Modern cryptology

Anne Canteaut
INRIA
Projet CODES
France
`Anne.Canteaut@inria.fr`

Françoise Lévy-dit-Véhel
École Nationale Supérieure
des Techniques Avancées
France
`levy@ensta.fr`

Graham Norton
The University
of Queensland
Australia
`ghn@maths.uq.edu.au`

Cryptography is an ancient discipline. Messages were already being encrypted in the archaic period of Greece. One of these, dating from the sixth century BC, consisted of wrapping a roll of paper around a cylinder and then writing the message on the paper. The unrolled paper was then sent off to the receiver, who could easily decrypt the message if he knew the diameter of the original cylinder.

For many years, Cryptography was the exclusive reserve of military and diplomatic circles. Literature on the subject was very limited. The first fundamental publication was the 1949 article of Claude Shannon - *"The communication theory of secrecy systems"* [Sha]- which lays down a mathematical basis for cryptographic systems, beginning with the definition of a new model : Information Theory. Feistel made a significant subsequent contribution with the publication in the beginning of the seventies of his work on iterative block-cipher systems [Fei1, Fei2]. This led to the DES encryption algorithm being proposed in 1997 as a secret-key encryption standard for non-classified applications. The increase in computer power challenged the security of DES, which was replaced by a new standard called AES in October 2000. This algorithm is the result of recent research, especially in Cryptanalysis.

But the major advance in Cryptography was incontestably the 1976 publication of *"New directions in cryptography"* [Dif], by Whitfield Diffie and Martin Hellman. This article introduced the revolutionary concept of *public-key* cryptography. Even though the authors did not give a practical realisation of a public-key system, the properties were nonetheless clearly enunciated. Moreover, they presented a protocol by which two entities could agree a secret key from preliminary knowledge of public data only. The first realisation of a public-key system was due to Ronald Rivest, Adi Shamir and Leonard Adleman in 1978 : the RSA system [Riv]. Since then, the literature on this topic has not ceased to develop.

More recently, Cryptography has had to offer additional capabilities in order to meet new threats arising from the development of information networks and the massive digitisation of documents. The principal examples of these capabilities are guaranteeing the authenticity of messages (their provenance and their contents) and certifying a person's identity. The former are realised by digital signature algorithms and the latter by identification techniques.

This introductory article presents first of all the two principal categories of cryptographic procedures which are most frequently used: *encryption algorithms*, which serve to protect the confidentiality of data and *signature algorithms* which, just like handwritten signatures, guarantee the provenance and the integrity of messages. The article details several practical implementation aspects of these procedures. It must be noted that the desired capabilities of a particular application need to be listed in advance before looking for an appropriate cryptographic solution.

1 Encryption

An encryption algorithm transforms a message, called the *cleartext*, into a *ciphertext* which will be readable by its legitimate receiver only. This transformation is done via an encryption function which is parametrised by an encryption key. A privileged interlocutor can then decrypt the ciphertext by using the deciphering function, provided that he knows the corresponding deciphering key. Such a system is secure only insofar as it is impossible for an intruder to deduce the cleartext from the ciphertext and *a fortiori* to recover the deciphering key.

This formalisation was used for just over a century. Cryptographers have now realised that it is unrealistic for the security of a cipher system to rest solely on the hypothesis that an attacker does not know the ciphering method used. The recent publication of specifications of proprietary algorithms on the internet, such as those used in the GSM system, has once more shown us that it is impossible keep an algorithm secret over the long term. Therefore, the security of an encryption algorithm must reside uniquely on the secrecy of the deciphering key. Furthermore, the fact that encryption and decryption methods are made public offers a certain guarantee of system security, insofar as each new cryptographic algorithm is immediately confronted by the inventiveness of the scientific community.

We distinguish two main types of encryption algorithms according to whether their keys are secret or public. Each class of algorithms has its advantages and its disadvantages. Secret-key systems require the sharing of a secret among the interlocutors. The discovery in 1976 of public-key systems allowed this constraint to be removed. However, this has not delivered a perfect solution as public-key encryption algorithms are slow and do not permit online encryption. In the majority of current applications, the best solution is a hybrid system which combines both types of algorithm.

1.1 Secret-key encryption

In *secret-key (or symmetric or conventional) encryption algorithms* the sender and the receiver share the same secret key — in other words, the encryption and decryption keys are identical. The use of a secret-key algorithm for communication thus requires a preliminary exchange of a secret between the two protagonists across a secure channel via other cryptographic techniques.

An essential parameter for the security of a secret-key system is the size of the keyspace. Actually it is always possible to recover a secret key by launching a so-called *exhaustive* attack. This consists of simply enumerating all possible keys and attempting to decrypt the ciphertext by using each one in turn. If the keyspace corresponds to the set of k-bit words, the average number of calls to the decryption function required by an exhaustive attack is 2^{k-1}. Such an attack thus becomes infeasible as soon as the keyspace is sufficiently large. In view of current computing power, one considers that a secret key must be at least 64 bits long (which means that an exhaustive attack would require on the average 2^{63} attempts). We note that this limit evolves with technology. To give an order of size, an exhaustive attack of the DES encryption system, which uses a 56-bit secret key, was achieved in January 1998 in 39-days on 10,000 Pentium processors in parallel, then in 56 hours in July 1998 with the aid of a dedicated machine consisting of 1500 DES components [1]. The computing time for an exhaustive attack is evidently exponential in the size of the secret key. It is 2^{64} times, i.e.

[1]`http://www.eff.org/descracker.html`

18446744073709551616 times harder to break a system possessing a 128-bit key than to break a system using a 64-bit key (which is already very difficult).

There are other types of attacks on secret-key encryption systems. The majority consist of exploiting certain structural pecularities of the algorithm or certain statistical biases in the distribution of clear/ciphertext pairs. The most well-known are differential cryptanalysis, invented by the Israeli cryptographers Biham and Shamir in 1991 and linear cryptanalysis proposed by the Japanese cryptographer Matsui in 1993. One generally considers that a secret-key encryption offers good security if there is no attack of complexity less than an exhaustive search. At the moment, the security of secret-key systems rests uniquely on empirical statements that they are difficult to cryptanalyse. One can show that an encryption algorithm resists classical attacks, but one cannot exclude the appearance of efficient new attacks.

Only the so-called 'block-cipher' techniques are considered here. A cipher system is said to be by blocks if it divides the cleartext into fixed-size blocks and enciphers one block at a time. The blocksize is generally 64 or 128 bits.

DES Until very recently, the most famous and most often used secret-key cipher system was DES (Data Encryption Standard). It was adopted as an American standard for commercial communications in 1977 (FIPS 46 standard[2]) and later by ANSI in 1991. DES operates on 64-bit blocks and uses a 56-bit secret key. It is thus vulnerable to future exhaustive attacks.

This is why the majority of applications now use it in the form of a triple DES with two keys, consisting of three successive DES encryptions with two secret keys. This technique permits doubling of the secret keysize (to 112 bits). More precisely, to encrypt with triple DES, one first does a DES encryption with a key of 56 bits, then a DES encryption parametrised by a second key and then a further DES encryption with the first key. Only two keys are used as the use of three different secret keys does not increase the algorithm security. It is of note that triple DES with two keys has been adopted in the ANSI X9.17 and ISO 8732 standard. It is heavily used for financial applications.

Other applications, such as the PGP (Pretty Good Privacy) system, prefer IDEA (International Data Encryption Algorithm) conceived by Lai and Massey in 1992. This algorithm works with 64-bit blocks, but uses a 128-bit key.

AES AES (Advanced Encryption Standard) is the new secret-key encryption standard, chosen in October 2000 from fifteen systems proposed in response to a call for proposals by NIST (National Institute of Standards and Technology). This algorithm, initially called RIJNDAEL, was conceived by two Belgian cryptographers, V. Rijmen et J. Daemen. It works with 128-bit message blocks and is available in three different keysizes : 128, 192 and 256 bits. The specifications of these three versions as well as several implementations are available from the NIST website [3].

As with most block algorithms, the AES encryption procedure consists of interating a permutation which is parametrised by a secret 'subkey', which changes at each iteration. The different subkeys are generated by a special algorithm and each is 128 bits long. For a 128-bit key, AES iterates the function described in Figure 1 ten times. The first iteration is preceded

[2]`http://csrc.nist.gov/cryptval/des.htm`
[3]`http://csrc.nist.gov/encryption/aes/rijndael/`

by a bitwise exclusive-or of the cleartext and subkey 0 , the original secret key; in the same way, the last iteration is slightly different from the preceding ones.

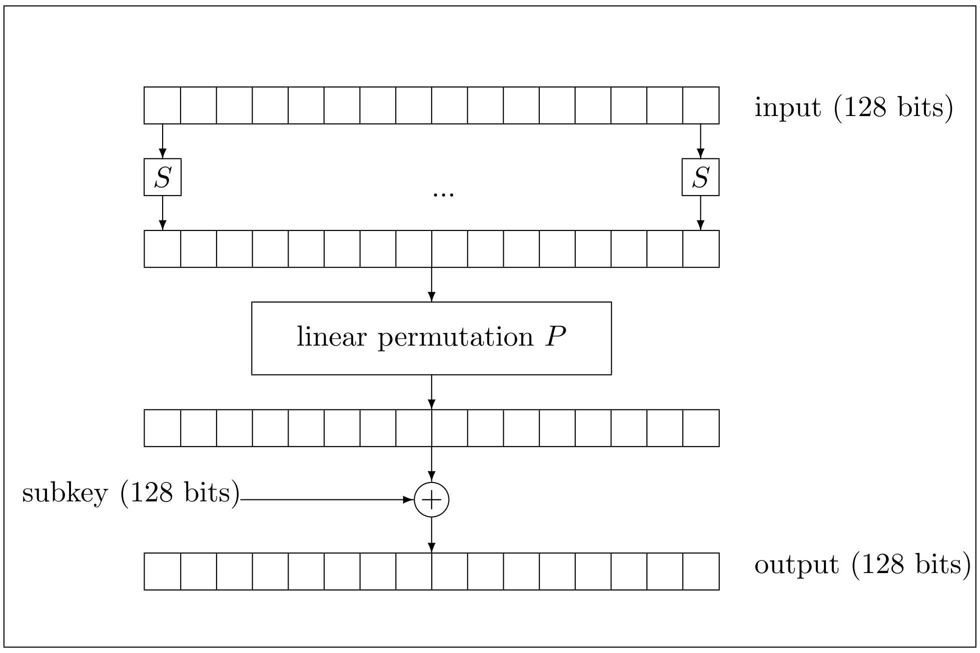

Figure 1: One AES iteration

The iterated function consists of three steps, conforming to the fundamental principles of confusion and diffusion enunciated by Shannon. The first step, called the confusion step, consists of applying the same permutation S to each of the sixteen bytes of the 128-bit input. This function corresponds to inversion in the finite field of 2^8 elements (in practice, the function is table-driven) and it ensures that the algorithm will be able to resist classical attacks (differential cryptanalysis, linear cryptanalysis ...). Then during the diffusion phase, one permutes the bits of the word obtained by applying another function P, also composed of simple operations in the field of 2^8 elements. Finally, one performs a bitwise exclusive-or between the result and the current subkey.

The 128-bit subkeys, numbered 0 to 10, are derived from the secret key in the following way : subkey 0 is the secret key ; subkey i (used in the ith iteration) is obtained from subkey $i-1$ using the algorithm described in Figure 2.

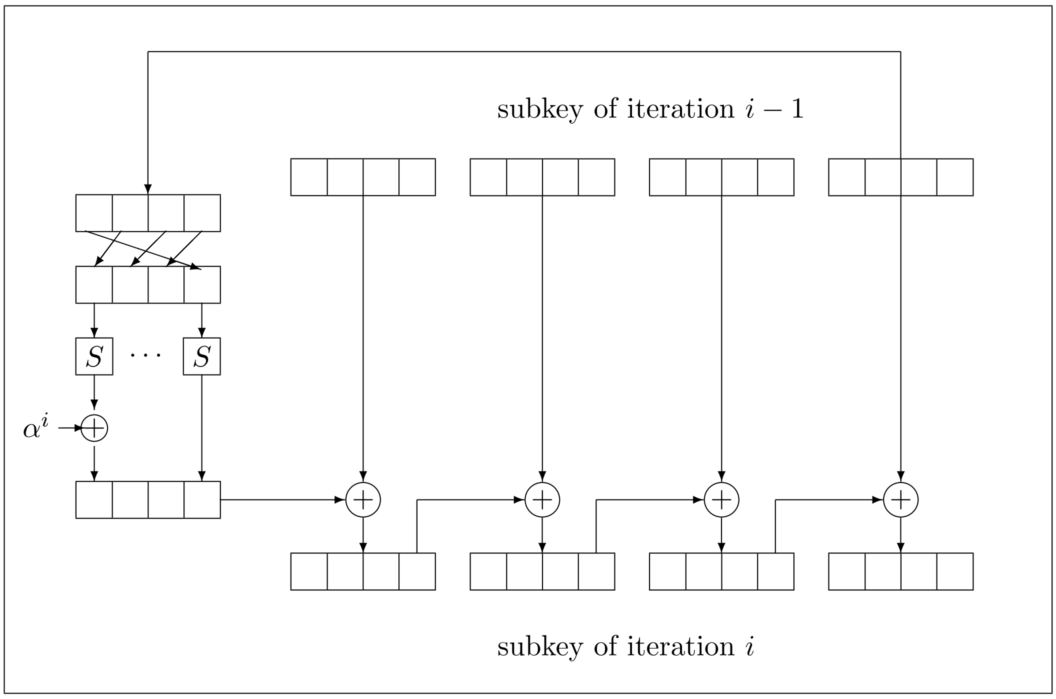

Figure 2: AES subkey generating algorithm

One first permutes the last four bytes of key number $i-1$, then one applies the function S. After adding a constant (depending on i) to the first byte, one performs a bitwise exclusive-or to the four bytes thus obtained and the first four bytes of the preceding subkey. The three other blocks of four bytes of subkey i are then simply the result of an exclusive-or between the block corresponding to subkey $i-1$ and the preceding block of subkey i.

The fact that AES is composed solely of simple operations on bytes makes it extremely fast, both for the 8-bit processors used in smart cards and for software implementations. It attains for example an enciphering rate of 70 Mbits/sec in a C++ implementation on a 200 MHz Pentium[4].

1.2 Public-key encryption

Public-key or asymmetric cryptography avoids secret-sharing between the two interlocutors. In a public-key cipher system, each user has a pair of keys, (i) a *public key* (in general, universally available in a directory) and (ii) a confidential *secret key*. To send a confidential message to Bob, Alice encrypts the cleartext using Bob's public key. The latter, together with the corresponding secret key, is the only way to decrypt the received message.

If one were to compare the two types of encryptions for exchanging confidential messages in concrete terms, a secret-key system would be a strong box, whereas a public-key system would correspond to a letter box. Let us imagine that Alice would like to communicate a document to Bob by depositing it in a strong box. In this case, Alice and Bob must share a secret, viz. the secret combination of the strong box. This simultaneously allows Alice to deposit the documents and Bob to retrieve them. Charlie can use the same strong box

[4]`\tthttp://fp.gladman.plus.com/cryptography\_technology/rijndael/`

to communicate with Bob only if Charlie also knows the secret combination. But the latter solution also provides Charlie with the possibility of reading all documents placed in the box, even those not intended for him. Suppose now that Bob asks his interlocutors to transmit confidential documents to him by depositing them in his letter box, which can be locked with a unique key. Everyone who knows Bob's address (which is public) can deposit messages for him. On the other hand, only Bob has the key to open the letter box and read the messages which are destined for him.

The essential notion on which public-key encryption rests is that of a *one-way trapdoor function.* A function is called *one-way* if it is easy to calculate but impossible to invert. (Here 'impossible' means infeasible in a realistic timeframe, given reasonable computing power. For example, a calculation requiring a billion years on a billion parallel processors is considered impossible.) Such a function is called a *trapdoor function* if the calculation of its inverse function becomes easy once one knows supplementary information (the trapdoor).

It is very easy to construct a public-key cipher system given a one-way trapdoor function. The encryption procedure consists of simply applying the function to the cleartext. Being one-way, it is very difficult to invert i.e. to determine the cleartext from the ciphertext unless one knows the trapdoor (which corresponds to the secret key of the receiver). The difficulty resides entirely in finding these very particular functions. Their construction relies on the reputedly difficult mathematical problems. The most famous is that of factoring large integers, which is the basis for the RSA system.

RSA This the most often-used public-key system. Strictly speaking, RSA is not a standard but its use is described and recommended in a large number of official standards and for bank applications in particular. Examples are the French standard ETEBAC 5 and the American standard ANSI X9.31.

Its operation depends on results from classical arithmetic. In what follows, for integers a and n, the notation $a \bmod n$ denotes integer division of a by n. Consider an integer n equal to the product of two prime numbers p and q. By Euler's theorem, if a is an integer such that $a \bmod (p-1)(q-1) = 1$, then for any non-zero integer x strictly less than n, one has :

$$x^a \bmod n = x\ .$$

The RSA principle is as follows : the public key of a user is the pair of integers (e, n), where n is the product of two prime numbers p and q, and e is relatively prime to $(p-1)(q-1)$. The pair (e, n) is published in a directory.

The secret key is an integer d satisfying $ed \bmod (p-1)(q-1) = 1$. It is very easy to find such a d given e, p and q. By hypothesis, e is relatively prime to $(p-1)(q-1)$, so by Bezout's theorem, there are non-zero integers A and B such that

$$A(p-1)(q-1) + Be = \gcd((p-1)(q-1), e) = 1\ .$$

The secret key d is then the positive integer B modulo $(p-1)(q-1)$.

In RSA, the message blocks are represented by integers between 0 and $n-1$. To send a message m to Bob, Alice looks up Bob's public key and calculates the ciphertext c given by : $c = m^e \bmod n$.

When he receives the ciphertext c, Bob obtains the cleartext by calculating $c^d \bmod n = m$. For e and d satisfy $ed \equiv 1 \bmod (p-1)(q-1)$ by definition, so :

$$c^d \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n = m.$$

.

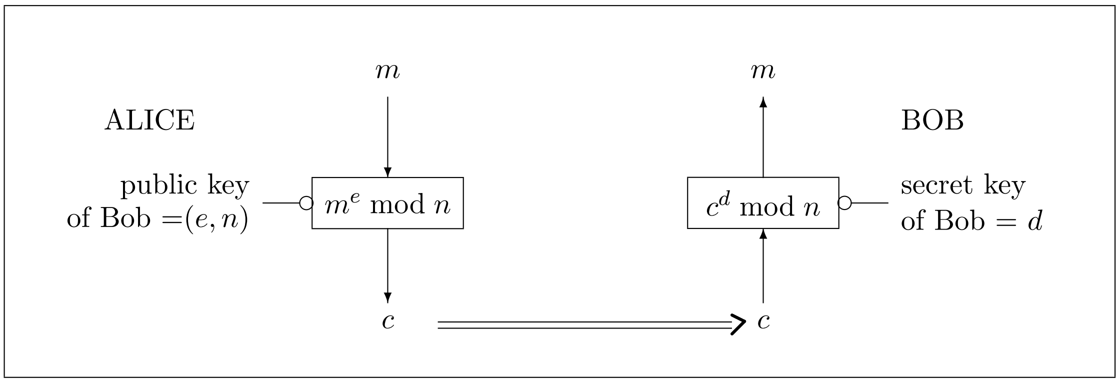

Figure 3: RSA public-key encryption

Attacking the RSA system therefore consists of obtaining the cleartext m from knowledge of the ciphertext $c = m^e \bmod n$ and from the public key (e, n). No efficient algorithms are known for solving this problem. The only known general attack for decrypting RSA consists of finding the secret key d from the public values (e, n). One can show that solving this problem is equivalent to factorising the integer n. Currently no fast factorisation algorithms exist. The largest ordinary number factorised to date has 512 bits (155 decimal digits). This record was established in 1999 through the collaboration of eleven scientific teams[5]. The factorisation required a two-and-a-half-month distributed computation (on 300 computers), and 224 hours on a Cray-C916. Recent results show that in order to be protected from factorisation algorithms, it is imperative that integers p and q with a product of at least 768 bits be used for RSA. Further, one should in general use prime numbers with a 1024-bit product.

Cryptosystems using elliptic curves Besides integer factorisation, another often-used problem in cryptography is the extraction of discrete logarithms, which can be described as follows. Let G be a finite group with cardinality $|G|$ and generator g. We write G multiplicatively. Given an element β in G, find an integer x such that $\beta = g^x$. (Thus $0 \leq x \leq |G| - 1$.)

This problem is the origin of the 1978 Diffie-Hellman key-exchange protocol of the 1984 El Gamal ciphersystem, and of several signature schemes (cf. Section 2). Its adaptation to other groups has been envisaged for fifteen years and gave rise to a number of others, called elliptic curve cipher systems. The idea, due to N. Koblitz and V. Miller in 1985, is to let G be the additive group of rational points of an elliptic curve over a finite field $\mathbf{F}$. Without going into the precise details involved, we cite the two principal reasons motivating such an approach :

- One can generate a large number of groups in this way without changing the field $\mathbf{F}$; thus one can conceive of an arithmetic processor optimised for specific $\mathbf{F}$ computations. These could be used for different implementations of the same ciphersystem (or of different cipher systems based on the same field).

[5]`http://ultralix.polytechnique.fr/Labo/Francois.Morain/rsa155.html`

- There is no known sub-exponential algorithm for solving the discrete logarithm problem is this context[6]. On the other hand, the best known solution for the discrete logarithm problem in a finite field with q elements has algorithmic complexity $\exp(O((\lg q)^{1/3}(\lg \lg q)^{2/3}))$.

This last observation importantly allows the use of significantly smaller keys, compared to those necessary for cipher systems based on the discrete logarithm in classical groups or those of RSA [7]. For example 170-bit keys suffice for ensuring the same level of security as a 1024-bit RSA encryption. The memory necessary for storing these keys, as well as the size of the data concerned, is thus reduced. We note that currently, the record for extracting discrete logarithms on elliptic curves concerns a 108-bit key. This was established by Robert Harley of INRIA, at a computation cost similar to that needed to break a 512-bit RSA key [8].

Some remarks on the keysize We record here that the keysize in no way has the same significance as for secret-key systems. The best possible attack for breaking a public-key system is not an exhaustive search, even if this is the case for a secret-key system. To find a 512-bit RSA secret key, it would be absurd to test each of the 2^{512} 512-bit keys, which is evidently infeasible. Factorisation techniques are much faster. Furthermore, the attacks on public-key systems are generally not exponential in the keysize [9]. For a secret-key system, it is twice as hard to break an algorithm having a 65-bit key as for an algorithm with a 64-bit key. This property is no longer true for RSA. Thus a secret-key system using 128 bits is considered secure, whereas a public-key system using a key of the same length is extremely weak.

2 Digital signatures

In many communications, the actual confidentiality of data is of little consequence. However, ensuring its provenance and its integrity is important. In other words, one needs to verify that the data has not been modified during transmission.

2.1 The signature principle

A *digital signature* process consists of appending a small number of bits to the cleartext. These bits depend simultaneously on the message and its author. To have the same guarantee as a handwritten signature, everyone needs to be able to verify the signature, but no one should be able to imitate it.

A signature scheme is thus composed of signature and verification functions. The signature function is parametrised by a secret key owned by the signer and assigns a signature to each cleartext. The verification function does not require any secret knowledge. Starting with the cleartext and the signature, it allows the authenticity of the cleartext to be verified from the latter.

Thus a signature scheme must possess a certain number of properties. In particular, it must be impossible to forge a signature in practice: only the owner of the secret key can sign

[6]Excepting certain well-known classes of curves.

[7]The keysizes in these two systems are effectively comparable and the underlying difficulty of the two problems above seem to be of the same order.

[8]`\tthttp://cristal.inria.fr/harley/ecdl7/`

[9]Except for cipher systems based on the discrete logarithm problem on elliptic curves, and subject to algorithmic advances in this area.

in his name. The signature must be invalid if the cleartext has been modified ; it must be impossible to reuse a signature. Finally, the signer must be unable to deny signing a message.

A signature thus guarantees :

- the identity of the sender ;
- the integrity of the received data, i.e. that the message has not been modified during transmission ;
- the non-repudiation of the message, i.e. that the sender cannot deny being the author of the message.

This is why digital signature procedures are just as binding as handwritten ones. Their validity is now recognised by French Law (Law 2000-230 of 13 March 2000).

2.2 Principal signature schemes

RSA Signature Certain 'reversible' public-key cipher systems can be used to construct signature schemes. The signature function corresponds to the decryption function parametrised by the user's secret key and the verification function is derived from the encryption function.

Thus in the RSA signature scheme for example, a user signs a message m by applying the RSA decryption function to his secret key d. To verify the signature, it suffices to apply the RSA encryption function (parametrised by the associated public key (e, n) and to verify that the result of this calculation does indeed correspond to the cleartext sent. The conditions imposed by the size of the integers p and q are the same in the signature context as in that of encryption.

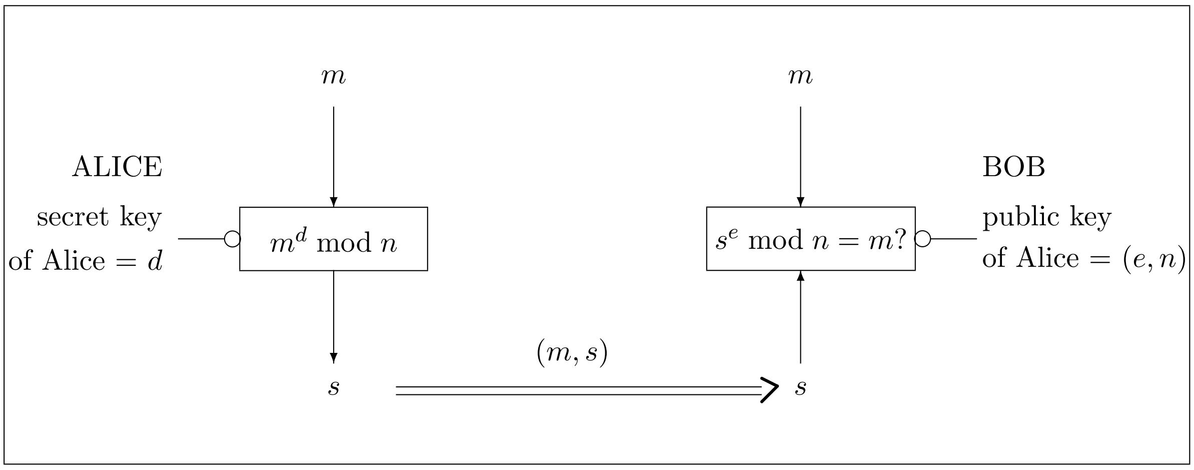

Figure 4: The RSA signature

DSA The DSA scheme[10] forms part of a family of signature algorithms based on the discrete logarithm problem. In 1994 it became the American digital standard for the protection of non-classified information. The performance of this scheme is comparable to that of the RSA

[10]Digital Signature Algorithm

as a signature scheme but is notably more costly for the verification phase. In contrast — and this is its major interest — it produces short signatures [11] (compared to RSA signatures, typically 1024 bits), namely 320 bits, all the while offering analogous security. The level of security here measures the difficulty of forging a signature, i.e. fabricating a valid signature without knowing the secret key.
It is also possible to construct signature schemes based on the discrete logarithm problem on elliptic curves : ECDSA, the adaptation of DSA to this context, being an example.

3 Cryptography in practice

So far, we have presented the basic cryptographic primitives. Let us now look at how these can be used in practice.

Hybrid ciphers When one wants to ensure the confidentiality of exchanged messages, one does not in general have access to a single type of cipher system. In effect, the complexity of the operations involved in public-key systems renders the cipher system extremely slow compared to a secret-key system. (for example, in hardware, RSA is on the order of 1000 times slower than DES). On the other hand, only a public-key scheme allows a secure exchange of a secret without preliminary exchange of a shared secret. Thus one would prefer to use a public-key algorithm to exchange a secret key. This key will serve to encrypt the exchange of information with the aid of a symmetric algorithm. This combination of the two techniques permits both the speed of secret-key encryption and the resolution of the problem of exchanging secret keys between the two interlocutors. This is notably the encryption solution used in the PGP program [12]. More generally, public-key systems are used in practice to encrypt very short messages.

Signature and hash functions To sign a message one has recourse to cryptographic hash functions before applying one of the algorithms mentioned in Section 2.2. Such functions, which have completely public descriptions, transform arbitrary length binary strings into fixed length *compressed* or *hashed* binary strings (generally 128 or 160 bits). Two reasons motivate their use : the first, as we have seen, is the slowness of public-key systems : the messages to be signed are relatively long and it would be too costly to sign them as is. One therefore applies a hash function to them and one signs the hashed message, not the message itself.

The second reason, related to security, is to prevent or obstruct certain types of attacks on signature schemes. For example, in the case of RSA, the signature of the product of two messages [13] is the product of the two individual signatures. This particular feature of RSA renders it vulnerable to forging the messages, (certainly, most often unintelligible), but can still represent a threat, above all when the signature mechanism is used to gain identification. Hashing the message before signing it allows this weakness to be mitigated.

To be usable for cryptographic applications, a hash function must however satisfy the following constraint : a *collision* between messages must be avoided in practice i.e. two

[11] This property is very desirable, in particular when digital signatures are used for certification (cf. Section 3), their significant size rendering the mechanism all the more cumbersome to manage.
[12] `http://www.pgp.net/pgpnet/pgp-faq/`
[13] A message is regarded as a positive integer strictly less than the modulus n.

messages must not hash to the same bit string. Since finding collisions requires at least 2^{64} attempts, the hashed string must be at least 128 bits long.

Most hash functions currently in use are improvements of the MD4 (Message Digest 4) function. The latter was frequently used before being used in Cryptography in 1996. Among the principal hash functions, one may cite the function MD5. This function compresses a message into 128 bits. Certain weaknesses in its construction were revealed recently, but these do not compromise its security. All the same, certain people prefer to avoid using it. Thus some prefer the American standard SHA-1 (Secure Hash Algorithm), used in the DSA signature scheme, or the function RIPEMD-160, conceived in the setting of a European project[14]. These two functions also have the advantage of compressing to 160 bits.

Certification of public keys By avoiding secret sharing between the protagonists, public-key cryptography is confronted by another, extremely difficult problem : how to guarantee the validity of public keys ? As soon as a user wants to encrypt a message using a public-key algorithm or to verify a signature, he must obtain the public key of his interlocutor or that of the signer. If the public keys are stored in insecure directories, they risk being intercepted and replaced by other keys.

Thus it is possible to fabricate false signatures simply by replacing the public key of a user. Thus if Charlie has replaced Alice's public key by his own public key, everyone receiving a message signed by Charlie will think that this message carries Alice's authentic signature. Before verifying a signature, a user has to ensure the validity of the signer's public key.

This problem, crucial for all of public-key cryptography, may be resolved by introducing a third party, called a witness, who validates the link between users and their public keys. Formally, a *public-key certificate* consists of cleartext and a signature. The cleartext contains in particular a public key and a character string identifying the key owner. The signature corresponds to the digital signature via the witnessing of the preceding text. If this signature is authentic, it validates the link between the user's identity and his public key.

Once a system has a large number of users, an infrastructure to manage the public keys thus becomes necessary. There are very hierarchical systems, such as that described in standard ISO X509-v3, in which a user's key is witnessed, whose key is in turn certified by a higher witness ... In contrast, the PGP system uses a system without authentication and is based on trust. One accepts the user's public key because it is signed by someone whose key is itself signed by another party whom one knows and trusts. All of these techniques are cumbersome to put into operation. But they are fundamental because the security of a public-key system rests in large part on the management of public keys.

A word on bank cards These fundamental cryptographic notions suffice for understanding the recent attacks made on bank cards and for decrypting the ensuing polemics. When one uses a bank card to pay for a small purchase, the operation is done off-line, without exchanging any information with the bank (bank information is assembled and communicated at the end of the day). The unique control at the moment of payment (besides the confidential code), consists in verifying that the card being used is valid i.e. that it was emitted by a recognised bank. This procedure is done using an RSA signature : each card has an identifier which has been signed by the bank. It is this signature, written on the chip, which is verified at each transaction by the business terminal. Each card bearing a valid signature is therefore

[14]`http://www.esat.kuleuven.ac.be/bosselae/ripemd160.html`

considered authentic since the bank is the only authority which has the RSA secret key allowing the signing.

The unique fault of this system, which could be exploited to fabricate false bank cards, is the RSA keysize used. The signature was produced using a 320 bit key, so a key of this size can be easily factorised in several days with the current power of a large public computer (in contrast, this operation demanded considerable computer power a dozen years ago). To fabricate false cards, it thus suffices to obtain the public key of the bank (the number n, the product of two prime numbers), which is relatively easy since this data is present in every terminal accepting a payment. The factorisation of this number then directly furnishes the corresponding secret key which can be used to 'imitate' the bank signature. It only remains to choose an identifier arbitrarily and to write the associated signature into a blank card. Such a card would be accepted by a terminal. Since it is in fact not a genuine bank card but a decoy, the protection of a confidential code no longer exists.

Neither the security of a smart card, nor that of the RSA signature algorithm are challenged by this attack. The only problem is the keysize used, which is happily 792 bits in the most recent cards. In any cipher system worthy of the name, evidently the keysize must evolve concurrently with computer power.

References

[Dif] W. Diffie, M.E. Hellman: *New directions in cryptography*, IEEE Transactions on Information Theory, 22 (1976), pp. 644-654.

[Fei1] H. Feistel: *Cryptography and computer privacy*, Scientific American, 228, May 1973, pp. 15-23.

[Fei2] H. Feistel: *Block cipher cryptographic system*, U.S. patent 3,798,359, 19 March 1974.

[Riv] R.L. Rivest, A. Shamir, L.M. Adleman: *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, 21 (1978), pp. 120-126.

[Sha] C.E. Shannon: *Communication theory of secrecy systems*, Bell System Technical Journal, 28 (1949), pp. 656-715.

Let us also cite:

- S. Singh: *The code book – The Secret History of Codes and Code-breaking* . Fourth Estate, 1999.
- A.J. Menezes, P.C. van Oorschot, et S.A. Vanstone: *Handbook of Applied Cryptography.* CRC Press, 1997. It can be downloaded from `http://www.cacr.math.uwaterloo.ca/hac/`.