Title: **CRYPTOGRAPHY AND DIGITAL SIGNATURE**

Authors :

Alessandro Languasco
Dipartimento di Matematica Pura e Applicata
Università di Padova
Via Belzoni 7
35131 Padova, Italy
e-mail: languasco@math.unipd.it

Alberto Perelli
Dipartimento di Matematica
Università di Genova
Via Dodecaneso 35
16146 Genova, Italy
e-mail: perelli@dima.unige.it

CRYPTOGRAPHY AND DIGITAL SIGNATURE

A. LANGUASCO A. PERELLI

The problem to assure a secure transmission of informations played a fundamental role in the human history. During the centuries several different techniques was used; famous examples are the translitteration method of Julius Caesar and the Enigma machine (which was used by the German army during the second world war). The methods used until the middle seventies of the last century was not completely satisfactory (one should remind, for example, how the Enigma code was broken in the fourties by the english mathematician A.Turing and his group [1]); in 1976 W. Diffie and M.E. Hellman proposed a revolutionary code (called public-key code) which had the theoretical chance to assure security and authentication. Two years later R.L. Rivest, A. Shamir e L.M. Adleman, using some well-known prime numbers properties, were able to practically implement Diffie-Hellman's idea. So we can say that modern cryptography born in the seventies and, in few years, replace almost completely the methods obtained with classic techniques.

Our goal here is to briefly present some underlying ideas on this topic, focusing our attention on the theoretical differencies among classic and modern cryptography. Moreover, in the last section, we say few words about one of the most important applications of the public-key cryptography: the digital signature.

1. Cryptography

Cryptography's field of investigation is the study of the methods that can be used to send informations in a disguised form such that only the intended receiver can remove the disguise and read the plain message. Denote respectively with

$$\mathcal{M} = \{\text{plain messages}\} \quad \text{and} \quad \mathcal{C} = \{\text{enciphered messages}\}$$

the sets of the plain messages and of the enciphered messages. A *cryptographic transformation* is an injective function $f : \mathcal{M} \longrightarrow \mathcal{C}$. So the inverse function f^{-1} is well-defined and satisfies the following diagram:

$$\mathcal{M} \xrightarrow{f} f(\mathcal{M}) \xrightarrow{f^{-1}} \mathcal{M}.$$

We remark that the injectivity of f is very important since we don't want any ambiguity in the deciphering operations. We will call

f: *enciphering* function and f^{-1} : *deciphering* function.

A *cryptosystem* is the quadruplet $(\mathcal{M}, \mathcal{C}, f, f^{-1})$.

A short history.

One of the first cryptographic method was used by the Roman emperor Julius Caesar. Now we can describe it as follows. It was based on the arithmetic in the modulus n (where n denotes the number of letter of the used alphabet). Once one has fixed a bijective correspondence between the alphabet and $\mathbb{Z}_n$, the enciphering operation was made by shifting the letters of a fixed quantity m; *e.g.*, if $n = 21$ and $m = 5$, to process the message M (for simplicity we can considered it of one letter) was used the enciphering function $C \equiv M+5 \pmod{21}$ and the deciphering function $M \equiv C$-5 (mod 21). If such a method is used on a sufficiently long text it is easy to see that a frequence analysis on the letter of the enciphered result would allow a third person to break the system (*i.e.*, to obtain M from C by reconstructing the deciphering function from the frequence analysis).

In the sixteenth century Blaise de Vigenère developed a variation of this method which is harder to break. He considered k-letters block (every word corresponds with an element of $(\mathbb{Z}_n)^k$) and defined, as enciphering operation, the simultaneous shift of all the letters in the block by a fixed "password" of k-letters (*i.e.*, we add, as vectors, the message $M \in (\mathbb{Z}_n)^k$ to a fixed vector $B \in (\mathbb{Z}_n)^k$). As in the previous case, this system can be broken by a suitable frequence analysis.

In 1931 Lester Hill defined as f the product for an invertible matrix. Letting $\mathcal{M} = \mathcal{C} = (\mathbb{Z}_n)^k$, he used an invertible matrix A whose entries are in $\mathbb{Z}_n$. The enciphering function is then obtained multiplying A and $M \in \mathcal{M}$, while the deciphering one involves the product of $C \in \mathcal{C}$ by A^{-1}. Hill's method can be broken using the linear algebra to the modulus n.

In the twenties a german entrepreneur, A. Scherbius, built the first version of a cryptographic machine which later will be called Enigma. In fact the enciphering fuction was based on an ingeniuous iteration of Julius Caesar's method. Every single shift was realized by a rotor which electromechanically connected the input letter with the output one; such an output letter was then passed to another rotor which made another shift an so on until the final result. The first versions of Enigma had three rotors, but in the most recent ones their complexity was increased by using five rotors whose starting positions and operating sequences was interchangeable. The german government was so sure of Enigma's inviolability that based its whole reserved information traffic, both military than commercial, on it. The polish mathematician M. Rejewski was the first which found some weakness in Enigma. For some period of time he was able to decipher the german messages doing calculations by hand. During the second world war, the United Kingdom secret service recognised that breaking the Enigma code should be a considerably strategic advantage. So they created a specialized cipher bureau, whose leader was the english mathematician Alan Turing, to study the encrypted german traffic of informations. Even if the complexity of Enigma was increased by the previously mentioned variations, Turing's group was able to develop some electromechanical computers (which in fact can be considered the ancestors of the modern electronic computer) to analyze and, in many cases, decipher the Enigma-coded messages. Many historical experts think that such a bureau played a fondamental role to assure Allies' victory in the second world war.

In fact we can say that early cryptosystems used only elementary topics of Algebra and Number Theory; essentially this was the situation until the seventies.

Classic Cryptography.

The examples we have seen are from *classic cryptography* (also called *secret-key cryptography* or *symmetric cryptography*). Anyone who has enough informations to encipher a message can easily (or with a little effort) decipher it. Moreover who wants secretely communicate has to agree (and to exchange in a secure way) on the enciphering and deciphering keys. We remark that the classical methods has the following property: their *enciphering and deciphering transformations are computationally equivalent* (*i.e.,* their computational complexities are equal or of the same order). Such a characteristic allow us to made a classification based on computational complexity:

Classic cryptography: the cryptosystems in which (after knowing the enciphering key) the deciphering transormation can be implemented in approximatively the same computational complexity needed to encipher.

If the deciphering function f^{-1} has a computational complexity that has a larger order than the corresponding enciphering function f, such a cryptosystem has the potential to be transformed into a *public-key* system (in the following we will be more precise).

Public key cryptography.

Public key cryptography (also called *asymmetric*) was discovered by W. Diffie and M.E. Hellman [2] in 1976. To clarify the definition we gave in the previous paragraph, we remark that in a public key cryptosystem who knows only how to encipher cannot use the enciphering key to rebuild the deciphering one without performing a prohibitive amount of computations. In other words, $f: \mathcal{M} \longrightarrow \mathcal{C}$ can be easily evaluated knowing K_E (enciphering key), but, without an additional information (the deciphering key K_D), the evaluation of $f^{-1}: f(\mathcal{M}) \longrightarrow \mathcal{M}$ is an hard computational problem. Such kind of functions are called *one-way functions* (o *trap-door functions*).

At the present state of knowedge there's no functions whose one-way property is *proved* (even if, for some functions, commonly used in practice, it is conjectured that it holds). We also remark that some of the functions that today can be considered one-way, could, in the next future, loss (from a practical point of view) this characteristic due to the improvement of microprocessors' performances.

Finally we note that a public key cryptosystem can also be used as a safe method for the key-exchange problem of a classic system. In fact, usually, classic systems keys are shorter than the public ones and so, using a public-key system to periodically exchange the classic keys, one can have a comfortable security and a faster implementation (since, in practice, a shorter time is needed to perform enciphering and deciphering).

RSA cryptosystem.

One of the most important examples of a conjectural one-way function (and also the first developed public-key cryptosystem) was given R.L. Rivest, A. Shamir and L.M. Adleman [3] in 1978. They used the large (conjectural) difference among the computational complexities of primality and factorization algorithms in $\mathbb{Z}$.

The RSA system can be briefly described as follows: every user X have to

- choose randomly two big prime numbers p,q (at present 300 digital digits are considered a safe choice) and calculate $n=pq$;
- calculate $\varphi(n)=(p-1)(q-1) = \# \mathbb{Z}_n^*$ (with $\#A$ we denote the cardinality of the set A);
- choose randomly an integer e, $1< e < \varphi(n)$, coprime with $\varphi(n)$;
- calculate $d \equiv e^{-1} \pmod{\varphi(n)}$.

We firstly remark that one of the most important request to build a safe system is that the needed choices should be done in a *random* way. Moreover, it can be easily verify that the whole amount of computations in the previous design is low.

For every user X of the cryptosystem are then defined the following quantities:

Public key $K_E = (n,e)$: every other user can know it, so anyone of them can use it to encipher the messages for X;

Private key $K_D = d$: X has to keep it secret since it allows anyone who knows it to decipher the messages for X ;

Enciphering function: $f : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ defined by $f(M) \equiv M^e \pmod{n}$;

Deciphering function: $f^{-1}: \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ defined by $f^{-1}(C) \equiv C^d \pmod{n}$.

An user A who wishes to send a message M to the user B, can use B's public key (n_B,e_B) to compute

$$C \equiv M^{e_B} \pmod{n_B}$$

and to send C to B. B, after receiving C, can use his own secret key d_B to compute

$$C^{d_B} \pmod{n_B} \equiv M^{e_B d_B} \pmod{n_B}.$$

Since $e_B d_B \equiv 1 \pmod{\varphi(n_B)}$, Fermat-Euler's Theorem[1] implies (in this case) that

$$C^{d_B} (\mathrm{mod}\, n_B) \equiv M (\mathrm{mod}\, n_B)$$

and hence B can read the clear text M.

To break the system RSA (*i.e.*, to compute f^{-1} without knowing K_D) one should be able to compute d knowing only e and n; but if someone knows e, to evaluate d he should know $\varphi(n)$ starting just by n. In this case the calculations needed to evaluate $\varphi(n)$ are *computationally equivalent* to the ones needed to factor n. Hence to break the RSA system one should be able to efficiently factorize n (*i.e.*, in a reasonable amount of time). As we will see in the following paragraph, at present the computational complexity of the primality algorithms (used to prove the primality of p and q) is less than the one of the factorization algorithms (used to compute from n his prime factors p and q). For this reasons, in practice, the RSA system is considered a safe one.

Some words about primality and factorization algorithms.

One of the crucial steps in RSA system is the choice of $n=pq$. In fact one has not to choose p and q of a "special form" (e.g. classical special form are 2^k-1 or 2^k+1) because in such cases there exist very fast factorization algorithm. Now we discuss some characteristics of the *primality algorithms (or tests).* These algorihms can prove if a given integer n is prime (without searching for its factors). First of all we remind that the number of digital digits of an integer n is approximatively log n, that log n is used as the "measure unit" for the computational complexity of the algorithms which work on the number n and that an algorithm is called *polynomial* if its complexity is $O(\log^c n)$ for some absolute constant $c>0$.

Some of the best known primality tests are based on congruence properties and, in particular, on Little Fermat Theorem: if n is prime then $a^{n-1} \equiv 1 \pmod{n}$ for every $a \in \mathbb{Z}_n^*$. But such a property does not hold only for prime numbers; in fact there exist the *Carmichael numbers*, *i.e., composite* integers n such that $a^{n-1} \equiv 1 \pmod{n}$ holds for every $a \in \mathbb{Z}_n^*$. For example, 561 = 3*11*17 is a Carmichael number, and in 1994 W.R. Alford, A. Granville and C. Pomerance [4] proved that there are infinite Carmichael numbers. Hence, the easiest primality tests can furnish just a *probabilistic* estimate of the primality of n (in other words: the probabilty that n is prime is very near to 1), even if the Miller-Rabin test (see [4]) can prove the primality of n with a computational complexity $O(\log^5 n)$. By the way such an algorithm assume that the Extended Riemann Hypothesis (see Davenport [6]), a famous conjecture in number theory, holds. Without

[1] Euler-Fermat Theorem: Let $n \in \mathbb{N}$, $n \neq 0$. If g.c.d.$(a,n)=1$ then $a^{\varphi(n)} \equiv 1 \pmod{n}$. Moreover, if n is squarefree, then, for every $a \in \mathbb{Z}_n$, we get $a^{1+k\varphi(n)} \equiv 1 \pmod{n}$ for every $k \in \mathbb{Z}$.

assuming any unproved hypothesis, the best primality test nowadays known[2] was developed in 1983 by L.M. Adleman, C. Pomerance and R. Rumely [7] and it has a computational complexity (completely proved using some analytic number theory techniques)

$$O((\log n)^{c \log \log \log n}),$$

where $c > 0$ is a suitable absolute constant; in other words, such a test is "almost-polynomial".

Now we consider the *factorization algorithms*. In the latest twenty years many researchers tried to solve the problem of fast factorization of integers, but, even if some sophisticated mathematical techniques were used, at present this problem seems to be essentially more difficult than the primality one. In fact, the best factorization algorithm (J.M. Pollard [8] had the main idea in 1993, and then other researchers improved it; see the book of H. Lenstra - A.K. Lenstra [9]) has a computational complexity that is of higher order than the primality one. Such a method is based on some properties of number fields and its (conjectural) computational complexity is

$$O(e^{c(\log n)^{1/3}(\log\log n)^{2/3}})$$

where $c > 0$ is a suitable absolute constant, which is considerably larger than the primality tests one.

As a practical example, we remark that a personal computer that anyone of us can buy in a computer store can construct a 140-decimal digits integer which is the product of two "general" primes in few seconds. To factorize the same number, using parallel computers, a month of computer time is needed! So it is a good idea to use sufficiently big integers; nowadays a 220-decimal digits integer (product of two "general" primes) is considered a safe choice for a personal use, but, for a professional use, larger integers are a better one. We finally remark that, since the existence of sufficiently large lower bounds for the general case of the factorization is an open problem, the RSA cryptosystem can, in fact, be considered a public-key system just using a conjectural point of view, even if, in practice, it is commonly used.

2. Digital signature

During our life, frequently happens that one has to sign a document or a cheque. In these cases (and in many others) the signature certifies our identity and, sometimes, it moreover assures the non-ripudiability (as for a cheque). Our identity is then certified by our handwriting since it is considered a personal identification. It is clear that we can realize it only on a piece of

[2] **Remark** (added in August 2002). Recently three indian researcher, M.Agrawal, N. Kayal and N.Saxena, see [10], proved that there exists a unconditional deterministic primality algorithm whose complexity is $O((\log n)^{12+\epsilon})$. The main tool used is a polynomial identity (in some sense similar to Little Fermat Theorem) in a suitable subfield of $\mathbb{Z}_p[x]$.

paper and so the problem is: how to build a tool for some non-material support (as bits are) which is easy to use and efficient? To this end a concept of digital signature was introduced to certify any sequence of binary digits. Classic cryptography cannot be used for reach such an authentication, so we have to use the public key one; in fact in this way an easy algorithm of digital signature can be described (with general notations) as follows.

Let A and B two users of a public key system. We know that their enciphering functions f_A and f_B are public and the corresponding deciphering functions f_A^{-1} are f_B^{-1} secret; we further assume that f_A and f_B are bijective functions. If A wishes to send a M to B, he has to send $f_B(M)$ and, to certify his identity, he also sends as an attachment the quantity $f_B(f_A^{-1}(name_A))$. $f_A^{-1}(name_A)$ represents the digital signature of and $name_A$ is a nickname of A. B deciphers the message using f_B^{-1} (which is only known by himself) and $f_B^{-1}(f_B(M)) = M$. To check if the sender is really A, B applies $f_A\ f_B^{-1}$ to the attachment and he obtains $f_A f_B^{-1}(f_B f_A^{-1}(name_A)) = name_A$. Such a method runs well sinche only A can sign the message in this way (because he/she is the only one who knows f_A^{-1}).

But this design does not allow us to know if it is really A that signs using f_A^{-1}. A third person C could public a key saying that A is its owner; C is a impostor and in this way could be able to read the messages for A. Such a problem can be solved introducing a *Public Key Certifier* to which every user can ask if the digital signature previously obtained coincides with the one checked, certified and testified by the Certifier.

Moreover, it is also possible that an intruder can identify $f_A^{-1}(name_A)$ using a wide number of intercepted messages. A good idea is then to build a signature that depends also from the message; the digital signature becomes $f_A^{-1}(mark_M)$, where $mark_M$ is a bit-sequence of fixed length (usually 160 binary digits) which is obtained from the message M using a suitable function whose main characteristics are that none can rebuild M just knowing $mark_M$ and that it does not cause collisions with high probabilty (good functions are those whose probabilty that $mark_M = mark_{M'}$ with $M \neq M'$ is less than 10^{-50}). Such functions are called *hash functions* and are furnished to every user by the Certifier since just one of them is used for the whole cryptosystem. B will check that the message was in fact sent by A and he will get $mark_M$; to be sure that, with high probability, there are no falsifications, B recompute the mark of M using the hash function of the cryptosystem.

It it clear that this seems to introduce some weakness in a system that should be inattackable; so why we don't use the whole message as a mark of itself? Surely the security degree is, in this case, higher but, in practice, a longer time is needed to encipher and decipher. One has to choose among an absolute safety with an high cost in time and a relative security (however of high degree) with a less amount of time. The choice depends essentially from the kind of information in M, and also by some personal preferences.

There are some other kinds of digital signature in which time is preferred with respect to security. For example some digital signature methods use $f_A^{-1}(mark_M)$ attached to the message M which is sent as a clear text. Other methods assure only who is the sender without any grants on the message integrity (one signs with $f_A^{-1}(name_A)$ attached to the message M sent as a clear text).

We finally remark that the introduction of the Certifier allow us to attach a *time-mark* to the message; its importance is clear because the time provided by a computer is not significative. If A needs a time-marking of a message, sends to the Certifier $f_B(f_A^{-1}(mark_M))$ (from that is not possible to rebuild the clear text of the message); the Certifier attachs date and time and applies its own enciphering function (which is private). The time-marked quantity is then sent back to A which attachs it to the message M and sends all to B enciphering using the rules of the public key system. In some sense the time-marking is an analogue of the authentication of a document and assure that such a document cannot be changed with a different one.

Bibliography

[1] A. Hodges (2000), *Alan Turing: The Enigma*, Walker Publishing Company.

[2] W. Diffie, M.E. Hellman (1976), New directions in cryptography, *IEEE Trans. Information Th.* 22, pp. 644-654.

[3] L.M. Adleman, R.L. Rivest, A. Shamir (1978), A method for obtaining digital signature and public key cryptosystems, *Comm. of ACM* 21, pp. 120-126.

[4] W.R. Alford, A. Granville, C. Pomerance (1994), There are infinitely many Carmichael numbers, *Ann. Math.* 140, pp. 703-722.

[5] N. Koblitz (1987), *A Course in Number Theory and Cryptography*, Springer-Verlag Berlin, Heidelberg, New York.

[6] H. Davenport (2001), *Multiplicative Number Theory*, third ed., Springer-Verlag Berlin, Heidelberg, New York.

[7] L.M. Adleman, C. Pomerance, R. Rumely (1983), On distinguishing prime numbers from composite numbers, *Annals of Math.* 117, pp. 173-206.

[8] J.M. Pollard (1993), Factoring with cubic integers, in: *The development of the Number Field Sieve*, A.K. Lenstra e H.W. Lenstra (eds), Springer Lecture Notes, pp. 4-10.

[9] A.K. Lenstra, H.W. Lenstra (eds) (1993), *The development of the Number Field Sieve*, Springer Lecture Notes.

[10] M.Agrawal, N. Kayal, N.Saxena (2002), Primes in P, preprint 2002, electronic version: www.cse.iitk.ac.in/news/primality.html

Suggested lectures

For Number Theory we suggest the excellent book by

H. Davenport (2000), *The Higher Arithmetic*, seventh ed., Cambridge University Press, which also contains a chapter on cryptography. More historical details can be found in

D. Kahn (1967), *The Codebreakers, the Story of Secret Writing*, Scribner, New York, and in the recent book by

S. Singh (2000), *The Code Book: the science of Secrecy from Ancient Egypt to Quantum Cryptography*, Anchor Books.

Excellent references on the mathematical modelization of public-key cryptography can be found in the books:

N. Koblitz (1998), *Algebraic Aspects of Cryptography*, Springer-Verlag Berlin, Heidelberg, New York, and Koblitz [5].

Concerning primality and factorization algorithms, we suggest the books by Koblitz [5], by

H. Cohen (1993), *A Course in Computational Algebraic Number Theory*, Springer-Verlag Berlin Heidelberg New York, and by

H. Riesel (1994), *Prime Numbers and Computer Method for factorization*, Birkhäuser Basel, Boston, Berlin.

Another excellent and recent reference on this topic is

C. Pomerance, R. Crandall (2000), *Prime Numbers: A Computational Perspective*, Springer-Verlag Berlin, Heidelberg, New York.