Université de Marseille II, M1 2nd Semester 2009

Cryptology course

By
Christophe RITZENTHALER

These are notes of a course taught at Marseille in Second Semesters 2006-2009. They can be found on `www.iml.univ-mrs.fr/~ritzenth`.
Please send comments and corrections to me at `ritzenth@iml.univ-mrs.fr`.

This course is deeply inspired by the following books :
P. Barthélemy, R. Rolland, P. Véron, cryptographie, principes et mises en œuvre, Lavoisier (2005).
J. A. Buchman, introduction to cryptography, second edition, Springer-Verlag 2004.

Contents

Preface

This course is aimed at French student in fourth year at university with a mathematical background. On a meta-mathematical point, cryptography is divided into two parts : the maths games based on number theory, algebraic geometry, combinatorics and the real world, using network, chips, certificate, protocols .... Nowadays, these two parts are of equal importance but as a mathematician, my course deals mainly with the first aspect. Still a few chapters will be devoted to real cryptography.

The mathematics used in this course are elementary, which does not mean that the course is easy ! Indeed, cryptography twists most of the use we could make on general results and it can be difficult to see at the end why a given scheme gives what we want. It may also be difficult for standard mathematicians which are not accustomed to judge a theorem according to its effectivity or to its efficiency. For instance, we all agree that there are infinitely many prime numbers, but can one produce one with 1000-digits ? And if yes, how fast can you do that ? One second or one year ?

Chapter 1

Why modern cryptography ?

1.1 Introduction

We have an intuitive idea of what cryptography is : its fundamental objective (but not unique) is to enable two people, usually referred to as Alice (A) and Bob (B) [1], to communicate over an insecure channel in such a way that an opponent, Oscar (O) (or Eve for Eavesdropper), cannot understand what is being said. This channel can be a telephone line or a computer network for instance. Note that the science of secrecy includes two different approaches : the design of proper tools to insure secrecy which is *stricto sensu cryptography* (**cryptographie**) and the attacks of these tools to find out their weaknesses, called *cryptanalysis* (**cryptanalyse**). Thus the science that covers these two parts is nowadays called *cryptology* (**cryptologie**).
Cryptography has been used for long, mainly for military purposes. Going back to Caesar one finds the example of 'the shaved slave' : to transmit a message, one shaved the skull of a slave, wrote the message on top of it, waited till hair has grow again (!) and sent the slave. Besides the obvious slowness of the method, another drawback is that the secrecy of the information is based on the secrecy of the method : if for instance the slave speaks, the message is known. This important notion was first written down by Auguste Kerckhoffs (1883). He gave the basic principles of (secret key) cryptography :

'*Il faut bien distinguer entre un système d'écriture chiffré, imaginé pour un échange momentané de lettres entre quelques personnes isolées, et une méthode de cryptographie destinée à régler pour un temps illimité la correspondance des différents chefs d'armée entre eux. Ceux-ci, en effet, ne peuvent à leur gré et à un moment donné, modifier leurs conventions ;* [...] *Un grand nombre de combinaisons ingénieuses peuvent répondre au but qu'on veut atteindre dans le premier cas ; dans le second, il faut un système remplissant certaines conditions exceptionnelles, conditions que je résumerai sous les six chefs suivants :*

1. *Le système doit être matériellement, sinon mathématiquement indéchiffrable ;*

[1] listen to the song (!) on `http://www.iro.umontreal.ca/~crepeau/CRYPTO/Alice-Bob.html`

2. *Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi;* [...]'

Hence, if the secrecy cannot be based on the method (this is still not always understood even nowadays by certain companies, see for instance the 'secret' $A5$ algorithm of the GSM norm for mobiles which has been reveled on internet), it has to be based on additional secret information, called *keys*.

We will see in this chapter classical cryptosystems. Besides introducing the basic terminology, the aim of this chapter is to answer the questions : why this course ? Why do we have to use advanced mathematical tools ? We will see that the classical cryptosystems are not satisfactory, either due to their mathematical weaknesses or due to the changes in our needs.

1.2 Classical cryptography ...

Let us introduce some terminology. The information that Alice wants to send to Bob will be called *plaintext* (**Message clair**). She will *cipher* or *encrypt* (**chiffrer**) the plaintext to obtain a ciphertext (**texte chiffré ou cryptogramme**) with the help of a key. Bob who knows the key will then *decipher* (**déchiffrer**) the ciphertext, whereas Oscar, who does not, can try to *decrypt* (**décrypter**) it, in order to obtain the plaintext.

Definition 1.2.1. *A (secret key)* cryptosystem *(**cryptosystème**) is a three-tuple* $(\mathcal{P}, \mathcal{C}, \mathcal{K})$ *where the following conditions are satisfied :*

1. $\mathcal{P}$ *is a finite set of possible plaintexts;*
2. $\mathcal{C}$ *is a finite set of possible ciphertexts;*
3. $\mathcal{K}$ *is a finite set of possible keys called* keyspace *(**espace des clés**);*
4. *For each* $K \in \mathcal{K}$*, there is an* encryption rule *(**fonction de chiffrement**)* $\mathcal{E}_K : \mathcal{P} \to \mathcal{C}$ *and a* decryption rule *(**fonction de déchiffrement**)* $\mathcal{D}_K : \mathcal{C} \to \mathcal{P}$ *such that* $\mathcal{D}_K \circ \mathcal{E}_K = Id$.

Remark 1. *We see already that the function* $\mathcal{E}_K$ *must be injective. Moreover if* $\mathcal{P} = \mathcal{C}$*,* $\mathcal{E}_K$ *and* $\mathcal{D}_K$ *are bijective : they are permutations, inverse of each other.*

Alice and Bob will employ the following protocol to use a specific cryptosystem. First they choose a random key $K \in \mathcal{K}$. This is done when they are in the same place and not being observed by Oscar, or, alternatively, when they do have access to a secure channel. At a later time, suppose Alice wants to communicate a message to Bob over an insecure channel. We suppose that this message is a string

$$x = x_1 x_2 \dots x_n$$

for some integer $n > 0$, where each plaintext $x_i \in \mathcal{P}$. Each x_i is encrypted using $\mathcal{E}_K$ as $y_i = \mathcal{E}_K(x_i)$. She sends the resulting ciphertext

$$y = y_1 y_2 \dots y_n$$

to Bob over the channel. When Bob receives y, he deciphers it using $\mathcal{D}_K$. If we apply this protocol, the privacy is in the difficulty for Oscar to find the key K. Let us present now some classical examples. We will first start with mono-alphabetic ciphers (the elements of $\mathcal{P}$ are letters) and then move to more sophisticated poly-alphabetic ciphers.

1.2.1 The shift cipher

Let represent the 26 usual letters by elements in $\mathbb{Z}/26\mathbb{Z}$ (for instance $A \mapsto 0, B \mapsto 1, \dots Z \mapsto 25$). We consider then $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}/26\mathbb{Z}$. If $K \in \mathcal{K}$ one defines

$$\mathcal{E}_K : x \mapsto x + K.$$

The decryption rule is given by

$$\mathcal{D}_K : y \mapsto y - K.$$

Remark 2. *The case $K = 3$ is known historically as Caesar cipher. If we apply it to the plaintext* SAUNA *we obtain* VDXQD.
In spite of its extreme weakness (there are only 26 possibilities for the key, so one can test all of them till obtaining a coherent plaintext), it was used by South officers during the American Civil War and even by the Russian army in 1915.
Oddly, this cipher was used again at the beginning of internet, in forums. It was called ROT-13 (with $K = 13$). The idea was not to cipher texts but to prevent someone to read involuntarily a message (end of a movie,...).

1.2.2 The substitution cipher

The previous method can be seen as a particular case of permutation of letters. Allowing a random permutation is then an easy way to increase dramatically the number of keys : we can obtain $26! \approx 4 \cdot 10^{26}$ of them. Formally we have here $\mathcal{P} = \mathcal{C} = \mathbb{Z}/26\mathbb{Z}$ and $\mathcal{K} = \mathcal{S}_{26}$. The encryption and decryption rules for $\sigma \in \mathcal{K}$ are

$$\mathcal{E}_\sigma : x \to \sigma(x), \;\; \mathcal{D}_\sigma : y \to \sigma^{-1}(y).$$

1.2.3 The Vigenère cipher

This is our first case of poly-alphabetic ciphers. Let m be some fixed positive integer. Define $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}/26\mathbb{Z})^m$. For a key $K = (k_1, \dots, k_m) \in \mathcal{K}$ we define

$$\mathcal{E}_K(x_1, \dots, x_m) = (x_1 + k_1, \dots, x_m + k_m)$$

and

$$\mathcal{D}_K(y_1, \ldots, y_m) = (y_1 - k_1, \ldots, y_m - k_m).$$

Stricto sensu, the value m should be public. But we can make the system more difficult by assuming that the value m is secret as we will do (formally, it makes the accurate definitions of the sets a bit more complicated). Note that the case $m = 1$ is the shift cipher.

1.2.4 The Hill cipher

Let m be some fixed positive integer. Define $\mathcal{P} = \mathcal{C} = (\mathbb{Z}/26\mathbb{Z})^m$ and $\mathcal{K} = \mathrm{GL}_m(\mathbb{Z}/26\mathbb{Z})$. Note that a matrix with coefficients in $\mathbb{Z}/26\mathbb{Z}$ is invertible if and only if its determinant is coprime to 26. For a key $K \in \mathcal{K}$ we define

$$\mathcal{E}_K(x) = xK \text{ and } \mathcal{D}_K(y) = yK^{-1}.$$

As a special case we obtain the so called permutation cipher by using a permutation matrix (i.e. matrices with exactly one 1 on a given line and column and 0 elsewhere).

Remark 3. *We can notice that the Hill cipher is a generalization of Vigenère cipher. Indeed the Vigenère cipher with key $K = (k_1, \ldots, k_m)$ is a Hill cipher with a matrix of size $m + 1$*

$$\begin{pmatrix} 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \\ k_1 & \ldots & k_m & 1 \end{pmatrix}$$

and an entry $x = (x_1, \ldots, x_m, 0)$. Note that there will be in the final text a 0 at every $m + 1$ multiple positions that has to be canceled.

In conclusion, we see that this cryptographic constructions are based on two principles (that we can mix) : substitutions of letters and affine transformations. We will see now that how to attack them.

1.3 . . . and classical cryptanalysis

We have already seen in 1.2.1 that the shift cipher does not resist to a exhaustive research. Of course, as a set of key is always finite, exhaustive research is always an

option and thus any cryptosystem can be theoretically broken. However, the opponent has not an infinite power of computation or unlimited time so if the keyspace is too big (which means more than 2^{60} keys) one must have to try clever methods. We present here attacks based on statistical properties.

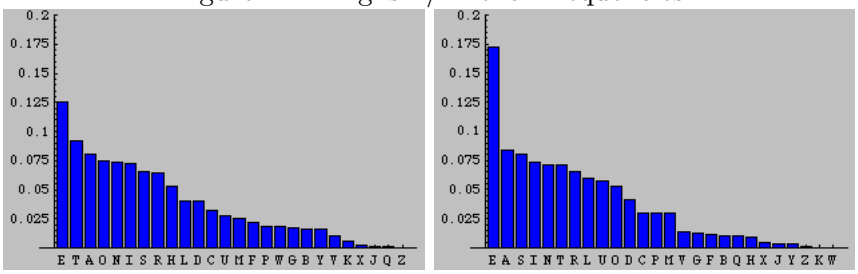
1.3.1 The substitution cipher

A way to decrypt a ciphertext is to use statistical properties of the language. Various people have estimated the relative frequencies of the 26 letters (or of any group of 2 or 3 letters). For the case of English and French these probabilities are compiled in Tab. 1.3.1, Fig. 1.1 and the most frequent pairs are in the table below.

letter	English	French	letter	English	French
A	.082	0.084	N	.067	0.071
B	.015	0.010	O	.075	0.052
C	.028	0.030	P	.019	0.03
D	.043	0.041	Q	.001	0.009
E	.127	0.172	R	.060	0.065
F	.022	0.011	S	.063	0.080
G	.020	0.012	T	.091	0.07
H	.061	0.009	U	.028	0.057
I	.070	0.073	V	.010	0.013
J	.002	0.003	W	.023	0.0004
K	.008	0.0005	X	.001	0.004
L	.040	0.06	Y	.020	0.003
M	.024	0.02	Z	.001	0.001

English	French
TH	ES
HE	DE
IN	LE
ER	EN
AN	RE
RE	NT
ES	ON
ON	ER
ST	TE
NT	EL
EN	AN
ED	SE
ND	ET

Figure 1.1: English / French frequencies



One can use these tables to find the permutation key like we will show on the following example.

Example 1. *Let us consider the following French ciphertext :*
WVZJXOJUKVNAPZEIKXZJTPKHOJXKEPFZVNTKIKTOVXIKZZAAIKJXI KCKZVBOVAXKBQOTK-TWVOJTZENAZIXKTRKJTWVEPKTERJOIKJN.

The frequency analysis gives the following results : K (16), Z, J, T (9), V (8), O, I (7), X (6), A, E (5).
K is likely to be E but then it is difficult to decide between Z, J, T. We use then pairs containing K.
IK (7), KT (4), KJ, XK (3).
Among the pairs starting with E the most frequent is ES. So we will assume that T is S. The next one is EN so we will assume that J maps to N. Among the letters with second frequency, only Z is unassigned. We can suppose that Z maps to A.
Among the pairs finishing with E the most frequent are DE,LE,RE. So I must be D,L or R. Now I is quite frequent as a letter so we will assume that I maps to the most frequent letter among them which is R. We can also think that X maps to D.
So far we have

plain	*A*	*D*	*E*	*N*	*R*	*S*
cipher	*Z*	*X*	*K*	*J*	*I*	*T*

which gives

..AND...E.....RED..S.E..NDE.....TSERES..DRE....RENDRE.E.....DE...SES
...NS.....RDES.ENS...ES..N.REN.

Analysing the beginning of the sentence one recognize QUAND *and the last letter must be* T. *It is then easy to complete the key :*

plain	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
cipher	*Z*	*C*	*B*	*X*	*K*	*F*	*R*	*Q*	*E*	.	.	*P*	*H*

plain	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
cipher	*J*	*O*	*A*	*W*	*I*	*T*	*N*	*V*	*U*	.	.	*Y*	.

and to decipher :

QUAND ON VEUT PLAIRE DANS LE MONDE, IL FAUT SE RÉSOUDRE À APPRENDRE BEAUCOUP DE CHOSES QU'ON SAIT PAR DES GENS QUI LES IGNORENT *(Chamfort, Maximes et Pensées).*

1.3.2 The Vigenère cipher

The first step is to determine the integer m. A method is the so-called *Kasiski test* and another one uses the *index of coincidence*.
The Kasiski test was described by Friedrich Kasiski in 1863 (and by Babbage before him). It is based on the observation that two identical segments of plaintext will be encrypted to the same ciphertext whenever their occurrence in the plaintext is d positions apart, where $d \equiv 0 \pmod{m}$. Conversely, if we observe two identical segments of ciphertext, each of length at least three, say, then there is a good chance that they do correspond to identical segments of plaintext.
The Kasiski test works as follows. We search the ciphertext for pairs of identical segments of length at least three, and record the distance between the starting positions of the two segments. If we obtain several such distances d_i then we will conjecture that m divides the gcd of the d_i's.
Further evidence for the value of m can be obtained by the index of coincidence. This concept was defined by Wolfe Friedman in 1920, as follows

Definition 1.3.1. *Suppose $x = x_1 \ldots, x_n$ is a string of n alphabetic characters. The* index of coincidence *of x, denoted $I_c(x)$, is defined to be the probability that two random elements of x are identical.*

Suppose we denote the frequencies of $A, \ldots, Z$ in x by $f_0, \ldots, f_{25}$ respectively. We can choose two elements of x in $\binom{n}{2}$ ways. For each i, there are $\binom{f_i}{2}$ ways of choosing both elements to be i. Hence we have the formula

$$I_c(x) = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}.$$

Now suppose that x is a string of a given language text. Denote the expected probabilities of occurrence of the letters $A, \ldots, Z$ by $p_0, \ldots, p_{25}$ respectively. Then we would expect that

$$I_c \approx \sum_{i=0}^{25} p_i^2 = 0.065, 0.074 \text{ English/French},$$

since the probability that two random elements both are A is p_0^2, etc. The same reasoning applies if x is a ciphertext obtained by means of any monosyllabic cipher (as the probabilities are shifted).
Suppose we start with a ciphertext $y = y_1 \ldots y_n$ that has been constructed by using a Vigenère cipher. Define m substrings $Y_1 \ldots Y_m$ of y of lengths n/m by writing y in columns in a $m \times n/m$ matrices and by taking the m rows (for instance with $m = 5$) :

$$\begin{array}{lllll} Y_1: & y_1 & y_6 & \ldots \\ Y_2: & y_2 & y_7 & \ldots \\ & & & \\ Y_5: & y_5 & y_{10} & \ldots \end{array}$$

If this is done and if m is indeed the key length then each $I_c(Y_i)$ should be roughly equal to 0.065 (English) or 0.074 (French). On the other hand if m is not the key length, then the substrings Y_i will look much more random, since they will have been obtained by shift encryption with different keys. Observe that a complete random string will have

$$I_c' = 26(1/26)^2 = 1/26 \approx 0.038.$$

The two values are sufficiently far apart that we will often be able to determine the correct key length.

Once the key length obtained, how do we get the key ? It is useful to consider the mutual index of coincidence of two strings.

Definition 1.3.2. *Suppose that $y = y_1 \ldots y_n$ and $y' = y_1' \ldots y_{n'}'$ are strings of n and n' alphabetic characters respectively. The* mutual index of coincidence *of y and y', denoted $MI_c(y, y')$ is defined to be the probability that a random element of y is identical to a random element of y'.*

Suppose we denote the frequencies of $A, \ldots, Z$ in y by $f_0, \ldots, f_{25}$ and by $f'_0, \ldots, f'_{25}$ in y'. Then

$$MI_c(y, y') = \frac{\sum_{i=0}^{25} f_i f'_i}{nn'}.$$

Suppose that $K = (k_1, \ldots, k_m)$ is the key. Let us see if we can estimate $MI_c(Y_i, Y_j)$. The probability that both characters are A in Y_i, Y_j is $p_{-k_i} p_{-k_j}$, the probability that both are B is $p_{1-k_i} p_{1-k_j}$,etc. Hence we estimate that

$$MI_c(Y_i, Y_j) \approx \sum_{h=0}^{25} p_{h-k_i} p_{h-k_j} = \sum_{h=0}^{25} p_h p_{h+k_i-k_j}.$$

Observe that the value of this estimate depends only on the difference $k_i - k_j$. For $k_i = k_j$, this value is 0.065 (English) or 0.078 (French) whereas for a non zero difference this value is less than 0.045. The idea is now to fix an Y_i (with frequencies (f_h)), to shift it by any of the 26 letters g (let us call Y_i^g the result), then to compute for each Y_j (with frequencies (f'_h))

$$MI_c(Y_j, Y_i^g) = \frac{\sum_{h=1}^{25} f_{h-g} f'_h}{nn'} = \frac{\sum_{h=1}^{25} f_h f'_{h+g}}{nn'} \approx \sum_{h=1}^{25} p_{h-k_i} p_{h-k_j+g} = \sum_{h=1}^{25} p_h p_{h+k_i-k_j+g}.$$

When $g = k_j - k_i$ the value of MI_c should be close to 0.065 (English) or 0.078 (French). We obtain in this way (by varying also Y_i) a set of differences and we can express all the k_i in terms of one of them. It is then enough to make another 26 attempts.

Remark 4. *The Y_i have been obtained by a shift cipher which is a particular case of substitution. An alternative is then to apply the usual statistical methods of 1.3.1.*

Example 2. *Let us consider the French ciphertext*
BTCRIJKIETASXYJXAIBWCLCGEMRAIEBBGXCCKMATBXYIJQUTMPDWMHWMRKGEMWV
KIEAQTTVRBMCGMPNXPKBXXTSNUDBASXUPKA.
We must first determine the length of the key : between two XY *we have an interval of length* 27 *and between two* KIE *we have* 54*. We can then suppose that the key is of length* 3*. Let us compute the coincidence index :*
Y_1 =BRKTXXBLEABXKTYQMWWKMKATBGNKXNBXK *gives* $I_c(Y_1) \approx 0.079$*;*
Y_2 =TIIAYAWCMIBCMBIUPMMGWIQVMMXBTUAUA *gives* $I_c(Y_2) \approx 0.075$*;*
Y_3 =CJESJICGREGCAXJTDHREVETRCPPXSDSP *gives* $I_c(Y_3) \approx 0.050$.
Even if the last result is quite bad, we can believe that the value of m is correct.

We apply now the remark above :

- *For Y_1, the most frequent letter is* K *which should correspond to* E*. But then* B *which is also quite frequent should correspond to* V *which is not so likely. However if we assume that* X *(which is also quite frequent) corresponds to* E *then* B *corresponds to* I*. So we say that the first letter of the key is* T*.*

- *For Y_2, the most common letter is* M *which we assume corresponds to* E. *So the second letter should be* I.

- *For Y_3 we observe the sequence of the frequencies :*

$$1, 0, 4, 2, 4, 0, 2, 1, 1, 3, 0, 0, 0, 0, 0, 3, 0, 3, 3, 2, 0, 1, 0, 2, 0, 0.$$

  The gap $0, 0, 0, 0, 0$ is characteristic of the letters V,W,X,Y,Z *so the letter corresponding to* P *must be* A.

The key is then TIP *and the plaintext:*
IL N Y AURA PAS DE QUESTION SUR LE CHAPITRE UN RELATIF AUX METHODES DE CRYPTOGRAPHIE ANCIENNE AU PARTIEL DU MOIS DE MARS.

1.3.3 A known plaintext attack on the Hill cipher

So far we have considered only attacks on a single ciphertext. But let assume that the opponent knows several plaintexts and their ciphers. More exactly let us assume that he knows m plaintexts X_i and corresponding ciphertexts Y_i. Writing

$$\begin{pmatrix} Y_1 \\ \vdots \\ Y_m \end{pmatrix} = \begin{pmatrix} X_1 \\ \vdots \\ X_m \end{pmatrix} K$$

one sees easily that one can find in general the key K (if the matrix of the X_i's is invertible).

Remark 5. *If you're looking for a real challenge, look at* `http://www.fbi.gov/page2/nov07/code112107.html`.

1.4 New requests

We have seen that the previous cryptosystems were quite sensitive to different kinds of attacks, mainly statistical attacks. A good cryptosystem should avoid these attacks.
Can we imagine a good mono-alphabetic cryptosystem ? As we have seen the size of the keyspace is bound by 26! which starts being a bit small according to the current computation power. But these cryptosystems are also weak for another reason. The systems being open (like on internet) it is not rare to find protocols were the attacker can know pairs of plaintexts and their ciphers. Of course in this context mono-alphabetic cryptography is not possible. Neither is poly-alphabetic cryptography based on linear systems as we have seen in 1.3.3. Thus modern cryptosystem must include a non-linear behavior (see Chap. **??**).
Another important fact is that symmetric key cryptography is not sufficient anymore. With internet, it is not possible to assume that people can meet somewhere to exchange the key. New cryptosystems have then to be designed. They belong to a new family

called *public key cryptography* (**cryptographie à clé publique**) (see Chap. 3). Note also that we want these cryptosystems to be as fast as possible. Thus the size of the key is going to be a constraint.
Besides the encryption of the text we have to deal with issues on the protocol : how to be sure that Alice is the sender ? How to be sure that the message has not been modified ? Encryption is not enough for that and we will need to introduce new notions (signature for instance). See Chap. 2.
Our cryptography techniques are based on mathematical tools which are convenient for computers applications. However cryptography is not limited to the present content and we will briefly mention below other directions in cryptography.

1.4.1 Steganography

Unlike cryptography which ciphers text to make them incomprehensible, *steganography* (**stéganographie**) hides texts in a support which can be a casual text or a picture. The case of Caesar's slave may be seen as an example of steganography. Special invisible inks are another one and were used during second World War. A more recent example is the five messages published in Libération newspaper by the mysterious group AZF in March 2004. We refer to `http://www.apprendre-en-ligne.net/crypto/menu/index.html` for beautiful examples, among other :

- La dame de la Lune.
- Les lettres de Georges Sand.
- Les images numériques.

Note that the last one arouses a lot of interests nowadays in relation with the problem of copyright. This application is known under the name of *watermarking* (**tatouage électronique**). The idea is to mark an image in such a way that one cannot removed this mark. This has a lot of constrains :

- To respect the quality of the picture, i.e. it has to be invisible for the user.
- To be resistant to transformations (compressions, filters, partial selection, or print/scan).
- To be resistant to direct attacks.

The most promising methods rely on frequency analysis (via Fourier transformation), in order to write the information in medium frequency (invisible for the eye but which cannot be modified by compression).

1.4.2 Future perspectives

Can we imagine perfect secrecy ? The perfect cryptography exists : it is called *Vernam one-time-pad* (**masque jetable**) and was used to secure the red line between Washington and Moscow. It is simply a Vigenère cipher whose key is as long as the plaintext

and used only once (see 4.2.3) ! Of course, this is not the perfect solution : the problem is the transport of the key and this is not suitable for large internet protocols.
Quantum cryptography may solve this issue. The principle is to use the quantum behavior of photon to make impossible the reading of the message by an opponent without destroying it. For technical details see `http://www.bibmath.net/crypto/moderne/quantique.php3` or `http://www.apprendre-en-ligne.net/crypto/menu/index.html`.
At the same time, quantum physic may announce the end of current cryptography : indeed, existence of quantum computers may solve factorization problems (and break RSA) in polynomial time !
Let us mention that cryptography has also looted other areas of mathematics : braids (completely broken since, see [CJ03]), chaos, algebraic geometry,... See a nice presentation in *Dossier Pour la Science, l'art du secret - La cryptographie, 36, Juillet 2002* .

Chapter 2

What is cryptography ?

2.1 Transmission of information

Transmission of information raises at least three categories of issues :

- Representation of information, how to code it, how to compress it. We study how to use words in a given alphabet to represent different information (text, image,etc.). As underlying problem, one wants to represent this data with the minimal space. This is the problem of data compression.
- Integrity of data, detection and correction of errors. Note that in the second problem, the errors come from noise on the channel and are not errors made on purpose.
- Security of information, confidentiality.

The third point is the aim of this course.

Several mathematical tools are used to solve these problems :

- logic, complexity theory ;
- combinatorics, probability, statistics, entropy ;
- arithmetic ;
- commutative algebra, algebraic geometry.

2.1.1 Security

The following list gathers a short description of the main problems we are going to deal with, some of them are strongly related :

- secrecy, confidentiality, cipher. The data have to be understandable only by the authorized persons. More precisely : the *confidentiality* (**confidentialité**) is the

property that an information is not available or disclosed to unauthorized persons, entities or processes (Norm ISO 7498-2). In particular, during a communication, the purpose is to prevent a third party to get information contained in a message transmitted on a non secure channel.

- data integrity. One wants to avoid that transmitted data may be modified or forged by an opponent. More precisely, *integrity* (**intégrité**) is the way to prevent an unauthorized modification of the data. Attacks against integrity are called *substitutions*.

- authentication. *Authentication* (**authentification**) consists of checking the identity of the different elements involved in a communication. It can be a person (in that case one speaks also about *identification*) or a machine (in a open network -web- or in a close network -cash machine-). In the case of a message one wants to prove its origin. Attacks against authentication are called *masquerade* (**mascarades**).

- non-repudiation. *Non-repudiation* is a mechanism to prevent to deny a contract. One wants for instance to prove that a message was well sent or well received. The author cannot deny that he wrote it.

- signature. *Signature* is a mechanism to prove authentication of the sender, integrity of data and non-repudiation. Attacks again signatures are called *forgery* (**falsification**).

- certification. *Certification* is the way, a trusted entity validates a certain information.

- key management. *key management* (**gestion des clés**) is the difficult problem of distribution, integrity,... of the keys.

- proof. In general, an entity (prover, P) wants to prove to another one (verifier, V) that it is in possession of a secret (password for instance). In some cases, one requests that this *proof* (**preuve**) is done without giving any information on the initial data. One speaks then about *zero-knowledge proof* (**preuve à divulgation nulle**).

In this course we are going to focus on elementary bricks with which we will build more complex systems :

- Cipher ;

- Signature ;

- Authentication ;

- Key exchange.

2.2 Cryptography technics

2.2.1 Encryption scheme

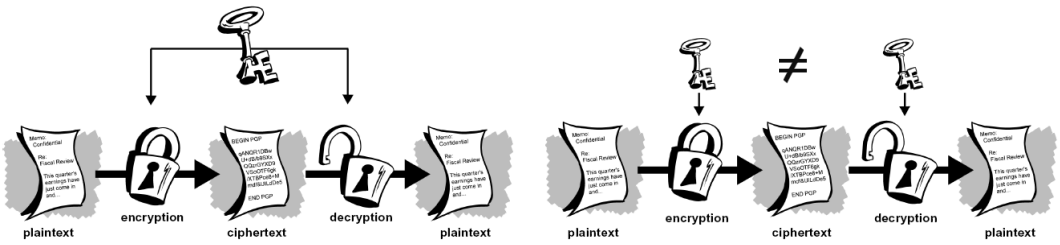
We have already seen a definition in Def. 1.2.1. Here we will extend the latter in order to include asymmetric cryptosystems.

Definition 2.2.1. *A* cryptosystem *(**cryptosystème**) is a three-uplet* $(\mathcal{P}, \mathcal{C}, \mathcal{K})$ *such that :*

1. $\mathcal{P}$ *is a finite set of possible plaintexts ;*
2. $\mathcal{C}$ *is a finite set of possible ciphertexts ;*
3. $\mathcal{K}$ *is a finite set of possible keys ;*
4. *for each* $e \in \mathcal{K}$*, there exists a* $d \in \mathcal{K}$ *such that there exists an encryption function* $\mathcal{E}_e : \mathcal{P} \to \mathcal{C}$ *and a decryption function* $\mathcal{D}_d : \mathcal{C} \to \mathcal{P}$ *satisfying* $\mathcal{D}_d \circ \mathcal{E}_e = Id$*.*

In the case where $e = d$*, one speaks about* symmetric *or* secret key *cryptosystems. Obviously, the key* e *must be secret. We have seen that the secure key exchange is then an important issue.*
If e *and* d *are distinct, one speaks about* asymmetric *or* public key *cryptosystems. In such system the encryption key can be made public, so* e *is called the* public key *and* d *is called the* private key*.*

Figure 2.1: symmetric vs asymmetric systems



One can wonder why secret key cryptography has no completely disappeared with the introduction of public key cryptography. Indeed the latter offers a solution to the problem of key exchange and moreover in a open system with n entities, it is easy to see that the number of keys to communicate is $2n$ with asymmetric when it is $n(n-1)$ for symmetric cryptography. The reason is simple : asymmetric cryptography is about 1000 slower than symmetric cryptography. Most of the present systems work then with both encryption : first the entity exchanges via an asymmetric protocol a message which is then used as a key to establish a symmetric protocol between the two entities.

2.2.2 Secret key cryptosystem

Important construction principles are *confusion* and *diffusion*. They were introduced by Shannon who studied the security of cryptosystem in the context of information theory. The confusion of a block cipher is high, if the statistical distribution of the ciphertexts depends in a complicated way on the distribution of the plaintexts. So an attacker cannot extract information on the plaintexts from the distribution on the ciphertexts. For instance shift cipher has a very low confusion and more generally linear ciphers are bad. For this reason, modern ciphers always include a non linear operation.
The diffusion of a block cipher is high if each bit of the plaintext and each bit of the key have an influence on many bits of the ciphertext.

Example 3. *Here is a list of examples, some of them will be studied in the course.*

- *DES (Data Encryption Standard, see* **??***). It is ANSU X3.92 standard suggested in 1974, published in the Federal Register in 1975 and accepted as standard in 1977. It uses* 56 *bits keys (completed with* 8 *bits of parity checking) and plaintexts/ciphertexts are coded on* 64 *bits. This system is considered now as weak because of the small size of the keys. It is still used in* 3*-DES.*

- 3*-DES. One composes three circuits in the following way : one uses two DES keys* k_1, k_2*. One ciphers the plaintext with* k_1*, deciphers with* k_2 *and ciphers again with* k_1*. See in* **??** *why one cannot use a double DES.*

- *AES (Advanced Encryption Standard, see* **??***). The new American standard suggested to replace DES. Build in 1998 by J. Daemen and V. Rijmen under the name Rijndael (pronunce 'Rhine dahl'), it became the standard FIPS-197 in 2001. There is a choice for the key :* $128, 192$ *or* 256 *bits. Although Rijndael proposes a choice for the size of blocks, the standard AES has kept only the size* 128 *bits.*

- *MISTY1, Camellia, SCHACAL-2 are other cryptosystems which have been selected by the European project NESSIE (New European Schemes for Signatures, Integrity and Encryption).*

2.2.3 Public key cryptography

To Construct a secure asymmetric cryptosystems is mathematically equivalent to find a good *one-way function* (**fonction à sens unique**). It is a function f which is easy to compute but difficult to inverse (i.e. $f^{-1}(y)$ is difficult to compute for almost all y). See 4.3.4 for a precise definition. Intuitively, we will say that something is easy to compute if one can do it in polynomial time and something is difficult if it is not. Good candidates seems to be problems in the class NP. However this is not completely true as an NP-problem can be hard in bad cases but easy for a big number of other examples. Moreover we do not know if P $\neq$ NP. The question is difficult. Actually, nobody knows if there exists one-way functions but a lot of functions seems to play this role in practice. Note also that one can distinguish between exponential and sub-exponential algorithms.

This distinction has direct consequences on the size of the keys (160 or 1024 bits). Thus, good cryptosystems are the one for which no subexponential attack is known.
To obtain an efficient cryptosystem, one has to require a last condition : the function f must become easy to inverse if one knows a secrete value : such functions are called *trapdoor one-way functions* (**fonction à sens unique avec trappe**).

We give here the most used problems :

Factorization. Given a number n which is the product of two big prime numbers $n = pq$, determine p and q. There are several (non-polynomial algorithms) to attack this problem, see **??** :

- Eratosthenes sieve : trying all prime numbers up to $\sqrt{n}$ is of course an exponential method.
- Pollard ρ-method ;
- Pollard $p-1$ method : It is an exponential method but very efficient when $p-1$ has only small prime factors ;
- continued fractions ;
- elliptic curves methods ;
- multiple polynomial quadratic sieve ;
- number field sieve.

These last four are subexponential. This problem is used in RSA, RSA signature (see 3.1, 3.4.1).
Square roots. Let p be a prime number greater than 2 and x a square in $\mathbb{Z}/p\mathbb{Z}$. It is possible to compute square roots. If $p \equiv 3 \pmod 4$, $x^{\frac{p+1}{4}}$ is a solution. If $p \equiv 1 \pmod 4$, Shanks algorithm is used (see **??**). If $n = pq$ is the product of 2 big prime numbers distinct and if p and q are known, the square root can be computed using the *Chinese remainder theorem* (**théorème du reste chinois**). But if p and q are unknown, the problem is difficult (in some sense as difficult as factorization). This is used in Rabin encryption, see 3.3.
Discrete logarithm problem (DLP). Let G be a commutative group and H a finite cyclic subgroup generated by an element a. Given a^n find n is called the *discrete logarithm problem* (**problème du logarithme discret**). In some groups this problem is difficult, for instance the multiplicative group of $\mathbb{Z}/p\mathbb{Z}$ (or more generally of a finite field). There are also several way to attack the problem, see **??** :

- Pollard ρ method ;
- giant step, baby step (Shanks) ;
- Pohlig and Hellman method ;

- Index calculus : Adleman, Coppersmith.

Index calculus methods leads to sub-exponential algorithms for finite fields. However for other groups (elliptic curves) it is not the case. This problem is used in 3.2 and 3.4.2 and 3.4.

Remark 6. *Shoup [SHO97], generalizing a result of Nechaev has shown that if p is the largest prime dividing the group order, a generic algorithm to solve the DLP with a probability bounded away from zero has to perform $\mathcal{O}(\sqrt{p})$ operations.*

Example 4. *Some asymmetric cryptosystems:*

- *Diffie-Hellman. This was the first protocol using public key cryptography. It is used for key exchange (see 3.5).*
- *RSA (Rivest-Shamir-Adleman). This system is based on the difficulty of factorization of a product of two prime numbers. Associated with KEM (Key encapsulation Mechanism) (**encodage**) it has been selected by NESSIE.*
- *ElGammal. This system is based on the difficulty of the Discrete Logarithm problem in some groups.*
- *PSEC (Provable Secure Elliptic Curve Encryption). It is based on computation on the group of points of an elliptic curve. Associated with KEM, it belongs to NESSIE.*
- *Rabin. This system is based on the difficulty to compute the square root modulo a product n of two unknown prime numbers.*
- *McEliece. This system is based on the difficulty to decode in the error-coding theory.*

2.2.4 Random functions

Cryptographic protocols often use random elements. A difficult question is how to produce these elements. On one hand, 'nature' supplies hardware-based generators like the radioactive decay or the time between two keyboard strokes.
However it is often too time-consuming to generate true random numbers, then pseudo-random numbers are used. Intuitively a pseudorandom generator can be described as an algorithm that, given a short sequence of bits, produces a long sequence of bits that looks random. Of course since the algorithm is deterministic, this explains the 'pseudo'. M. Blum and S. Micali formalized this notion.

Definition 2.2.2. *If D, D' are two distributions, a* distinguer *(**distingueur**) of D and D' is a probabilistic polynomial algorithm A such that there exists a polynomial P satisfying*

$$|Pr(A(y) = 1) - Pr(A(y') = 1)| > \frac{1}{P(n)}$$

for all $y \in D, y' \in D'$ of length less than n.

Definition 2.2.3. *Two distributions D, D' are called* undistinguishable *(**indistinguables**) if for all probabilistic polynomial algorithm A and for n big enough, A is not a distinguer of D and D'. If D' is the uniform distribution, D is called a* pseudorandom *(**pseudo-aléatoire**) distribution.*

We assume now that D' is a uniform distribution.

Definition 2.2.4. *If D is a distribution of bits, a* predicator *(**prédicateur**) of D is a probabilistic polynomial algorithm A such that there exists a polynomial P satisfying*

$$Pr(A(y_1 \dots y_i) = y_{i+1}) > \frac{1}{2} + \frac{1}{P(n)}$$

for all $y \in D$ of length n and $i < n$.
D is called unpredictable *(**imprédictible**) if such an A does not exist.*

Theorem 2.2.1. *A distribution D is pseudorandom if and only if it unpredictable.*

Definition 2.2.5. *A* pseudorandom generator *(**générateur pseudo-aléatoire**) is a deterministic polynomial algorithm G such that for all input x of G, one has $|G(x)|_2 > |x|_2$ and $G(x)$ is pseudorandom when $|x|_2 \mapsto \infty$.*

A pseudorandom generator has been constructed by L. Blum, M. Blum and M. Shub. Let p, q be two big prime numbers congruent to 3 modulo 4. Let x_0 be a quadratic residue modulo $n = pq$. The distribution $D = y_1 \dots y_n$ is defined by $x_i \equiv x_{i-1}^2 \pmod{n}$ and $y_i = x_i \pmod 2$. This generator has been proved secure. Unfortunately it is too slow for several applications.

2.2.5 Hash function

Definition 2.2.6. *Let Σ be an alphabet. Let denote by Σ^* the set of all words. A* hash function *(**fonction de hachage**) is a map $h : \Sigma^* \to \Sigma^n$ for some n.*

Example 5. *The map that sends $b_1 \dots b_k$ in $\{0,1\}^*$ to $b_1 \oplus \dots \oplus b_k$ is a hash function.*

For cryptographic purposes, we add two conditions :

- it must be hard to determine a pre-image of an element.
- it must be resistant to *collisions* which means that it is practically impossible to find two messages x_1, x_2 such that $h(x_1) = h(x_2)$.

As h is never injective, these two properties can be realized only because of the inextricability of the computations. Due to the *birthday paradox* (**paradoxe des anniversaires**), the second condition requires a size of n which is nowadays 128 bits at least (see 4.2.4).

Example 6. *Among the current hast functions, let us mention : SHA0, SHA1 (Secure Hash Algorithm) on $128, 160, 224, 256, 384, 512$ bits, MD5 (Message Digest) on 128 bits, Whirlpool on 512 bits (this one has been selected by NESSIE).*
Note that some of these functions have been recently successfully attacked (i.e. finding collisions): SHA0 and MD5 have been broken (see Le Monde 01/01/09).

2.3 Protocols

Besides the classical encryption-decryption scheme, we will describe the following protocols.

2.3.1 Signature

Signature algorithms are made of several parts and can be schematized by:

$$\begin{cases} \text{Message} + \text{Alice's private key} & = \text{Signature} \\ \text{Message} + \text{Signature} + \text{Alice's public key} & = \text{Yes/No} \end{cases}$$

Let us assume that A wants to sign an already ciphered message y. The user A has a public key e_A and a secret key d_A. A digest m of the message y is computed with a hash function h, $m = h(y)$. There is then a signature function $\mathcal{S}$ with which one computes $s = \mathcal{S}(d_A, m)$ called the *appendix of the signature*. A transmits then the couple (y, s). To verify the signature, the system produces also a verification function $\mathcal{V}$ such that $\mathcal{V}(e_A, y, s)$ is 'true' (or 1) if and only if the message y is signed by A. We will denote by $\mathcal{S}_A$ the signature function of A, i.e. the function $\mathcal{S}_A(y) = (y, \mathcal{S}(d_A, h(y)))$, and

$$\mathcal{V}_A(y, s) = \begin{cases} 0 \text{ if } (y, s) \neq \mathcal{S}_A(y), \\ 1 \text{ if } (y, s) = \mathcal{S}_A(y). \end{cases}$$

Notice that this process implies non-repudiation. Indeed, since everybody knows the key e_A and since only A can produce $S_A(y)$, everybody can check that the signature is A's one.

Example 7. *Here are some examples of signature with appendix.*

- *RSA, see 3.4.1. It is a 'reverse' RSA : the one who signs does it with its private key. Everybody can check with the public key of A that it is the right signature. Associated with the encapsulation PSS (Probabilistic Signature scheme), this was selected by NESSIE.*
- *DSS (Digital Signature Standard, see 3.4). Use DSA (Digital Signature Algorithm) based on the discrete logarithm in $(\mathbb{Z}/p\mathbb{Z})^*$ or ECDSA (Elliptic Curve Digital Signature Algorithm) based on the discrete logarithm in an elliptic curve. ECDSA has been selected by NESSIE.*

2.3.2 Authentication

One can think about using signature to obtain authentication. But the requirements are a bit different. For instance, authentication does not require non-repudiation. Moreover with signature everybody can verify the signature. However, authentication may require to prevent *re-play* (**rejeu**), i.e. prevent that data can be reused (for instance that a check is shown twice). This can be achieved by a *dating* (**datation**) system.

Example 8. *Here are some of the most well known authentication systems:*

- *Feige-Fiat-Shamir authentication system based on the difficulty to extract a square root modulo $n = pq$, see 3.6.3.*
- *It is possible to construct MAC (Message Authentication Code) by using secret ciphers (DES, AES).*
- *UMAC (accepted by NESSIE) which uses a interesting family of hash functions (see [BHKKR99]).*

2.3.3 Key management

All these protocols assume at some points that one can trust that the couple (d_A, e_A) really belongs to A and is valid. These problems are part of the PKI (*Public key infrastructure* (**Infrastructure de gestion des clés**)). The key creation can be done locally or through a *Key Distribution Center* (**centre de distribution des clés**). Then the public key is registered by a *certifying authority* (**autorité de certification**) which gives a *certificate* (**certificat**) to the user. It contains the public key, the name of the user, the name of the certifying authority and a deadline. Eventually this certificate is signed by the certifying authority with its own private key. The other users know the public key and can check the certificate.

2.3.4 More complex protocols

Here are some concrete applications of the previous tools :

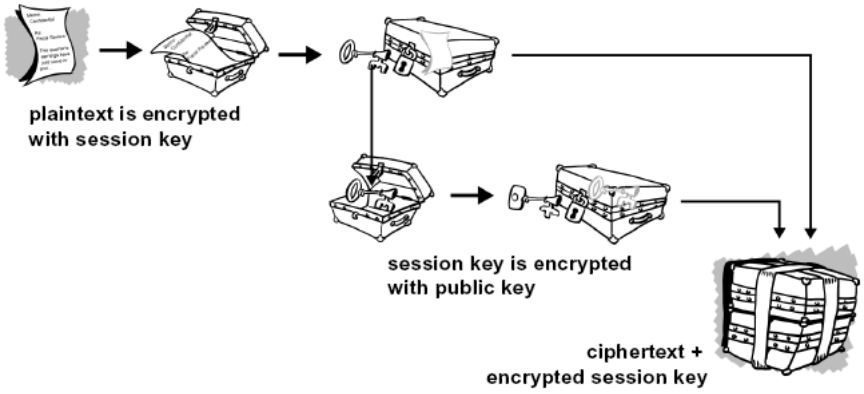
- PGP (Pretty Good Privacy) : this system provides encryption and signature. It can be used for emails or commercial transactions.
- Kerberos : a complete system with which the clients get from a key distribution center a session key to communicate with a server.
- Station to station protocol : this system uses symmetric and asymmetric encryption and also authentication procedure.
- SSH : protocol for networks to get a secure connection with a machine.

2.3.5 Normalization and standardization

These questions are always important in computer science. Recall that standards are defined *de facto* when the norms are defined *de jure* by an organism :

- NIST (National Institute of Standards and Technology) which publishes FIPS (Federal Information Processing Standards), for instance FIPS 140-1 and 140-2;
- ISO (International Standards Organization); Concerning information security, let us mention the norms ISO 17799 and ISO 15408;

Figure 2.2: How PGP encryption work



- ANSI (American National Standards Institute);
- ITU (International Telecommunication Union);
- ECMA (European Computer Manufacturers Association);
- IEEE (Institute of Electrical and Electronics Ingineers);
- AFNOR (Association Française pour la NORmalisation).

In the case of internet, standards are called RFC (Requests For Comments). One can access them on `http://www.rfc-editor.org`. There are two noteworthy subcategories : the STD which are the official standards and the FYI (For Your Information) which are tutorials.

2.4 Cryptanalysis

Cryptanalysis is the part of cryptology that deals with attacking cryptosystem. It is of course a way to make them more secure by finding their weaknesses. The main aim is of course to break the cryptosystem, i.e. to determine the plaintext or the secret key. One can also ask to find parts of the text. If the ciphertext does not allow to find any partial information on the text, it is said *semantically sure* (**sémantiquement sûr**). But one can also see other purposes : forge a signature or modify a message without being spotted. We then have to define the level of security we want (see **??**).
To developed a security theory one has also to define the power of the opponent. The security is then based on the difficulty to manage some computations.
Currently an average computer can realize during a human time $2^{50} - 2^{60}$ elementary operations. Thus the size of the keyspace has to be adapted according to the best attack known (see 2.4.2).

Remark 7. *Let us emphasize that there is no absolute proof of security. But under some assumptions and attacks, one can prove that breaking a system is as hard as to solve a problem, known to be hard. This game is the based of provable security.*

2.4.1 Attacks classification

One has first to distinguish the level of the attack : directly against the core functions, against the protocol (mathematical or logical), or even against its implementation.

Example 9. *For RSA, we have mentioned the attack against factorization (core function attack). We will see other types (see 3.1.3) : protocol attacks, common module, small private exponent, small public exponent or against the implementation (timing or power analysis attack).*
For classical cryptography we have seen in Chap. 1 several types of statistical attacks. We will add in **??** *:* differential or linear cryptanalysis *(**cryptographie différentielle ou linéaire**).*

Furthermore, one has also to decide what kind of information the opponent can get. Following Kerckhoffs' principle, the basic hypothesis is that the opponent knows everything about the system except the secret key. One distinguishes several types :

- *ciphertext-only attack* (**attaque à chiffré seul**). The opponent knows only the ciphertext.
- *known plaintext attack* (**attaque à textes clairs connus**). The opponent knows pairs of plaintext and ciphertext.
- *chosen plaintext attack* (**attaque à textes clairs choisis**) (CPA). The opponent can get ciphers of chosen plaintexts but only before the plaintext to attack (called *challenge*.)
- *adaptive chosen plaintext attack* (**attaque adaptative à textes clairs choisis**). The opponent can get ciphers of chosen plaintexts before and after the challenge.
- *chosen ciphertext attack* (**attaque à textes chiffrés choisis**) (CCA1). The opponent can decipher chosen ciphertexts before the challenge. For instance such an attack is possible if a cryptosystem is used for identification. This works as follows. Alice wants to make sure that she is connected to Bob. She sends an encrypted random number to Bob for which only Bob knows the decryption key. Bob decrypt the message and sends it back to Alice. An attacker can try to impersonate Alice. Instead of sending random numbers he sends messages of his choice.
- *adaptative chosen ciphertext attack* (**attaque adaptative à textes chiffrés choisis**) (CCA2). The opponent can decipher chosen ciphertexts before and after the challenge (apart from this latter).

In public cryptography, as everybody is allowed to cipher, CCA2 is the most powerful attack. CCA attacks are called active attacks in contrast with the first three which are 'passive' attacks (or CPA attacks). Nowadays security level for public key cryptography is to ask that a system is semantically secure against CCA2 attacks.

2.4.2 Keys length : where size matters

Let us take the example of DES. To find the key, require on average $\#\mathcal{K}/2$ attempts. As the secret key is 56 bits long, one has $2^{55} \approx 10^{17}$ keys to try. So it would take :

- 39 days on 10000 Pentium (done in 1997).
- 2.5 days on a \$250000 machine (1998).
- 35 minutes for \$1000000.

The next results are extracted from a table created by K. Lenstra and E.R. Verheul. Considering that DES was in 1982 sufficiently secure for commercial applications, this table gives an estimation of the size of keys in the future for an equivalent security. For a symmetric system with a key of size b, the attack we consider is the one which run over 2^{b-1} elements (on average). For asymmetric systems, computations are based on the best factorization or discret logarithm algorithms. To complete the table one applies Moore law (the computer power doubles every 18 months).

year	symmetric cryptosystems	RSA module or DL in $(\mathbb{Z}/p\mathbb{Z})^*$	size of the group for general DL	elliptic curves	Number of years on a PII 450 Mhz
1982	56	417	102	105	1.11×10^3
2005	74	1149	131	139	2.26×10^8
2010	78	1369	138	146	3.22×10^9
2015	82	1613	145	154	4.59×10^{10}
2020	86	1881	151	188	6.54×10^{11}
2025	89	2174	158	169	9.33×10^{12}
2030	93	2493	165	176	1.33×10^{14}
2040	101	3214	179	191	2.7×10^{16}

Chapter 3

Public key cryptography

In this chapter we will study with more details different asymmetric cryptosystems. The idea was first presented in the seminal paper of Diffie and Hellman entitled *New directions in Cryptography*. Although Diffie and Hellman invented the concept of public key cryptography in 1976 it was until a year or so later that the first system, namely RSA, was invented.
The previous paragraph is how the 'official' history of public key cryptography goes. However in the late 1990s an unofficial history came to light. It turned out that in 1969, a cryptographer called James Ellis, working for the British government's communication headquarters GCHQ, invented the concept of public key cryptography (or non-secret encryption as he called it). Ellis did not however have a system.
The problem of finding such a public key encryption system was given to an new recruit to GCHQ called Clifford Cocks in 1973. Within a day Cocks invented what was essentially the RSA algorithm . In 1974 another employee at GCHQ, Malcolm Williamson invented the concept the concept of Diffie-Hellman key exchange. Hence by 1974 the British security services has already discovered the man techniques in public key cryptography.

3.1 RSA

3.1.1 Encryption-decryption scheme

Let $n = pq$ be the product of two distinct big prime numbers. If ϕ denotes Euler function (see 4.1.4), one has

$$\phi(n) = (p-1)(q-1).$$

Let e be a number coprime with $\phi(n)$. By BÃ©zout theorem, there exists d such that

$$ed \equiv 1 \pmod{\phi(n)}.$$

Lemma 3.1.1. *Let* $\mathcal{E}_e(x) \equiv x^e \pmod{n}$. *This function defines a permutation of* $(\mathbb{Z}/n\mathbb{Z})^*$ *with inverse* $\mathcal{D}_d(y) \equiv y^d \pmod{n}$.

Proof. By Fermat's little theorem (see Cor. 4.1.3), one has

$$\mathcal{D}_d \circ \mathcal{E}_e(x) \equiv x^{de} \pmod{n} \equiv x^{k\phi(n)+1} \pmod{n} \equiv x \pmod{n}$$

so the function $\mathcal{E}_e$ is injective, thus bijective and $\mathcal{D}_d$ is its inverse. □

The cryptosystem RSA is given by

- $\mathcal{P} = \mathcal{C} = (\mathbb{Z}/n\mathbb{Z})^*$;
- $\mathcal{K}$ is a subset of $\mathbb{Z}/n\mathbb{Z}$ given by the elements e prime to $\phi(n)$.
- For $e \in \mathcal{K}$ one denotes by d an element such that $ed \equiv 1 \pmod{\phi(n)}$. Then $\mathcal{E}_e(x) \equiv x^e \pmod{n}$ and $\mathcal{D}_d(y) \equiv y^d \pmod{n}$.

A priori the security of RSA relies on the computation of $\phi(n)$. Clearly, if the factorization of n is known, $\phi(n)$ also.

3.1.2 Practical construction

We have to find big prime numbers p, q. In general this is done by random choice of an integer and then by a primality test. The fact that this procedure is efficient rely on the two following facts :

- The density of primes is high enough, see 4.1.2;
- There are fast algorithms to test primality. Usually one uses a probabilistic polynomial algorithm. Recently, a deterministic polynomial algorithm has been found. However its constants are still not as good as the probabilistic one. One picks a odd number and test if it is prime. If it is not one increases iteratively by 2 until one finds one. See **??**.

How does man find the encryption key e ? A way to do it is also to pick e randomly and to test if $\gcd(e, \phi(n)) = 1$. If it is not the case, one picks another one. The number m of distinct primes dividing $\phi(n)$ is less than $m = \log_2(\phi(n))$ since

$$\phi(n) \geq \prod_{p|\phi(n)} p \geq 2^m.$$

By 4.1.2 we know that the number of primes less than $\phi(n)$ is equivalent to $\phi(n)/\log(\phi(n))$ so the probability to find an integer $1 < e < \phi(n)$ and coprime with $\phi(n)$ is greater than

$$\frac{\phi(n)/\log(\phi(n)) - m}{\phi(n)} \sim \frac{1}{\log(\phi(n))}.$$

Such a number e can then be find in $\log(\phi(n))$ attempts.

Another solution is to construct a prime number strictly bigger than p and q and smaller than $\phi(n)$. Or one may fix e from the beginning (for instance $e = 3$). One constructs then p, q such that e does not divides $p-1$ or $q-1$. All these methods are feasible in a good time.
The computation of the exponent d is done by Euclid algorithm (see 4.3.3).
The computation of $x^e \pmod n, y^e \pmod n$ can be done by the fast exponentiation method. One can make it even faster using the factorization of n and the Chinese remainder theorem. One gains a factor 4 but one has to be careful (see 3.4.1).

3.1.3 Attacks

As we have seen, attacks can be made at different levels.
On the level of the function itself, it is enough to know $\phi(n)$ to break the cipher. But

Proposition 3.1.1. *Knowing $\phi(n)$ is equivalent to the factorization of n.*

Proof. If p, q are known, one has $\phi(n) = (p-1)(q-1)$.
Conversely, one has

$$\begin{aligned} (p-1)\left(\frac{n}{p} - 1\right) &= \phi(n) \\ p^2 - (n+1-\phi(n))p + n &= 0 \end{aligned}$$

This equation can be solved in polynomial time. □

So can we break RSA without factorizing n ? If not, it means that the difficulty of the problem depends on the best known factorization algorithms, which are so far subexponential. The best current record is 200 digits (RSA200) which has 624 bits. Thus n must have 1024 bits.
Unfortunately, the answer to this question is not known and one can give only some related results.

Theorem 3.1.1. *There is a Las Vegas algorithm taking the module n and e and using an oracle which gives d and having for output the factorization of n.*

Proof. Let us note that the equation

$$x^2 \equiv 1 \pmod n$$

is equivalent to

$$\begin{aligned} x^2 &\equiv 1 \pmod p \\ x^2 &\equiv 1 \pmod q. \end{aligned}$$

So the first equation admits 4 solutions, two of them are the trivial solutions ± 1.
If x is a non-trivial solution, n divides $(x+1)(x-1)$ but none of the factors. Thus

$$\begin{aligned} \gcd(x+1, n) &= p \text{ or } q, \\ \gcd(x-1, n) &= q \text{ or } p. \end{aligned}$$

Hence, if we know a non-trivial square root of 1 modulo n we can factorize n in polynomial time. How to compute such a root ?
Let us pick a random element $1 < \omega \leq n-1$. If $\gcd(\omega, n) > 1$ we have a factor of n. Otherwise, using the oracle, we write $ed - 1 = 2^s r$ where $s \geq 1$ and r odd. We have

$$\omega^{2^s r} = \omega^{ed-1} = \omega^{k\phi(n)}$$

so

$$\omega^{2^s r} \equiv 1 \pmod{n}.$$

Let t_0 be the smallest integer such that

$$\omega^{2^{t_0} r} \equiv 1 \pmod{n}.$$

If $t_0 > 0$, we define

$$v_0 = \omega^{2^{t_0-1} r}.$$

Hence $v_0 \not\equiv 1 \pmod{n}$ but $v_0^2 \equiv 1 \pmod{n}$. Now if $v_0 \not\equiv -1 \pmod{n}$ we have a non-trivial root of 1 modulo n.
The algorithm fails if $t_0 = 0$ or if there exists $0 \leq t \leq s-1$ such that $\omega^{2^t r} = -1 \pmod{n}$. The number of such ω is less than $\phi(n)/4$ according to Rabin's theorem (see **??**). So the probability at each round to get a good ω is bigger than $3/4$. As each round needs a polynomial time we can conclude. □

Note that this seems to be a good argument for the difficulty of RSA. But we have not proved that it is not possible to decrypt without knowing d !
We move now to attacks on the protocol.

- common module. It is a very simple attack. Let us assume that A_1 and A_2 are two users with the same RSA module n. Let e_{A_1} (resp. e_{A_2}) the public key of A_1 (resp. A_2). First, A_1 and A_2 can know each other private key. But there is worst. Let us assume now that e_{A_i} are coprime. One can find x_1, x_2 such that

  $$x_1 e_{A_1} + x_2 e_{A_2} = 1.$$

  If B sends a message x to A_1, A_2 one has

  $$y_{A_i} \equiv x^{e_{A_i}} \pmod{n}$$

  so O can compute x by

  $$x \equiv y_{A_1}^{x_1} y_{A_2}^{x_2} \pmod{n}.$$

  So two different users cannot use the same RSA module.

- small private exponent. In order to speed up the computations, in particular on small devices like smart card, one may want to choose a small exponent d. However it is not a good choice as there exists an attack due to Wiener, based on

continued fractions (see 4.5). We assume that $p > \sqrt{n} > q$ and $1 < p/q < 2$ and $1 < e < \phi(n)$. Consider

$$ed \equiv 1 \pmod{\phi(n)},$$

this relation can be rewritten

$$ed = 1 + k(n - (p+q) + 1)$$

where $k < d$ by BÃ©zout algorithm (since $e < \phi(n)$). So dividing by nd

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \frac{|1 + k(1 - (p+q))|}{nd},$$

so

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{k(p+q)}{dn},$$

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{kq(p/q+1)}{dn},$$

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{3kq}{dn},$$

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{3}{\sqrt{n}}.$$

If $d \leq n^{\frac{1}{4}}/\sqrt{6}$ then

$$\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{1}{2d^2}.$$

This inequality shows that k/d is a reduction of the continued fractions development of e/n (see 4.5). It is enough to make the continued fraction development of e/n and to check for each reduction if the denominator is a good candidate for d (by trying to factorize n). There exist improvements of this method, in particular Boneh-Durfee attack which works with $d < n^{0.292}$.

- small public exponent, broadcast attack. Let us first look at a simple case. Let A_1, A_2, A_3 be three users having for public keys $(n_i, 3)$. If B sends a message x to A_is such that $x < \inf(n_1, n_2, n_3)$ then Oscar gets by eavesdropping

$$\begin{aligned} y_1 &\equiv x^3 \pmod{n_1}, \\ y_2 &\equiv x^3 \pmod{n_2}, \\ y_3 &\equiv x^3 \pmod{n_3}. \end{aligned}$$

Using the Chinese remainder theorem, we get $y \equiv x^3 \pmod{n_1 n_2 n_3}$. But as $x^3 < n_1 n_2 n_3$, one has $y = x^3$ in $\mathbb{N}$, Oscar can find x by taking a cubic root in $\mathbb{N}$. Let us consider now the case of k users A_i with public keys (n_i, e_i) where the n_is are coprime. Let us assume that B sends a message $x < \inf(n_i)$ to the A_i's. In order

to avoid the previous attack, he adds noise with certain monic public polynomials f_i (which induces bijections on $\mathbb{Z}/n_i\mathbb{Z}$, called permutation polynomials), i.e

$$y_i \equiv (f_i(x))^{e_i} \pmod{n_i}.$$

Let $g_i = f_i^{e_i} - y_i$. If $k \geq \max(e_i \deg(f_i))$, Oscar can find x. In order to do so, Oscar uses the Chinese remainder theorem and construct the polynomial function

$$g(u) = \sum_{i=1}^{k} u_i g_i(u),$$

where

$$u_i \equiv \begin{cases} 1 \pmod{n_i} \\ 0 \pmod{n_j} \text{ if } j \neq i. \end{cases}$$

By multiplying each g_i by an appropriate power of x, we can assume that $\deg(g_i) = \deg(g_j)$ for $i \neq j$. As $\sum u_i$ may be taken equal to 1 (with $u_i = \alpha_i \prod_{j \neq i} n_j$ and α_i defined by $\sum \alpha_i (\prod_{j \neq i} n_j) = 1$) so g is monic.
If $n = n_1 \cdots n_k$, x is the unique solution of

$$g(u) \equiv 0 \pmod{n}$$

such that

$$x < \inf(n_i) < n^{1/k} \leq n^{1/deg(g)}.$$

Oscar uses then the following theorem due to Coppersmith:

Theorem 3.1.2. *Let n be an integer and $g \in \mathbb{Z}[x]$ a monic polynomial. Let $X = n^{1/\deg(g) - \epsilon}$ for a certain $\epsilon > 0$. It is then possible to compute in an efficient way the integers x such that $g(x) = 0$ and $|x| < X$. The computation time is bound by the time of the LLL algorithm on a lattice of dimension $\mathcal{O}(\min(1/\epsilon, \log_2(n)))$.*

We move now to the last category of attacks.

- timing or power analysis, see [KJJ99]. If RSA is implemented on a smart card, one can use an attack due to Kocher to find the key. The attack takes into account the usual multiplication and power d algorithms modulo n (for instance, a fast exponentiation algorithm). The difference of timing between these operations can be used to estimate the bits of d. Actually, the attack is not so easy to perform. Kocher has also invented an attack based on the power consummation called *power cryptanalysis* (**analyse de courant**).

3.1.4 Security

What is the security of RSA ? Recall that intuitively semantic security means that it is not possible to get any information from the ciphertext. It can be proved that a

deterministic encryption cannot be semantically secure. For instance, if e is odd, one has for the Jacobi symbol (see 4.4.1):

$$\left(\frac{x^e}{n}\right) = \left(\frac{x^e}{p}\right)\left(\frac{x^e}{q}\right) = \left(\frac{x}{p}\right)\left(\frac{x}{q}\right) = \left(\frac{x}{n}\right).$$

Thus RSA is not semantically secure against passive attacks. On the other hand even this basic RSA does not give away some important information, as the parity.

Theorem 3.1.3. *Let f be the function defined by*

$$f(y) = \begin{cases} 0 \text{ if } \mathcal{D}_d(y) \text{ is even,} \\ 1 \text{ if } \mathcal{D}_d(y) \text{ is odd.} \end{cases}$$

If f can be computed practically, it is possible to decrypt every message.

Proof. First let us show that to compute f is equivalent to compute g defined by

$$g(y) = \begin{cases} 0 \text{ if } \mathcal{D}_d(y) \leq n/2 \\ 1 \text{ if } \mathcal{D}_d(y) > n/2. \end{cases}$$

To do so, let $0 \leq x < n$ such that $y \equiv x^e \pmod{n}$. Denote by $y_1 \equiv 2^e y \pmod{n}$ and $x_1 \equiv 2x \pmod{n}$. We have

$$\begin{aligned} \mathcal{E}_e(x_1) &\equiv (2x)^e \pmod{n} \\ &\equiv 2^e y \pmod{n} \\ &\equiv y_1 \pmod{n} \end{aligned}$$

If $0 \leq x \leq n/2$ then $x_1 = 2x$ and x_1 is even. If $n/2 < x < n$ then $x_1 = 2x - n$ so x_1 is odd and reciprocally. So

$$f(2^e y) = f(y_1) = g(y).$$

Now

$$g(2^{-e} y) = f(2^e 2^{-e} y) = f(y).$$

Thus, a polynomial algorithm to compute g gives a polynomial algorithm to compute f and reciprocally.
Let us define now a polynomial algorithm in the size of n which uses the function g as an oracle. For all $k \geq 1$, let denote

$$\begin{aligned} x_k &\equiv 2^k x \pmod{n}, \\ y_k &\equiv \mathcal{E}_e(2^k x) \pmod{n} \\ I_k &= \bigcup_{q=0}^{2^{k-1}-1} \left[\frac{2qn}{2^k}, \frac{(2q+1)n}{2^k}\right[. \end{aligned}$$

We have

$$y_k \equiv (2^k x)^e \pmod{n} \equiv 2^{ke} x^e \pmod{n} = 2^{ke} y \pmod{n}.$$

If $x \in I_{k+1}$ then there exists q such that

$$\begin{aligned} \frac{2qn}{2^{k+1}} &\leq x &< \frac{(2q+1)n}{2^{k+1}}, \\ qn &\leq 2^k x &< qn + \frac{n}{2}, \end{aligned}$$

so $0 \leq 2^k x \pmod n < \frac{n}{2}$, $0 \leq x_k < \frac{n}{2}$ and $g(y_k) = 0$. Reciprocally, if $g(y_k) = 0$ then $x \in I_{k+1}$.
In the same way, if $x \notin I_{k+1}$, then there exists q such that

$$\begin{aligned} \frac{(2q+1)n}{2^{k+1}} &< x &< \frac{2(q+1)n}{2^{k+1}}, \\ qn + \frac{n}{2} &\leq 2^k x &< (q+1)n, \end{aligned}$$

so $\frac{n}{2} \leq 2^k x \pmod n < n$, $\frac{n}{2} \leq x_k < n$ and $g(y_k) = 1$. Reciprocally, if $g(y_k) = 1$ then $x \notin I_{k+1}$.
If y is given one can then compute in polynomial time for each value of k the value y_k and $g(y_k)$. One can then know if $x \in I_{k+1}$ or not. Denote

$$A_N = \cap_{k=1}^{N} \chi_k$$

where $\chi_k = I_k$ (resp. $\complement I_k$) if $x \in I_k$ (resp. $x \notin I_k$). The set A_N is an interval of length $n/2^N$, so for $N > \log_2(n)$, the length of the interval is less than 1 and it contains at most one integer, x. □

Remark 8. *A bit of the plaintext which is as difficult to compute as the whole plaintext is called a* hard-core bit *(**hard-core bit**). We have shown that the last bit is a hard-core bit.*

On the contrary if the adversary is more powerful, RSA falls completely.

Proposition 3.1.2. *RSA is not secure against a CCA2 attack.*

Proof. Suppose the message Oscar wants to break is $y = x^e \pmod n$. Oscar creates $y' = 2^e y$ and asks the oracle to decrypt y' and get x'. Now Oscar can compute $x'/2 = (2^e y)^d/2 = x$. □

3.1.5 Encryption and decryption scheme

The previous section showed us the necessity to perform RSA with a specific scheme to get security. Given a plaintext as a block of characters, how do we cipher it with RSA ? The difficulty of inversion of RSA is not enough to ensure the security of the system. For instance if we decide to apply RSA on blocks of length 1 then a simple statistical analysis is enough to decrypt the ciphertext.
We show here the RSAES-OAEP scheme (RSA Encryption Scheme- Optimal Asymmetric Encryption Padding), recommended in the directive PKCS#1v2.1 by RSA society.

We start with M a message given by a sequence of octets. To encode we use the following functions:

$$M \overset{\text{EME-OAEP}}{\longrightarrow} m \overset{\text{OS2IP}}{\longrightarrow} x \overset{\mathcal{E}_e}{\longrightarrow} y \overset{\text{I2OSP}}{\longrightarrow} C,$$

and to decode

$$C \overset{\text{OS2IP}}{\longrightarrow} y \overset{\mathcal{D}_d}{\longrightarrow} x \overset{\text{I2OSP}}{\longrightarrow} m \overset{\text{EME-OAEP}}{\longrightarrow} M.$$

Let us describe the missing procedures.
The procedure OS2IP (Octet String to Integer Primitive) is very easy. If $m = m_1 \dots m_l$ where m_i are octets (i.e. numbers between 0 and $2^8 = 255$), one constructs

$$x = \sum_{i=0}^{l-1} m_{l-i} 256^i.$$

The value of $|m|_{256}$ depends on the size of the module n, for instance $|m|_{256} = |n|_{256} - 1$. The procedure I2OSP is the inverse, i.e. it corresponds to the decomposition in base 256 (see 4.1.3).
The procedure EME-OAEP for encryption and decryption requires a hash function h (which gives a sequence of octets of length $|h|_{256} > 16$ i.e. which is without collisions), a public parameter P (which is a sequence of octets) and a pseudo-random function g (which takes a sequence of octets and an integer l and gives a sequence of octets of length l). One computes then a sequence PS of null octets of length $|m|_{256} - |M|_{256} - 2|h|_{256} - 1$ which is also bigger than 16. We concatenate

$$DB = h(P)||PS||00001111||M \quad (|DB|_{256} = |m|_{256} - |h|_{256}).$$

Now one picks randomly a sequence s of length $|h|_{256}$ and one computes

$$\begin{aligned} DBmask &= g(s, |m|_{256} - |h|_{256}) \\ maskedDB &= DB \oplus DBmask \\ smask &= g(maskedDB, |h|_{256}) \\ maskeds &= s \oplus smask \\ m &= maskeds||maskedDB \end{aligned}$$

For decoding, one starts with m. The first $|h|_{256}$ octets of m give us $maskeds$, the others are $maskedDB$. One compute then

$$\begin{aligned} smask &= g(maskedDB, |h|_{256}) \\ s &= smask \oplus maskeds \\ DBmask &= g(s, |m|_{256} - |h|_{256}) \\ DB &= DBmask \oplus maskedDB \end{aligned}$$

Is is now easy to reconstitute M.
It has been shown that RSA-OAEP is semantically secure against a CCA2 attack (this relies on the hardness of the partial one-wayness of RSA, see [FOPS01]). This is noticeable as OAEP combined with another trapdoor one-way function is sure against CCA1 but may not be against CCA2.

Remark 9. *Originally it was believed that RSA-OAEP was* plaintext-aware*, i.e. that it is difficult to construct a valid ciphertext without being given the corresponding plaintext to start with. This implies that one cannot mount a CCA attack. Since to write down a ciphertext requires you to know the plaintext, so why would you ask the decryption oracle to decrypt the ciphertext ? It is then a very strong notion of security but this property is not valid.*

3.2 ElGammal encryption

Let p be a prime number. We know (see Prop. 4.1.9) that the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic. Let α be a generator and define

$$\mathcal{E}_\alpha(x) \equiv \alpha^x \pmod{p}.$$

The inverse of $\mathcal{E}_\alpha$ is called the *discrete logarithm* (**logarithme discret**), DL. The *discrete logarithm problem*, denote DLP, is finding the discrete logarithm of α^x knowing p, α. This problem is NP and known to be difficult. The function $\mathcal{E}_\alpha$ is a one-way function and no trapdoor is known so far. Its use in a public cryptosystem will not be direct.

3.2.1 The Diffie-Hellman problem

There are two types, the computational and decisional one. The *computational Diffie-Hellman problem* (**problÃ¨me de Diffie-Hellman calculatoire**), called CDH, is :

- **Input :** p is a prime number, α a generator of $(\mathbb{Z}/p\mathbb{Z})^*$, $\alpha^a \pmod{p}, \alpha^b \pmod{p}$;
- **Output :** compute $\alpha^{ab} \pmod{p}$.

If one can solve the DLP in feasible time, we can clearly solve the CDH problem. The reciprocal is not known but one does not know examples where CDH is easy and not DLP. For some prime numbers, it has been proved that the two problems are equivalent. More precisely, if $p-1$ is not divisible by the square of a big prime number, then CDH is not easier than DLP (see [MW99]).
The decisional Diffie-Hellman problem (**problÃ¨me de dÃ©cision de Diffie-Hellman**) is:

- **Input :** p is a prime number, α a generator of $(\mathbb{Z}/p\mathbb{Z})^*$, $\alpha^a \pmod{p}, \alpha^b \pmod{p}$ and y an integer;
- **Output :** Is $y = \alpha^{ab} \pmod{p}$?

If one can solve the computational Diffie-Hellman problem, one can solve the decisional one. The converse is generally not known. However, one knows groups where the decisional problem is easy but where one does not know simple algorithm to solve the computational one (for instance a supersingular elliptic curve).

3.2.2 ElGammal encryption scheme

Let p be a big prime number such that $p-1$ has a big prime factor q. Let α be an element of order q in $(\mathbb{Z}/p\mathbb{Z})^*$ (such an element exists because of Rem. 22). One calls H the subgroup generated by α.

$\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^*$;
$\mathcal{C} = H \times (\mathbb{Z}/p\mathbb{Z})^*$;
Public key : p, α a generator of H of order q, $\beta \equiv \alpha^a \pmod p$;
Secret key : $a \in [0, q-2]$;
$\mathcal{E}_e(x) \equiv (y_1, y_2) \equiv (\alpha^k \pmod p, x\beta^k \pmod p)$ for a $k \in [0, q-2]$ which changes for every x;
$\mathcal{D}_d((y_1, y_2)) \equiv y_2 y_1^{-a} \pmod p$.

Note that the encryption is not deterministic : the parameter k is chosen randomly. This function has a trapdoor : if someone knows the DL a of β, he can easily construct x knowing the relation $y_2 = xy_1^a$ but he cannot find k. There are some variants :

- It is possible to work on the entire $(\mathbb{Z}/p\mathbb{Z})^*$ by taking α a primitive element. Pohlig-Hellman algorithm (**??**) shows that this does not provide a better security and one has to work with bigger exponents. Moreover the determination of a primitive element requires in any case the construction of a prime number q dividing $p-1$ (see **??**).
- The modified ElGammal encryption is to restrict the plaintext space to H.
- A last variant works with an hash function h. One has for instance

$$y_1 = \alpha^k \pmod p, \quad y_2 = h(\beta^k \pmod p) \oplus x.$$

3.2.3 Practical construction

We have to construct p, q, α.

- Generate q. We pick a random number of a given size and apply a primality test.
- Construct p. We look for a prime number p of the given size in the arithmetical progression $aq+1$. If it does not work, we modify the value of q. This procedure can be carried out in average time of the size of p.
- Find α. We have $p-1 = sq$. Pick randomly $0 < \beta < p$ until $\beta^s \not\equiv 1 \pmod p$. Write $\alpha = \beta^s$.

3.2.4 Attacks and security

What are the size of p and q ? For any cyclic group G of order n, there exist two attacks : one is in $\mathcal{O}(\sqrt{n})$ (Shanks) and the other in $\mathcal{O}(\sqrt{t})$ (Pohlig-Hellman) where t is the biggest prime factor of n (see **??**). For this reason, q has 160 bits. In the particular case where $G = (\mathbb{Z}/p\mathbb{Z})^*$, there exists also a subexponential algorithm in the size of p called *index calculus attack* (**mÃ©thode de l'index**) that we can apply also in a cyclic subgroup H of order q. But one does not know how to apply it directly on H. So one has to balance the values of p and q, for instance p of size 1024 and q of size 160 bits.
What can we say about semantical security ? First note that the Legendre symbol of α is 1 (see 4.4.1). Indeed as q is big it is not 2, so q divides $(p-1)/2$ and

$$\left(\frac{\alpha}{p}\right) \equiv \alpha^{(p-1)/2} \equiv (\alpha^q)^{(p-1)/(2q)} \equiv 1 \pmod{p}.$$

So the Legendre symbol of y_2 equal the Legendre symbol of x and the system is not semantically secure.
However the modify version (with $x \in H$) has always Legendre symbol equal to 1 (as x is a power of α). More precisely one can show that the semantical security against known plaintext attack is equivalent to the decisional Diffie-Hellman problem. But if the attacker is more powerful (CCA2), then the system is not semantically secure (for CCA1 it is not known), see **??**. Moreover, another issue is to translate a plaintext in an element of H. This is inextricable apart from the case of Sophie Germain primes $p = 2q + 1$: indeed the elements of H are the quadratic residues modulo p. So if $0 \leq x < q$ is a quadratic residue we keep it. Otherwise we change it in $p - x$. One has then :

$$\left(\frac{p-x}{p}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{x}{p}\right) = 1$$

because -1 is not a quadratic residue ($(p-1)/2$ is odd). But this case is not really interesting because the exponents are big.
What happens if the messages are short ? D. Boneh, A. Joux and P. Nguyen have shown in [BJN00] that the version with hash function is not secure.

Remark 10. *One can think about other groups for the DL problem for which there is no subexponential algorithm known. It is the case with an elliptic curve and the key size is then less than* 200 *bits. The operations are a bit more difficult and all elliptic curves are not secure (see [FL05]).*

3.3 Rabin encryption

Given a prime number p it is easy to know if a number x is a quadratic residue, i.e. a square modulo p and to find its roots if it is so (see 4.4). It is not the same if one works with $n = pq$ for p, q two big secret prime numbers. Indeed

Proposition 3.3.1. *For every $n = pq$ with p, q prime numbers and for every x which is a square modulo n, there exists a Las Vegas algorithm which computes the square roots of x if the factorization of n is known.*

Proof. One can compute the square roots of x modulo p and q (if necessary with Shanks algorithm) and find the result with the Chinese remainder theorem. □

And conversely:

Proposition 3.3.2. *For every $n = pq$ with p, q prime numbers there is a Las Vegas algorithm which computes p and q if one uses an oracle giving a square root modulo n of any given square.*

Proof. One picks randomly u and let $a \equiv u^2 \pmod{n}$. The oracle gives a square root v of a. There is half a chance that $u \not\equiv \pm v \pmod{n}$ and in this case one knows that $u^2 - v^2 = kn$, i.e. n divides $(u+v)(u-v)$ but any of the factor. Thus $\gcd(n, u+v) = p$ or q and the same for the other one. □

Remark 11. *On the contrary, nobody knows if to know if $x \pmod{n}$ is a square or not is equivalent to the factorization of n. Note that the Jacobi symbol can be computed in polynomial time but if it is equal to 1 one cannot conclude.*

3.3.1 Rabin encryption scheme

Let $n = pq$ for p, q two big prime numbers congruent to 3 modulo 4.

$\mathcal{P} = (\mathbb{Z}/n\mathbb{Z})$; $\mathcal{C} = H \times \{-1, 0, 1\} \times \{0, 1\}$ where H is the subset of quadratic residues; Public key : n; Secret key : p, q; $\mathcal{E}_e(x) = (y, J, b) = (x^2 \pmod{n}, J, b)$ where J is the Jacobi symbol of x and $b = 1$ if $x \geq n/2$ and 0 otherwise; Compute the four roots of y : $x_1, -x_1, x_2, -x_2$. With J determine if $x \in \{-x_1, x_1\}$ or in $\{-x_2, x_2\}$. Once the right pair determined, use b to determine the sign.

The decipher process relies on the following lemma:

Lemma 3.3.1. *If p, q are prime numbers congruent to 3 modulo 4 then y, J, b as above determine a unique root of y modulo n.*

Proof. y has four roots $\pm x_i$ modulo n. If u, v are such that $u^2 \equiv y \pmod{p}$ and $v^2 \equiv y \pmod{q}$ then one can assume that $\pm x_1 \equiv \pm u \pmod{p}$ (resp. $\pm v \pmod{q}$) and $\pm x_2 \equiv \pm u \pmod{p}$ (resp. $\pm x_2 \equiv \mp v \pmod{q}$). Thus

$$\left(\frac{\pm x_1}{n}\right) = \left(\frac{\pm x_1}{p}\right)\left(\frac{\pm x_1}{q}\right) = \left(\frac{x_1}{p}\right)\left(\frac{x_1}{q}\right) = \left(\frac{u}{p}\right)\left(\frac{v}{q}\right)$$

because of the congruences of p and q modulo 4. Also

$$\left(\frac{\pm x_2}{n}\right) = -\left(\frac{u}{p}\right)\left(\frac{v}{q}\right)$$

so J allows us to choose between $\pm x_1$ and $\pm x_2$. Now b allows us to determine the sign. □

3.3.2 Security

As the Rabin problem is as difficult as the factorization problem, the cryptosystem is sure against a chosen plaintexts attack. In the contrary it is not secure against a chosen ciphertext attack as we can use the decryption oracle to factorize n.

3.4 Signature

NIST has defined in FIPS 186-2 the signature standard DSS (Digital Signature Standard). This standard is based on three types of signatures : RSA or ECDSA (Elliptic Curve Digital Signature Algorithm), DSA (Digital Signature Algorithm).
We will recall also the initial ElGammal signature algorithm.

3.4.1 RSA

Any public cryptosystem for which there exists a hash function with value in $\mathcal{C}$ can be transform into a signature scheme (such a function can be difficult to build for instance if $\mathcal{C} = (\mathbb{Z}/n\mathbb{Z})^*$. It is the case for RSA.

Public key : e;
Private key : d;
Signature : $y \to (y, s) = (y, \mathcal{D}_d(h(y)))$;
Verification : check that $\mathcal{E}_e(s) = h(y)$.

Of course one has to follow a protocol (almost identical to RSA) for the signature (called RSASSA-PSS) in order to obtain sufficient security.
It is also obvious that any attack against RSA is an attack against this signature.

Note that the hash function plays an important role. If it is not a one-way function, Oscar can realize the following attack: he computes $h' = r^e \pmod n$ for some random r. He can then compute a pre-image of h', i.e. $h' = h(y')$. He can then pretend that Alice signed y' since (y', r). Of course y' has probably no meaning but in certain protocols that does not play a role. Such a forgery is called an *existential forgery* (**falsification existentielle**).

If it is not collision resistant, Alice can break the repudiation condition: she finds two messages y, y' such that $h(y) = h(y')$ and sign (y, s). Later, she can say that she

signed (y', s).

We can finally mention the following *default attack* (**attaque par dÃ©faut**). The signature is realized by a smartcard. Let assume that to speed up the signature, the device uses the following method :

- Let $d_p \equiv d \pmod{p-1}$, $d_q \equiv d \pmod{q-1}$,
- one has $d_p e \equiv 1 \pmod{p-1}$ and $d_q e \equiv 1 \pmod{q-1}$ so $s \equiv h(y)^{d_p} \pmod{p}$ and $s \equiv h(y)^{d_q} \pmod{q}$,
- one computes $s \pmod{pq}$ with the Chinese remainder theorem.

If it happens that during the computation of s, one compute $s' \equiv h(y)^{d_p} \pmod{p}$ and $s' \not\equiv h(y)^{d_q} \pmod{q}$, one has $\gcd((s')^e - h(y), n) = p$ and one has factorized n. One can force this mistake on the device by applying a short electric impulse at the right time for instance. Of course, modern cards intertwine the computation of the two elements in order to prevent this attack.

3.4.2 ElGammal signature

Let p be a big prime number and h a hash function with values in $[0, p-2]$. We will use the full multiplicative group $G = (\mathbb{Z}/p\mathbb{Z})^*$. Let g be a primitive element.

Public key : (p, g, b) with $b \equiv g^a \pmod{p}$;
Secret key : $a \in [0, p-2]$;
Signature : $y \to (y, r, s)$ with $r \equiv g^k \pmod{p}$, $s \equiv k^{-1}(m - ra) \pmod{p-1}$ with $m = h(y)$ and $k \in [1, p-2]$ different for each signature;
Verification : check that $0 \leq r \leq p-1$. If not reject the signature ; compute $m = h(y)$; compute $v \equiv g^m \pmod{p}$ and $w \equiv b^r r^s \pmod{p}$; Check that $v = w$.

To find the secret key is equivalent to solve a case of the DLP. Also it is not difficult to see that if a message y and r are fixed, finding a valid s if also a case of the DLP (from $v \equiv b^r r^s \pmod{p}$). Last, if one fixes a message y and s one has to solve

$$g^m \equiv b^r r^s \pmod{p}$$

for which no algorithm is known.

It is important to check that $0 \leq r \leq p-1$. Otherwise, Oscar may create signatures of meaningful documents starting from an old signature (this is call *selective forgery* (**falsification sÃ©lective**)). Let (y, r, s) a valid signature and y' a new message. To sign y', Oscar compute

$$u \equiv h(y')h(y)^{-1} \pmod{p-1}.$$

Here we assume $h(y)$ coprime to $p-1$. Oscar also computes

$$s' \equiv su \pmod{p-1}$$

and using the Chinese remainder theorem, he determines r' with

$$r' \equiv ru \pmod{p-1}, \quad r' \equiv r \pmod{p}.$$

The signature (y', r', s') is valid. Indeed

$$b^{r'}(r')^{s'} \equiv b^{ru} r^{su} \equiv g^{u(ar+ks)} \equiv g^{h(y')} \pmod{p}.$$

Moreover $r' \equiv r \pmod{p}$ but $r' \equiv ru \not\equiv r \pmod{p-1}$ (otherwise h is not without collisions) implies $r' \neq r$ so $r' > p$.

It is important to change the value of k. Suppose that y_1 and y_2 are signed by s_1, s_2 with the same k. Then the number $r \equiv g^k \pmod{p}$ is the same. Therefore $s_1 - s_2 \equiv k^{-1}(h(y_1) - h(y_2)) \pmod{p-1}$. From this congruence k can be determined if $h(y_1) - h(y_2)$ is invertible modulo $p-1$. Then the adversary can find the secret key a.

This signature may be attacked if g is a small divisor of $p-1$. More precisely, one assumes now that $p = 4u+1$ and that $p - 1 = gt$. g^t is a primitive element of a cyclic subgroup H of order g and b^t belongs to H. We suppose g small enough such that the DL in H is easy. We can then compute z such that $g^{tz} = b^t$. As $g \in G$ is primitive $g^{(p-1)/2} \equiv -1 \pmod{p}$. Since $gt \equiv -1 \pmod{p}$, $(gt)^{(p-3)/2} \equiv (-1)^{(p-3)/2} \equiv -1 \pmod{p}$. Then

$$g^{-1} g^{(p-1)/2} t^{(p-3)/2} \equiv -1 \pmod{p},$$

so $t^{(p-3)/2} \equiv g \pmod{p}$.
An attacker who wants to imitate the signature of A compute $m = h(y)$, takes $r = t$ and $s \equiv (p-3)(m - tz)/2 \pmod{p-1}$. Under these conditions

$$\begin{aligned} \omega &\equiv b^t (t^{(p-3)/2})^{(m-tz)} \pmod{p} \\ &\equiv b^t g^{m-tz} \pmod{p}, \\ &\equiv b^t g^m b^{-t} \pmod{p}, \end{aligned}$$

so $\omega = v$. This forged signature is then accepted as a signature from A.

Remark 12. *If no hash function is used, there exist two other types of attacks during which the attacker chooses particular couples (r, s) and makes a existential forgery. Let us present the first one. Let i, j be two integers such that $0 \leq i \leq p-2$ and $0 \leq j \leq p-2$. We look for $r \equiv g^i b^j \pmod{p}$. We want to satisfy*

$$g^y \equiv b^r (g^i b^j)^s \pmod{p}$$

which is equivalent to $g^{y-is} \equiv b^{r+js} \pmod{p}$. This is the case for instance if

$$\begin{cases} y - is & \equiv 0 \pmod{p-1} \\ r + js & \equiv 0 \pmod{p-1} \end{cases}.$$

These equations can be solved if $\gcd(j, p-1) = 1$. *We obtain*

$$\begin{cases} r & \equiv g^i b^j \pmod{p} \\ s & \equiv -rj^{-1} \pmod{p-1} \\ y & \equiv -rij^{-1} \pmod{p-1}. \end{cases}$$

The second case is an existential forgery with known signature. Let assume that (r, s) *is a signature for the text* y. *Oscar can then sign other texts. Let* h, i, j *integers with* $0 \leq h, i, j \leq p-2$ *and* $\gcd(hr - js, p-1) = 1$. *We compute*

$$\begin{cases} \lambda & \equiv r^h g^i b^j \pmod{p} \\ \mu & \equiv s\lambda(hr - js)^{-1} \pmod{p-1} \\ y' & \equiv \lambda(hy + is)(hr - js)^{-1} \pmod{p-1}. \end{cases}$$

It is tedious but easy to check that (y', λ, μ) *is valid.*

3.4.3 DSA

DSA is based on the DLP. It is an improvement of the ElGammal scheme. Indeed in the previous one, the size of the appendices is about twice the size of p (i.e. 2048 bits). We want to work with shorter elements.
Let $512 \leq t \leq 1024$ be a multiple of 64. Let $2^{t-1} < p < 2^t$ be a prime number. Let $2^{159} < q < 2^{160}$ be a prime divisor of $p-1$.

Public key : (p, q, α, β) with α an element of order q of $(\mathbb{Z}/p\mathbb{Z})^*$ and $\beta \equiv \alpha^a \pmod{p}$; Secret key : $a \in [0, q-2]$; Signature : $y \to (y, r, s)$ with $r \equiv \alpha^k \pmod{p}$, $s \equiv k^{-1}(m + ar) \pmod{q}$ with $m = h(y)$ and $k \in [1, q-2]$ different for each signature; Verification : check that $(r, s) \in [0, p-1] \times [0, q-1]$. If not reject the signature ; compute $m = h(y)$; compute $\omega \equiv s^{-1} \pmod{q}$, $u_1 \equiv m\omega \pmod{q}$ and $u_2 \equiv r\omega \pmod{q}$; compute $v \equiv \alpha^{u_1}\beta^{u_2} \pmod{p}$ and check that $v = r$.

Unfortunately there is no proof of security for DSA (i.e. against an existential forgery with passive attacker).

3.5 Key exchange

3.5.1 First version

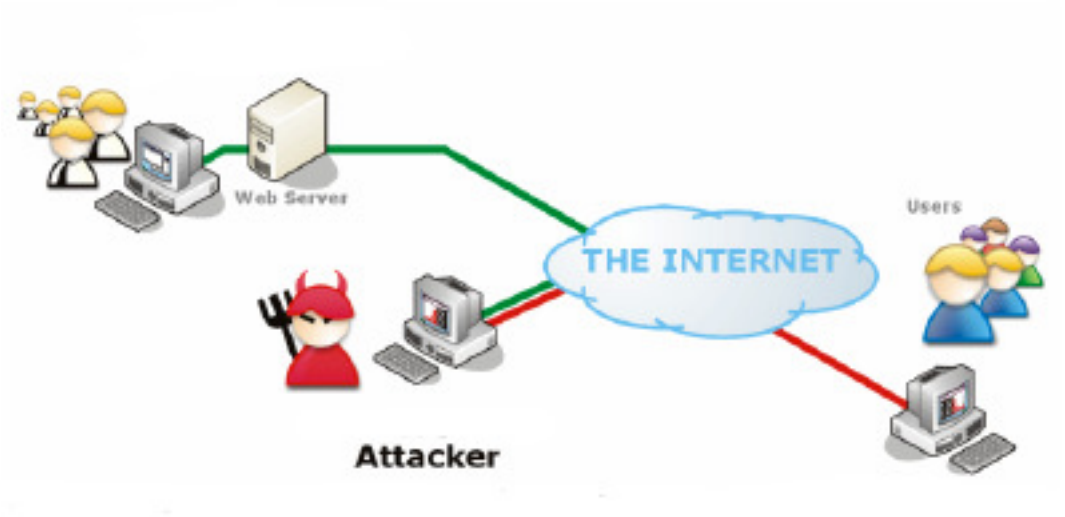
As we said, encryption of long messages is done generally by secret key systems, much faster than public ones. The secret key systems are used to exchange the symmetric key between the users. RSA or ElGammal can be used in this direction. We describe here the elementary Diffie-Hellman key exchange protocol.
A and B want to construct a symmetric 128 bits key k. They take a prime number p of 1024 bits such that $p-1$ has a prime factor of at least 160 bits, a primitive elements

$g \in (\mathbb{Z}/p\mathbb{Z})^*$ and a hash function h which output has 128 bits.

A chooses $0 < n < p-1$ which is kept secret;
B chooses $0 < m < p-1$ which is kept secret;
A sends g^n to B and B sends g^m to A ;
A and B computes $k = h(g^{mn})$.

The difficulty of the Diffie-Hellman problem insures the security of the system. However for a general i the problem of the ith bit is not solved : it is not known which are the bits of g^{mn} which are as difficult to find as the whole g^{mn}. One knows that the first bits are easy to compute. One knows also that to compute more than $\sqrt{\log_2(p)}$ last bits is as difficult as to compute g^{mn}. Using the hash function h allows to walk round this issue.
The *middle-man attack* (**attaque de l'homme du milieu**) (see Fig. 3.1) is an attack which can be made on this protocol by an active attacker : the attacker O intercepts the exchanges between A and B. He constructs then a common key with A and a common key with B. He can then recreate for A, B the impression of a secure communication whereas he can of course decipher all the messages.

Figure 3.1: Middle-man attack



3.5.2 How to avoid this attack ?

Signatures enable to avoid this attack. Indeed, Alice and Bob not only exchange g^n and g^m but send these messages with their signature.

The drawback is that the size of the data exchanged is now quite big (at best $2\cdot(160+160)$ bits).

Remark 13. *There exist variants (for instance the MQV protocol) for which the size is only* 160 *bits.*

We have now moved to the issue of 'key distribution problem of securely distributing secret keys' to 'one of authentically distributing public keys'. This problem is more or less solved by PKI which ensures that a given public key really belongs to its owner. Currently there are two big types of PKI certificates which we will describe briefly.

PGP

The email encryption program Pretty Good Privacy (PGP) takes a bottom-up approach to the distribution of trust. The public key management is done by the users themselves who build a 'Web of Trust'. For instance Alice can sign Bob's public key and then Bob can give the sign 'certificate' to Charlie, in which case Alice is acting as a CA for Bob. If Charlie trusts Alice's judgement with respect to signing people's keys then she will trust that Bob's key really does belong to Bob. It is really up to Charlie to make this decision. As users keep doing this cross-certification of each other's keys, a web of trusted keys grows from the bottom up. Another possibility to enter the web of trust is to attend what is called a keysigning party (see `http://en.wikipedia.org/wiki/Key_signing_party`.
PGP uses a block cipher called IDEA for bulk transmission. Digital signature is produced either with RSA or DSA.
Key revocation is still a problem. The ad-hoc method is that if your key is compromised then you should tell all your friends, who tell their friends and so on.

SSL

The Secure Socket Layer, SSL, was driven by commercial requirements, namely to provide a secure means for web based shopping and sales. It allows various protocols to be transparently layered on top, for example HTTP, FTP, SSH.
After an initial handshake all subsequent traffic is encrypted. The server side of the communication, namely the website, is always authenticated for the benefit of the client. Optionnaly the client may be authenticated o the user. The server is authenticated since it provides the client with an X509 public key certificate. This, for web shopping transactions is signed by some global CA whose public key comes embedded into the user's web browser. During SSH session, the server side is usually a self-signed certificate from the host computer.
The following is a simplified overview of how SSL can operate.

- The client established connection with the server on a special port number so as to signal this will be a secure session.
- The server sends a certified public key to the client.
- The client verifies the certificate and decides whether it trusts this public key.

- The client chooses a random secret.
- The client encodes this with the server's public key and sends this back to the server.
- The client and server now securely share the secret.
- The server now authenticates itself to the client by responding using the shared secret.

X509 is a standard which defines a structure for public key certificates. A CA assigns a unique name to each user and issues a signed cerificate. The CAs are connected in a tree structure, which each CA issuing a digital certificate for the one beneath it. In addition cross-certification between the branches is allowed. The following recors are always in a certificate.

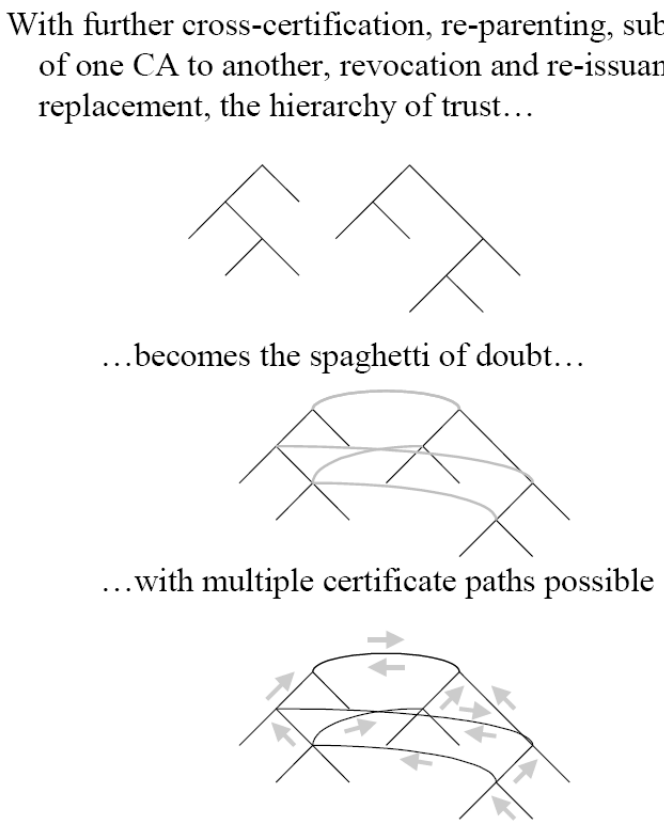
- The version number of the X509.
- The certificate serial number.
- The CA's signing algorithm indetifier.
- The issuer's name.
- The validity period in the form of a not-before and not-after date.
- The subject's name, i.e. whose public key is being signed. This could be an email address or a domain name.
- The subject's public key.
- The issuer's signature on the subject's public key and all data that is to be bound to the subject's public key, such as the subject's name.

A constraint often presents is the path length, i.e. the number of certificates which are allowed to appear in a chain (in X509v3). This option was not taken in account by Internet Explorer 5 and 6. This can lead to an attack 'man in the middle' by extending a chain of certification.

The major issue with PKI is certificate revocation. The default strategy is to publish a list (CRL) which cancels a certification (equivalent to 1970s-era credit card blacklist booklets). Now, where can they be found ? How can we fetch them ? Note that if 10M clients download a 1MB CRL issue once a minute gives a 150GB/s traffic !
Another big issue is cross-certification for several hierarchies : certification paths can contain paths,.... One solution would be to create a super-root above the hierarchies but what happens if they are incompatible (for instance one is CIA, the other Mossad) ?

In conclusion, PKIs run as a protection racket : they require of the user

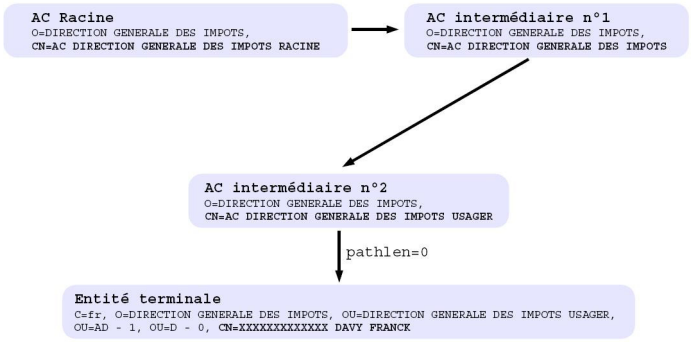
Figure 3.2: Cross-certification



- certificate management software to be installed and configured;
- payment for each certificate;
- significant overhead in managing keys and certificates.

But '... disclaims any warranties... makes no representation that any CA or user to which it has issued a digital ID is in fact the person or organization it claims to be... makes no assurances of the accuracy, authenticity, integrity, or reliability of information'. They then advertise as follows : 'if you fail to renew your Server ID prior to the expiration date, operating your Web site will become far riskier than normal... your Web site visitors will encounter multiple, intimidating warning messages when trying to conduct secure transactions with your site. This will impact customer trust and could result in lost business for your site'.

Figure 3.3: TÃ©lÃ©dÃ©claration des impÃ´ts



Remark 14. *In response to some of the problems associated with X509, another type of certificate format has been proposed called SPKI, or Simple Public Key Infrastructure. Each SPKI certificate has an issuer and a subject both of which are public keys and not names. This is because SPKI's authors claim that it is a key which does something and not a name. SPKI does not assume the global CA hierarchy. It assumes a more ground-up approach like PGP. However it is currently not used much commercially since PKI vendors have a lot of investment in X509.*

3.6 Commitment and authentification

3.6.1 Commitment

Suppose Alice and Bob want to make some decision based on a random coin flip but they are not at the same place. It can be done by a *commitment scheme* (**mise en gage**) as follows : let h be a hash function. Alice takes a random word R_A and choose the result of the coin flip x. She computes then $h_A = h(R_A||x)$ and sends this result to Bob. Bob makes up his mind about the result and send it to Alice. Alice sends R_A, x to Bob. Such a scheme has two properties :

1. Bob cannot determine the value x ; the scheme is said *concealing* (**dissimulant**).
2. Alice cannot change her mind ; the scheme is said *binding* (**liant**).

Here the two properties are only obtained in a computationally way. It is however possible to design schemes where at least one of these property is true whatever the computing power of Alice and Bob is.

Remark 15. *Here is another possibility, without hash functions. Bob sends $n = pq$ to Alice with p, q two big prime numbers congruent to 3 modulo 4. Alice chooses x and compute the coin flip $\left(\frac{x}{n}\right)$. She sends $y \equiv x^2 \pmod{n}$ to Bob. Bob guesses the result ± 1 and sends its answer to Alice. Alice sends then x to Bob and Bob sends (p, q) to Alice.*

3.6.2 Authentication : zero-knowledge proof

In various applications (email, bank exchanges) it may be necessary that the sender can check the identity of the receiver before starting the transaction. This is of course an authentication problem. An authentication scheme must allow the verifier V to check the identity of the prover P in polynomial time, as often as he wants without being able to steal his identity. This cannot be achieved by the standard methods (password, access code) but in 1985 S. Goldwasser, S. Micali and C. Rackoff introduced the notion of zero-knowledge interactive proof. They showed that the minimal amount of information that an user must reveal to prove (with a huge probability) that he knows a secret can be zero. In 1986, A. Fiat and A. Shamir showed how this can be use in the context of authentication by giving a scheme based on the factorization problem (see [QG90] for an educative introduction).
The general principle is the following : each user has a secret s which satisfies a predicate $P(I, s)$ for a case I of a difficult problem. The verifier asks the provers a series of questions. If the last one knows the secret s he can answer all the questions correctly. Otherwise, he can answer only randomly with a probability $0 < q < 1$ of success. Itering the process, one can make the probability of cheating as small as desired.

3.6.3 Fiat-Shamir protocol

Let $n = pq$ with p, q two big prime numbers. We will use the difficulty of the square root problem in $G = \mathbb{Z}/n\mathbb{Z}$. P has a public key $v \equiv a^2 \pmod{n}$ for $1 \leq a \leq n-1$ coprime to n. P and V use the following protocol.

Public key of P : $v \equiv a^2 \pmod{n}$; Private key of P : $a \in [1, n-1]$; P chooses a random $r \in [1, n[$ and sends $x \equiv r^2 \pmod{n}$ to V ; V sends a random bit b to P ; if $b = 0$, P sends r to V and V checks that $x \equiv r^2 \pmod{n}$; if $b = 1$ then P sends $y \equiv ra \pmod{n}$ to V and V checks that $x \equiv y^2 v^{-1} \pmod{n}$.

If the user P does not know the value a, he cannot satisfies both cases with probability 1 (if he could, he would know a). At each trial, he has $1/2$ chance to be unmasked. By itering again one sees that a proof made by a fair prover is always accepted, the system is said *consistent* (**consistant**). A proof made by an unfair prover has a tiny chance to be accepted, the system is said *meaningful* (**significatif**). It can be shown that this proof is zero-knowledge : r being random, ra and x are also and no information on a is divulged.

3.6.4 Schnorr identification

This system uses a multiplicative group $G = (\mathbb{Z}/p\mathbb{Z})^*$ where p is a 1024 bits prime number and H a subgroup of order q at least 160 bits. Let α be a generator of H.

Public key of P : $v \equiv \alpha^{-s} \pmod{n}$; Private key of P : $s \in \mathbb{Z}/p\mathbb{Z}$; P chooses a random $r \in \mathbb{Z}/p\mathbb{Z}$ and sends $x \equiv \alpha^r \pmod{p}$ to V ; V sends a challenge $e \in [0, 2^k[$ $(k = \lvert p\rvert_2)$ to P ; P sends $y \equiv r + es \pmod{q}$ to V ; V checks that $x \equiv \alpha^y v^e \pmod{p}$.

If the prover can answer two different cases e, e' then he knows s (and he is able to compute the DL of v). Indeed

$$\begin{aligned} x &\equiv \alpha^y v^e \equiv \alpha^{y'} v^{e'} \pmod{p}, \\ v^{e'-e} &\equiv \alpha^{y-y'} \pmod{p}, \\ \alpha^{s(e-e')} &\equiv \alpha^{y-y'} \pmod{p}, \\ s &\equiv \frac{y-y'}{e-e'} \pmod{q}. \end{aligned}$$

So the prover cannot answer more than one of the 2^k different questions. He can cheat successfully with a probability less than $1/2^k$. If $k = 20$ the system is meaningful. Moreover it can be shown that it is zero-knowledge.

3.7 Blue card

We study here an application of the protocols we described. Ship-cards were invented by Moreno in 1974 (although, as usual, there were pre-ideas, see wikipedia article on the subject).

First we give an old version which was broken and then we discuss the improvements to avoid the attacks (see `http://interstices.info/jcms/c_33835/le-protocole-cryptographi`

3.7.1 The protocol

From outside, when Alice (A) wants to pay with her ship-card (C) on a terminal (T). Alice's bank (B) will be part of the protocol also it is a simplification, since in reality it is a central server.

1. A introduces C in T;
2. The seller enters the amount of the payment on T
3. T identifies C and asks for the code;
4. A enters the code (say 3456)
5. T sends the code to C;
6. if the code is correct, C acknowledges it to T;
   Until here, everything is local. If the amount is greater than 100 euros and in 20% of the cases approximately, a checking is required.

7. T asks B for the amount and the card C;
8. B agrees and the amount is paid.

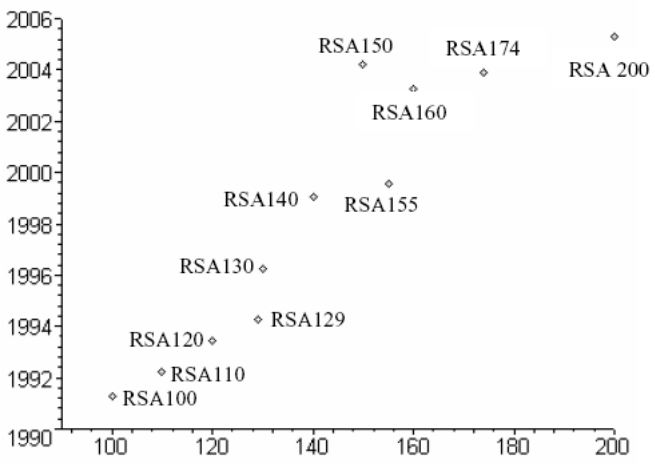
From inside, this can be translated in the following steps (we simplified, not taking into account hash functions)

1. $T \mapsto C$: identification;
2. $C \mapsto T$: $Data, s_B(Data)$ where $s_B(Data)$ is a signature pre-computed as an RSA-signature on $Data$ by the bank with its private key d_B (d_B is not on C). $Data$ contains a name, the card number and a validity period. T knows the bank public key e_B.
3. $T \mapsto A$: Code ?
4. $A \mapsto T$: 3456;
5. $T \mapsto C$: 3456; C knows the code;
6. $C \mapsto T$: ok.

3.7.2 Attack

The attack we describe was used by Serge Humpich in 1998. This attack was known, presented by Louis Guillou in 1988 but was underestimated by the banks. Moreover, it can be used only when the bank is not involved in the protocol, which excludes money withdraw.
Initially, the security was heavily based on the protocol secret and on the difficulty to recreate a card. The secret was broken by a careful analysis of a payment terminal and the second difficulty was surmounted by the release of easy tools which could be used to programm a card.
Two weaknesses were exploited by Humpich to create Yescard

1. a logic weakness : it is possible to copy the data and the signature from a valid card to a new one and to force this new card to answer ok whatever code is entered.
2. primitive weakness : already in 1988, a 320 bits key wasn't secured anymore. It was then easy for Humpich to break the system and find the secret key d to create completely fake data.



3.7.3 Improvements

The first thing was to take a longer key in 1999 : today it is a 230 digits number

$$
\begin{aligned}
&155088080278376929842392150075130787847102\\
&021520671110279311199011387539455345999975\\
&760530467173585609159755538979740893817334\\
&404367470478098639006990667909672893308140\\
&504493596951450867623994249344075058927001\\
&5739962374529363251827.
\end{aligned}
$$

Until end of may 2007, it was still possible to copy the information from a valid card and to create a yes card (one swindle for more than 640 000 euros was revealed the 9th of February 2007). But the consortium EMVCo (Europay, Mastercard, Visa) has published on the web the information on three new protocols, one of them (DDA, Dynamic Data Authentication) preventing this attack. We describe it here, without technical details (see `http://www.emvco.com/specifications.asp`).

1. $T \mapsto C$: authentication ;
2. $C \mapsto T$: $s_{CA}(e_B), s_B(e_C), Data, s_B(Data)$ where $s_{CA}(e_B)$ is a signature of the public key of the bank given by a certification authority (CA), and $s_B(e_C)$ is a signature on the public key of the card. Using only e_{CA}, the terminal can compute e_B and then e_C;
3. $T \mapsto C$: a challenge N which is a random number;
4. $C \mapsto T$: $r = \mathcal{E}_{d_C}(N)$. Note that d_C is kept secret in a private zone of the card. The terminal can then check if $N = \mathcal{D}_{e_C}(r)$;

5. $T \mapsto A$: code ?
6. $A \mapsto T$: 3456;
7. $T \mapsto C$: $\mathcal{E}_{e_C}(3456)$. The code is ciphered, so it cannot be easily intercepted.
8. $C \mapsto T$: ok.

One of the problems is that the card has to be able to run a RSA encryption which is quite heavy for such a small device. The use of elliptic curve cryptography is then of importance here.

Remark 16. *Other attacks are still possible on the hardware (see Sec. 3.1.3). One can mention also 'relay attack' (the terminal is fake and information are transferred to another terminal which withdraws more money). By measuring round-trip latency, the distance between the legitimate card and terminal can be securely established, defeating relay attacks*

Chapter 4

Reviews and complements

4.1 Integers, congruences, $\mathbb{Z}/n\mathbb{Z}$

4.1.1 Divisibility

Definition 4.1.1. *We say that a* divides *n if there is an integer b with $n = ab$.*

Proposition 4.1.1. *1. if $a|b$ and $b|c$ then $a|c$.*

2. if $a|b$ then $ac|bc$ for all c.

3. if $c|a$ and $c|b$ then $c|da + eb$ for all d and e.

4. if $a|b$ and $b|a$ then $|a| = |b|$.

Proof. Let us show the last one. $a = 0$ if and only if $b = 0$, so the proposition is proved in this case. If $a \neq 0$, there exists $a' \neq 0$ such that $aa' = b$ so $|aa'| = |a||a'| = |b|$. Since $|a'| \geq 1$, $|a| \leq |b|$. By symmetry, $|b| \leq |a|$ and we have the equality. □

The following result is very important. It shows that division with remainder of integers is possible ($\mathbb{Z}$ is euclidian).

Proposition 4.1.2. *If a and b are integers, $b > 0$, then there are uniquely determined integers q and r such that $a = qb + r$ and $0 \leq r < b$.*

Proof. One can check that $q = \lfloor \frac{a}{b} \rfloor$ and $r = a - bq$ satisfy the hypothesis. □

We define the greatest common divisor of two integers.

Definition 4.1.2. *Among all the common divisors of two integers a, b which are not both zero, there is exactly one greatest divisor (with respect to $\leq$). It is called the* greatest common divisor *(**plus grand multiple commun**) of a and b and denoted* $\gcd(a, b)$ *(or sometimes (a, b)).*

For completeness we set $\gcd(0, 0) = 0$.

Proposition 4.1.3. *The set of all integer linear combinations of a and b is the set of all integer multiples of* $\gcd(a,b)$*, i.e.*

$$a\mathbb{Z} + b\mathbb{Z} = \gcd(a,b)\mathbb{Z}.$$

Proof. We can assume a or b different from 0. Set $I = a\mathbb{Z} + b\mathbb{Z}$. Let g be the smallest positive integer in I. We claim that $I = g\mathbb{Z}$. To see this, choose a non-zero element $c \in I$. By Prop. 4.1.2 there are q, r such that $c = qg + r$ and $0 \leq r < g$. Therefore $r = c - qg$ belongs to I. But since g is the smallest positive integer in I, we must have $r = 0$ and $c = qg$.
It remains to show that $g = \gcd(a,b)$. Since $a, b \in I$ it follows from $I = g\mathbb{Z}$ that g is a common divisor of a, b. Moreover since $g \in I$ there are x, y with $g = ax + by$. Therefore if d is a common divisor of a and b then d divides g. Then $|d| \leq |g|$ and g is the greatest common divisor of a, b. □

Corollary 4.1.1 (Bézout)**.** *For all a, b, n the equation $ax+by = n$ is solvable in integers x, y if and only if* $\gcd(a,b)|n$.

Proof. If there are x, y with $n = ax + by$ then $n \in a\mathbb{Z} + b\mathbb{Z}$ and by Prop. 4.1.3 we have $n \in \gcd(a,b)\mathbb{Z}$.
Conversely, if n is a multiple of $\gcd(a,b)$ then $n \in \gcd(a,b)\mathbb{Z} = a\mathbb{Z} + b\mathbb{Z}$. □

If the $\gcd(a,b) = 1$ one says that a, b are *coprime* (**premiers entre eux**).
One will introduce also the *least common multiple* (**plus petit commun multiple**) $\operatorname{lcm}(a,b) = ab/\gcd(a,b)$. One can check that this number is the smallest common multiple of a and b and that any multiple of a and b is a multiple of $\operatorname{lcm}(a,b)$.

Remark 17. *Euclid algorithm, which gives a fast method to compute the gcd, will be presented in 4.3.3.*

4.1.2 Prime and factorization

Definition 4.1.3. *An integer $p > 1$ is called a* prime number *(**nombre premier**) if it has only* 1 *and p as divisors.*

Example 10. *There are infinitely many prime numbers. The biggest generic one is*

$$(((((((((2^3+3)^3+30)^3+6)^3+80)^3+12)^3+450)^3+894)^3+3636)^3+70756)^3+97220$$

Interested readers may read `http://www.cs.uwaterloo.ca/journals/JIS/VOL8/Caldwell/caldwell78.html` *for the origin of this number. It has 20,562 decimal digits and the proof was built using fastECPP on several networks of workstations.*

We will write $\mathbb{P}$ the set of prime numbers. To estimate the efficiency of some algorithms, we need results on density of primes.

Theorem 4.1.1. *Let $\pi(x) = \#\{p \leq x \text{ prime}\}$. One has*

$$\pi(x) \sim \frac{x}{\log x}.$$

Let $n \geq 2$ be an integer and c an integer prime to n. Let $\pi_{n,c}(x) = \#\{p \leq x \text{ prime}, p = kn + c\}$. One has

$$\pi_{n,c}(x) \sim \frac{1}{\phi(n)} \frac{x}{\log x}.$$

Remark 18. *For $x \geq 17$, one has $\pi(x) > x/\log x$ and for $x > 1$ one has $\pi(x) < 1.25506(x/\log x)$.*

- To find a prime number, the number of attempts is then of the size of x. Indeed, the probability to fail in k attempts is $(1-1/\log(x))^k$ so the probability to succeed

$$1 - (1 - 1/\log(x))^k \sim 1 - e^{-k/\log(x)}$$

  which is closed to 1 for any $k = \log(x)^{1+\epsilon}$.

- To construct a couple (p, q) of prime numbers such that q divides $p - 1$, one can apply the previous lemma in the case $n = q$ and $c = 1$. There are less than $p/2q$ values of k for which $p = kq + 1$ may be prime (since we exclude k odd). In the series $kq + 1$ there are $\pi_{q,1}(p)$ values which give a prime number p, so the proportion of good k among the $p/2q$ possible is

$$\frac{\pi_{q,1}(p)}{p/2q} = \frac{2q}{q-1} \frac{1}{\log p}.$$

  Here again one finds a number of attempts of the size of p.

- To construct a couple (p, q) of prime numbers such that $p = 2q + 1$. It is not known if there exists infinitely many (Sophie Germain) primes p. However it is conjectured that the number of Sophie Germain primes less than n is equivalent to

$$\frac{2Cn}{(\log(n))^2},$$

  where $C \approx .66$.

Let us finish with the fundamental result.

Theorem 4.1.2. *Every integer $a > 1$ can be written as the product of prime numbers*

$$a = \prod_{p \in \mathbb{P}} p^{e(p)}$$

with $e(p) \geq 0$ and $e(p) = 0$ except for finitely many primes p. Up to permutation, the factors in this product are uniquely determined.

4.1.3 Representations of integers

In books, integers are written in decimal expansion. On computers, binary expansion is used. More generally, integers can be represented using the so-called g-adic expansion.

Proposition 4.1.4. *Let $g > 1$ be an integer. For each positive integer a, there is a uniquely determined positive integer k and a uniquely determined sequence*

$$(a_1, \ldots, a_k) \in [0, \ldots, g-1]^k$$

with $a_1 \neq 0$ and

$$a = \sum_{i=1}^{k} a_i g^{k-i}. \tag{4.1}$$

In addition $k = \lfloor \log_g a \rfloor + 1$.

Proof. Let a be a positive integer. If a can be represented as in (4.1) then $g^{k-1} \leq a \leq g^k - 1 < g^k$. Hence k is unique. We prove the existence and uniqueness of the sequence $(a_1, \ldots, a_k)$ by induction on k.
For $k = 1$, set $a_1 = a$.
For $k > 1$. We first prove the uniqueness. If there is a representation as in (4.1), then $0 \leq a - a_1 g^{k-1} < g^{k-1}$ and therefore $0 \leq a/g^{k-1} - a_1 < 1$. Therefore a_1 is the integral quotient of a divided by g^{k-1} and is hence unique. Set $a' = a - a_1 g^{k-1}$. If $a' = 0$ then $a_i = 0$, $2 \leq i \leq n$ or $a' = \sum_{i=2}^{k} a_i g^{k-i}$ is the uniquely determined representation of a' by the induction hypothesis. □

Definition 4.1.4. *The sequence $(a_1, \ldots, a_k)$is called the* g-adic expansion *of a (**représentation en base** g). Its length is $|a|_g = \lfloor \log_g a \rfloor + 1$.*

Example 11. *The sequence* 10110 *is the binary expansion of* $2^4 + 2^2 + 2 = 22$.
Let us compute the binary expansion of 105 *:* $2^6 < 105 < 2^7$ *so it is of length* 7. *We have* $a_1 = \lfloor 105/64 \rfloor = 1$. *Now* $105 - 64 = 41$ *so* $a_2 = \lfloor 41/32 \rfloor = 1$. $41 - 32 = 9$, $a_3 = \lfloor 9/16 \rfloor = 0$, $a_4 = \lfloor 9/8 \rfloor = 1$ *;* $9 - 8 = 1$ *so* $a_5 = a_6 = 0$ *and* $a_7 = 1$. *Hence the binary expansion of* 105 *is* 1101001.
Another way to compute this representation is to use the division algorithm, which we illustrate here :

$$\begin{aligned} 105 &= 2 \cdot 52 + 1 \\ 52 &= 2 \cdot 26 + 0 \\ 26 &= 2 \cdot 13 + 0 \\ 13 &= 2 \cdot 6 + 1 \\ 6 &= 2 \cdot 3 + 0 \\ 3 &= 2 \cdot 1 + 1 \\ 1 &= 2 \cdot 0 + 1 \end{aligned}$$

The result can now be read from bottom to top.
Another important representation in computer science is the hexadecimal *representation, with* $g = 16$. *When writing the hexadecimal expansion, we use instead of the digits* $10, 11, \ldots, 15$ *the letters* A, B, C, D, E, F *so* $A1C$ *is the hexadecimal expansion of* $10 \cdot 16^2 + 16 + 12 = 2588$.

4.1.4 Congruences and $\mathbb{Z}/n\mathbb{Z}$

Definition 4.1.5. *We say that* a *is* congruent *to* b *modulo* n *and we write* $a \equiv b \pmod{n}$ *if* $n|(b-a)$.

This relation is a equivalence relation between the integers. We can consider the class of $\mathbb{Z}$ (as a set) modulo this relation (for a fixed n). It is easy to see that if a is a element, the class of a is $a + n\mathbb{Z}$ so representative elements are $0, 1, \ldots, n-1$.
On the other hand we can consider the commutative ring $(\mathbb{Z}, +, \times)$. The set $n\mathbb{Z}$ is an ideal in $\mathbb{Z}$, we can consider the quotient $\mathbb{Z}/n\mathbb{Z}$ which is also a ring denoted $\mathbb{Z}/n\mathbb{Z}$. It is not difficult to see that the set we constructed before is in natural bijection with the ring we obtained here (by definition !), thus we can consider that the elements of $\mathbb{Z}/n\mathbb{Z}$ are classes of the representative elements $0, 1, \ldots, n-1$. The addition and multiplication in $\mathbb{Z}/n\mathbb{Z}$ are done using the usual addition and multiplication on the representative elements and reducing the result modulo n.

Example 12. *Let* $n = 6$. *If* $a, b \in \mathbb{Z}/n\mathbb{Z}$ *with* a *the class of* 3 *and* b *the class of* 5, *one can write* $a + b = class(3 + 5) = class(8) = class(2)$ *and* $ab = class(15) = class(3)$. *The sign* $=$ *is legitim as we are working in the ring. However we can perform directly the same operations on the integers and use the congruence notations :* $3 + 5 = 8 \equiv 2 \pmod{6}$ *and* $3 \cdot 5 = 15 \equiv 3 \pmod{6}$. *We will often work with the elements and use the congruence relation. Moreover by abuse of notations we will confuse the element and its class modulo* n *if the context is clear.*

Example 13. *What are the last two digits of* 3^{2005} *(Use a fast exponentiation algorithm) ?*
In base 2, $2005 = 11111010101$ *so the exponentiation can be realize according to the following scheme:*

$$\left|\begin{array}{ccccccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 81 & 41 & 21 & 61 & 81 & 41 & 21 & 61 & 81 & 9 & 3 \\ 43 & 3 & 83 & 23 & 43 & 3 & 3 & 43 & 43 & 3 & 3 \end{array}\right|$$

Prove that if a number is written in base 10 *as abcabc then it is divisible by* 13.

We will now study the ring $R = \mathbb{Z}/n\mathbb{Z}$. The neutral element for addition is (the class) of 0 and the neutral element for multiplication is (the class) of 1. As in any commutative ring, we can study the *invertible elements* (**éléments inversibles**) in R : $a \in R$ is invertible if and only if there is $b \in R$ such that $ab = 1$.

Proposition 4.1.5. *An element* $a \in R$ *is invertible if and only if* a *is coprime to* n.

Proof. $a \in R$ is invertible if and only if there exists b such that $ab = 1$ i.e. $ab \equiv 1 \pmod{n}$. So there exists $k \in \mathbb{Z}$ such that $ab - kn = 1$. By Cor. 4.1.1, this is equivalent to $\gcd(a, n) = 1$. □

Example 14. *A matrix $M \in M_m(R)$ is invertible if and only if* $\det(M)$ *is invertible in R.*

Corollary 4.1.2. *The ring R is a field (i.e. all the element different from* 0 *are invertible) if and only if n is prime.*

We denote this field $\mathbb{F}_p$. It is not difficult to see that the invertible elements form a group with respect to the multiplication. This group is denoted $R^* = (\mathbb{Z}/n\mathbb{Z})^*$ and its order $\phi(n)$, the *Euler function* . We will study this group in more details later.

Proposition 4.1.6. $(\mathbb{Z}/n\mathbb{Z}, +)$ *is a* cyclic group *(**groupe cyclique**), i.e. it is generated by a unique element. Any finite cyclic group is isomorphic to $(\mathbb{Z}/n\mathbb{Z}, +)$. The number of generators of $(\mathbb{Z}/n\mathbb{Z}, +)$ is $\phi(n)$.*

Proof. Let $G =< a >$ be a finite cyclic group of order n generated by a. Any element b of G can be written ra with a unique $0 \leq r < n$. We define $\psi : G \to \mathbb{Z}/n\mathbb{Z}$, $b \mapsto r$. It is easy to see that ψ is an isomorphism of group.
The element 1 is a generator of $G = \mathbb{Z}/n\mathbb{Z}$. So any invertible element is a generator of G. Conversely, if a is a generator of G there exists $n \in \mathbb{Z}$ such that $na = 1$, i.e. a is invertible in G. So the number of generators is $\phi(n)$. □

Remark 19. *The generators of $(\mathbb{Z}/n\mathbb{Z}, +)$ are the invertible elements. So the invertible elements are the elements of order n.*
Any cyclic ring is isomorphic (as a ring) to $(\mathbb{Z}/n\mathbb{Z}, +, \times)$.

We will study the decomposition of cyclic groups.

Lemma 4.1.1. *If m, n are coprime the condition*

$$\begin{cases} a \equiv b \pmod{n} \\ a \equiv b \pmod{m} \end{cases}$$

is equivalent to

$$a \equiv b \pmod{mn}.$$

Proof. If $a \equiv b \pmod{mn}$ then the two conditions hold. Conversely, there exist k_1, k_2 such that $a - b = k_1 m = k_2 n$. As m, n are coprime $m | k_2$ and so there exists k_3 such that $a - b = k_3 mn$, i.e. $a \equiv b \pmod{mn}$. □

Theorem 4.1.3 (Chinese Remainder Theorem)**.** *Let m, n coprime. The solutions*

$$\begin{cases} x \equiv a \pmod{m} \\ x \equiv b \pmod{n} \end{cases}$$

are of the following form : let u, v such that $um + vn = 1$ then a solution is $x_0 = bum + avn$ and all the other solutions are of the form $x_0 + kmn$ for $k \in \mathbb{Z}$.

Proof. One can check that x_0 is a solution. Now if x_1 is another solution then $x_1 \equiv x_0 \pmod{m}$ and $x_1 \equiv x_0 \pmod{n}$ so the previous lemma allows to conclude. □

This can be reformulated in the following proposition.

Proposition 4.1.7. *If m, n are coprime, the ring $\mathbb{Z}/mn\mathbb{Z}$ is isomorphic to $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$.*

Proof. By applying Th. 4.1.3, one can check that the map $T : x \pmod{mn} \mapsto (x \pmod{m}, x \pmod{n})$ is an isomorphism of rings. □

Example 15. *Solve the equation $2x \equiv 3 \pmod{45}$ using the Chinese Remainder Theorem.*
We can prove the following result : if $n = \prod_{i=1}^{k} p_i^{e(p_i)}$ is an odd integer, then the number of solutions of $x^2 \equiv 1 \pmod{n}$ is 2^k. By the Chinese Remainder Theorem, it is enough to prove that $x^2 \equiv 1 \pmod{p^e}$ for any p odd prime and $e \geq 1$ if and only if $x \equiv \pm 1 \pmod{p^e}$. It is true if $e = 1$ since $Zp\mathbb{Z}$ is a field. Let assume that it is true for e and let us prove it for $e + 1$. If $x^2 \equiv 1 \pmod{p^{e+1}}$ then it is true also modulo p^e and by hypothesis one has $x \equiv \pm 1 \pmod{p^e}$, i.e $x \equiv \pm 1 + kp^e \pmod{p^{e+1}}$. Now $(\pm 1 + kp^e)^2 \equiv 1 \pm 2kp^e \pmod{p^{e+1}}$ so $p|k$ and $x \equiv \pm 1 \pmod{p^{e+1}}$.

We will see now some properties of the Euler function.

Lemma 4.1.2. *1. $\phi(p) = p - 1$;*

2. $\phi(p^\alpha) = (p-1)p^{\alpha-1}$.

3. if m and n are coprime then $\phi(mn) = \phi(m)\phi(n)$.

4. for all $n > 1$,

$$\phi(n) = n \prod (1 - \frac{1}{p_i})$$

where p_i are the prime factors of n.

Proof. By Cor. 4.1.2, the number of invertible elements in $\mathbb{Z}/p\mathbb{Z}$ is $p-1$, so $\phi(p) = p-1$. In the same way, the elements of $\mathbb{Z}/p^\alpha\mathbb{Z}$ which are not coprime to p are the multiple of p, thus p elements. So the number of invertible elements is $(p-1)p^{\alpha-1}$.
By Prop. 4.1.7, one has $(\mathbb{Z}/mn\mathbb{Z})^* \simeq (\mathbb{Z}/m\mathbb{Z})^* \times (\mathbb{Z}/n\mathbb{Z})^*$ which gives the result. □

Remark 20. *[HW79] One has*

$$\underline{\lim} \frac{\phi(n) \log\log n}{n} = e^{-\gamma}$$

where γ is the Euler constant.
More elementary $\phi(n) \geq \sqrt{n}$ if $n > 6$.

Corollary 4.1.3 (Fermat's little theorem)**.** *For all $a \in (\mathbb{Z}/n\mathbb{Z})^*$ one has*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

In particular if $n = p$ prime, one has

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof. By definition, the order of the group $(\mathbb{Z}/n\mathbb{Z})^*$ is $\phi(n)$, which gives in the group $a^{\phi(n)} = 1$. □

Remark 21. *Another form of Fermat's little theorem is the following : for any a one has $a^p \equiv a \pmod{p}$.*
Here is another proof of Fermat's little theorem : prove first that $\binom{p}{k} \equiv 0 \pmod{p}$ for all $1 \leq k \leq p-1$. Develop $(a+1)^p \pmod{p}$. Then by induction on a.

Let us show a last property of the function ϕ.

Proposition 4.1.8. *For all $n \geq 1$, one can write*

$$n = \sum_{d|n} \phi(d).$$

Proof. Let us denote

$$D = \{\frac{1}{n}, \frac{2}{n}, \ldots, \frac{n}{n}\}.$$

For all $d|n$ define

$$D_d = \{\frac{k}{d},\ \gcd(k,d) = 1, \quad 1 \leq k \leq d\}.$$

By the unicity of the irreducible form of a fraction, one sees that the D_ds form a partition of D. Now the proposition follows, noting that $\#D_d = \phi(d)$. □

The structure of $(\mathbb{Z}/p\mathbb{Z})^*$ is particulary simple.

Proposition 4.1.9. *The group $G = ((\mathbb{Z}/p\mathbb{Z})^*, \times)$ is cyclic. The generators of this group are called* primitive *elements (**éléments primitifs**).*

Proof. Let d be a divisor of $p-1$. If there exists an element x of order d, it generates a subgroup H of order d of G. For all $y \in H$, $y^d - 1 = 0$. As the polynomial $Y^d - 1$ has at most d roots in $\mathbb{Z}/p\mathbb{Z}$, all the elements of order divising d are in H. Thus the number $N(d)$ of elements of order d is 0 or $\phi(d)$, in both cases $N(d) \leq \phi(d)$. Using Prop. 4.1.8, one sees that $N(d) = \phi(d)$ for all d, in particular $N(p-1) = \phi(p-1) > 0$ and G has an element of order $p-1$, i.e. a primitive element. □

Remark 22. *The proof shows also that for any divisor d of $p-1$ there is an element in G of order d.*

Proposition 4.1.10. *Let p be a odd prime and $e \geq 2$. Then*

$$(\mathbb{Z}/p^e\mathbb{Z})^* \simeq \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/p^{e-1}\mathbb{Z}.$$

Proof. First we prove by induction that if $k \in \mathbb{N}^*$ then $(1+p)^{p^k} = 1+\lambda p^{k+1}$ with $\lambda \in \mathbb{N}^*$ prime to p.
From this result we deduce that $1+p$ is an element of order p^{e-1} in $(\mathbb{Z}/p^e\mathbb{Z})^*$.
Now, we consider the surjective morphism $\psi : (\mathbb{Z}/p^e\mathbb{Z})^* \to (\mathbb{Z}/p\mathbb{Z})^*$. Let $x \in (\mathbb{Z}/p^e\mathbb{Z})^*$ an element such that $\psi(x)$ generated $(\mathbb{Z}/p\mathbb{Z})^*$. The order of x is a multiple of $(p-1)$ so there is an element y of order $p-1$. As the orders of y and $1+p$ are coprime, the element $y(1+p)$ is of order $p^{e-1}(p-1)$. So the group $(\mathbb{Z}/p^e\mathbb{Z})^*$ is cyclic and by the Chinese Remainder Theorem, we can write it as the proposition claims. □

Example 16. *Find the structure of $(\mathbb{Z}/16\mathbb{Z})^*$ and the primitive elements of $(\mathbb{Z}/13\mathbb{Z})^*$.*

Remark 23. *For $p = 2$ and $e \geq 3$ the groups $(\mathbb{Z}/p^e\mathbb{Z})^*$ are not cyclic. More precisely $(\mathbb{Z}/p^e\mathbb{Z})^* \simeq \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2^{e-2}\mathbb{Z}$.*

4.2 Probability

4.2.1 Introduction

Let S be a finite nonempty set. We call it the *sample space* (**espace des évènements**). Its elements are called *elementary events* (**évènement élémentaires**).

Example 17. *If we flip a coin, we either obtain heads H or tails T. The sample space is $S = \{H, T\}$.*

An *event* (**évènement**) is a subset of S. The certain event is S. The null event is $\emptyset$. Two events U, V are *mutually exclusive* (**incompatibles**) if $U \cap V = \emptyset$.

Definition 4.2.1. *A probability distribution on S is a map Pr that sends an event to a real number such that*

1. *$0 \leq Pr(A) \leq 1$ for all event A;*
2. *$Pr(S) = 1$;*
3. *$Pr(A \cup B) = Pr(A) + Pr(B)$ for two mutually exclusive events.*

$Pr(A)$ is called the probability *of A.*

One can show that $Pr(\emptyset) = 0$ and if $A \subset B$ then $Pr(A) \leq Pr(B)$.

Example 18. *The probability distribution on the set $\{1, 2, 3, 4, 5, 6\}$ which models a die maps each elementary event to $1/6$.*

The probability distribution that maps each elementary event $a \in S$ to the probability $1/|S|$ is called the *uniform distribution* (**distribution uniforme**).

4.2.2 Conditional probability

Let (S, Pr) be a finite set with a probability distribution.

Definition 4.2.2. *Let A, B be events with $Pr(B) > 0$. The* conditional probability *(**probabilité conditionnelle**) of A given B is defined to be*

$$Pr(A|B) = \frac{Pr(A \cap B)}{Pr(B)}.$$

Two events are called *independent* (**indépendants**) if

$$Pr(A \cap B) = Pr(A) \cdot Pr(B).$$

This is equivalent to $Pr(A|B) = Pr(A)$.

Proposition 4.2.1 (Bayes theorem)**.** *If A, B are two events such that $Pr(A) > 0$ and $Pr(B) > 0$ then*

$$Pr(B)Pr(A|B) = Pr(A)Pr(B|A).$$

Example 19. *We throw two dice. Determine the probability that they both show different numbers under the condition that the sum of both numbers is even (answer : $2/3$).*

Remark 24. *Conditional probabilities are subtle and may lead to paradoxes. Let quote the following introduce by Leslie and Carter. Let A be the event : apocalypse will happen before 2050 and B the event the apocalypse will happen after 2050. Let Z the event, I was there in 2005. In case A one can say that $1/10$ humans would have lived in 2005. In the case B this number will be smaller than $1/1000$. What is the probability of A knowing that I'm alive in 2005 ? One has $Pr(A) = .01, Pr(B) = .99$, $Pr(Z|A) = 0.1, Pr(Z|B) = .001$ so*

$$Pr(Z) = \frac{Pr(Z|A)Pr(A)}{Pr(A|Z)} = \frac{Pr(Z|B)Pr(B)}{1 - Pr(Z|A)}.$$

So we get $Pr(A|Z) > 1/2$! For a a possible explaination of this paradox see [Del03].
Another interesting problem is the problem of the 3 boxes : in a TV game there are 3 boxes, one of them containing a treasure. You can pick a box. The presenter, who knows where is the treasure, opens one of the two remaining box he knows empty. He asks you then if you want to reconsider your choice. Do you ?
A last example is the following : in a casino, you're asked to play the following game : you can choose a sequence of three colors (red, red, black for instance). Then your opponent chooses one also. The roulette is launched. The first sequence to appear wins. Is it a fair game ? It can be shown that whatever the sequence you choose, the opponent can make a better choice in proportions from 2 to 1 till 7 to 1 ! Here is the list and the probability of the opponent to win.

Your choice	*Opponent's choice*	*probability*
RRR	*NRR*	*7/8*
RRN	*NRR*	*3/4*
RNR	*RRN*	*2/3*
RNN	*RRN*	*2/3*
NRR	*NNR*	*2/3*
NRN	*NNR*	*2/3*
NNR	*RNN*	*3/4*
NNN	*RNN*	*7/8*

4.2.3 Perfect secrecy

We will study Shannon security for symmetric cryptosystems. We assume that the sets $\mathcal{P}$ and $\mathcal{K}$ are given with probability distributions $Pr_{\mathcal{P}}$ and $Pr_{\mathcal{K}}$. One denotes also Pr the product probability on $\mathcal{P} \times \mathcal{K}$. To simplify notations, we write $Pr(x)$ for $x \in \mathcal{P}$ meaning $Pr(\{x\} \times \mathcal{K}) = Pr_{\mathcal{P}}(\{x\})$ and the same for $k \in K$. Last, if $y \in \mathcal{C}$, one denotes $Pr(y)$ the probability of the event $\{(x,k),\ \mathcal{E}_k(x) = y\}$.
Note that

$$\{(x,k),\ \mathcal{E}_k(x) = y\} = \bigcup_{\substack{k \in \mathcal{K} \\ y \in \mathcal{E}_k(\mathcal{P})}} \{(\mathcal{D}_k(y), k)\}.$$

So

$$Pr(y) = \sum_{\substack{k \in \mathcal{K} \\ y \in \mathcal{E}_k(\mathcal{P})}} Pr(k) Pr(\mathcal{D}_k(y)).$$

Let us point out that we assume here that a new key k is chosen randomly for each new message x.

Definition 4.2.3. *Such a cryptosystem is* perfectly secure *(**parfaitement sûr**) if*

$$\forall x \in \mathcal{P}, \forall y \in \mathcal{C}, Pr(x|y) = Pr(x).$$

Proposition 4.2.2. *If for any $y \in \mathcal{C}$, $Pr(y) > 0$ and if the system is perfectly secure then*

$$\#\mathcal{K} \geq \#\mathcal{C} \geq \#\mathcal{P}.$$

Proof. Let $x \in \mathcal{P}$ such that $Pr(x) > 0$. For any $y \in \mathcal{C}$, one has $Pr(x|y) = Pr(x)$. By Prop. 4.2.1 $Pr(y)Pr(x|y) = Pr(y|x)Pr(x)$ so $Pr(y|x) > 0$ which means that there is at least one key $k \in \mathcal{K}$ such that $\mathcal{E}_k(x) = y$. So $\#\mathcal{K} \geq \#\mathcal{C}$. As $\mathcal{E}_k$ is injective $\#\mathcal{C} \geq \#\mathcal{P}$. □

Proposition 4.2.3. *Assume that the cryptosystem satisfies $\#\mathcal{K} = \#\mathcal{C} = \#\mathcal{P}$ and that $Pr(y) > 0$ for any $y \in \mathcal{C}$. It is perfectly secure if and only if*

- *all the keys have the same probability;*
- *for all $x \in \mathcal{P}$ and all $y \in \mathcal{C}$ there exists a unique key k satisfying $\mathcal{E}_k(x) = y$.*

Proof. If the conditions are satisfied, let us compute

$$\begin{aligned} Pr(y) &= \sum_{\substack{k\in\mathcal{K}\\ y\in\mathcal{E}_k(\mathcal{P})}} Pr(k)Pr(\mathcal{D}_k(y)), \\ &= \frac{1}{\#\mathcal{K}}\sum_{k\in\mathcal{K}} Pr(\mathcal{D}_k(y)), \\ &= \frac{1}{\#\mathcal{K}}\sum_{x\in\mathcal{P}} Pr(x) = \frac{1}{\#\mathcal{K}}. \end{aligned}$$

On the other hand, for $x \in \mathcal{P}$ and $y \in \mathcal{C}$, as there is exactly one key k such that $\mathcal{E}_k(x) = y$, one has

$$Pr(y|x) = \sum_{k|\ \mathcal{E}_k(x)=y} Pr(k) = \frac{1}{\#\mathcal{K}}.$$

Now

$$Pr(x|y) = \frac{Pr(x)Pr(y|x)}{Pr(y)} = Pr(x).$$

Conversely, assume that the security is perfect. As in the previous proposition, one shows that for all $(x, y) \in \mathcal{P} \times \mathcal{C}$ there exists a key k such that $\mathcal{E}_k(x) = y$. If x is fixed one has

$$\{\mathcal{E}_k(x),\ k \in \mathcal{K}\} = \mathcal{C}.$$

As $\#\mathcal{K} = \#\mathcal{C}$, the key k has to be unique.
Let k_1, k_2 be two different keys. Let $y \in \mathcal{C}$ and $x_i \in \mathcal{P}$ be the unique elements such that $\mathcal{E}_{k_i}(x_i) = y$. One has

$$P(y) = P(y|x_i) = \sum_{k|\ \mathcal{E}_k(x_i)=y} Pr(k) = Pr(k_i)$$

so $Pr(k_1) = Pr(k_2)$. □

Example 20. *The one-time-pad of Vernam (see 1.4.2) satisfies of course the conditions of the proposition. It is then perfectly secure.*
On the contrary, if x is of length $n > 1$ and A a invertible binary matrix of size $n \times n$ (picked randomly) then $y = Ax$ is not perfectly secure ! Indeed, $Pr(x = 0) = 1/2^n$ when $Pr(x = 0|y = 0) = 1$.

4.2.4 Birthday paradox

The problem is the following. Suppose a group of people are in a room. What is the probability that two of them have the same birthday ? This probability is astonishingly large.
We will make a slightly more general analysis. Suppose that there are n birthdays and k people. An elementary event is a tuple $(b_1, \ldots, b_k) \in \{1, 2, \ldots, n\}^k$. So we have n^k elementary events which are equally probable. Then the probability of an elementary

event is $1/n^k$.
We want to compute the probability p that two people among k in the room have the same birthday. In order to do that, one evaluates the probability $1-p$ that any two people have different birthdays. The event we are interested is the set E of all sequences $(g_1, \ldots, g_k)$ whose entries are pairwise distinct. The order of E can be computed as follow : the first entry can be any of the n possibilities. If the first entry is fixed then there are $n-1$ possibilities for the second entry and so on. Hence we obtain

$$|E| = \prod_{i=0}^{k-1}(n-i)$$

and

$$1 - p = |E|/n^k = \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right).$$

Now $1 + x \leq e^x$ for all real number so

$$1 - p \leq e^{-k(k-1)/(2n)}.$$

If $k \geq (1 + \sqrt{1 + 8n\log 2})/2$ then $p \geq 1/2$. For $n = 365$ one can choose $k = 23$.

Example 21. *Suppose that four-digit PINs are randomly distributed. How many people must be in a room such that the probability that two of them have the same PIN is at least* $1/2$ *?*

We can see now why we need at least 128 bits for hash functions : indeed $k \approx 2^{60}$ so n must be at least quadratic with respect to k, so $n > 2^{120}$ in order to avoid collisions.

4.3 Comparison of functions, complexity

To define complexity in a proper way, one should introduce Turing machines. However we will assume an intuitive understanding of an algorithm by a program on a computer. We will not consider operations at the level of the symbols of the alphabet of the machine. Instead we will consider elementary operations as addition, multiplications, etc.
It can be shown that the class of complexity (polynomial, exponential or subexponential) does not depend on the model we choose. Thus we will express the *size* (**taille**) of the input in terms of their length in bits. For instance if the input is an integer n, its size will be $|n|_2 = \lfloor \log_2(n) \rfloor + 1$.
We want to analyze the complexity of algorithms which will be essentially their running time. Indeed we want to

- evaluate the resources we need (CPU time).
- compare two different algorithms solving the same problem.
- give a upper bound on the size of the input for which the running time is feasible in practice.

We recall the following definitions.

Definition 4.3.1. *Let (D_n, Pr) the set of all input of size n of an algorithm A with probability distribution Pr. Let $C_n(d)$ be the number of elementary operations done by A on the input $d \in D_n$.*
The best cases *(**cas favorables**) are the inputs d_0 such that $C_n(d_0) = \min\{C_n(d), d \in D_n\}$.*
The worst cases *(**cas défavorables**) are the inputs d_0 such that $C_n(d_0) = \max\{C_n(d), d \in D_n\}$.*
The average running time *(**temps moyen d'éxécution**) of A is given by $\sum_d C_n(d)Pr(d)$.*

To analyze an algorithm is to try to compare the average cost with usual functions $\log_2(n), n, n^2, 2^n, \ldots$.

4.3.1 Asymptotic behavior

Let f, g be two positive functions. One says that f is *dominated by* (**dominée par**) g (near $+\infty$) if there exists a constant $c > 0$ and x_0 such that for all $x > x_0$, $f(x) \leq cg(x)$. One denotes $f = \mathcal{O}(g)$.
One says that f is *negligible* (**négligeable**) with respect to g if for all ϵ there exists x_0 such that for $x > x_0$ $f(x) \leq \epsilon g(x)$. One denotes $f = o(g)$. Also if g is not given, we say that f is negligible if for any integer n, $f(x) = o(x^{-n})$.
f and g are said *equivalent* if $f - g = o(g)$. One denotes $f \sim g$.

Example 22. $x^2 + 5x = \mathcal{O}(x^2)$, $x\log(x) = o(x^2)$, $x^3 + x + \log(x)^5 \sim x^3$.

Sometimes, we use also $\tilde{\mathcal{O}}(g(n))$ to mean $\mathcal{O}(g(n)\log(g(n))^k)$ for some k.

4.3.2 Complexity

Let n be the maximal size of the input.

Definition 4.3.2. *An algorithm is said* exponential *(**exponentiel**) if its average running time is $\mathcal{O}(e^{cn})$ for a constant $c > 0$.*
An algorithm is said polynomial *(**polynômial**) if its average running time is $\mathcal{O}(n^d)$ for a constant $d \geq 1$. The set of problems that can be solve by a polynomial algorithm is denoted P.*
An algorithm is said subexponential *(**sous-exponentiel**) id its average running time is $\mathcal{O}(L_n(u, v))$ where*

$$L_n(\alpha, c) = e^{cn^\alpha (\log n)^{1-\alpha}}$$

for some $0 < \alpha < 1$ and $c > 0$.

Some classical running-times :

- addition of two integers of size n : $\mathcal{O}(n)$.
- multiplication of two integers of size n : $\mathcal{O}(n^2)$.

- reduction of an integer of size $2n$ modulo an integer of size n : $\mathcal{O}(n^2)$.
- multiplication of two integers of size n modulo an integer of size n : $\mathcal{O}(n^2)$.
- Euclid algorithm for two integers of size at most n : $\mathcal{O}(n^2)$.
- computation of $x^c \pmod m$ where x, c, m are of size at most n : $\mathcal{O}(n^3)$.
- computation of $a^{-1} \pmod m$ where a, m are of size n : $\mathcal{O}(n^3)$.
- application of Gauss triangulation algorithm on a $k \times n$ matrix with binary coefficients $\mathcal{O}(nk^2)$.

Remark 25. *These running-times do not take in account the optimizations which may be dramatic. For instance, the product of two integers of size n can be realize using Fast-Fourier-Transform in* $\mathcal{O}(n(\log n)^3)$*, see [Knu98].*

4.3.3 Euclid algorithm

Euclidian division

We present two algorithms to do the euclidian division of two integers $a > b$.

```
B := b;
R := a;
Q := 0;
while R ≥ B do
      R := R − B;
      Q := Q + 1;
end while;
```

It is easy to check that at each step $a = Qb + R$ and that R decreases.
We have now a binary version. First one computes the smaller integer n such that $2^n b > a$.

```
B := b;
R := a;
Q := 0;
N := n;
Aux := 2^N B;
while N > 0 do
      Aux := Aux/2;
      N := N − 1;
      if R < Aux then
         Q := 2 ∗ Q;
      else
         Q := 2Q + 1;
         R := R − Aux;
      end if;
end while;
```

It is easy to check that at each step $a = AuxQ + R$ and that $0 \leq R < Aux$ and that when $N = 0$ one has $Aux = B$.

Remark 26. *The algorithms have not the same complexity. If b is fixed, the first one runs in maximal time of the order of a, so it is exponential in the size of a. For the second one, one has to determine n which can be done in polynomial time in the size of a. Then the number of loops is of the order of $|a|_2$ so linear in the size of a. Note that the loops contain in both cases only elementary operations.*

Euclid algorithm

```
R0 := |a|;
R1 := |b|; (b ≠ 0)
while R1 > 0 do
      R := DivisionRemainder(R0, R1);
      R0 := R1;
      R1 := R;
end while;
```

The output is $R1 := 0$ and $R0 := \gcd(a, b)$. This is clear because the set of common divisors of a and b is the set of common divisors of $R0$ and $R1$. Moreover if $R1 = 0$ then $\gcd(R1, R0) = R0$. Before analyzing the complexity of this algorithm, we give an extended form which has the same complexity.

```
R0 := a, a ≥ 0;
R1 := b; (b > 0);
U0 := 1; U1 := 0;
V0 := 0; V1 := 1;
while R1 > 0 do
        Q := DivisionQuotient(R0, R1);
        R := DivisionRemainder(R0, R1);
        U := U0 − Q ∗ U1;
        V := V0 − Q ∗ V1;
        R0 := R1; R1 := R;
        U0 := U1; U1 := U;
        V0 := V1;V1 := V;
end while;
```

As previously at the end, $R1 := 0$, $R0 := \gcd(a, b)$ and one has

$$U0a + V0b = R0.$$

As previously this can be shown by recursion on the equalities :

$$\begin{aligned} U0a + V0b &= R0 \\ U1a + V1b &= R1 \\ R1 &\geq 0 \end{aligned}$$

Example 23. *Use the extended euclidian algorithm to compute a solution of* $42x \equiv 1 \pmod{67}$ *(answer :* 8*).*

We have seen that (if we use the binary euclidian division) all the operations are polynomial. So to prove that Euclid algorithms are polynomial it is enough to prove that the number of loops is at worst polynomial in the size of a.

Proposition 4.3.1. *Let* $a > b > 0$ *and* $\Phi := (1+\sqrt{5})/2$*. Then the number of iterations in the euclidian algorithm is at most* $\log(a)/\log(\Phi) < 1.441 \log_2(a)$*.*

Proof. Denote by $R0_k$ (resp. $R1_k, R_k, Q_k$) the value of $R0$ (resp. $R1, R, Q$) at the kth round of the loop for $0 \leq k \leq n$ (so $R0_0 = a$ and $R0_n = \gcd(a, b)$, $R_n = R1_n = 0$ et $R0_k = R1_k Q_k + R_k$). We will prove by inverse induction that

$$R0_k \geq R0_n \Phi^{n-k}.$$

This is true for $k = n$. For $k = n - 1$ one has

$$R0_{n-1} = R1_{n-1}Q_{n-1} + R_n = R1_{n-1}Q_{n-1} = R0_n Q_{n-1} > 2R0_n > R0_n \Phi^1.$$

Indeed $Q_{n-1} \neq 1$ since otherwise we would have $R0_{n-1} = R1_{n-1}$ which implies $R_{n-2} = R1_{n-1}$ (impossible : the remainder must be smaller than the divider).
For $n - 2 \geq k \geq 0$ one has

$$R0_k = R0_{k+2} + Q_k R0_{k+1} \geq R0_{k+2} + R0_{k+1} \geq R0_n \Phi^{n-k-1} \left(1 + \frac{1}{\Phi}\right) = R0_n \Phi^{n-k}.$$

This proves the proposition since

$$\log(R0_0) = \log(a) \geq n \log(\Phi) + \log(\gcd(a, b)).$$

□

Remark 27. *The presence of the golden number is not a mystery : it is associated to the growing of the Fibonacci sequence which is the worst case for the euclidian algorithm.*

4.3.4 Probabilistic algorithms

All complexity notions can be extended to probabilistic algorithms, i.e. algorithms for which some decisions may be taken randomly. It might seem paradoxal but often these algorithms give faster answers than deterministic algorithms, the problem is that their running time is less easy to evaluate and that sometimes they do not stop. More precisely one distinguishes the following categories :

- a *Las Vegas* algorithm is a probabilistic polynomial algorithm which may run endlessly but if it stops it gives the right answer.
- a *Monte Carlo* algorithm is a probabilistic algorithm which worst running time is polynomial. Then it always stops but if it answers yes the answer is sure but it may answer no when it is yes.
- an *Atlantic city* algorithm is a probabilistic algorithm which worst running time is polynomial. Then it always stops but can be wrong if the answer if yes or no. Of course the probability of a wrong answer is smaller than the correct one.

Note that we have defined the different algorithms for decision problems. By dichotomy argument it is possible to reduce in linear time the problem of the value of a function $f : \mathbb{N} \to \mathbb{N}$ to a decision problem.

Remark 28. *We can always simulate a probabilistic algorithm (running on a non-deterministic Turing machine) by a standard algorithm on a standard Turing machine. However the reduction time may be exponential.*
Caution : probabilistic algorithms are a different notion from non deterministic algorithms. For instance Monte Carlo algorithms are the algorithms to solve a problem in the class RP (Randomized problem) when non deterministic solve problem in a larger one denoted NP (nondeterministic polynomial). Intuitively a NP (decision) problem is a problem for which we can check the positive answers in polynomial time using an oracle which delivers a specific certificate. For instance the problem 'IsComposite?' is clearly in NP: indeed to check that an integer is composite it's enough to give two factors and to make the product. The problem 'IsPrime?' is also in NP (even in P). RP is included in NP because at least half of the certificates must be valid.

Example 24. *We will give the precise definition of a one-way function. A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function if*

- *there is an Atlantic City algorithm which computes f;*
- *for all probabilistic algorithm (A, Pr) which runs in polynomial maximal time there exists a function ν satisfying $\lim_{k\to\infty}(\nu(k)k^m) = 0$ for all m, and k_0 such that for all $k > k_0$ and for all $x \in \{0,1\}^k$*

$$Pr(f(A(1^k, f(x))) = f(x)) \leq \nu(k).$$

The word 1^k is an artificial input to impose that the input is at length at least k even if $f(x)$ is smaller than x. To understand the interest of this constraint, let us give the following example. Let us define

$$f(x) = f(x_1 \dots x_k) = (x_1 \dots x_t)$$

for $t = \lfloor \log_2(k) \rfloor + 1$. It is very easy to find an algorithm giving a preimage of f. However as the input of A is $f(x)$, of length t, the writing of such a solution is exponential in t. So we would have an 'easy exponential' algorithm. To avoid this paradox, one imposes the input of A to be of length k and then we have an easy linear algorithm.

Remark 29. *An alternative definition is to ask that f is computable in polynomial time but that f^{-1} is not. Then it is clear that to find such an f would prove $P \neq NP$. However this definition which is convenient for theoretical studies is not accurate for cryptography.*
Complexity theory helps us to delimit problems which might be useful to build cryptosystem. However, complexity theory deals with difficulty in the worst case, whereas cryptography deals with difficulty in the average case. For instance knapsack problems that are on a complexity point of view the most difficult (NP-complete) were cryptographically broken (by lattice attacks, see [Odl90]).
To see that RSA for instance is not 'hard in worst case' and 'easy on average', there is a technique called random self reduction. *Suppose that we are given y and are asked to solve $y \equiv x^e \pmod N$ where the idea is that this is a worst case instance. We reduce this to an average problem by choosing $s \in (\mathbb{Z}/N\mathbb{Z})^*$ at random and setting $y' = s^e y$. Then we try to solve $y' \equiv x'^e \pmod N$. If the average case is easy, we can solve the hard case also since $x = x'/s$.*

4.3.5 Reduction of a problem

Definition 4.3.3. *Solving a problem $\mathcal{P}_1$, we say that we call a* $\mathcal{P}_2$-oracle *if the algorithm we use transforms the input of the problem $\mathcal{P}_1$ in an input of $\mathcal{P}_2$.*

Definition 4.3.4. *If $\mathcal{P}_1$ and $\mathcal{P}_2$ are two problems, we say that $\mathcal{P}_1$* reduces *(**se réduit**) to $\mathcal{P}_2$ (in polynomial time) if there exists a polynomial algorithm for $\mathcal{P}_1$ which uses a $\mathcal{P}_2$-oracle at most a polynomial times.*

Example 25. *See the proof of Th. 3.1.3.*

4.4 Quadratic residues

Definition 4.4.1. *Let a, n be two coprime integers. We say that a is a* quadratic residue *(**résidu quadratique**) modulo n if there is an integer x such that $x^2 \equiv a \pmod{n}$.*

We will deal with two problems : find out if a number is a quadratic residue; if it is so, find out a square root.

4.4.1 Legendre, Jacobi symbols

Let $n = p$ be an odd prime (the case $p = 2$ is trivial). We know (Prop. 4.1.9) that $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic. Let α be a primitive element.

Lemma 4.4.1.

$$\alpha^{(p-1)/2} = -1.$$

Moreover, the element α is not a quadratic residue.

Proof. By Fermat's little theorem $\alpha^{p-1} = 1$, so $\alpha^{(p-1)/2} = \pm 1$. But as α is of order $p-1$ one has $\alpha^{(p-1)/2} = -1$. So α is not a quadratic residue. Otherwise there would exist β such that $\beta^2 = \alpha$ and $\alpha^{(p-1)/2} = \beta^{p-1} = 1$. □

Lemma 4.4.2. *Let $n = p$ be an odd prime. The quadratic residue are the even powers of α and 0. Thus, half of the $p-1$ elements of $(\mathbb{Z}/p\mathbb{Z})^*$ are quadratic residue.*

Proof. It is enough to prove that α^{2k+1} is not a quadratic residue. Otherwise there would be u such that $u^2 = \alpha^{2k+1}$, so $\alpha = u^2/\alpha^{2k}$ would be a square. □

Definition 4.4.2. *Let $n = p$ be an odd prime number and a an integer. We define the* Legendre symbol *of a by:*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \text{ is divisible by } p, \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1 & \text{if } a \text{ is not a quadratic residue modulo } p. \end{cases}$$

Lemma 4.4.3. *If a is prime to p, then*

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$

Proof. Let α be a primitive element of $(\mathbb{Z}/p\mathbb{Z})^*$. One can write $a \equiv \alpha^k \pmod{p}$. Now

$$a^{(p-1)/2} \equiv (\alpha^{(p-1)/2})^k \equiv (-1)^k \pmod{p}.$$

Lem. 4.4.2 leads to the conclusion. □

The Legendre symbol of a is equal to 1 if and only if a is a quadratic residue. By the expression of the previous lemma, this property can be computed in polynomial time.

Definition 4.4.3. *Let $n \geq 3$ be an odd integer. We write $n = p_1 \cdots p_s$ where the p_i are prime numbers (not necessarily distinct). Let a be an integer. The* Jacobi symbol *of a is defined by*

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right) \cdots \left(\frac{a}{p_s}\right).$$

This symbol is multiplicative in both arguments:

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right),$$

$$\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right).$$

Remark 30. *The Jacobi symbol only depends on the class of a modulo n.*
The definition can be extended to even numbers n by multiplicativity.

The Jacobi symbol can be computed efficiently even if the factorization of n is not known.

Theorem 4.4.1. *Let $n > 3$ be an odd integer. Then*

$$\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2},$$

$$\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}.$$

Theorem 4.4.2 (quadratic reciprocity law)**.** *Let $m, n \geq 3$ be odd coprime integers. Then*

$$\left(\frac{m}{n}\right) = (-1)^{\frac{(n-1)(m-1)}{4}} \left(\frac{n}{m}\right).$$

Example 26. *Compute*

$$\begin{aligned}\left(\frac{88}{35}\right) &= \left(\frac{53}{35}\right) = \left(\frac{18}{35}\right) \\ &= \left(\frac{2}{35}\right)\left(\frac{9}{35}\right) = -1 \cdot 1 \cdot \left(\frac{35}{9}\right) \\ &= -1 \cdot \left(\frac{-1}{9}\right) = -1.\end{aligned}$$

Remark 31. *The Jacobi symbol does not characterize quadratic residues modulo n. Clearly if the Jacobi symbol of a is -1 then a is not a quadratic residue but if it is 1 we cannot say more.*
An even integer for n in the quadratic reciprocity law is forbidden. So if m is even, we have to take first power of 2 out, before using the formula.

4.5 Continued fractions

Continuous fractions are important in many branches of mathematics and particularly in the theory of approximation to real numbers by rational. We shall refer to [HW79, Chap.X] for the proofs.

4.5.1 Finite continued fraction

We shall describe the expression

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \quad \cdots \\ \qquad + \frac{1}{a_N}}}}$$

which we shall denote $[a_0, a_1, a_2, \ldots, a_N]$ and call a *continued fraction* (**fraction continue**). We will restrict to the case where

$$\text{the } a_i \text{ are positive integers (for } i > 0 \text{) and } a_0 \text{ is an integer.} \tag{4.2}$$

Theorem 4.5.1. *[HW79, Th.149] If p_n and q_n are defined by*

$$\begin{array}{lll} p_0 = a_0 & p_1 = a_1 a_0 + 1 & p_n = a_n p_{n-1} + p_{n-2} \\ q_0 = 1 & q_1 = a_1 & q_n = a_n q_{n-1} + q_{n-2} \end{array}$$

then

$$[a_0, a_1, \ldots, a_n] = \frac{p_n}{q_n}.$$

Theorem 4.5.2. *[HW79, Th.150] The functions p_n and q_n satisfy*

$$p_n q_{n-1} - p_{n-1} q_n = (-1)^{n-1}.$$

4.5.2 Infinite continued fraction

We start by 'taking the limit' of the previous process and call it an *infinite continued fraction.*

Theorem 4.5.3. *[HW79, Th.165] If (a_i) is a sequence of integers satisfying (4.2) then the sequence $[a_0, a_1, \ldots, a_i]$ converges and we denote its limit by $[a_0, a_1, \ldots]$.*

We can now inverse the process. Let x be a real number and let $a_0 = \lfloor x \rfloor$. We have then $x = a_0 + \xi_0$ with $0 \leq \xi_0 < 1$. We define now $a_1 = \lfloor 1/\xi_0 \rfloor$. We have $a_1 > 0$ and we define $1/\xi_0 = a_1 + \xi_1$ with $0 \leq \xi_1 < 1$. Iterating, we define in this way a sequence (a_i) satisfying (4.2).

Theorem 4.5.4. *The number $[a_0, a_1, \ldots]$ is equal to x. Moreover if x is irrational, it can be represented in a unique way as an infinite continued fraction.*

Example 27. *The continued fraction development of a rational number is finite. The periodicity of the development is equivalent to x being an irrational root of a quadratic equation with integral coefficients.*
$e = [2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, \ldots]$.
$\pi = [3, 7, 15, 1, 292, 1, 1, \ldots]$. *This sequence is much more mysterious. For instance nobody knows if the upper limit of the coefficients is $+\infty$ or not. It can be shown that this problem is equivalent to the density in $\mathbb{R}$ of the sequence $(n \sin n)$.*

Continued fractions are good approximations of numbers as the following theorem shows.

Theorem 4.5.5. *[HW79, Th.171,183,184] One has*

$$|x - \frac{p_n}{q_n}| < \frac{1}{q_n^2}$$

and either $p/q = p_n/q_n$ or $p/q = p_{n+1}/q_{n+1}$ satisfies

$$|x - \frac{p}{q}| < \frac{1}{2q^2}$$

Conversely if p/q is a fraction which satisfies

$$|x - \frac{p}{q}| < \frac{1}{2q^2}$$

then there is an n such that $p/q = p_n/q_n$.

4.6 Finite fields

In this section we describe a method for constructing finite fields.

4.6.1 Definitions

Definition 4.6.1. *Let p be a prime and n a positive integer n. We call the* finite field *(**corps fini**) over $\mathbb{Z}/p\mathbb{Z}$ of degree n the decomposition field of the polynomial $X^{p^n} - X$. We denote this field $\mathbb{F}_{p^n}$.*

It can be shown (see [Goz97, chap.VII]) that $\mathbb{F}_{p^n}$ is the only field with p^n elements up to isomorphism. In particular, if f be an irreducible polynomial of degree n over $\mathbb{F}_p$, it defines an extension of $\mathbb{F}_p$ of degree n, $\mathbb{F}_p[X]/f$, which can be identified to $\mathbb{F}_{p^n}$.

Example 28. *The polynomial $X^2 + X + 1$ is irreducible over $\mathbb{F}_2$ (otherwise it would have a root in $\mathbb{F}_2$). It defines then the extension $\mathbb{F}_4$.*

Proposition 4.6.1. *The extension $\mathbb{F}_{p^n}/\mathbb{F}_p$ is Galois with Galois group isomorphic to $\mathbb{Z}/n\mathbb{Z}$. A generator of this group is the automorphism $x \mapsto x^p$.*

4.6.2 Irreducible polynomials

Rabin's test relies on the following theorem.

Proposition 4.6.2. *If f is of degree n and irreducible then $f|X^{p^n} - X$.*

Proof. Let F be the extension of $\mathbb{F}_p$ defined by f. As F is Galois, f has all its roots in F. As f is irreducible, these roots have multiplicity one and we know that they are zeros of $X^{p^n} - X$. So $f|X^{p^n} - X$. □

So we can apply the following algorithm :

1. Test if $f|X^{p^n} - X$. If not f is not irreducible.
2. for each prime q dividing n compute $\gcd(f, X^{p^{n/q}} - X)$. f is irreducible if and only if all these numbers are equal to 1.

Example 29. *Let $p = 2$ and $f(X) = X^8 + X^4 + X^3 + X + 1$. We can check that $f|X^{2^8} - X$ and $\gcd(f, X^{2^4} - X) = 1$.*

4.6.3 Operations

The elements of the field are residue classes modulo f. The construction of those residue classes corresponds to the construction of residue classes in $\mathbb{Z}$. By euclidian division, a representative of the class is a polynomial of degree less than $\deg(f)$. Addition and multiplication can be done imitating the process in $\mathbb{Z}/n\mathbb{Z}$.

Example 30. *Representatves of the residue classes in $\mathbb{F}_2[X]/X^2+X+1$ are $0, 1, X, 1+X$. We obtain the following addition and multiplication tables.*

$+$	0	1	X	$X+1$
0	0	1	X	$X+1$
1	1	0	$X+1$	X
X	X	$X+1$	0	1
$X+1$	$X+1$	X	1	0

$\times$	1	X	$X+1$
1	1	X	$X+1$
X	X	$X+1$	1
$X+1$	$X+1$	1	X

We can also compute the inverse of an element using Bézout theorem. Indeed, since f is irreducible, for any $g \in \mathbb{F}_p[X]$ of degree less than $\deg(f)$, f and g are coprime. There exist then (u, v) such that $uf + vg = 1$. Thus v is the inverse of g modulo f.

Example 31. *Let $p = 2$ and $f(X) = X^8 + X^4 + X^3 + X + 1$. We determine the inverse of $X + 1$ modulo f. We have*

$$f(X) = (X + 1)(X^7 + X^6 + X^5 + X^4 + X^2 + X) + 1$$

so the class of $X^7 + X^6 + X^5 + X^4 + X^2 + X$ is an inverse of $X + 1$ modulo f.

Chapter 5

Travaux dirigés

5.1 TD 1 : chiffrement par substitutions et arithmétique

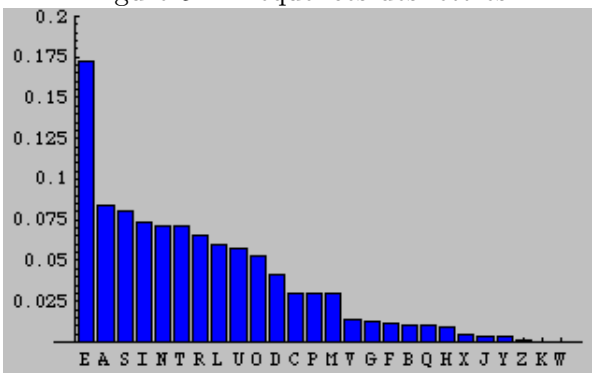
Déchiffrement

On considère le texte suivant, chiffré en utilisant la méthode de chiffrement par substitutions. Le décrypter.

WVZJXOJUKVNAPZEIKXZJTPKHOJXKEPFZVNTKIKTOVXIKZZAAIKJXI KCKZVBOVAXKBQOTKTWVOJTZENAZIXKTRKJTWVEPKTERJOIKJN.

On rappelle le graphique des fréquences pour les lettres et la table pour certains couples en ordre décroissant de fréquence.

Figure 5.1: fréquences des lettres



ES	DE	LE	EN	RE	NT	ON	ER	TE	EL	AN	SE	ET

Divisibilité et congruence

1. Montrer que le nombre qui s'écrit en base 10 $d_1 \dots d_k$ est divisible par 11 si et seulement si $\sum_{i=1}^{k} (-1)^{k-i} d_i$ est divisible par 11.

2. Soit $n \in \mathbb{N}^*$. Montrer que les nombres $n(n+1)$ et $n(n+2)$ ne sont jamais des carrés.

3. Pour tout $m, n \in \mathbb{N}^2$, montrer qu'on a $m!\, n! | (m+n)!$.

4. Montrer que si $(m, n) = 1$ alors

$$\frac{(m+n-1)!}{m!\, n!}$$

est un entier. Donner un contre-exemple si $(m, n) \neq 1$.

5. Montrer que $\text{pgcd}(a^m - 1, a^n - 1) = a^{\text{pgcd}(m,n)} - 1$ pour tout couple d'entiers positifs m, n et $a \in \mathbb{Z}$.

6. Combien d'essais (en fonction de la taille de p) sont nécessaires pour trouver deux nombres premiers (p, q) de même taille tels que q divise $p - 1$?

7. Ecrire la table d'addition et de multiplication de $\mathbb{Z}/7\mathbb{Z}$.

8. Quels sont le éléments inversibles de $\mathbb{Z}/25\mathbb{Z}$? Quels sont leur inverse ?

9. Expliquer le fonctionnement de l'algorithme d'exponentiation rapide. Calculer $2^{2005} \pmod 7$.

10. Quelles sont les solutions du système

$$\begin{cases} x & \equiv 1 \pmod 2 \\ x & \equiv 2 \pmod 3 \\ x & \equiv 3 \pmod 5 \\ x & \equiv 4 \pmod 7 \end{cases}$$

5.2 TD 2 : Structure de $(\mathbb{Z}/n\mathbb{Z})^*$, un protocole simple et probabilités

Groupe des éléments inversibles

1. Soit G un groupe et $g \in G$. Montrer que $g^n = 1$ ssi n est un multiple de l'ordre de g.

2. Soit G un groupe et $g \in G$. Montrer que si $g^n = 1$ et que pour tout p premier divisant n, $g^{n/p} \neq 1$ alors n est l'ordre de g.

3. Soient p et q deux nombres premiers distincts. A-t-on

$$p^{q-1} + q^{p-1} \equiv 1 \pmod{pq}?$$

4. Montrer que si $n = \prod_{i=1}^{k} p_i^{e(p_i)}$ (p_i premiers distincts) est impair, le nombre de solutions de $x^2 \equiv 1 \pmod{n}$ est 2^k.

5. Montrer le théorème suivant: p est premier si et seulement si

$$(p-1)! \equiv -1 \pmod{p}.$$

Un protocole d'échange

Alice veut envoyer à Bob le message $x \in \{0,1\}^n$. Alice (resp. Bob) possède une clé secrète a (resp. b) de même longueur que x. Ils effectuent le protocole suivant :

1. Alice envoie $A_1 = x \oplus a$ à Bob.

2. Bob envoie $B_1 = A_1 \oplus b$ à Alice.

3. Alice envoie $A_2 = B_1 \oplus a$ à Bob.

Montrer que Bob peut retrouver le message x. Montrer que si Oscar a intercepté tous les échanges, il peut également retrouver x.

Probabilités

Nous allons étudier la sécurité de Shannon pour les systèmes symétriques. On suppose que les ensembles $\mathcal{P}$ et $\mathcal{K}$ sont donnés avec des lois de probabilité notées respectivement $Pr_{\mathcal{P}}$ et $Pr_{\mathcal{K}}$. On note alors Pr la loi de probabilité produit sur $\mathcal{P} \times \mathcal{K}$ (i.e. $Pr(A \times B) = Pr_{\mathcal{P}}(A) \times Pr_{\mathcal{K}}(B)$). Pour simplifier les notations, on pourra aussi écrire $Pr(x)$ pour $x \in \mathcal{P}$ en identifiant $\{x\}$ avec $\{x\} \times \mathcal{K}$ (ainsi $Pr(\{x\} \times \mathcal{K}) = Pr_{\mathcal{P}}(\{x\})$ et de même pour $k \in K$. Enfin si $y \in \mathcal{C}$, on note $Pr(y)$ la probabilité de l'évènement $\{(x,k),\ \mathcal{E}_k(x) = y\}$.

1. On dit qu'un tel système symétrique est *parfaitement sûr* si

$$\forall x \in \mathcal{P}, \forall y \in \mathcal{C}, Pr(x|y) = Pr(x).$$

   Justifier la dénomination.

2. Montrer que si $\forall y \in \mathcal{C}$, $Pr(y) > 0$ et que si le système est parfaitement sûr alors

$$\#\mathcal{K} \geq \#\mathcal{C} \geq \#\mathcal{P}.$$

   On pourra utiliser la formule de Bayes et interpréter $Pr(y|x) > 0$.

3. Montrer que pour $y \in \mathcal{C}$ fixé,

$$\{(x,k),\ \mathcal{E}_k(x) = y\} = \bigcup_{\substack{k \in \mathcal{K} \\ y \in \mathcal{E}_k(\mathcal{P})}} \{(\mathcal{D}_k(y), k)\}$$

   et qu'ainsi

$$Pr(y) = \sum_{\substack{k \in \mathcal{K} \\ y \in \mathcal{E}_k(\mathcal{P})}} Pr(k) Pr(\mathcal{D}_k(y)).$$

4. On suppose maintenant que $\#\mathcal{K} = \#\mathcal{C} = \#\mathcal{P}$ et que $\forall y \in \mathcal{C}$, $Pr(y) > 0$. Montrer que le système est parfaitement sûr si et seulement si

   - toutes les clés ont la même probabilité ;
   - $\forall x \in \mathcal{P}$ et $\forall y \in \mathcal{C}$ il existe une unique clé k satisfaisant $\mathcal{E}_k(x) = y$.

   Pour montrer la sécurité parfaite, on pourra utiliser la formule de la question précédente.

5. On imagine le système de chiffrement suivant : soit $n > 1$ un entier et $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$. On suppose que $Pr_{\mathcal{P}}$ est telle que $Pr_{\mathcal{P}}(x) > 0$ pour tout x. Pour tout $x \in \mathcal{P}$ on tire aléatoirement (avec une distribution uniforme) une clé $k \in \mathcal{K}$ et on envoie $\mathcal{E}_k(x) = x \oplus k$. Montrer qu'un tel système est parfaitement sûr. Quels sont ses inconvénients ?

5.3 TD 3 : complexité

Algorithme d'Euclide

Division euclidienne

Voici deux algorithmes pour la division euclidienne de $a > b > 0$.

```
B := b;
R := a;
Q := 0;
while R ≥ B do
      R := R − B;
      Q := Q + 1;
end while;
```

- Pourquoi cet algorithme s'arrête-t-il ?
- Pourquoi donne-t-il le bon résultat ? On vérifiera qu'à chaque étape $a = Qb + R$.

En voici maintenant une version binaire. Soit n le plus petit entier tel que $2^n b > a$.

```
B := b;
R := a;
Q := 0;
N := n;
Aux := 2^N B;
while N > 0 do
      Aux := Aux/2;
      N := N − 1;
      if R < Aux then
         Q := 2 ∗ Q;
      else
         Q := 2Q + 1;
         R := R − Aux;
      end if;
end while;
```

- Vérifier qu'à chaque étape on a $a = Aux \cdot Q + R$ et $0 \leq R < Aux$ et que si $N = 0$ on a $Aux = B$.
- Comparer la complexité des deux algorithmes.

Algorithme d'Euclide

```
R0 := |a|;
R1 := |b|; (b ≠ 0)
while R1 > 0 do
        R := DivisionRemainder(R0, R1);
        R0 := R1;
        R1 := R;
end while;
```

- Montrer que lorsque $R1 := 0$ on a $R0 := \gcd(a, b)$.

Voici maitenant la version étendue de l'algorithme d'Euclide qui a la même complexité.

```
R0 := a, a ≥ 0;
R1 := b; (b > 0);
U0 := 1; U1 := 0;
V0 := 0; V1 := 1;
while R1 > 0 do
        Q := DivisionQuotient(R0, R1);
        R := DivisionRemainder(R0, R1);
        U := U0 − Q ∗ U1;
        V := V0 − Q ∗ V1;
        R0 := R1; R1 := R;
        U0 := U1; U1 := U;
        V0 := V1;V1 := V;
end while;
```

- Montrer que, en sortie, $R1 := 0$, $R0 := \gcd(a, b)$ et

$$U0a + V0b = R0.$$

On pourra raisonner par récurrence sur les relations :

$$\begin{aligned} U0a + V0b &= R0 \\ U1a + V1b &= R1 \\ R1 &\geq 0 \end{aligned}$$

- Utiliser l'algorithme pour calculer les solutions de $42x \equiv 1 \pmod{67}$.

Pour montrer que l'algorithme est polynômial, il est suffisant de montrer que le nombre de boucles est polynômial en la taille de a car toutes les opérations à l'intérieur de la boucle le sont.

Proposition 5.3.1. *Soit $a > b > 0$ et $\Phi := (1 + \sqrt{5})/2$. Alors le nombre d'itérations est au pire* $\log(a)/\log(\Phi) < 1.441 \log_2(a)$.

Pour se faire, on note $R0_k$ (resp. $R1_k, R_k, Q_k$) les valeurs de $R0$ (resp. $R1, R, Q$) à la k-ième itérations, pour $0 \leq k \leq n$.

- Quelles sont les valeurs initiales et finales ?
- Montrer par récurrence descendante que

$$R0_k \geq R0_n \Phi^{n-k}.$$

5.4 TD4 : révisions

Questions simples

1. Peut-on avoir un exposant de chiffrement e pair pour RSA ?
2. Soit la clé publique $n = 55, e = 13$ pour RSA. Calculer la clé secrète d puis déchiffrer $y = 4$. Pour indication $4^{2^i} \equiv 16, 36, 31, 26, 16 \pmod{55}$ pour $i = 1, 2, 3, 4, 5$.

Une attaque cyclique sur RSA

Soit (n, e) une clé publique pour RSA. Soit $x \in (\mathbb{Z}/n\mathbb{Z})^*$ un texte clair et y son chiffré. Montrer qu'il existe un entier k tel que

$$x^{e^k} \equiv x \pmod{n}$$

et que

$$y^{e^{k-1}} \equiv x \pmod{n}.$$

Pensez-vous qu'une telle attaque est dangereuse pour RSA ? Justifiez.

Une attaque sur les bits de plus haut poids

Soit p, q deux grands nombres premiers tels que $p < q < 2p$ et $n = pq$. Soit (d, e) un couple de clé privée/publique pour RSA tel que $e < \phi(n)$ et $d < \phi(n)$.

1. Montrer que $p + q \leq 3\sqrt{n}$.
2. Montrer que si on note $k = (ed - 1)/\phi(n)$

$$\left| \frac{kn + 1}{e} - d \right| < 3\sqrt{n}.$$

3. On suppose jusqu'à la fin que $e = 3$. Montrer qu'alors $k = 2$.
4. Soit $d' = \lfloor \frac{kn+1}{e} \rfloor$. Comparer d' et d. En conclure qu'environ la première moitié des bits de d ne sont pas des hard-core bits.

Une méthode de factorisation

On rappelle qu'un témoin de non-primalité d'un entier n impair pour le test de Miller-Rabin est un entier a premier à n tel que, si on écrit $n - 1 = 2^s d$ avec d impair, alors $a^d \not\equiv 1 \pmod{n}$ et quelque soit $r \in \{0, \ldots, s-1\}$ on a $a^{2^r d} \not\equiv -1 \pmod{n}$.

Montrer que s'il existe un entier a témoin de non-primalité pour n tel que $a^{n-1} \equiv 1 \pmod{n}$ alors on peut trouver un facteur de n (On pourra commencer par montrer qu'il existe $r \in \{0, \ldots, s-1\}$ tel que $a^{2^r d} \not\equiv 1 \pmod{n}$ et $(a^{2^r d})^2 \equiv 1 \pmod{n}$).

5.5 TD 5 : Un protocole de signature

Nous étudions ici une variante du protocole de signature, utile par exemple dans le cas de figure suivant. Un client veut accéder à un coffre dans une banque. Celle-ci demande au client de signer un document daté avant que l'autorisation soit accordée. Cependant, le client ne veut pas que la banque puisse révéler à quiconque quand il a eu accès au coffre. Par conséquent, il ne veut pas que la vérification de sa signature puisse se faire sans sa participation. Dans ce protocole, il faut donc rajouter un système de contrôle pour qu'Alice ne puisse signer un document puis déclarer que ce n'est pas le cas. On propose le protocole suivant :

1. Choix de la clé. La signataire A choisit un groupe cyclique G fini d'ordre premier q, un générateur g et un élément $a \in (\mathbb{Z}/q\mathbb{Z})^*$, secret. Elle calcule alors $b = g^a$. Elle choisit également une fonction de hachage $h : \{0,1\}^* \to G$ et publie (G, g, q, b, h). *Donner un exemple de choix de groupe et de paramètre résistant pour DLP.*

2. Signature. Etant donné un document $m \in \{0,1\}^*$, A calcule $s = h(m)^a$. *Quelles propriétés cette signature possède-t-elle pour l'instant?*

3. Vérification. A va convaincre Victor (V) que s est bien la puissance a-ième of $h(m)$ sans révéler sa clé secrète a.

   (a) V a le document m, la signature s et b. Il choisit aléatoirement $u, v \in \mathbb{Z}/q\mathbb{Z}$ et calculer $z = s^u b^v$. Il envoit le challenge z à A.

   (b) *Montrer que Alice peut calculer w tel que $w = h(m)^u g^v$.* Elle envoie w à Victor.

   (c) Victor compare w et $h(m)^u g^v$ et accepte la signature s'ils sont égaux.

   Nous allons montrer que ceci permet de vérifier que la signature est correcte.

   (a) Montrer que pour tout $z \in G$ il y a q couples (u, v) tels que $s^u b^v = z$.

   (b) Montrer que si $s \neq h(m)^a$ et si $z \in G$ alors pour tout $w \in G$ il y a exactement un couple (u, v) tel que $h(m)^u g^v = w$ et $s^u b^v = z$ pour un w et un z donnés.

   (c) Montrer que si s n'est pas correcte, la probabilité qu'elle soit acceptée est inférieure à $1/q$.

4. Non répudiation. Supposons maintenant que durant la vérification $z = s^u b^v$ et w aient été échangés et que la vérification échoue. Alice déclare que s n'est pas correcte. *Pourquoi doit-elle être en mesure de prouver ce fait ?* Pour le montrer, on réalise alors le protocole suivant.

   (a) Victor effectue une seconde vérification avec un challenge $z' = s^{u'} b^{v'}$.

   (b) La réponse w' lui est renvoyée.

   (c) Victor vérifie alors si
   $$(wg^{-v})^{u'} = (w'g^{-v'})^u \tag{5.1}$$
   et si l'égalité est vérifiée, il accepte que la signature est incorrecte.

Nous allons montrer que ce protocole est correct.

(a) *Montrer que si A réalise correctement les calculs de vérification alors (5.1) est vraie.*

(b) *Supposons maitenant que s est valide mais que A veut convaincre V que ce n'est pas le cas. Alors* $w \neq h(m)^u g^v$. *Montrer alors que pour tout challenge* z' *et réponse* w' *il existe un unique couple* (u', v') *tel que (5.1) soit vraie.*

(c) *Montrer que lorsque s est valide, A peut faire échouer le test de vérification tout en préservant l'égalité (5.1) avec une probabilité inférieure à* $1/q$.

(d) *En déduire que ce protocole permet la non-répudiation.*

5.6 TD 6 : La signature Elgamal

Soit p un grand nombre premier et h une fonction de hachage à valeurs entières dans $[0, p-2]$. Soit g un élément primitif de $G = (\mathbb{Z}/p\mathbb{Z})^*$. On considère le protocole de signature suivant pour Alice :

Clé publique : (p, g, b) avec $b \equiv g^a \pmod{p}$;
Clé secrète : $a \in [0, p-2]$;
Signature : $y \to (y, r, s)$ avec $r \equiv g^k \pmod{p}$, $s \equiv k^{-1}(m - ra) \pmod{p-1}$ avec $m = h(y)$ et $k \in [1, p-2]$ différent à chaque signature;
Vérification : on vérifie que $0 \leq r \leq p-1$. Si non, rejeter la signature ; calculer $m = h(y)$; calculer $v \equiv g^m \pmod{p}$ et $w \equiv b^r r^s \pmod{p}$; vérifier que $v = w$.

1. *Rappeler les critères que doit remplir une signature.*
2. *Montrer que si la signature est correcte, on a bien $v = w$.*
3. *Quel élément un attaquant devrait-il calculer à priori pour signer des textes à la place d'Alice ?*
4. *Quel est la taille du paramètre de sécurité p pour qu'une telle attaque soit impossible de nos jours ?*
5. *Peut-on alors effectivement (i.e. en terme de temps de calcul) utiliser ce protocole ?*

Dans la suite, nous allons voir certaines attaques contre ce protocole de signature lorsqu'on oublie certaines des recommandations.

L'importance de k

Montrer que si Alice se sert deux fois du même k pour signer deux messages y_1 et y_2 différents, alors Oscar peut en général retrouver le secret a (On regardera l'expression s).

L'importance de la condition $0 \leq r \leq p-1$

Supposons que notre protocole ne vérifie pas cette condition. On va montrer que Oscar peut alors créer des falsifications sélectives, i.e. des signatures de nouveaux messages ayant un sens à partir d'une ancienne signature. Soit donc (y, r, s) une signature valide produite par Alice. Oscar souhaite signer un message y' avec la signature d'Alice. Oscar calcule :

$$u \equiv h(y')h(y)^{-1} \pmod{p-1}.$$

Il calcule ensuite

$$s' \equiv su \pmod{p-1}.$$

Il trouve r' tel que

$$r' \equiv ru \pmod{p-1}, \quad r' \equiv r \pmod{p}.$$

1. *Montrer comment et sous quelles conditions il peut réaliser tous ces calculs.*
2. *Vérifier que (y', r', s') est une signature valide.*
3. *Montrer que si h est sans collision alors la condition $0 \leq r \leq p-1$ empêche la falsification.*

L'importance de la fonction h

Supposons que Alice n'utilise pas de fonction de hachage, i.e. $m = y$ dans notre protocole. Oscar peut alors réaliser une autre falsification comme suit :

1. Soit i, j des entiers tels que $0 \leq i, j \leq p-2$. On cherche r sous la forme $r \equiv g^i b^j \pmod{p}$.
2. *Montrer que la relation $v = w$ est équivalente à*

$$g^{y-is} \equiv b^{r+js} \pmod{p}.$$

3. Ceci est le cas en particulier si

$$\begin{cases} y - is & \equiv 0 \pmod{p-1} \\ r + js & \equiv 0 \pmod{p-1} \end{cases}.$$

4. *Donner la condition pour que ce système admette une solution puis déterminer (r, s, y) en fonction de i, j.*
5. *Pourquoi cette falsification est moins puissante que la précédente ?*

5.7 TD 7 : Partage d'un secret et vote électronique

Partage

Le protocole de partage d'un secret de Shamir est basé sur le lemme suivant

Lemma 5.7.1. *Soient $l \leq t$ deux entiers positifs, p un nombre premier et $(x_i, y_i) \in \mathbb{Z}/p\mathbb{Z}$, $1 \leq i \leq l$ des couples tels que les x_i sont distincts. Alors il y a exactement p^{t-l} polynômes $P \in \mathbb{Z}/p\mathbb{Z}[X]$ de degré au plus $t-1$ tel que $P(x_i) = y_i$.*

Nous présentons maintenant un protocole de partage d'un secret $s \in \mathbb{Z}/p\mathbb{Z}$ entre n personnes de telle sorte que t d'entre elles (mais pas moins) peuvent reconstituer le secret initial.

1. Initialisation : soit p premier plus grand que $n+1$ et $x_i \in (Z/p\mathbb{Z})^*$, n éléments distincts. Le distributeur D publie les x_i.
2. Le partage :
   (a) D choisit secrètement $a_j \in Z/p\mathbb{Z}$, $1 \leq j \leq t-1$ et construit le polynôme
   $$a(X) = s + \sum_{j=1}^{t-1} a_j X^j.$$
   (b) D calcule les parts $y_i = a(x_i)$ et distribue secrètement à chaque i-ème membre du groupe le couple (x_i, y_i).
3. La reconstruction du secret : *montrer que si t membres du groupe collaborent ils peuvent retrouver le secret.*
4. Sécurité : supposons que $m < t$ membres du groupe collaborent. *Montrer que pour tout $s' \in \mathbb{Z}/p\mathbb{Z}$ il existe exactement p^{t-m-1} polynômes $a'(X) \in \mathbb{Z}/p\mathbb{Z}[X]$ de degré $\leq t-1$ tel que $a'(0) = s'$ et $a'(x_i) = y_i$, $1 \leq i \leq m$. En déduire que ces membres ne peuvent pas obtenir d'information sur le secret s.*

Vote électronique

Le vote est une procédure complexe puisqu'on veut obtenir (au moins !) les sept propriétés suivantes:

1. Seules les personnes autorisées peuvent voter;
2. Personne ne peut voter plus d'une fois;
3. Aucune partie ne peut déterminer le vote d'un tiers;
4. Aucune partie ne peut dupliquer un vote;
5. Le compte total doit être correct;

6. Toutes les parties peuvent vérifier que le résultat a été correctement calculé;

7. Le protocole fonctionne même en présence d'erreurs.

Initialisation : On suppose que les n centres de vote C_i ont chacun un chiffrement à clé public avec une fonction de chiffrement $\mathcal{E}_i$ *(en proposer un exemple)* et qu'au plus $t-1 < n/2$ d'entre eux peuvent être malhonnêtes. On suppose aussi fixé un groupe cyclique $(G, \times)$ d'ordre q premier et deux éléments $g, h \in G$ tel que $h = g^x$. On suppose que ce problème du logarithme discret est difficile, i.e. personne (pas même les centres de vote) ne connaît x. *Proposer un exemple de tel groupe.*

Vote : On suppose que chacun des m votants a le choix entre deux candidats $\{\pm 1\}$. Chaque votant choisit une valeur $v_j \in \{\pm 1\}$ et une valeur aléatoire $a_j \in \mathbb{Z}/q\mathbb{Z}$ et publie le vote $B_j = g^{a_j} h^{v_j}$. *Montrer que la propriété (3) est satisfaite de manière inconditionnelle. Quel protocole ajouter pour assurer les propriétés (1) et (2) ?.*
Pour la suite, il va être important de s'assurer que $v_j = \pm 1$. On propose le schéma de mise en gage suivant:

1. Le votant V_j choisit aléatoirement $r, w \in \mathbb{Z}/q\mathbb{Z}$ et $d \in (Z/q\mathbb{Z})^*$.

2. si $v_j = 1$ alors il publie
$$\alpha_1 = g^r (B_j h)^{-d},\ \alpha_2 = g^w.$$
Sinon il publie
$$\alpha_1 = g^w,\ \alpha_2 = g^r (B_j h)^{-d}.$$

3. Un des centres de vote C_i au hasard lui envoie un challenge aléatoire $c \in \mathbb{Z}/q\mathbb{Z}$.

4. Si $v_j = 1$, V_j calcule $d' = c - d$ et $r' = w + a_j d'$ et poste $(d_1, d_2, r_1, r_2) = (d, d', r, r')$. Sinon il poste $(d_1, d_2, r_1, r_2) = (d', d, r', r)$.

5. C_i vérifie si
$$\begin{aligned} d_1 d_2 &\neq 0 \\ d_1 + d_2 &= c \\ g^{r_1} &= \alpha_1 (B_j h)^{d_1} \\ g^{r_2} &= \alpha_2 (B_j / h)^{d_2}. \end{aligned}$$

Montrer que la vérification est correcte avec une probabilité $> 1/q$ si et seulement si $v_j \in \{\pm 1\}$. Montrer ensuite que C_i n'a aucune information sur le vote.

Distribution du vote : chaque votant V_j utilise le protocole de Shamir pour partager les secrets v_j, a_j entre les n centres de vote de telle manière qu'il faut au moins $t < m$ d'entre eux pour récupérer les secrets. On note
$$R_j = v_j + r_{1j} X + \ldots + r_{tj} X^t,\ S_j = a_j + s_{1j} X + \ldots + s_{tj} X^t,$$

les polynômes utilisés avec des valeurs aléatoires r_{lj}, s_{lj} et

$$(u_{ij}, w_{ij}) = (R_j(i), S_j(i)),\ 1 \leq i \leq m$$

les valeurs envoyées à chaque C_i chiffrées avec la fonction $\mathcal{E}_i$. *Montrer que la propriété (4) est maintenant complètement satisfaite.*
Le votant V_j se lie au polynôme R_j en publiant

$$B_{lj} = g^{s_{lj}} h^{r_{lj}},\ 1 \leq l \leq t.$$

Chaque centre C_i vérifie si

$$B_j \prod_{l=1}^{t} B_{lj}^{i^l} = h^{u_{ij}} g^{w_{ij}}.$$

Montrer que si V_j a bien choisi R_j et S_j pour ses calculs, la vérification réussit. A quoi sert-elle ?

Décompte : Chaque centre de vote C_i publie

$$T_i = \sum_{j=1}^{m} u_{ij},\ A_i = \sum_{j=1}^{m} w_{ij}.$$

Montrer que toutes les autres parties peuvent vérifier que T_i et A_i sont corrects en calculant

$$\prod_{j=1}^{m} \left(B_j \prod_{l=1}^{t} B_{lj}^{j^l} \right) = h^{T_i} g^{A_i}.$$

Montrer que les conditions (6), (7) sont vérifiées en interpolant au moins $t+1$ des valeurs T_i.

Solutions

Vote électronique

Supposons que $v_j \notin \{\pm 1\}$. Alors $B_j h$ et B_j/h ont des DL difficiles même pour V_j. Si V_j veut passer les tests $g^{r_1} = \alpha_1 (B_j h)^{d_1}$ et $g^{r_2} = \alpha_2 (B_j/h)^{c-d_1}$ ceci n'est donc possible que s'il a choisi $\alpha_1 = g^w (B_j h)^{-d_1}$ (ce qui fixe la valeur de d_1) puis $\alpha_2 = g^{w'} (B_j/h)^{-c+d_1}$ ce qu'il ne peut faire qu'avec $1/q$ chance puisqu'il ne connaît pas encore la valeur de c. C_i voit passer des nombres aléatoires dans $\mathbb{Z}/q\mathbb{Z}$, il ne peut donc en déduire aucune information.

5.8 TD 8 : révisions

Question de cours

1. Qu'est ce qu'une attaque CCA1 ? En cryptographie à clé publique, quelle est l'attaque la plus puissante ?

2. Pourquoi le chiffrement à clé publique n'a-t-il pas complètement mis fin au chiffrement symétrique ?

3. Quel est l'ordre de grandeur de la taille du module pour un protocole RSA de nos jours ? Pourquoi ?

4. Montrer pourquoi un système de chiffrement basé sur le problème du logarithme discret dans $(\mathbb{Z}, \times)$ est cryptographiquement faible.

5. Est ce que 5 est un résidu quadratique modulo 107 ?

6. Sur quel problème supposé difficile repose le chiffrement de Rabin ?

7. Expliquer l'attaque de l'homme du milieu sur le protocole d'échange de clés de Diffie-Hellman. Comment l'empêcher ?

8. Expliquer l'algorithme de signature d'ElGammal. Quel oubli dans l'algorithme permet une falsification sélective ?

9. Comment fonctionne le mode CTR (Counter Mode) ?

10. Quel est le rôle des S-boxes dans l'algorithme DES ? Par quoi sont-elles remplacées dans AES ?

11. Décrire le triple DES.

12. Pourquoi le fait de trouver deux nombres x, y tels que $x^2 \equiv y^2 \pmod{n}$ permet de factoriser n avec une probabilité $1/2$?

13. Quelle est le défaut de la méthode des pas de bébé, pas de géant ? Quelle est l'algorithme qui permet de l'éviter ?

14. Quelle restriction sur l'ordre du groupe cyclique impose une attaque basée sur l'algorithme de Polhig-Hellman ?

15. Qu'appelle-t-on sécurité sémantique ? RSA est-il sémantiquement sûr contre une attaque CCA2 (Justifier) ? Citer une méthode de chiffrement sémantiquement sûr contre une attaque CCA2.

Signature en aveugle

On considère le protocole suivant:

1. Un signataire S produit une clé publique (n, e) pour la signature RSA et la clé privée d. *Quelle est la taille de n pour une sécurité suffisante ? Pourquoi ?*
2. Un utilisateur A produit un message $m \in \mathbb{Z}/n\mathbb{Z}$ et un élément aléatoire $k \in (\mathbb{Z}/n\mathbb{Z})^*$. Il calcule $m' = mk^e$ et envoie m' à S.
3. S signe le message m' avec sa clé secrète d et renvoie le résultat s' à A.
4. A calcule $s = k^{-1}s'$. *Montrer que s est la signature de m.*
5. A envoie (m, s) à un vérificateur V. *Comment V vérifie-t-il la signature ?*
6. *Montrer comment on peut utiliser ce protocole pour que A puisse acheter chez V sans que sa banque S puisse lier A à l'achat.* On supposera qu'Alice utilise le message m comme un billet d'un montant fixé (et elle pourra donc en utiliser plusieurs pour payer).

Une méthode de factorisation

On rappelle qu'un témoin de non-primalité d'un entier n impair pour le test de Miller-Rabin est un entier a premier à n tel que, si on écrit $n - 1 = 2^s d$ avec d impair, alors $a^d \not\equiv 1 \pmod{n}$ et quelque soit $r \in \{0, \ldots, s-1\}$ on a $a^{2^r d} \not\equiv -1 \pmod{n}$.

Montrer que s'il existe un entier a témoin de non-primalité pour n tel que $a^{n-1} \equiv 1 \pmod{n}$ alors on peut trouver un facteur de n (On pourra commencer par montrer qu'il existe $r \in \{0, \ldots, s-1\}$ tel que $a^{2^r d} \not\equiv 1 \pmod{n}$ et $(a^{2^r d})^2 \equiv 1 \pmod{n}$).

Prédicat difficile pour le DL

Soit $(G = (\mathbb{Z}/p\mathbb{Z})^*, \times)$ avec p premier congru à 3 modulo 4. Soit g un élément primitif de G, $0 < x < p - 1$ un entier (inconnu) et $y = g^x$.

- Combien y'a-t-il d'éléments primitifs ?
- Montrer que si y est un carré alors $y^{(p-1)/2} = 1$.
- Montrer que si y est un carré alors $y^{(p+1)/4}$ est une racine carrée de y. Quelle est la complexité de ce calcul en la taille de p ?
- Montrer que si on sait résoudre en temps polynomial le DLP dans G alors on peut calculer en temps polynomial la parité de x.
- Inversement, supposons qu'on dispose d'un oracle donnant la parité de x (i.e. une boîte noire calculant en temps polynomial la parité de x à partir de y). En déduire un algorithme polynomial en la taille de p pour le calcul de x. On s'inspirera des techniques d'exponentiation rapide.

Parier à distance

Alice et Bob désirent jouer au jeu suivant à distance : Alice choisit un bit (0 ou 1). Puis Bob choisit également un bit et le transmet à Alice. S'il est égal au bit d'Alice, Alice déclare que Bob a gagné sinon elle lui dit qu'il a perdu. Trouver un protocole pour que ce jeu soit possible (i.e. Bob ne doit pas pouvoir deviner le choix d'Alice et Alice ne doit pas pouvoir tricher en changeant son choix après le choix de Bob). Pour se faire, on pourra utiliser un générateur pseudo-aléatoire et une bonne fonction de hachage.

Questions de sécurité

1. Alice utilise un protocole RSA basique avec un module N et un couple de clés (e, d) sûrs. Par erreur, elle publie sa clé privée d. Elle choisit alors de générer un nouveau couple de clés (e', d'). *Le protocole est-il sûr avec ce nouveau couple ?*

2. On dit qu'un schéma de chiffrement à clé publique est *plaintext-aware* s'il est impossible d'obtenir un chiffré valide (c'est-à-dire accepté par le protocole mais qui peut très bien provenir d'un message clair dénué de sens) sans connaître le texte clair correspondant. On dit qu'un schéma de chiffrement est *non-malléable* si étant donné un chiffré c valide il est impossible de produire un chiffré $c' \neq c$ valide sans connaître les textes clairs correspondants à c et c'. *le chiffrement RSA est-il plaintext-aware ? Est-il non malléable ?*

Transfert sans mémoire

On souhaite un protocole réalisant l'opération suivante : Alice veut transférer un bit b à Bob de telle sorte que Bob reçoit effectivement ce bit dans la moitié des transferts mais Alice ne sait jamais si Bob l'a reçu ou non. On propose le protocole suivant :

1. Alice choisit deux grands premiers p, q congrus à 3 modulo 4 et calcule $N = pq$. Elle choisit également un élément aléatoire $x \in [1, \ldots, N-1]$ pair si $b = 0$ et impair si $b = 1$. Elle chiffre x sous RSA avec le module N et un couple de clés (d, e). On appelle y le chiffré qu'elle transmet à Bob ainsi que N. *Quelle est la taille de N pour laquelle la primitive RSA est sûre ? Pourquoi ?* On suppose N de cette taille. *Est-il difficile pour un adversaire de trouver le parité de x ?*

2. Bob choisit aléatoirement $a \in [2, \ldots, N-1]$ et envoie $w \equiv a^2 \pmod{N}$ à Alice.

3. *Montrer qu'Alice peut calculer en temps polynomial les racines carrées de w mais pas Bob.* Elle en choisit une aléatoirement et la retourne à Bob.

4. *Montrer qu'avec une probabilité au moins* $1/2$ *Bob peut alors déterminer le bit b* (on suppose bien sûr que Bob sait que x pair (resp. impair) veut dire $b = 0$ (resp. $b = 1$).

5. *Est-ce qu'Alice ne sait effectivement pas si Bob a reçu le message ? Pensez-vous que la probabilité pour Bob est exactement* $1/2$ *?*

Un algorithme de signature

On considère le protocole de signature suivant. Soit p un grand nombre premier tel que le DLP dans $(\mathbb{Z}/p\mathbb{Z})^*$ soit difficile. Soit $x < p$ la clé privée, g un élément primitif et $y \equiv g^x \pmod{p}$ la clé publique. On suppose également donnée une fonction de hachage $h : \{0,1\}^* \to S$ où $S = \{t \in [1, \ldots, p-2],\ \text{pgcd}(t, p-1) = 1\}$. Pour signer un message $M \in \{0,1\}^*$ on calcule $h = h(M)$ puis $z = x/h \pmod{p-1}$. L'appendice de la signature est $s \equiv g^z \pmod{p}$. Pour vérifier la signature on vérifie si $s^h \equiv y \pmod{p}$.

1. *Donner les conditions sur p pour que le DLP $y \equiv g^x \pmod{p}$ soit difficile.*

2. *Montrer que si les opérations sont correctement effectuées alors la signature est valide.*

3. *Montrer qu'un adversaire peut produire une falsification sélective.*

Bibliography

[BHKKR99] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, P. Rogaway, UMAC: fast and secure massage authnetication, Advances in cryptology - crypto '99, Lecture Notes in Computer Science **1666**, Springer-Verlag, (1999), 216-233.

[BJN00] D. Boneh, A. Joux, P. Nguyen, Why textbook ElGamma and RSA encryption are insecure, Advances in Cryptology-Asiacrypt '00, **1976** in Lecture Notes in Computer Science, Springer-Verlag, (2000), 30-43.

[CGH04] R. Canetti, O. Goldreich, S. Halevi, the random oracle methodology revisited, Journal of ACM, **51**, (2004), 557-594.

[CJ03] J.H. Cheon, B. Jun, a polynomial time algorithm for the braid Diffie-Hellman conjugacy problem, Advances in cryptology- Crypto 2003, lecture notes in computer science **2729**, Springer-Verlag, (2003), 212-225.

[Del03] J.-P. Delahaye, la Belle au bois dormant, la fin du monde et les extraterrestres, Pour la Science **309**, Juillet 2003, 98-103.

[FO99] Fujisaki E., Okamoto T., how to enhance the security of public key encryption at minimum cost, Public Key cryptography PKC '99, Lecture notes in computer science **1560**, Springer-Verlag, (1999), 53-68.

[FOPS01] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, RSA-OAEP is secure under the RSA assumption, In advances in Cryptology, Crypto 2001, **2139**, Springer-Verlag, 2001, 260-274.

[Goz97] I. Gozard, *Théorie de Galois*, Mathématiques 2ième cycle, Ellipses, 1997.

[HW79] G.H. Hardy, E.M. Wright, an introduction to the theory of numbers, Oxforf Science Publications, fifth edition 1979.

[Knu98] D.E. Knuth, the art of computer programming, vol. II, seminumerical alorithms, Addison-Wesley, 1998.

[KJJ99] P. Kocher, J. Jaffe, B. Jun, differential power analysis, advances in cryptology-crypto '99, lectures notes in computer sciences **1666**, Sprinver-Verlag, 1999.

[MW99] U. Maurer, S. Wolf, The relationship between breaking the Diffie-Hellman protocol and computing the dicrete logarithm, SIAM Journal on Computing, **28**, (1999), 1689-1721.

[Mon06] R. Montenegro, Title: A canonical path approach to bounding collision time for Pollard Rho algorithm, available on `http://arxiv.org/abs/math/0611586`.

[Odl90] A. Odlyzko, The rise and fall of knapsack cryptosystems, in Cryptology and Computational Number theory, Proc. Symposia in Applied Maths **42**, 1990.

[QG90] J.-J. Quisquater, L. Guillou, How to explain Zero-knowledge protocols to your children, advances in cryptology-crypto '89, lectures notes in computer science **435**, Springer-Verlag, 1990, 628-631.

[SHO97] V. Shoup, lower bounds for discrete logarithms and related problem, Eurocrypt 1997, Lecture Notes in computer science **1233**, Springer-Verlag, 1997, 256-266.

[FL05] Frey, Lange, crypto elliptique