. To achieve security, it should be difficult for the eavesdropper to cryptanalyze, that is, to recover the plaintext from the ciphertext. However, to guarantee usefulness, the legitimate receiver should be able to recover the plaintext. Such an operation is called decryption and uses a key k. To guarantee that only the legitimate receiver is able to decrypt, obviously this key must remain secret. If the sender wants to send data to different receivers, the encryption algorithm must use a parameter k', specifying the receiver, as extra input. For historical reasons this parameter has been called a (encryption) key, which is discussed in more detail later.

The person who attempts a cryptanalysis, called a cryptanalyst, may in some circumstances know a previously encrypted plaintext when trying to break the current ciphertext. Such an attack is called a known-plaintext attack, distinguishing it from the more basic ciphertext-only attack in which only the ciphertext is available to the cryptanalyst. Even more powerful attacks, especially in the commercial world, are feasible, such as a chosen-plaintext attack, in which the cryptanalyst chooses one (or more) plaintext(s). A company achieves this by sending a ciphertext to a local branch of a competing company that will most likely send the corresponding plaintext to its headquarters and encrypt it with a key the first party wants to break (1). In a variant of this type of attack the cryptanalyst sends a chosen ciphertext to the receiver. The plaintext is likely to be garbled and thrown in the bin. If the garbage collectors collaborate with the cryptanalyst, the latter has started a chosen-ciphertext attack. In the strongest subtype of chosen-text attacks the text chosen may depend on (previous or) other texts, and therefore it is called adaptive.

Authenticity

A document is authentic if it originated from the claimed source and if its content has not been modified. So, now the cryptanalyst is allowed to try to inject fraudulent messages and attempt to alter the data. Therefore one calls the cryptanalyst an active eavesdropper. To protect the data, one appends a message authentication code, abbreviated as MAC. If there is no concern for privacy, the message itself is sent in the clear. Only the legitimate sender should be allowed to generate a MAC. Therefore the sender needs to know a secret key k. If the key were not secret, anybody could impersonate the sender. So, the authenticator generation algorithm has the message and the sender's secret key as input. To check the authenticity of a message, the receiver runs a verification algorithm. If the algorithm's outputs "false," then the message is definitely not authentic and must be rejected and discarded. If the output is "satisfactory," very likely the message is authentic and is accepted. One cannot give a 100% guarantee that the message is authentic because the active eavesdropper could be very lucky, but one can approach the 100% margin as closely as desired. If the receiver wants to verify the authenticity of messages originating from different senders, the verification algorithm must use a parameter k', specifying the sender, as extra input. For historical reasons this parameter has been called a key, which is discussed in more detail later.

In all types of attacks the active eavesdropper is allowed to see one (or more) authenticated message(s). In chosen-text attacks, the cryptanalyst can choose a text which the sender will authenticate and/or send messages with a (fictitious) MAC(s). In the latter case, it is assumed that the active eavesdropper can find out whether the message was accepted or rejected.

Public Key Systems

One can wonder whether k' must remain secret, which is discussed now. If it is easy to compute k from k', it is obvious that k' must also remain secret. Then the key must be unique to a sender–receiver pair. This introduces a key management problem, since this key has to be transmitted in a secure way. In this case, the cryptosystem is called a conventional or symmetric cryptosystem and k, k' usually coincide.

On the other hand, if it is hard to compute k from k' and hard to compute a $\bar{k}$, which allows partial cryptanalysis, then the key k' can be made public. This concept was invented by Diffie and Hellman (2) and independently by Merkle (3). Such a system is called a public key (or sometimes an asymmetric cryptosystem). This means that for privacy protection each receiver R publishes a personal k_R, and for authentication, the sender S makes k_S public. In the latter case the obtained authenticator is called a digital signature because anyone who knows the correct public key k_S can verify the correctness.

Note that the sender can claim that the secret key was stolen or that k_S was published without consent. That would allow a denial of ever having sent a message (4). Such situations must be dealt with by an authorized organization. If high security is desired, the MAC of the message must be deposited with a notary public. Another solution is digital time stamping (5) based on cryptography (the signer needs to alert an authority that his public key must have been stolen or lost).

If the public key is not authentic, the one who created the fake public key can decrypt messages intended for the legitimate receiver or can sign claiming to be the sender (6). So then the security is lost. In practice, this problem is solved as follows. A known trusted entity(ies), for example, an authority, certifies that the key K_S corresponds to S, and therefore signs (S, K_S). This signature is called a certificate.

Security Levels

There are different levels of security in modern cryptography, depending on whether information theory, physics (in particular quantum physics), computational complexity theory, or heuristics, has been used. To be more precise, when the computer power of the opponent is allowed to be unbounded and one can mathematically prove that a formal definition of security is satisfied, then one is speaking about unconditional security. Information theory and probability theory is used to achieve this level of security. Evidently the formal definition of security must sufficiently model real-world security need(s).

In quantum cryptography one assumes the correctness of the laws of quantum physics (7).

A system or protocol is proven secure, relative to an assumption, when one can mathematically prove the following statement. The latter being, if the assumption is true, then a formal security definition is satisfied for that system or protocol. Such an assumption is typically an unproven claim in computational complexity theory, such as the presumed hardness of factoring large integers, or to compute discrete logarithm in finite groups. In this model the users and the opponent have only a computer power bounded by a polynomial in function of the length of a security parameter and one states that a system is secure if it requires superpolynomial (that is, growing faster to infinity than any polynomial) time to break it. One should note that this model is limited. Indeed, when using a cryptosystem, one needs to choose a security parameter which fixes the length.

In practice, a system is secure if the enemy needs the computer time of all computers on earth working in parallel, and the user needs, varying from application to application, 1 nanosecond up to a few minutes. However, modern theoretical computer science cannot guarantee that a certain number of basic operations are needed to break a cryptosystem. So, new algorithms may be developed that break cryptosystems faster than the previously best algorithms. Moreover, new technology makes computers faster each day. The impact of new algorithms and new hardware is clear from the following example. In 1977, it was estimated that factoring a 129 digit integer (product of two primes) would take 40 quadrillion (that is 4×10^{16}) years, whereas it was actually factored in 1993–1994 using the idle time of approximately 1600 computers on the Internet for 234 days (8).

A cryptosystem or protocol is as secure as another if one can mathematically prove that a new attack on the first scheme implies a new attack against the other and vice versa.

Finally, the weakest form of security is called heuristic. A system is heuristically secure if no (significant) attack has been found. Many modern but practical cryptosystems have such a level of security.

TOOLS

Many tools are used to achieve the desired security properties. These are based on discrete mathematics from several disciplines (mainly algebra, combinatorics, number theory, and probability theory) and our state-of-the-art knowledge of computer science (in particular, the study of (efficient) algorithms, algorithmic number theory, and computational complexity). Software engineering is used to design software implementations. Electrical engineering plays a role in hardware implementations, and information theory is also used, in particular, to construct unconditionally secure cryptosystems. Some of the main tools are explained briefly now.

The One-Time Pad

The one-time pad (9), also called the Vernam scheme, was originally designed to achieve privacy. Shannon (10), who invented information theory to study cryptography, proved the unconditional security of the scheme when used for privacy. The scheme has become a cornerstone of cryptography and is used as a principle in a wide range of seemingly unrelated contexts.

Shannon defined an encryption system as perfect when, for a cryptanalyst not knowing the secret key, the message m is independent of the ciphertext c.

In the original scheme the plaintext is represented in binary. Before encrypting the binary message, the sender and receiver have obtained a secret key, a binary string chosen uniformly at random. When m_i is the ith plaintext bit, k_i the ith key bit and c_i the ith ciphertext bit, in the Vernam scheme $c_i = m_i \oplus k_i$, where $\oplus$ is the exclusive-or, also known as exor. To decrypt, the receiver computes $m_i = c_i \oplus k_i^{-1}$, where in the case of the exclusive-or $k^{-1} = k$. The key is used only once. This implies that if the sender needs to encrypt a new message, then a new key is chosen, which explains the terminology: one-time pad. In modern applications, the exor is often replaced by a group operation.

Secret Sharing

A different interpretation of the one-time pad has recently been given (11–13). Suppose that one would like to make a backup of a secret m with bits m_i. If it is put into only one safe, a thief who breaks open the safe will find it. So, it is put in two safes so that a thief who breaks open one safe is unable to recover the secret.

The solution to this problem is to choose a uniformly random string of bits k_i (as many as there are bits in the message). One stores the bits k_i in the first safe and the bits $c_i = m_i \oplus k_i$ in the second. Given the content of both safes, one can easily recover the secret.

In the previous discussion, it is assumed that two safes would not be broken into, but only one at the most. If one fears that the thief may succeed in opening more, one could proceed as follows. Choose uniformly random $(t - 1)$ elements $s_1, s_2, \ldots, s_{t-1}$ in a finite group $S(+)$ and (assuming $m \in S$) construct $s_t = m - (s_1 + s_2 + \cdots + s_{t-1})$. Put s_i $(1 \leq i \leq t)$ in safe i. An example of such a group is $GF(2^n)(+)$ where n is the length of the message m. When $t = 2$, this corresponds to the one-time pad. One calls s_i $(1 \leq i \leq t)$ a share of the secret m, and the one who knows the share is called a shareholder or participant. Then it is easy to prove that the eavesdropper who opens $(t - 1)$ safes learns nothing about the secret. Only by opening all the safes is one able to recover the secret m.

A major disadvantage of this scheme is that it is unreliable. Indeed if one share is destroyed, for example, by an earthquake, the secret m is lost. A t-out-of-l secret sharing scheme is the solution. In such a scheme, one has l shares, but only t are required to recover the secret, whereas $(t - 1)$ are useless. An example of such a secret sharing scheme is discussed later on.

The concept of secret sharing was generalized, allowing one to specify in more detail who can recompute the secret and who cannot (14). Although previous secret sharing schemes protect reliability and privacy, they do not protect correctness and authenticity. Indeed, a shareholder could reveal an incorrect share, which (very likely) implies the reconstruction of an incorrect secret. When one can demonstrate the correctness of the shares, it is called verifiable secret sharing.

One-Way Functions

Cryptography based on computational complexity relies on one-way functions. A function(s) f is one-way if it is easy to compute f, and, given an image y, it is hard to find an x such that $y = f(x)$.

The state-of-the-art of computational complexity does not allow one to prove that one-way functions exist. For some functions f no efficient algorithm has been developed so far to invert f, and in modern cryptography it is often assumed that such functions *are* one-way.

One-way functions have many applications in modern cryptography. For example, it has been proven that a necessary and sufficient condition for digital signatures is a one-way function(15,16).

Block Ciphers

A blockcipher is a cryptosystem in which the plaintext and ciphertext are divided into strings of equal length, called blocks, and each block is encrypted one at a time with the same key.

To obtain acceptable security, a block cipher requires a good mode (17). Indeed, patterns of characters are very common. For example, subsequent spaces are often used in text processors. Common sequences of characters are also not unusual. For example, “ from the ” corresponds to 10 characters, which is 80 bits. In the Electronic Code Book (ECB) mode, the plaintext is simply divided into blocks that are then encrypted. Frequency analysis of these blocks allows one to find such very common blocks. This method allows one to find a good fraction of the plaintext and often the complete plaintext if the plaintext that has been encrypted is sufficiently long. Good modes have been developed based on feedback and feedforward.

Many block ciphers have been designed. Some of the most popular ones are the US Data Encryption Standard (DES), the Japanese NTT (Nippon Telegraph and Telephone Corporation), Fast Encipherment ALgorithm (FEAL), the “International Data Encryption Algorithm” (IDEA) designed by Lai (Switzerland), RC2, and RC5. DES (18), an ANSI (American National Standards Institute) and NIST (National Institute of Standards and Technology, US) standard for roughly 20 years, is being replaced by the Advanced Encryption Standard (AES), currently under development.

Hash Function

A hash function h is a function with n bits of input and m bits of output, where $m < n$. A cryptographic hash function needs to satisfy the following properties:

1. It is a one-way function.
2. Given x, it is hard to find an $x' \neq x$ such that $h(x) = h(x')$.
3. It is hard to find an x and an $x' \neq x$ such that $h(x) = h(x')$.

Note that the second property does *not* necessarily imply the third.

Several modes of block ciphers allow one to make cryptographic hash functions. A cryptographic hash function is an important tool for achieving practical authentication schemes. When signing a message digitally, first one pads it, and then one uses a cryptographic hash function before using the secret key to sign.

Universal hash functions are another type of hash function. These are used in unconditionally secure settings.

When referring to a hash function in applied cryptography, one means a cryptographic hash function.

Pseudonoise Generators and Stream Ciphers

A problem with the one-time pad is that the key can be used only once. The key must be transported by a secure path. In the military and diplomatic environment, this is often done by a trusted courier (using secret sharing, trust in the courier can be reduced). However, these requirements are unrealistic commercially.

The goal of a pseudonoise (or pseudorandom) generator is to output a binary string whose probability distribution is (computationally) indistinguishable from a uniformly random binary string. The pseudonoise generator starts from a *seed,* which is a relatively short binary string chosen uniformly random.

When one replaces the one-time key in the Vernam scheme by the output of a pseudorandom generator, this is called a stream cipher. Then the sender and receiver use the seed as the secret key. It has been proven that if the pseudonoise is (computationally) indistinguishable from uniform, the privacy protection obtained is proven secure. This means that if an unproven computational complexity hypothesis is satisfied, no modern computer can find information about the plaintext from the ciphertext. It has also been demonstrated that a one-way function is needed to build a pseudorandom generator. Moreover, given any one-way function, one can build a pseudorandom generator. Unfortunately, the latter result is too theoretical to be used for building efficient pseudorandom generators.

Linear-feedback shift-register sequences are commonly used in software testing. However, these are too predictable to be useful in cryptography and do not satisfy the previous definition. Indeed, using linear algebra and having observed a sufficient number of outputs, one can compute the seed and predict the next outputs.

Many practical pseudorandom generators have been presented. Some of these have been based on nonlinear combinations of linear-feedback shift-registers others on recurrent linear congruences. Many of these systems have been broken. Using the output feedback (OFB) mode (17) of a block cipher one can also obtain pseudonoise generators. An example of a pseudonoise generator based on number theory is discussed later on.

Key Distribution

Public key systems, when combined with certificates, solve the key distribution problem. In many applications, however, replaying old but valid signatures should be impossible. Indeed, for example, one should not allow a recorded and replayed remote authenticated login to be accepted in the future. A solution to this problem is to require a fresh session key, used only for a particular session. Another reason to use session keys is that public key systems are slow, and so sender and receiver need to agree on a common secret key.

When conventional cryptography is used, the problem of key management is primary. Freshness remains important. The problem is how two parties who may have never communicated with each other can agree on a common secret key.

Many protocols have been presented. Designing secure ones is very tricky. Different security levels exist. A key distribution protocol based on number theory is discussed further on.

Zero-Knowledge

In many practical protocols one must continue using a key without endangering its security. Zero-knowledge (19) has been invented to prevent a secret(s) which has been used in a protocol by party (parties) A to leak to other parties B.

If B is untrusted, one gives the dark side of B the name B′. More scientifically, machines B adhere to their specified protocol. To specify parties that will interact with A, but behave differently, we need to speak about B′.

When untrusted parties (or a party), let us say specified by B′, are involved in a protocol, they see data being communicated to them and they also know the randomness they have used in this protocol. This data pulled together is called the view of B′. To this view corresponds a probability distribution (a random variable), because of the randomness used in the protocol. When both parties A and B have x as common input, this random variable is called $\text{View}_{A,B'}(x)$. If x is indeterminate, we have a family of such random variables, denoted $\{\text{View}_{A,B'}(x)\}$. One says that the protocol is zero-knowledge (does not leak anything about the secret of A) if one can simulate the view of B. This means that there is a computer (polynomial-time machine) without access to the secret that can generate strings with a distribution that is indistinguishable from $\{\text{View}_{A,B'}(x)\}$. One form of indistinguishability is called perfect, meaning that the two distributions are identical. There is also statistical and computational indistinguishability.

So, zero-knowledge says that whatever party B′ learned could be simulated off-line. So party B did not receive any information it can use after the protocol terminated. This is an important tool when designing proven secure protocols.

Commitment and Interactive Proofs

In many cryptographic settings, a prover A needs to prove to a verifier B that something has been done correctly, for example, demonstrate that a public key was chosen following the specifications. A straightforward, but unacceptable solution, would be to reveal the secret key used.

The solution to this problem is to use interaction (19). In many of these interactive protocols, the prover *commits* to something. The verifier asks a question [if the question is chosen randomly then the protocol is called an Arthur–Merlin game (20)]. Then the prover replies and may be asked to open the commitment. This may be repeated.

To be a (interactive) proof, it is necessary that the verifier will accept if the statement is true and the prover and verifier follow the described protocol. This property is called completeness. It is also necessary that the verifier will reject the proof if the statement is false, even if the prover behaves differently than specified and the dishonest prover A' has infinite computer power. This requirement is known as soundness. In a variant of interactive proofs, called arguments, the last condition has been relaxed.

An important subset of interactive proofs are the zero-knowledge ones. Then the view of a possibly dishonest verifier can be simulated, so the verifier does not learn any information that can be used off-line. Zero-knowledge interactive proofs have been used toward secure identification (entity authentication) protocols. An example of such a protocol is discussed later.

Note that several mechanisms for turning interactive zero-knowledge proofs into noninteractive ones have been studied both from a theoretical and practical viewpoint.

Cryptanalysis

Cryptanalysis uses its own tools. The classical tools include statistics and discrete mathematics.

Even if a cryptographic scheme is secure (that is, has not been broken), an inappropriate use of it may create a security breach. A mode or protocol may allow a cryptanalyst to find the plaintext, impersonate the sender, etc. Such problems are called "protocol failures." An incorrect software implementation often enables a hacker to make an attack, and a poor hardware implementation may imply, for example, that the plaintext or the key leaks due to electromagnetic radiation or interference.

The most popular modern cryptanalytic tool against asymmetric cryptosystems, based on the geometry of numbers, is the Lenstra–Lenstra–Lovasz (LLL) lattice reduction algorithm (21). It has, for example, been used to break several knapsack public key systems and many protocols (22). When analyzing the security of block ciphers, the differential (23) and linear cryptanalytic (24) methods are very important. Specially developed algorithms to factor and compute discrete log have been developed, for example, the quadratic sieve method (25).

ALGORITHMS BASED ON NUMBER THEORY AND ALGEBRA

Although many of these algorithms are rather slow, they are becoming very popular. Attempts to break them have allowed scientists to find better lower bounds on the size of keys for which no algorithm exists and unlikely will be invented in the near future to break these cryptosystems. However, if a true quantum computer can be built, the security of many of these schemes is in jeopardy.

When writing $a \in_R S$, one means that a is chosen uniformly random in the set S.

We assume that the reader is familiar with basic knowledge of number theory and algebra.

RSA

RSA is a very popular public key algorithm invented by Rivest, Shamir, and Adleman (26).

To generate a public key, one chooses two random and different primes p and q which are large enough (512 bits at least). One computes their product $n := p \cdot q$. Then one chooses $e \in_R Z^*_{\phi(n)}$, where $\phi(n) = (p - 1)(q - 1)$, computes $d := e^{-1} \bmod \phi(n)$ and publishes (e, n) as a public key. The number d is the secret key. The numbers p, q, and $\phi(n)$ must also remain secret or be destroyed.

To encrypt a message $m \in Z_n$, one finds the authentic public key (e, n) of the receiver. The ciphertext is $c := m^e \bmod n$. To decrypt the ciphertext, the legitimate receiver computes $m' := c^d \bmod n$ using the secret key d. The Euler–Fermat theorem (and the Chinese Remainder theorem) guarantees that $m' = m$.

To sign with RSA, one processes the message M, hashes it with h to obtain m, computes $s := m^d \bmod n$, and sends (M, s), assuming that h has been agreed upon in advance. The receiver, who knows the correct public key (e, n) of the sender, can verify the digital signature. Given (M', s'), one computes m' from M', using the same preprocessing and hash function as in the signing operation, and accepts the digital signature if $m' = (s')^e \bmod n$. If this fails, the receiver rejects the message.

Many popular implementations use $e = 3$, which is not recommended at all for encryption. Other special choices for e are popular, but extreme care with such choices is called for. Indeed many signature and encryption schemes have suffered severe protocol failures.

Diffie–Hellman Key Distribution

Let $\langle g \rangle$ be a finite cyclic group of large enough order generated by g. We assume that q, a multiple of the order of the ord(g) (not necessarily a prime), is public.

The first party, let us say A, chooses $a \in_R Z_q$, computes $x := g^a$ in this group, and sends x to the party with which it wants to exchange a key, say B. Then B chooses $a \in_R Z_q$, computes $y := g^b$ in this group, and sends y to A. Now both parties can compute a common key. Indeed, A computes $z_1 := y^a$ in this group, and B computes $z_2 := x^b$ in this group. Now $z_2 = z_1$, as is easy to verify.

It is very important to observe that this scheme does not provide authenticity. A solution to this very important problem has been described in Ref. 27.

The cryptanalyst needs to compute $z = g^{\log_g(x) * \log_g(y)}$ in $\langle g \rangle$. This is believed to be difficult and is called the Diffie–Hellman search problem.

An example of a group which is considered suitable is a subgroup of Z^*_p, the Abelian group for the multiplication of elements modulo a prime p. Today it is necessary to have at least a 1024 bit value for p, and q should have a prime factor of at least 160 bits. Other groups being used include elliptic curve groups.

ElGamal Encryption

The ElGamal scheme (28) is a public key scheme. Let g and q be as in the Diffie–Hellman scheme. If g and q differ from user to user, then these should be extra parts of the public key.

To make a public key, one chooses $a \in_R Z_q$, computes $y := g^a$ in this group, and makes y public. To encrypt $m \in \langle g \rangle$, knowing the public key y_A, one chooses $k \in_R Z_q$, computes $(c_1, c_2) := (g^k, m \cdot y_A^k)$ in the group, and sends $c = (c_1, c_2)$. To decrypt, the legitimate receiver (using the secret key a) computes $m' := c_2 \cdot (c_1^a)^{-1}$ in this group.

The security of this scheme is related to the Diffie–Hellman problem.

ElGamal Signatures

The public and secret key are similar as in the ElGamal encryption scheme. The group used is Z^*_p, where p is a prime.

Let M be the message and m the hashed and processed version of M. To sign, the sender chooses $k \in_R Z^*_{p-1}$, computes $r := g^k \bmod p$, computes $s := (m - ar)k^{-1} \bmod (p - 1)$, and sends (M, r, s). To verify the signature, the receiver computes m from M and accepts the signature if $g^m = r^s \cdot y^r \bmod p$; otherwise rejects.

Several variants of this scheme have been proposed, for example, the US Digital Signature Standard (29).

Pseudonoise Generator

Several pseudorandom generators have been presented, but we discuss only one. In the Blum–Blum–Shub (30) generator, a large enough integer $n = pq$ is public, where p and q have secretly been chosen. One starts from a seed $s \in Z^*_n$ and sets $x := s$, and the first output bit b_0 of the pseudorandom generator is the parity bit of s. To compute the next output bit, compute $x := x^2 \bmod n$ and output the parity bit. More bits can be produced in a similar manner.

More efficient pseudorandom generators have been presented (31).

Shamir's Secret Sharing Scheme

Let t be the threshold, m be the secret, and l the number of shareholders.

In this scheme (12), one chooses $a_1, a_2, \ldots, a_{t-1} \in_R GF(q)$, and lets $f(0) = a_0 = m$, where $f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_{t-1} \cdot x^{t-1}$ is a polynomial over $GF(q)$ and $q \geq l + 1$. The share $s_i = f(x_i)$ where $x_i \neq 0$ and the x_i are distinct. This corresponds to a Reed–Solomon code in which the message contains the secret and $(t - 1)$ uniformly chosen elements. Given t shares it is easy to compute $f(0)$, the secret, using Lagrange interpolation. One can easily prove that given $(t - 1)$ (or less shares), one has perfect secrecy, that is, any $(t - 1)$ shares are independent of the secret m.

GQ

Fiat and Shamir (32) suggested using zero-knowledge proofs to achieve identification. We discuss a variant of their scheme invented by Guillou and Quisquater (33).

Let $n = pq$, where p and q are distinct primes and v is a positive integer. To each prover one associates a number I, relatively prime to n which has a vth root. The prover, usually called Alice, will prove that I has a vth root and will prove that she knows a vth root s such that $s^v I \equiv 1 \mod n$. If she can prove this, then a receiver will conclude that the person in front must be Alice. One has to be careful with such a conclusion (34). The zero-knowledge interactive proof is as follows.

The verifier first checks whether I is relatively prime to n. The prover chooses $r \in_R Z_n^*$, computes $z := r^v \mod n$, and sends z to the verifier. The verifier chooses $q \in_R Z_v$ and sends it to the prover. If $q \notin Z_v$, the prover halts. Else, the prover computes $y := rs^q \mod n$ and sends y to the verifier. The verifier checks that $y \in Z_n^*$ and that $z = y^v I^q \mod n$. If one of these tests fails, the protocol is halted.

This protocol must be repeated to guarantee soundness. Avoiding such repetitions is a practical concern, addressed in Ref. 35. If the protocol did not halt prematurely, the verifier accepts the prover's proof.

CONCLUSION

More encryption schemes and many more signature schemes exist than we were able to survey. The tools we discussed are used in a broad range of applications, such as electronic funds transfer (36), electronic commerce, threshold cryptography (37,38) (which allows companies to have public keys and reduce the potential of abuse by insiders), private e-mail. Cryptography has evolved from a marginally important area in electrical engineering and computer science to a crucial component.

READING LIST

Several books on practical cryptography have appeared in the last few years. The book by Menezes et al. (39) can be considered the best technical survey on the topic of applied cryptography printed so far. A more academic book, although not so exclusive, is Stinson's (40). Several interesting chapters, in particular the one on cryptanalysis (22) have appeared in the book edited by Simmons (41). Several chapters on cryptography will appear in Ref. 42.

Unfortunately, no good book on theoretical cryptography has appeared so far. Books which have appeared in this area are only readable by experts in the area, or their authors have only written about their own contributions.

Although outdated, the tutorial by Brassard (43) balances theory and practical aspects and is still worth reading. The book by Kahn (1) overviews historical cryptosystems. Although the new edition discusses modern cryptography, there are too few pages on the topic to justify buying the new edition if one has the old one. Other more general books are available (44,45).

The main conferences on the topic of cryptography are Eurocrypt and Crypto. Nowadays, there are many specialized or more local conferences, such as *Asiacrypt* (which absorbed *Auscrypt*), the *Workshop on Fast Software Encryption,* the *Workshop on Cryptographic Protocols,* the *ACM Conference on Computer and Communications Security,* and the *IMA Conference on Cryptography and Coding* in Britain. The proceedings of many of these conferences are published by Springer Verlag. Many conferences have sessions on cryptography, such as *IEEE-ISIT, IEEE-FOCS, ACM-STOC.* Articles that have appeared in journals are scattered. Unfortunately, some prestigious journals have accepted several articles of poor quality.

BIBLIOGRAPHY

1. D. Kahn, *The Codebreakers,* New York: Macmillan, 1967.
2. W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Trans. Inf. Theory,* **IT-22**: 644–654, 1976.
3. R. C. Merkle, Secure communications over insecure channels, *Commun. ACM,* **21**: 294–299, 1978.
4. J. Saltzer, On digital signatures, *ACM Oper. Syst. Rev.,* **12** (2): 12–14, 1978.
5. S. Haber and W. S. Stornetta, How to time-stamp a digital document, *J. Cryptol.,* **3** (2): 99–111, 1991.
6. G. J. Popek and C. S. Kline, Encryption and secure computer networks, *ACM Comput. Surv.,* **11** (4): 335–356, 1979.
7. C. H. Bennett and G. Brassard, An update on quantum cryptography, *Lect. Notes Comput. Sci.,* **196**: 475–480, 1985.
8. D. Atkins et al., The magic words are squeamish ossifrage, *Lect. Notes Comput. Sci.,* **917**: 263–277, 1995.
9. G. S. Vernam, Cipher printing telegraph systems for secret wire and radio telegraphic communications, *J. Amer. Inst. Electr. Eng.,* **45**: 109–115, 1926.
10. C. E. Shannon, Communication theory of secrecy systems, *Bell Syst. Tech. J.,* **28**: 656–715, 1949.
11. G. R. Blakley, Safeguarding cryptographic keys, *AFIPS Conf. Proc.,* **48**: 313–317, 1979.
12. A. Shamir, How to share a secret, *Commun. ACM,* **22**: 612–613, 1979.
13. G. R. Blakley, One-time pads are key safeguarding schemes, not cryptosystems, *Proc. IEEE Symp. Security Privacy,* CA, 1980, pp. 108–113.
14. M. Ito, A. Saito, and T. Nishizeki, Secret sharing schemes realizing general access structures, *Proc. IEEE Global Telecommun. Conf. (GLOBECOM '87),* 1987, pp. 99–102.
15. M. Naor and M. Yung, Universal one-way hash functions and their cryptographic applications, *Proc. 21st Annu. ACM Symp. Theory Comput. (STOC),* 1989, pp. 33–43.
16. J. Rompel, One-way functions are necessary and sufficient for secure signatures, *Proc. 22nd Annu. ACM Symp. Theory Comput. (STOC),* 1990, pp. 387–394.
17. National Bureau of Standards, *DES Modes of Operation,* FIPS Publ. No. 81 (Fed. Inf. Process. Stand.), Washington, DC: US Department of Commerce, 1980.
18. National Bureau of Standards, *Data Encryption Standard,* FIPS Publ. No. 46 (Fed. Inf. Process. Stand.), Washington, DC: US Department of Commerce, 1977.
19. S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM J. Comput.,* **18** (1): 186–208, 1989.
20. L. Babai, Trading group theory for randomness, *Proc. 17th Annu. ACM Symp. Theory Comput. (STOC),* 1985, pp. 421–429.
21. A. K. Lenstra, Jr., H. W. Lenstra, and L. Lovasz, Factoring polynomials with rational coefficients, *Math. Ann.,* **261**: 515–534, 1982.
22. E. F. Brickell and A. M. Odlyzko, Cryptanalysis: A survey of recent results, in G. J. Simmons (ed.), *Contemporary Cryptology,* New York: IEEE Press, 1992, pp. 501–540.
23. E. Biham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems, *J. Cryptol.,* **4** (1): 3–72, 1991.
24. M. Matsui, Linear cryptanalysis method for DES cipher, *Lect. Notes Comput. Sci.,* **765**: 386–397, 1994.

25. C. Pomerance, The quadratic sieve factoring algorithm, *Lect. Notes Comput. Sci.,* **209**: 169–182, 1985.
26. R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, *Commun. ACM,* **21**: 294–299, 1978.
27. P. C. van Oorschot, W. Diffie, and M. J. Wiener, Authentication and authenticated key exchanges, *Des., Codes Cryptogr.,* **2**: 107–125, 1992.
28. T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inf. Theory,* **31**: 469–472, 1985.
29. National Institute of Standards and Technology, *Digital Signature Standard,* FIPS Publ. No. 186 (Fed. Inf. Process. Stand.), Springfield, VA: U.S. Department of Commerce, 1994.
30. L. Blum, M. Blum, and M. Shub, A simple unpredictable pseudo-random number generator, *SIAM J. Comput.,* **15** (2): 364–383, 1986.
31. A. W. Schrift and A. Shamir, The discrete log is very discreet, *Proc. 22nd Annu. ACM Symp. Theory Comput. (STOC),* 1990, pp. 405–415.
32. A. Fiat and A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, *Lect. Notes Comput. Sci.,* **263**: 186–194, 1987.
33. L. C. Guillou and J.-J. Quisquater, A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory, *Lect. Notes Comput. Sci.,* **330**: 123–128, 1988.
34. S. Bengio et al., Secure implementations of identification systems, *J. Cryptol.,* **4** (3): 175–183, 1991.
35. M. V. D. Burmester, An almost-constant round interactive zero-knowledge proof, *Inf. Process. Lett.,* **42** (2): 81–87, 1992.
36. S. Brands, Electronic money, in M. Atallah (ed.), *Handbook of Algorithms and Theory of Computation,* Boca Raton, FL: CRC Press, in press, 1998.
37. Y. G. Desmedt, Threshold cryptography, *Eur. Trans. Telecommun.,* **5** (4): 449–457, 1994.
38. Y. Desmedt, Some recent research aspects of threshold cryptography, *Lect. Notes Comput. Sci.,* **1396**: 158–173, 1997.
39. A. Menezes, P. van Oorschot, and S. Vanstone, *Applied Cryptography,* Boca Raton, FL: CRC Press, 1996.
40. D. R. Stinson, *Cryptography: Theory and Practice,* Boca Raton, FL: CRC Press, 1995.
41. G. J. Simmons (ed.), *Contemporary Cryptology,* New York: IEEE Press, 1992.
42. M. Atallah (ed.), *Handbook of Algorithms and Theory of Computation,* Boca Raton, FL: CRC Press, in press, 1998.
43. G. Brassard, Modern Cryptology, *Lect. Notes Comput. Sci.,* Springer-Verlag, New York, 1988, p. 325.
44. B. Schneier, *Applied Cryptography. Protocols, Algorithms, and Source Code in C,* 2nd ed. New York: Wiley, 1996.
45. C. P. Schnorr, Efficient signature generation for smart cards, *J. Cryptol.,* **4** (3): 239–252, 1991.

YVO G. DESMEDT
University of Wisconsin—Milwaukee
University of London