Combinatorial and Computational Mathematics (15 – 17 february 2000, Pohang, South Korea)
S. P. Hong, J. H. Kwak, K. H. Kim and F. W. Roush Eds.
World Scientific Publishing, New Jersey, 2001.

Number Theory and Public-Key Cryptography

David Pointcheval

LIENS – CNRS, École Normale Supérieure,
45 rue d'Ulm, 75230 Paris Cedex 05, France – CNRS.
David.Pointcheval@ens.fr – http://www.di.ens.fr/~pointche

Abstract. For a long time, cryptology had been a mystic art more than a science, solving the confidentiality concerns with secret and private techniques. Automatic machines, electronic and namely computers modified the environment and the basic requirements. The main difference was the need of public mechanisms to allow large-scale communications with just a small secret shared between the interlocutors, but that furthermore resist against adversaries with more powerful computers. Unfortunately, the security remained heuristic: with a permanent fight between designers (the cryptographers) and breakers (the cryptanalysts).

In 1976, Diffie and Hellman claimed the possibility of achieving confidentiality between two people without any common secret information. However, they needed quite new objects: (trapdoor) one-way functions. Hopefully, mathematics, with algorithmic number theory, have been realized to provide such objects. A new direction in cryptography was under investigations: asymmetric cryptography and provable security.

In this paper we review the main problems that cryptography tries to solve, and how it achieves these goals thanks to the algorithmic number theory. After a brief history of the ancient and conventional cryptography, we review the Diffie-Hellman's suggestion with the apparent paradox. Then, we survey the solutions based on the integer factorization or the discrete logarithm, two problems that nobody knows how to efficiently solve.

Keywords: Number Theory, Public Key Cryptography, Digital Signatures, Public Key Encryption

1 INTRODUCTION

The need of confidentiality has always existed. Thus, one can find some "cryptographic" techniques in a quite far past, even before the ancient Greek civilization. But confidentiality, at that period, only relied on the secrecy of the techniques, which were various. Some of those techniques are reviewed in the main books about cryptography [65, 76, 41].

1.1 Brief History of Ancient Cryptography

The *Lacedemonian Scytale* is one of the oldest technique, used during the 5th century B.C. It consists in rolling up a papyrus around a piece of wood, then writing the message and getting off the papyrus from that piece of wood. The resulting message contains all the letters but scrambled according to the diameter of the piece of wood. Of course, the security completely vanishes against an adversary who knows the technique, even if he does not exactly know the diameter of the piece of wood.

Another well-known cipher, the *shift cipher*, a.k.a. the Caesar's cipher, has been used by Julius Caesar during the 1st century B.C. It simply consisted in

shifting the letters through a constant number k of characters: *i.e.* with $k = 3$, 'A' was replaced by 'D', 'B' by 'E', 'C' by 'F', etc. Once again, the knowledge of the general technique is enough to break the confidentiality. However, this *Caesar's cipher* gives a good taste of the cryptographic techniques used during the two following millennia (except the last two decades). Indeed, up to a recent past, confidentiality was achieved using more or less intricate combinations of permutations and substitutions. The *conventional cryptography*, currently in use, is still in the same vein.

Transpositions/Permutations

For a *transposition/permutation cipher* of length ℓ, a message is split into blocks of size ℓ, on which one applies the same permutation π among the indices between 1 and ℓ: for a block $m = m_1 \dots m_\ell$, the ciphertext is $c = m_{\pi(1)} \dots m_{\pi(\ell)}$. The permutation π has to be kept secret between interlocutors. However, such a cipher is weak because it preserves the frequency distribution of each character. The *Lacedemonian scytale* provides an example of *permutation cipher*.

Substitutions

Another technique mostly used to design block ciphers is the *substitution* of blocks of characters. If the substituted blocks are of size 1, one talks about a *mono-alphabetic substitution*. The *Caesar's cipher* is the most famous example, which replaces each letter m_i of the message $m = m_1 m_2 \dots m_n$ by the letter $c_i = m_i + k \bmod 26$, where k is the shift parameter. According to folklore, Caesar used $k = 3$.

The *affine cipher* generalizes this latter cipher to $c_i = am_i + b \bmod 26$, where (a, b) is the secret parameter, also called the *secret key*.

But, the substitution may involve larger blocks. For example, the *Hill cipher* encodes blocks of size ℓ as vectors in $M \in \mathbb{Z}_{26}^{\ell}$. Then one encrypts by multiplying the plaintext vector M by an invertible square matrix K, $C = KM$.

More intricate variants appeared using many substitution mappings, instead of only one as for previous examples. The *Vigenère cipher* is a classical *poly-alphabetic substitution*. It involves several distinct *shifting ciphers* according to the position of the character: $c_i = m_i + k_{i \bmod \ell} \bmod 26$, where $k = k_0 \dots k_{\ell-1}$ is a keyword of length ℓ (the secret key). Beaufort has designed a variant of Vigenère, $c_i = k_{i \bmod \ell} - m_i \bmod 26$, which is its own inverse.

Unfortunately, when one uses such techniques to encrypt messages in a natural language, the high redundancy may help an adversary to recover the keyword. Indeed, some statistics of the plaintext are preserved in the ciphertext. Therefore, Kasiski provided a general technique for cryptanalyzing poly-alphabetic ciphers, with repeated keywords, such as the Vigenère cipher: if the period between two identical mappings is not two large, one can recover that period as well as each mapping which simply consists of a mono-alphabetic substitution.

Specialized Devices and Rotor-Machines

In the 18th century, appeared dedicated devices to encrypt and decrypt more efficiently. The most famous is the *Jefferson cylinder*, which implements a poly-alphabetic substitution at no computational cost for the parties: it consists of 36 disks on which the 26 letters A–Z are written in a random ordering, distinct for each disk. All the disks rotate around a same axis.

The sender rotates the wheels so that the plaintext appears along a reference line, along the cylinder's length. The 25 other line positions each defines a ciphertext. Then, with the same cylinder device, the recipient rotates the wheels to obtain the ciphertext written in a line, and the plaintext is the only intelligible message among the 25 other lines. Alternatively, they also may agree on a common offset to uniquely define the plaintext.

Then appears the most dominant technique of the World War II: the rotor-based machines. A plaintext is encrypted through the successive rotors which each performs a mono-alphabetic substitution, for a fixed position. Therefore, for fixed rotor positions, the rotors implement a mono-alphabetic substitution which is the composition of the substitutions defined by individual rotors. To provide a poly-alphabetic substitution, after any encipherment of a character, rotors move, which therefore provides a new mapping. The most important property of the rotor-based machines is the long period between two identical mappings, which avoids Kasiski attacks.

As said above, many rotor-based machines have been implemented during the World War II, such as the famous German cipher device, Enigma.

Kerckhoffs' Principles

Unfortunately, none of all those ciphers resisted against adversaries who exactly knew the mechanism of the transformation. Indeed, the breaking of Enigma by Alan Turing has been helped by the robbery of a machine in a submarine. However in 1883, Kerckhoffs claimed that the security of a cryptosystem should not rely on the secrecy of this latter, but just on the secrecy of a small parameter, the *secret key*.

Nevertheless, until a recent past, cryptographic applications were limited to military people. Even if they claim not to require the secrecy of the schemes for security concerns, they still keep them secret. But for such a specific class of people, it is possible to assume the secrecy of the mechanisms. However, since several decades, many people have become aware of the importance and the need of secrecy and authentication, not only for military applications. First industrial/commercial people wanted to be able to exchange critical data secretly. And now, everybody would like to be able to sell and buy over the Internet, or simply discuss while preserving his private life.

1.2 Conventional Cryptography

In the mid 1970s, the American government took care of that industrial need of confidentiality, and asked for an Encryption Standard. Such a standard has

therefore been developed in cooperation with IBM. This "Data Encryption Standard" [50], the well-known DES, is the first commercial-grade algorithm, officially defined by the American Standard FIPS 46–2, with openly and fully specified implementation details, as required by the Kerckhoffs' principles. The confidentiality thus relies on a 56-bit secret key shared between the two parties. Then followed FEAL (Fast Data Encipherment Algorithm) [71] the Japanese alternative to DES, IDEA (International Data Encryption Algorithm) [34], SAFER (Secure and Fast Encryption Routine) [38], etc. Unfortunately, no formal security can really be proven, even if a theory is beginning to capture some kinds of attacks (*e.g.* the decorrelation technique [78]).

Such a conventional encryption can be modelled as presented on Figure 1, where $\mathcal{E}$ is the encryption device, $\mathcal{D}$ the decryption device and k the common secret, shared between both parties. The basic security requirement is the impossibility, for anybody who does not know the secret key k, to recover the plaintext m from the ciphertext c.

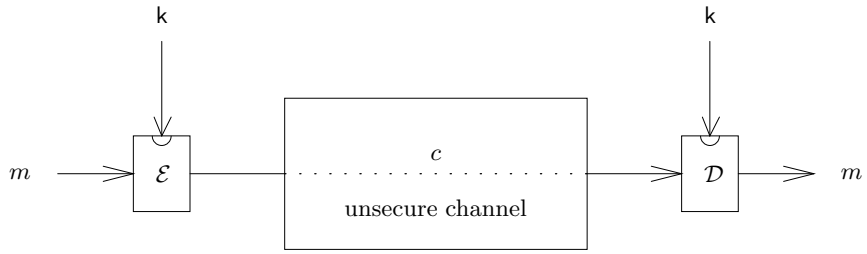

Fig. 1. Conventional Encryption

More formally, one would like that no information about the plaintext m would be leaked in the ciphertext c. Unfortunately, Shannon [70] showed that such a security level can only be reached if one uses a secret key k as long as the message to encrypt. Furthermore, this key has to be renewed for each new message to be encrypted. Such a *perfect encryption scheme* exists, and is called the *Vernam Cipher* [79], a.k.a. the *one-time pad*:

- let $m \in \{0,1\}^n$ be an n-bit long message to be encrypted;
- the two parties share a common secret $\mathsf{k} \in \{0,1\}^n$ of the same length as the message m;
- the ciphertext is simply $c = m \oplus \mathsf{k}$;
- it can easily be decrypted by the recipient since $m = c \oplus \mathsf{k}$.

However, that perfect security is not practical, and cannot be used for large-scale communications.

1.3 Practical/Provable Security: the Limit of Mathematics

Hopefully, that impossibility of *perfect secrecy* does not close the cryptographic research: adversaries are not powerful but limited in both computational power and time. Therefore, we can consider *practical security* that prevents attacks from real adversaries.

With *provable security*, one would like, to prevent, at least, any kind of attacks, known and unknown, that an adversary could perform in "reasonable" time. Unfortunately, as said above, no general security analysis can be driven about conventional cryptographic schemes: one can just prevent some restrictive classes of attacks. Therefore, nothing guarantees that no attack can ever be found against a scheme. This limitation is mainly due to the mathematics. Indeed, mathematics are the main tool to analyze the security of cryptographic schemes, trying to study the probability distributions of the ciphertexts, the plaintexts and the keys. But such analyses cannot take advantage of any limitation in time of the adversary.

1.4 Asymmetric Cryptography: on the Importance of Mathematics

In 1976, Diffie and Hellman [15] suggested to extend the Kerckhoffs' principles, with the remark that in an encryption scheme, one just wants to protect the decryption process. Why the encryption phase should be secret or use secret information?

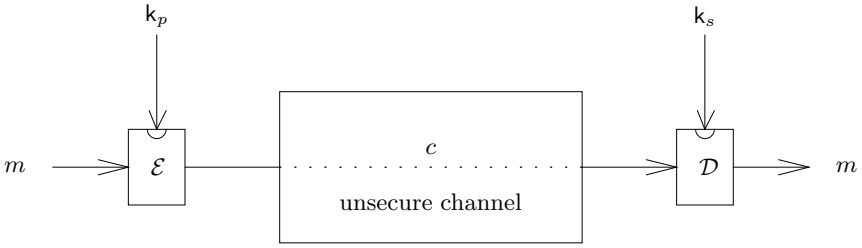

Fig. 2. Public-Key Encryption

Let us follow that suggestion with the encryption model described on Figure 2. It consists of

- an encryption phase $\mathcal{E}$, which allows anybody to transform a plaintext m into an unintelligible ciphertext c
- a decryption phase $\mathcal{D}$, which allows the owner of the secret data k_s to recover the plaintext from the ciphertext c.

Of course, the encryption phase $\mathcal{E}$ has be related to the secret data k_s, since it is clear that any encryption process has to be specific to the recipient, but in a public way. Therefore, it uses a public data k_p. The pair $(\mathsf{k}_p, \mathsf{k}_s)$ is a pair of matching secret/public keys, associated to a specific user. Hence, the name of *public-key cryptography*.

However, whereas k_s and k_p have to be related, it should be *impossible* to recover k_s from k_p: this impossibility can be guaranteed by a computational problem, *i.e.* a problem that is difficult to solve in practice.

1.5 Outline of the Paper

In this paper, we first develop this Diffie-Hellman's suggestion of public-key cryptography while precising the new requirements. Then, we show how mathemat-

ics, and particularly the algorithmic number theory, helped to actually provide public-key cryptosystems.

2 PUBLIC-KEY CRYPTOGRAPHY

In public-key cryptography, each people, say Alice, owns a pair of matching secret and public keys $(\mathsf{k}_p, \mathsf{k}_s)$, where k_p has to be widely published to belong to Alice, while k_s has to be kept secret by Alice. Thanks to these keys, one hopes to be able to achieve confidentiality with an encryption scheme and authentication with a digital signature scheme.

2.1 Public-Key Encryption

The aim of a public-key encryption is to allow anybody who knows the public key k_p of Alice to send her a message that she will be the only one able to recover it, thanks to her private key k_s.

Definition

A public-key encryption scheme can be formally defined by the three following algorithms (as depicted on Figure 2):

- The *key generation algorithm* G. On input 1^k, where k is the security parameter, the algorithm G produces a pair $(\mathsf{k}_p, \mathsf{k}_s)$ of matching public and secret keys. Algorithm G is probabilistic.
- The *encryption algorithm* $\mathcal{E}$. Given a message m and a public key k_p, $\mathcal{E}$ produces a ciphertext c of m. This algorithm may be probabilistic.
- The *decryption algorithm* $\mathcal{D}$. Given a ciphertext c and the secret key k_s, $\mathcal{D}$ gives back the plaintext m. This algorithm is necessarily deterministic.

Basic Security

Informally, one would like that nobody can recover the whole plaintext from a ciphertext, except the designated recipient. But in some cases, that security notion is not enough: let us consider the situation where we know that the ciphertext encrypts "yes" or "no" (or "sell"/"buy"), one bit of information about the plaintext reveals the whole plaintext. Therefore, one could furthermore require that nobody can get *any* information about the plaintext from a ciphertext. Both security notions will be more formally defined later, under the name of *one-wayness* and *semantic security* (a.k.a. *indistinguishability of encryptions* or *polynomial security* [24]) respectively.

On the other hand, the adversary may have access to some additional information. In a public-key setting, she can get the encryption of any plaintext of her choice. But one can furthermore assume that the adversary can get the decryption of some ciphertexts of her choice. The first scenario is called the *Chosen-Plaintext Attack* while the second one is named the *Chosen-Ciphertext Attack* [48, 61].

2.2 Digital Signature

Digital signature schemes are the electronic version of handwritten signatures for digital documents: a user's signature on a message m is a string which depends on m and the secret key k_s of the user, in such a way that anyone can check the validity of the signature by using the public key k_p only.

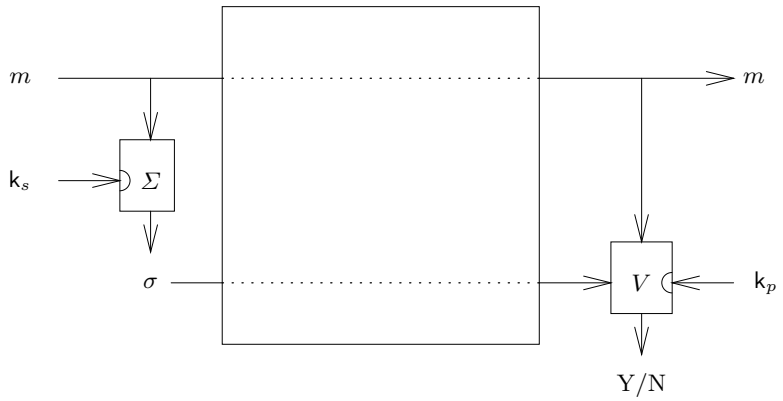

Fig. 3. Digital Signature

Definition

A signature scheme is usually defined by the three following algorithms (as depicted on Figure 3):

- The *key generation algorithm* G. On input 1^k, where k is the security parameter, the algorithm G produces a pair $(\mathsf{k}_p, \mathsf{k}_s)$ of matching public and secret keys. Algorithm G is probabilistic.
- The *signing algorithm* Σ. Given a message m and a pair of matching public and secret keys $(\mathsf{k}_p, \mathsf{k}_s)$, Σ produces a signature σ. The signing algorithm might be probabilistic.
- The *verification algorithm* V. Given a signature σ, a message m and a public key k_p, V tests whether σ is a valid signature of m with respect to k_p. In general, the verification algorithm need not be probabilistic.

Basic Security

As above, everybody has an intuition about the security notion that a signature scheme should satisfy. First, one would like that only the owner of the secret key k_s related to the public key k_p could produce an accepted signature σ for a message m. But according to the choice of the message m, many kinds of security notions have been defined. Furthermore, according to the additional information the adversary may have (access or not to a signature oracle), many scenarios of attacks have been formalized [27], as we will see later.

3 NEW REQUIREMENTS

Above descriptions just follow the Diffie-Hellman's suggestion but do not give any solution. Since we have given a more precise explanation of the *public-key cryptography*, we can focus on some new tools which are the basis of an actual achievement: the *one-way functions* and the *trapdoor one-way functions*.

3.1 One-Way Functions

The pairs of matching public/secret keys have to be related. The public key k_p is derived from the secret key k_s. But since k_p is thereafter published, to remain secret, k_s has to be computationally unrecoverable from k_p.

DEFINITION 1 (One-Way Functions). A function f is said *one-way* if for any x one can easily compute $y = f(x)$. But for a given $y = f(x)$, nobody can recover any z such that $y = f(z)$.

But what does that mean, *easy* and *difficult*? Mathematics only study the existence of a pre-image but do not care about the means to get it. Hopefully, the complexity theory addresses this problem with the classes of complexity $\mathcal{P}$ and $\mathcal{NP}$, namely the $\mathcal{NP}$-complete problems. Indeed, those classes can informally be seen as follows

- the class $\mathcal{P}$ contains the problems which can be solved in polynomial time (in the size of the data)
- the class $\mathcal{NP}$ contains the problems for which a solution can be checked in polynomial time
- the $\mathcal{NP}$-complete problems are the strongest problems in the class $\mathcal{NP}$.

More precisely, any problem in $\mathcal{NP}$ can be polynomially reduced to any $\mathcal{NP}$-complete problem: if one $\mathcal{NP}$-complete problem can be solved in polynomial time, then any problem in $\mathcal{NP}$ can be solved in polynomial time, and therefore $\mathcal{NP} = \mathcal{P}$. However, this latter equality is the strongest open problem of the complexity theory. But both classes are widely believed to be distinct: the $\mathcal{NP}$-complete problems cannot be solved in polynomial time. Furthermore, no one knows better algorithms than exponential ones to solve any $\mathcal{NP}$-complete problem.

Such an $\mathcal{NP}$-complete problem [22] seems a good candidate as a one-way function. Indeed, in general, given a solution x, it is easy to define an instance y which admits x as a solution. However, given that instance y, the $\mathcal{NP}$-completeness seems to claim that there is no efficient algorithm to find a solution.

Unfortunately, the $\mathcal{NP}$-completeness only deals with the worst case, whereas the recovery of the secret key k_s should be always difficult, except maybe for a negligible fraction of cases.

3.2 Trapdoor One-Way Functions

Let us come back to encryption, and more precisely to the Diffie-Hellman's suggestion [15]. They claimed that the encryption phase should be available to anybody: $c = \mathcal{E}(\mathsf{k}_p, m)$ where $\mathcal{E}$ is the public encryption process and k_p the public key of the recipient. While the decryption can only be proceeded by the designated recipient: $m = \mathcal{D}(\mathsf{k}_s, c)$ where $\mathcal{D}$ is the public decryption algorithm which requires the secret key k_s to correctly decrypt c.

For such an application, one needs a function $f(\cdot) = \mathcal{E}(\mathsf{k}_p, \cdot)$ that anybody can easily compute, but such that the inversion is impossible for anybody, except for the one who knows k_s, a trapdoor.

DEFINITION 2 (Trapdoor One-Way Functions). A function f is said *trapdoor one-way* if it is a *one-way* function, except for those who know a *trapdoor information* t: knowing t, for any given $y = f(x)$, one can easily compute a $z = g(t, y)$ which satisfies $y = f(z)$.

As we have previously seen, the complexity theory provides convenient definitions and classes to encompass "easy" and "difficult" tasks. But the $\mathcal{NP}$-complete problems cannot all be used. Firstly, because complexity theory analyzes problems in an asymptotic framework. Therefore, a problem may become difficult in practice only for very large instances, which cannot be used for actual cryptographic protocols. Secondly, because $\mathcal{NP}$-completeness only says that the worst cases cannot be efficiently solved. But the worst cases may be rare, whereas one would like a problem for which almost all the instances are difficult to solve.

Nevertheless such convenient $\mathcal{NP}$-complete problems have been identified: the problem of decoding an arbitrary linear code [40], the Knapsack or Subset Sum problem [43], the Permuted Kernel Problem [68], the Syndrome Decoding [75] or the Permuted Perceptrons Problem [53]. However, their application to cryptography is mainly restricted to interactive authentication (by zero-knowledge proofs of knowledge [25]), but cannot be efficiently used for encryption or signature.

4 THE ALGORITHMIC NUMBER THEORY

Hopefully, even if mathematics cannot help to analyze the security of cryptographic schemes, they provide candidates as one-way and trapdoor one-way problems with namely the integer factorization and the discrete logarithm problem.

4.1 The Integer Factorization

A first simple candidate that may come to mind is the factorization of integers: while it is easy to multiply two prime integers p and q to get the product $n = p \cdot q$, it is not simple to factorize n into its prime factors p and q.

Indeed, the multiplication between two integers p and q, both of size k, just requires a quadratic amount of time in k. However, the factorization of any

integer n, which consists of writing n as a product of prime integers $n = \prod p_i^{v_i}$ is a little more intricate.

First, it is a well-known result of arithmetic that any integer $n \geq 2$ has a factorization, as a product of prime powers $n = \prod p_i^{v_i}$, where the p_i are distinct primes (which can only be divided by 1 and themselves) and v_i are the valuations, represented by positive integers. Furthermore, this factorization is unique up to a permutation of the factors.

Generic Techniques

For a long time, many methods have existed for factorizing integers, from the trial-division to the Pollard's methods [57]. But their complexity is very bad: let us see some of them for an integer n, for which p is the smallest prime factor

- the trial-division requires p divisions to find the first prime factor, and up to $\sqrt{n}$ divisions to fully factorize n
- the Pollard's ρ method [57], later improved by Brent [11], finds p after $\mathcal{O}(\sqrt{p})$ iterations and fully factorizes n in $\mathcal{O}(n^{0.106})$, on average
- then some methods have been dedicated to special integers, such as the $p-1$-method which is quite fast when $p-1$ is smooth, and the $p+1$-method [80].

Anyway, all these methods that simply consist in trying to divide n by many primes provide algorithms which require an exponential amount of time w.r.t. k the size of n ($k = \ln n$), typically $\exp(k/2)$ or $\exp(k/4)$ in the special case that $n = pq$, the product of two primes of similar size. Indeed, those numbers seem the strongest to factorize, then they will be used for cryptographic purpose.

Improved Techniques

More recently, new methods appeared:

- First, using the remark that the $p-1$ method is quite general, one can use it on any group related to the prime factors of the integer n to factorize, such as an elliptic curve modulo n [46]. Therefore, on an elliptic curve modulo n, we can use the addition law as if it would be modulo a prime. But at some time, this addition is impossible. Such an accident reveals a factor of n. This method is called ECM and finds a prime factor p of n in $\mathcal{O}(L(p)^{\sqrt{2}})$ where $L(x) = \exp(\sqrt{\ln x \ln \ln x})$.
- Then come the methods based on congruential relations which likely lead to a factor of n, such as $x^2 = y^2 \bmod n$. The first algorithm using such relations is CFRAC (Continued Fraction Algorithm [47, 60]) which exploits some properties of the continued fraction development of $\sqrt{n}$. This method factorizes n in $\mathcal{O}(L(n)^{\sqrt{7}/2})$. It has been used to perform the first record: F_7, a 39-digit number.
- Other mechanisms have been used to collect such relations, by sieving. First the quadratic sieve [59] has been proposed, with many optimizations (Multiple Polynomial QS, Large Prime Variation). That technique, with a time complexity in $\mathcal{O}(L(n))$ has been used to establish many records, up to a 129-digit number in 1994 [1].

Currently, the most efficient algorithm is based on sieving on number fields. The Number Field Sieve (NFS) method [36] has a complexity in

$$\mathcal{O}(\exp((1.923 + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3})).$$

In practice, it becomes more efficient than the quadratic sieve for 130-digit numbers. It has been used to establish the last record, in august 1999, by factorizing a 155-digit integer, product of two 78-digit primes [12].
The factorized number, indicated by RSA-155, was taken from the "RSA Challenge List", which is used as a yardstick for the security of the RSA cryptosystem (see later) which is used extensively in hardware and software to protect electronic data traffic such as in the international version of the SSL (Security Sockets Layer) Handshake Protocol.
This latter record is very important since 155 digits correspond to 512 bits. And this is the size which is in use in almost all the implementations of the RSA cryptosystem (namely for actual implementations of SSL on the Internet).

```
RSA-155 =
  109417386415705274218097073220403576120037329454492O\
  599091384213147634998428893478471799725789126733249 7\
  625752899781833797076537244027146743531593354333897
  =   102639592829741105772054196573991675900\
      71656780803806680334193352179071130777 9
    * 106603488380168454820927220360012878679\
      207958575989291522270608237193062808643
```

Therefore, from the current state of the science, factorization is believed to be a difficult problem, especially for products of two primes of similar sizes larger than 384 bits.

4.2 The Discrete Logarithm Problem

Let $\mathcal{G}$ be a cyclic group of order q, with an internal law denoted multiplicatively. This means that

$$\mathcal{G} = \langle g \rangle = \{1, g, g^2, \ldots, g^{q-1}\},$$

for some g, called a generator of the cyclic group $\mathcal{G}$. Therefore, for any $y \in \mathcal{G}$, there exists at least an $x \in \mathbb{Z}$ such that $y = g^x$. One defines

$$\log_g y \stackrel{\text{def}}{=} \min\{x \in \mathbb{Z} \,|\, y = g^x\}.$$

Thanks to the *square-and-multiply* technique [30], for any integer x, it is easy to compute g^x: if $x = \sum x_i 2^i$, $g^x = \prod g_i^{x_i}$ where $g_0 = g$ and $g_i = g_{i-1}^2 (= g^{2^i})$. Indeed, if $\ell = |x|$, this method requires ℓ squares and less than ℓ multiplications (but just $\ell/2$ on average).

However, how do we compute $\log_g y$ for a given $y \in \mathcal{G}$?

Generic Techniques

Many methods are known for computing discrete logarithms (the reader is referred to a recent review [77] for more details), such as the *Baby Steps, Giant Steps* technique [69]: $x = \log_g y$ is known to belong in $\{0, \ldots, q-1\}$, and therefore there exist $a, b \in \{0, \ldots, s\}$, where $s = \lceil \sqrt{q} \rceil$, such that $x = a + bs$. Thus, building the two sets

$$\mathcal{S}_1 = \{g^a \mid a \in \{0, \ldots, s\}\}$$
$$\mathcal{S}_2 = \{y(g^{-s})^b \mid b \in \{0, \ldots, s\}\}$$

after a sort, one can easily find a collision $c \in \mathcal{S}_1 \cap \mathcal{S}_2$: $c = g^a = yg^{-sb}$ for some pair (a, b), which leads to $y = g^{a+sb}$. However, the time complexity of this method is in $\mathcal{O}(\sqrt{q} \ln \sqrt{q})$, because of the sort. Furthermore, the space complexity is also in $\mathcal{O}(\sqrt{q})$. This latter, which quickly becomes very huge, makes this technique impractical as soon as q is over hundred bits.

Pollard's ρ and λ methods [58] avoid this large space storage and the sort. The ρ method is well-known and provides a (heuristic) time complexity in $\mathcal{O}(\sqrt{q})$. The λ method is less known (or a.k.a. the *method for catching kangaroos*). They are still both impractical as soon as q is over hundred and twenty bits.

When the order of g is not prime, a divide-and-conquer technique in all the subgroups of order the factors of q can be applied: the *Pohlig-Hellman decomposition* [52]. For example, let us assume that $q = \prod_1^\ell q_i$ (this can be extended to greater valuations), one computes $x_i = \log_{g_i} y$, where $g_i = g^{q/q_i}$, for $i = 1, \ldots, \ell$. The Chinese Remainder Theorem (which will be recalled later) gives a solution of the simultaneous congruences, $x = x_i \bmod q_i$, for $i = 1, \ldots, \ell$. The overall complexity is dominated by the cost of finding the discrete logarithm for the largest prime factor.

Suitable Groups

This discrete logarithm problem needs a suitable cyclic group $\mathcal{G}$, of order with at least a large prime factor. The first group in use in cryptography has been a cyclic subgroup of multiplicative groups of finite fields $\langle g \rangle \subset \mathbb{Z}_p^\star$, where p is a large prime such that $p - 1$ admits a large prime factor q.

More recently, other groups have been introduced, such as the algebraic varieties of (hyper)-elliptic curves and some ideals in number fields [29]. Indeed, in 1985, N. Koblitz [31] and V. Miller [45] have proposed to use elliptic curves in cryptography, where the underlying problem is the discrete logarithm over points $(x, y) \in \mathbb{F}^2$ which satisfy an equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}),$$

equipped with an Abelian group structure. In 1988, N. Koblitz was the first to suggest using hyperelliptic curves [32, 33].

Specific Techniques

Of course, specific techniques have been designed to address the particularities of the underlying group. Therefore, in the particular case of subgroups of $\mathbb{Z}_p^\star$, the quadratic and number field sieves can be used. This later provides a time complexity in

$$\mathcal{O}(\exp((1.923 + o(1))(\ln p)^{1/3}(\ln \ln p)^{2/3})).$$

However, the quadratic sieve [35] is still the most efficient for current records, and has been used by R. Lercier and A. Joux (CELAR, France) to establish the last one: discrete logarithms modulo a 90-digit prime. More precisely,

```
p = \lfloor 10^{89} \pi \rfloor+ 156137
  = 31415926535897932384626433832795028841971693993\
    75105820974944592307816406286208998629596 19
g = 2
y = \lfloor 10^{89} e \rfloor
  = 27182818284590452353602874713526624977572470936\
    99959574966967627724076630353547594571382 17
  = g^176713807211421696273204823407162027230205795\
      24499141574938447166779186585383741881010 93
```

About elliptic curves, for a long time, only the generic techniques were known, but some new algorithms also appear in some particular cases: for anomalous elliptic curves [64], for supersingular curves, where the discrete logarithm problem can be reduced to the finite field setting, because of the Frobenius map which has a trace zero [42], but also for curves of trace one [74] and more recently when many automorphisms exist on the curve [17].

However, the generic ρ method with distinguished points has been used to establish the latest record, by R. Harley (INRIA, France). This record solves the so-called ECC2K-108 challenge from the list provided by the Canadian company Certicom. This challenge can be defined as follows:

- Let the curve C be $y^2 + xy = x^3 + x^2 + 1$ over $\mathbb{F}_{2^{109}}$.
- Represent $\mathbb{F}_{2^{109}}$ as $\mathbb{F}_2[t]/(f(t))$ where $f(t) = t^{109} + t^9 + t^2 + t + 1$ and is irreducible over $\mathbb{F}_2$.
- Then the following two points:

```
P = (0x0478C46CC96338CED91565E17257,
     0x1E7965E4A3AFB73A48FC9AB790E9)
Q = (0x1FF0CE5EC61893F2119C3077C59E,
     0x1F20E9B010AC691C9B87B438241D)
```

  are on C, where the coordinates have been written as hexadecimal integers by reducing modulo f and setting t to 2.

The problem is to find the logarithm of `Q` to the base `P`. The problem takes place in the subgroup of order $q = 324518553658426701487448656461467$, which is a prime. The solution is $474556618962230452997483160189 41 \bmod q$.

Hyper-elliptic curves (or more precisely their Jacobian) have been proposed in the hope of using smaller fields, thanks to a high genus which increases the number of points. And thus, both the computation load and the size of data are smaller. However, new methods [17, 23] have been recently proposed to compute more efficiently discrete logarithms in those groups.

Therefore, from the current state of the art of the discrete logarithm algorithms, one uses subgroups $\langle g \rangle$ of either

- the multiplicative group $\mathbb{Z}_p^\star$, where p is a 512-bit prime such that $p-1$ admits a 160-bit prime factor, which is the order of g;
- an elliptic curve, over the field $\mathbb{F}_{p^k}$ where p is a prime and $k \times |p| \simeq 160$, such that the cardinality admits a large prime factor.

5 TRAPDOOR ONE-WAY PROBLEMS

Unfortunately, both problems, the integer factorization and the discrete logarithm problem, are just one-way. And no information can make them easier.

However, some algebraic structures are based on the factorization of an integer n, where some computations are difficult without the factorization of n, but easy with it: the finite ring $\mathbb{Z}_n$ which is isomorphic to the product ring $\mathbb{Z}_p \times \mathbb{Z}_q$ if $n = p \cdot q$.

About the discrete logarithm, it helps to solve the so-called Diffie-Hellman problem [15], the first to have been proposed for cryptology purpose.

5.1 Finite Rings: the RSA problem

Thanks to the relation of equivalence, mod$n$, for any integer $n \in \mathbb{Z}$, one can define the quotient ring $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$, where $a, b \in \mathbb{Z}$ are in the same class if $a = b \bmod n$. In the following, we always identify a class of $\mathbb{Z}_n$ with any integer of this class.

Addition, Subtraction, Multiplication, Division

Many results are known about this quite rich structure. First, one can efficiently check equality, add, subtract or multiply two elements. The following results claim that the division is also easy, by any invertible element.

Theorem 3 (Bezout). *Let a and b be two integers in $\mathbb{Z}$ then*

$$(\exists u, v \in \mathbb{Z})(au + bv = 1) \Leftrightarrow \gcd(a, b) = 1.$$

More generally,

Theorem 4. *Let a and b be two integers in $\mathbb{Z}$ then*

$$(\exists u, v \in \mathbb{Z})(au + bv = d) \Leftrightarrow \gcd(a, b) \mid d.$$

where $x \,|\, y$ means that there exists z such that $y = xz$. In particular, for any pair $(a, b) \in \mathbb{Z}^2$ there exists a pair $(u, v) \in \mathbb{Z}^2$ such that $au + bv = \gcd(a, b)$. Furthermore, the (extended) Euclidean algorithm is an efficient algorithm which, given such a pair (a, b), provides both $\gcd(a, b)$ and the pair (u, v) such that $au + bv = \gcd(a, b)$. Its time complexity is linear in the size of the input, *i.e.* in $\mathcal{O}(|a| + |b|)$.

As a consequence, for any $x \in \mathbb{Z}_n$, x is invertible (relies in $\mathbb{Z}_n^\star$) if and only if

$$\begin{aligned} & (\exists y \in \mathbb{Z}_n) \quad xy = 1 \bmod n \\ \Leftrightarrow\ & (\exists y \in \mathbb{Z}_n)(\exists k \in \mathbb{Z}) \quad xy + kn = 1 \\ \Leftrightarrow\ & \gcd(x, n) = 1 \end{aligned}$$

Therefore, the extended Euclidean algorithm efficiently provides the inverse of any $x \in \mathbb{Z}_n^\star$, which allows an efficient division.

Furthermore, the following corollary comes:

COROLLARY 5. *p is a prime $\Leftrightarrow$ $\mathbb{Z}_p$ is a field.*

Powers and Roots modulo n

When n is not a prime, $\mathbb{Z}_n$ is not a field, but the Chinese Remainder Theorem provides the explicit structure of $\mathbb{Z}_n^\star$:

Theorem 6 (Chinese Remainder Theorem). *Let $n = m_1 m_2$ be a composite integer, where $\gcd(m_1, m_2) = 1$. Then the ring $\mathbb{Z}_n$ is isomorphic to the product ring $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2}$ and the following morphisms,*

$$\begin{aligned} f : \mathbb{Z}_n &\longrightarrow \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \\ x &\longmapsto (x \bmod m_1 \ , \ x \bmod m_2) \end{aligned}$$

$$\begin{aligned} g : \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} &\longrightarrow \mathbb{Z}_n \\ (a \ , \ b) &\longmapsto aum_2 + bvm_1 \bmod n \\ \text{where } u = m_2^{-1} \bmod m_1 &\text{ and } v = m_1^{-1} \bmod m_2 \end{aligned}$$

are isomorphisms of rings and $f \circ g = Id_{\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2}}$ while $g \circ f = Id_{\mathbb{Z}_n}$.

About the multiplicative group, one gets the following corollary

COROLLARY 7. *Let $n = m_1 m_2$ be a composite integer, where $\gcd(m_1, m_2) = 1$.*

$$(\mathbb{Z}_n, +, \times) \simeq (\mathbb{Z}_{m_1}, +, \times) \times (\mathbb{Z}_{m_2}, +, \times),$$

and thus

$$(\mathbb{Z}_n^\star, \times) \simeq (\mathbb{Z}_{m_1}^\star, \times) \times (\mathbb{Z}_{m_2}^\star, \times).$$

When we have a group, it is useful to know its cardinality to apply the Lagrange's theorem.

Theorem 8 (Lagrange's Theorem). *Let $\mathcal{G}$ be a group denoted multiplicatively. If we denote by c its cardinality, for any element $x \in \mathcal{G}$, $x^c = 1$.*

Therefore, we denote by $\varphi(n)$ the cardinality of the multiplicative group $\mathbb{Z}_n^\star$:

DEFINITION 9 (Euler's Totient Function).

$$\varphi(n) \stackrel{\text{def}}{=} \text{Card}(\mathbb{Z}_n^\star).$$

Thanks to the Chinese Remainder Theorem, this function is weakly multiplicative:

$$\gcd(m_1, m_2) = 1 \Rightarrow \varphi(m_1 m_2) = \varphi(m_1)\varphi(m_2).$$

Since $\varphi(p) = p - 1$ for any prime p, one can deduce that for any $n = \prod_1^\ell p_i^{v_i}$,

$$\varphi(n) = n \times \prod_1^\ell \left(1 - \frac{1}{p_i}\right).$$

Theorem 10. *The computation of $\varphi(n)$ is polynomially equivalent to the factorization of n.*

Proof. It is clear that the factorization of n easily leads to the value of $\varphi(n)$ with above formula. Furthermore, the Miller's algorithm [44] outputs the factorization of any n, given a multiple of $\varphi(n)$. □

Euler applied the Lagrange's theorem to the particular situation of the multiplicative group $\mathbb{Z}_n^\star$:

Theorem 11 (Euler's Theorem). *Let n be any integer, for any element $x \in \mathbb{Z}_n^\star$, $x^{\varphi(n)} = 1 \bmod n$.*

Therefore,

- let e be any integer relatively prime to $\varphi(n)$
- one takes $d = e^{-1} \bmod \varphi(n)$, which means that there exists an integer $k \in \mathbb{Z}$ such that $ed + k\varphi(n) = 1$
- for any $x \in \mathbb{Z}_n^\star$,

$$(x^e)^d = x^{ed} = x^{1-k\varphi(n)} = x \cdot (x^{\varphi(n)})^{-k} = x \bmod n.$$

As previously seen, the e-th power can be easily computed using the *square-and-multiply* method. Above relation allows to easily compute e-th roots, by computing d-th powers, where $ed = 1 \bmod \varphi(n)$. However, to compute e-th roots, one requires to know an integer d such that $ed = 1 \bmod \varphi(n)$. And therefore, $ed - 1$ is a multiple of $\varphi(n)$ which is equivalent to the knowledge of the factorization of n (Theorem 10).

The RSA Problem and the RSA Assumption

In 1978, Ronald Rivest, Adi Shamir and Leonard Adleman [63] defined the following problem.

DEFINITION 12 (The RSA Problem). Let $n = pq$ be the product of two large primes and e an integer relatively prime to $\varphi(n)$. For a given $y \in \mathbb{Z}_n^\star$, compute $x \in \mathbb{Z}_n^\star$ such that $x^e = y \bmod n$.

We have seen above that with the factorization of n (the trapdoor), this problem can be easily solved. However nobody knows whether the factorization is required, but nobody knows how to do without it either:

DEFINITION 13 (The RSA Assumption). For any product of two large primes, $n = pq$, large enough, the RSA problem is intractable (presumably as hard as the factorization of n).

5.2 The Diffie-Hellman Problem

In 1976, when Whitfield Diffie and Martin Hellman [15] suggested the asymmetric cryptography, they proposed the following problem:

DEFINITION 14 (The Diffie-Hellman Problem).
Let $\langle g \rangle \subset \mathcal{G}$ be a cyclic group. Given $A = g^a, B = g^b \in \langle g \rangle$, compute $C = g^{ab}$.

This problem is assumed to be intractable for any suitable group, where the discrete logarithm problem is intractable. However, even if with the discrete logarithm a of A in basis g it is easy to compute C, which is equal to B^a, the Diffie-Hellman problem is not proven equivalent to the discrete logarithm problem, or strictly easier either.

Another problem has been more recently defined [10], known as the *Decisional Diffie-Hellman Problem*:

DEFINITION 15 (The Decisional Diffie-Hellman Problem). Let $\langle g \rangle \subset \mathcal{G}$ be a cyclic group. Given $A = g^a, B = g^b, C = g^c \in \langle g \rangle$, decide whether $C = g^{ab}$.

6 RECAPITULATION

Let us make a final review of all the problems we've just seen, splitting them in two families, the factorization-like problems and the discrete logarithm-like ones.

6.1 The Factorization

With the current knowledge from algebraic theory and algorithmic, the factorization problem is believed to be intractable, at least for RSA-integers (products of two primes of similar size) over more than 700 bits:

DEFINITION 16 (Factorization – **FACT**(n)).
Given an integer $n = pq$ product of two large primes p and q, find these prime factors p and q.

This provides a *one-way function*. The factorization also provides a trapdoor to the following problem

DEFINITION 17 (RSA(n, e)-Problem – **RSA**(n, e)).
Let us given an integer $n = pq$ product of two large primes p and q, as well as an exponent e, relatively prime with $\varphi(n)$. For any $y \in \mathbb{Z}_n^\star$, find $x \in \mathbb{Z}_n^\star$ such that $x^e = y \bmod n$.

Indeed, the RSA(n, e)-problem is believed to be intractable without the factorization of n. However, given the factorization of n, it becomes quite easy. Therefore, the RSA function for any suitable pair (n, e)

$$\begin{aligned} f : \mathbb{Z}_n^\star &\longrightarrow \mathbb{Z}_n^\star \\ x &\longmapsto x^e \bmod n \end{aligned}$$

is a permutation onto $\mathbb{Z}_n^\star$, whose inversion is intractable unless one knows the factorization of n. Hence the name of *trapdoor one-way permutation*, which is a very useful property. However, RSA is the sole candidate for that kind of nice object!

Since $\mathbf{FACT}(n)$ provides a trapdoor to $\mathbf{RSA}(n, e)$ for any e, clearly we have $\mathbf{FACT}(n) \geq \mathbf{RSA}(n, e)$, where "$A \geq B$" means that a machine that can efficiently solve A, can be used to efficiently solve B.

6.2 The Discrete Logarithm Problem

Let us consider a suitable cyclic group of order q ($\langle g \rangle \subset \mathbb{Z}_p^\star$ or a subgroup of an elliptic curve, etc), with a generator g. The discrete logarithm problem on $\mathcal{G}$ is considered intractable:

DEFINITION 18 (The Discrete Logarithm Problem – $\mathbf{DL}(\mathcal{G})$).
Given $y \in \mathcal{G}$, find $x \in \mathbb{Z}_q$ such that $g^x = y$.

However, it provides a trapdoor to solve the Diffie-Hellman problem:

DEFINITION 19 (The Diffie-Hellman Problem – $\mathbf{DH}(\mathcal{G})$).
Given $A = g^a, B = g^b \in \mathcal{G}$, find $C = g^{ab}$.

It can also be stated as follows: let us given $A = g^a \in \mathcal{G}$, for any $B = g^b$ find $C = g^{ab}$. This problem is intractable, unless one knows a, since $C = B^a$. Furthermore, about this problem, even if one knows a candidate for the solution C, one cannot check its correctness:

DEFINITION 20 (The Decisional Diffie-Hellman Problem – $\mathbf{DDH}(\mathcal{G})$).
Given $A = g^a, B = g^b, C = g^c \in \mathcal{G}$, decide whether $c = ab \bmod q$.

Once again, the discrete logarithm provides a trapdoor and therefore

$$\mathbf{DL}(\mathcal{G}) \geq \mathbf{DH}(\mathcal{G}) \geq \mathbf{DDH}(\mathcal{G}).$$

REMARK 21. If $\mathcal{G} \subset \mathbb{Z}_n^\star$, for some $n = pq$,

$$\mathbf{DL}(n) \geq \mathbf{FACT}(n) + \mathbf{DL}(p) + \mathbf{DL}(q),$$

where $\mathbf{DL}(k)$ denotes $\mathbf{DL}(G)$ for some group $G \subset \mathbb{Z}_k^\star$. Therefore the discrete logarithm in some subgroup in $\mathbb{Z}_n^\star$ is stronger than both the factorization of n and the discrete logarithm in the subgroups modulo the prime factors of n: it combines the difficulty of both problems.

7 APPLICATION TO PUBLIC KEY CRYPTOGRAPHY

Since we have one-way problems (the factorization and the discrete logarithm) and trapdoor one-way problems (the e-th root and the Diffie-Hellman problems), asymmetric cryptography becomes reality.

7.1 Public Key Encryption

A public key encryption scheme is a triple $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ as defined in the second part. Let us more formally define the security notions that an encryption scheme should satisfy.

Security Notions

The first common security notion that one would like an encryption scheme to satisfy is the *one-wayness*: with just public data, an attacker cannot get back the whole plaintext of a given ciphertext. More formally, this means that for any adversary $\mathcal{A}$, her success in inverting $\mathcal{E}$ without the secret should be negligible, over the choice of m and the random coins of $\mathcal{E}$ and herself:

$$\mathsf{Succ}_{\mathcal{A}} = \Pr_{m}[\mathcal{A}(\mathcal{E}_{\mathsf{k}_p}(m)) = m].$$

However, many applications require more from an encryption scheme, namely the *semantic security* (a.k.a. *polynomial security/indistinguishability of encryptions* [24]), as already remarked in the introduction. This security notion requires the computational impossibility to distinguish between two messages, chosen by the adversary, which one has been encrypted, with probability significantly better than one half: her advantage, defined as below, where the adversary is seen as a 2-stage Turing machine $(\mathcal{A}_1, \mathcal{A}_2)$, should be negligible.

$$\mathsf{Adv}_{\mathcal{A}} = \Pr_{b}\left[\begin{array}{l}(\mathsf{k}_p, \mathsf{k}_s) \leftarrow G(1^k)\\ (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{k}_p) : \mathcal{A}_2(m_0, m_1, s, c) = b\\ c = \mathcal{E}_{\mathsf{k}_p}(m_b)\end{array}\right] - \frac{1}{2}.$$

Another notion has been thereafter defined, the so-called *non-malleability* [16], but this notion is equivalent to the above one in some specific scenarios [4, 8], then we don't detail it.

On the other hand, an attacker can play many kinds of attacks: she may just have access to public data, and then encrypt any plaintext of her choice (*chosen-plaintext attacks*) or moreover query the decryption algorithm (*adaptively/non-adaptively chosen-ciphertext attacks* [49, 61]). A general study of these security notions and attacks has been recently driven [4], we therefore refer the reader to this paper for more details.

Another scenario has been recently studied which involves many users that receive encryption (under their respective public keys) of related messages. So many encryptions may reveal some information about the plaintexts to an adversary that intercepts all the ciphertexts. However, recent papers [3, 2] have proven that if the scheme is semantically secure, it does not leak any information even in the multi-user scenario.

Some Examples

The RSA Encryption. When they defined the RSA problem, Rivest–Shamir–Adleman [63] wanted to propose a public-key encryption, thanks to the "trapdoor one-way permutation" property of the RSA function: the key generation algorithm produces a large composite number $n = pq$, a public key e, and a secret key d such that $e \cdot d = 1 \bmod \varphi(n)$. The encryption of a message m, encoded as an element in $\mathbb{Z}_n^\star$, is simply $c = m^e \bmod n$. This ciphertext can be easily decrypted thanks to the knowledge of d, $m = c^d \bmod n$. Clearly, this encryption scheme is *one-way*, under *chosen-plaintext attacks*, relative to the RSA problem. However, since it is deterministic, it cannot be semantically secure. Therefore, it cannot be proven secure in the multi-user scenario either.

Moreover, it is well-known to be weak against the *broadcast* attack when using a small exponent e [28], *e.g.* $e = 3$: with the encryptions of a message m under 3 different public keys $(n_1, e_1 = 3)$, $(n_2, e_2 = 3)$ and $(n_3, e_3 = 3)$, denoted by $c_i = m^3 \bmod n_i$, for $i = 1, 2, 3$, respectively. The Chinese Remainder Theorem provides the solution c of the simultaneous congruences $c = c_i \bmod n_i$. But $m^3 \bmod n_1 n_2 n_3$ is also a solution to this system. The uniqueness proves that $c = m^3 \in \mathbb{Z}$, since $m < m_i$ which implies that $m^3 < n_1 n_2 n_3$. An easy third root in $\mathbb{Z}$ helps to recover m.

The El Gamal Encryption. In 1985, El Gamal [18] designed a public-key encryption scheme based on the Diffie-Hellman key exchange protocol [15]: given a large prime p, and an element g in $\mathbb{Z}_p^\star$ of large prime order q, the key generation algorithm produces a random element $x \in \mathbb{Z}_q^\star$ as secret key, and a public key $y = g^x \bmod p$. The encryption of a message m, encoded as an element of $\langle g \rangle$, is a pair $(c = g^a \bmod p, d = y^a m \bmod p)$ for a random $a \in \mathbb{Z}_q$. This ciphertext can be easily decrypted thanks to the knowledge of x, $m = d/c^x \bmod p$. This encryption scheme is well-known to be *one-way*, under *chosen-plaintext attacks*, relative to the Diffie-Hellman problem (**DH**). It is also *semantically secure*, under *chosen-plaintext attacks*, relative to the Decisional Diffie-Hellman problem (**DDH**).

However, we had to wait a long time before any efficient proposal provably semantically secure under adaptively chosen-ciphertext attacks. The most famous one was just proposed two years ago [14], whose security is relative to the Decisional Diffie-Hellman problem. Many other have also been proposed [73, 20, 21, 54] but under the additional assumption of the random oracle model [5].

7.2 Digital Signature Schemes

As above, a digital signature scheme (G, Σ, V) admits various levels of security.

Forgeries and Attacks

In this subsection, we formalize some security notions [26, 27] which capture the main practical situations. On the one hand, the goals of the adversary may be various:

– *total break* : Disclosing the secret key of the signer.

– *universal forgery* : Constructing an efficient algorithm to sign any message.
– *existential forgery* : Providing a new message-signature pair.

In many cases this latter forgery, the *existential forgery*, is not dangerous, because the output message is likely to be meaningless. Nevertheless, a signature scheme which is not existentially unforgeable does not guarantee by itself the identity of the signer. For example, it cannot be used to certify randomly looking elements, such as keys.

On the other hand, various means can be made available to the adversary, helping her for her forgery. We focus on two specific kinds of attacks against signature schemes: the *no-message attacks* and the *known-message attacks*. In the first scenario, the attacker only knows the public key of the signer. In the second one, the attacker has access to a list of valid message-signature pairs. According to the way this list was created, we usually distinguish many subclasses, but the strongest is the *adaptively chosen-message attack*, where the attacker can ask the signer to sign any message that she wants. She can therefore adapt her queries according to previous message-signature pairs.

DEFINITION 22 (Secure Signature Scheme). A signature scheme is *secure* if an existential forgery is computationally impossible, even under an adaptively chosen-message attack.

Some Examples

The first secure signature scheme was proposed by Goldwasser et al. [26] in 1984. It uses the notion of claw-free permutations pairs, and provides polynomial algorithms with a polynomial reduction between the research of a claw and an existential forgery under an adaptively chosen-message attack. However, the scheme is totally unpractical. Hopefully, some practical schemes have been proposed.

The RSA Signature. In the same paper as their public key encryption scheme, Rivest–Shamir–Adleman [63] proposed the first signature scheme based on the "trapdoor one-way permutation paradigm", using the RSA function: the key generation algorithm produces a large composite number $n = pq$, a public key e, and a secret key d such that $e \cdot d = 1 \bmod \varphi(n)$. The signature of a message m, encoded as an element in $\mathbb{Z}_n^\star$, is its e^{th} root, $\sigma = m^{1/e} = m^d \bmod n$. The verification algorithm simply checks whether $m = \sigma^e \bmod n$.

However, the RSA scheme is not secure by itself since it is subject to existential forgery: it is easy to create a valid message-signature pair, without any help of the signer, first randomly choosing a certificate σ and getting the signed message m from the public verification relation, $m = \sigma^e \bmod n$.

Nevertheless, a classical way to increase security is to use the "hash-and-sign" paradigm: to sign any message m, one first hashes it, using any hash function such as MD5 [62] or SHA-1 [51], into $h = H(m)$, encoded as an element in $\mathbb{Z}_n^\star$. The signature of m is therefore the e^{th} root of h, $\sigma = h^{1/e} = h^d \bmod n$. The verification algorithm simply checks whether $\sigma^e = H(m) \bmod n$. Even if this was originally only heuristic, under some assumptions about the hash function

(namely that it behaves like a random function [5]) it can be proven secure [7, 13].

The Schnorr Signature. In 1986 a new paradigm for signature schemes was introduced. It is derived from fair zero-knowledge identification protocols involving a prover and a verifier [25], and uses hash functions in order to create a kind of virtual verifier. The first application was derived from the Fiat–Shamir [19] zero-knowledge identification protocol, based on the hardness of extracting square roots modulo n (which is equivalent to the factorization of n), with a brief outline of its security. Another famous identification scheme [66], together with the signature scheme [67], has been proposed later by Schnorr, based on that paradigm, under the discrete logarithm problem: the key generation algorithm produces two large primes p and q, such that $q \geq 2^k$, where k is the security parameter, and $q \mid p-1$, as well as an element g of $\mathbb{Z}_p^\star$ of order q. It also creates a pair of keys, $x \in \mathbb{Z}_q^\star$ and $y = g^{-x} \bmod p$. The signer publishes y and keeps x secret. The signature of a message m is a triple (r, e, s), where $r = g^K \bmod p$, with a random $K \in \mathbb{Z}_q^\star$, the "challenge" $e = H(m, r) \bmod q$ and $s = K + ex \bmod q$. It satisfies $r = g^s y^e \bmod p$ with $e = H(m, r)$, or simply $e = H(m, g^s y^e \bmod p)$, which is checked by the verification algorithm.

The security results for that paradigm have been considered as folklore for a long time but without any formal validation. But this formal validation appeared few years ago [55, 56] under the random oracle assumption [5].

8 CONCLUSION AND OPEN PROBLEMS

After the failure of mathematics to help in proving the security of public-key cryptosystems, because of a "computational" point of view, they have been realized to provide useful objects to make practical the suggestion of Diffie and Hellman.

Indeed, they provide one-way and trapdoor one-way problems which are the foundations of the public-key cryptography, for solving the confidentiality and the authentication concerns as seen above, but also for many other applications [41]. However, many problems remain open.

- About the factorization-like problems:
  - Do there exist algorithms to solve the factorization problem more efficiently than the NFS?
  - What is the exact relation between **RSA** and **FACT**? Even if they are likely not equivalent [9], nobody has ever shown any gap.
- About the Discrete Logarithm-like problems:
  - Is this problem stronger over an elliptic curve than in $\mathbb{Z}_p^\star$ (for similar sizes)? The research has just begun recently, and better algorithms are appearing.
  - Are there other suitable groups?
  - What about the real difficulty of the **DH** and **DDH** problems? Some relations are known, but in particular cases [39, 72]

– Are there other candidates as one-way and trapdoor one-way problems? Such new problems would immediately lead to new cryptosystems thanks to generic conversions [5, 6, 20, 21, 54].

On the other hand, mathematics provided many tools for breaking cryptosystems, such as LLL [37] or NFS [36]. Maybe they can provide some other tools to cryptographers.

REFERENCES

1. D. Atkins, M. Graff, A. K. Lenstra, and P. C. Leyland. THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE. In *Asiacrypt '94*, LNCS 917, pages 263–277. Springer-Verlag, Berlin, 1995.
2. O. Baudron, D. Pointcheval, and J. Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. In *Proc. of the 27th ICALP*, LNCS. Springer-Verlag, Berlin, 2000.
3. M. Bellare, A. Boldyreva, and S. Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In *Eurocrypt '2000*, LNCS. Springer-Verlag, Berlin, 2000.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.
5. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
6. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
7. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996.
8. M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Crypto '99*, LNCS 1666, pages 519–536. Springer-Verlag, Berlin, 1999.
9. D. Boneh and R. Venkatesan. Breaking RSA May Not be Equivalent to Factoring. In *Eurocrypt '98*, LNCS 1403, pages 59–71. Springer-Verlag, Berlin, 1998.
10. S. A. Brands. An Efficient Off-Line Electronic Cash System Based on the Representation Problem. Technical Report CS-R9323, CWI, Amsterdam, 1993.
11. R. P. Brent. An Improved Monte Carlo Factorization Algorithm. *BIT*, 20:176–184, 1980.
12. S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann. Factorization of a 512-bit RSA Modulus. In *Eurocrypt '2000*, LNCS. Springer-Verlag, Berlin, 2000.
13. J. S. Coron. On the Exact Security of Full-Domain-Hash. In *Crypto '2000*, LNCS. Springer-Verlag, Berlin, 2000.
14. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Crypto '98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998.
15. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
16. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *Proc. of the 23rd STOC*. ACM Press, New York, 1991.
17. I. Duursma, P. Gaudry, and F. Morain. Speeding Up the Discrete Log Computation on Curves with Automorphisms. In *Asiacrypt '99*, LNCS 1716. Springer-Verlag, Berlin, 1999.
18. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
19. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987.
20. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC '99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999.
21. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.
22. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.
23. P. Gaudry. An Algorithm for Solving the Discrete Log Problem on Hyperelliptic Curves. In *Eurocrypt '2000*, LNCS. Springer-Verlag, Berlin, 2000.

24. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
25. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304. ACM Press, New York, 1985.
26. S. Goldwasser, S. Micali, and R. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Proc. of the 25th FOCS*, pages 441–448. IEEE, New York, 1984.
27. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
28. J. Håstad. On Using RSA with Low Exponent in a Public Key Network. In *Crypto '85*, LNCS 218, pages 403–408. Springer-Verlag, Berlin, 1986.
29. D. Hühnlein, M. J. Jacobson, S. Paulus, and T. Takagi. A Cryptosystem Based on Non-Maximal Imaginary Quadratic Orders with Fast Decryption. In *Eurocrypt '98*, LNCS 1403, pages 294–307. Springer-Verlag, Berlin, 1998.
30. D. E. Knuth. *The Art of Computer Programming – Seminumerical Algorithms*, volume 2. Addison–Wesley, Readings, Massachusetts, London, 1981.
31. N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.
32. N. Koblitz. A Family of Jacobians Suitable for Discrete Log Cryptosystems. In *Crypto '88*, LNCS 403, pages 94–99. Springer-Verlag, Berlin, 1989.
33. N. Koblitz. Hyperelliptic Cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.
34. X. Lai and J. L. Massey. A Proposal for a New Block Encryption Standard. In *Eurocrypt '90*, LNCS 473, pages 389–404. Springer-Verlag, Berlin, 1991.
35. B. A. LaMacchia and A. M. Odlyzko. Computation of Discrete Logarithms in Prime Fields. *Designs, Codes and Cryptography*, 1(1):47–62, May 1991.
36. A. Lenstra and H. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.
37. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
38. J. L. Massey. SAFER K-64: a Byte-Oriented Block Ciphering Algorithm. In *Proc. of the 1st FSE*, LNCS 809, pages 1–17. Springer-Verlag, Berlin, 1994.
39. U. M. Maurer. Diffie Hellman Oracles. In *Crypto '96*, LNCS 1109, pages 268–282. Springer-Verlag, Berlin, 1996.
40. R. J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. *DSN progress report*, 42-44:114–116, 1978. Jet Propulsion Laboratories, CALTECH.
41. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Available from `http://www.cacr.math.uwaterloo.ca/hac/`.
42. A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *IEEE Transactions on Information Theory*, 39:1639–1646, 1993.
43. R. C. Merkle and M. E. Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. *IEEE Transactions on Information Theory*, IT–24:525–530, 1978.
44. G. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.
45. V. Miller. Uses of Elliptic Curves in Cryptography. In *Crypto '85*, LNCS 218, pages 417–426. Springer-Verlag, Berlin, 1986.
46. P. L. Montgomery. Speeding the Pollard and Elliptic Curve Methods for Factorization. *Mathematics of Computation*, 48(177):243–264, January 1987.
47. M. A. Morrison and J. Brillhart. A Method of Factoring and the Factorization of F_7. *Mathematics of Computation*, 29(129):183–205, January 1975.
48. M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *Proc. of the 21st STOC*, pages 33–43. ACM Press, New York, 1989.
49. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437. ACM Press, New York, 1990.
50. National Bureau of Standard U.S. Data Encryption Standard, 1977.
51. NIST. *Secure Hash Standard* (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.
52. S. C. Pohlig and M. E. Hellman. An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. *IEEE Transactions on Information Theory*, IT–24(1):106–110, January 1978.
53. D. Pointcheval. A New Identification Scheme Based on the Perceptrons Problem. In *Eurocrypt '95*, LNCS 921, pages 319–328. Springer-Verlag, Berlin, 1995.

54. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '2000*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.
55. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, Berlin, 1996.
56. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, to appear.
    Available from `http://www.di.ens.fr/~pointche`.
57. J. M. Pollard. A Monte Carlo Method for Factorization. *BIT*, 15:331–334, 1975.
58. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.
59. C. Pomerance. The Quadratic Sieve Algorithm. In *Eurocrypt '84*, LNCS 209, pages 169–182. Springer-Verlag, Berlin, 1985.
60. C. Pomerance and S. S. Wagstaff. Implementation of the Continued Fraction Integer Factoring Algorithm. *Congressus Numerantium*, 37:99–118, 1983.
61. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992.
62. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, The Internet Engineering Task Force, April 1992.
63. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
64. T. Satoh and K. Araki. Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves. *Comment. Math. Helv.*, 47(1):81–92, 1998.
65. B. Schneier. *Applied Cryptography, Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc, 1994.
66. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990.
67. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
68. A. Shamir. An Efficient Identification Scheme Based on Permuted Kernels. In *Crypto '89*, LNCS 435, pages 606–609. Springer-Verlag, Berlin, 1990.
69. D. Shanks. Class Number, a Theory of Factorization, and Genera. In *Proceedings of the Symposium on Pure Mathematics*, volume 20, pages 415–440. AMS, 1971.
70. C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
71. A. Shimizu and S. Miyaguchi. Fast Data Encipherment Algorithm FEAL. In *Eurocrypt '87*, LNCS 304, pages 267–278. Springer-Verlag, Berlin, 1988.
72. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266. Springer-Verlag, Berlin, 1997.
73. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt '98*, LNCS 1403, pages 1–16. Springer-Verlag, Berlin, 1998.
74. N. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, 12(3):193–196, 1999.
75. J. Stern. A New Identification Scheme Based on Syndrome Decoding. In *Crypto '93*, LNCS 773, pages 13–21. Springer-Verlag, Berlin, 1994.
76. D.R. Stinson. *Cryptography Theory and Practice*. CRC Press, 1995.
77. P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Eurocrypt '96*, LNCS 1070, pages 332–343. Springer-Verlag, Berlin, 1996.
78. S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. In *STACS '98*, LNCS 1373, pages 249–275. Springer-Verlag, Berlin, 1998.
79. G. S. Vernam. Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications. *Journal of the American Institute of Electrical Engineers*, 45:109–115, 1926.
80. H. C. Williams. A $p + 1$ Method of Factoring. *Mathematics of Computation*, 39(159):225–234, July 1982.