

Lecture Note 1

Date: 11.10.2007

Contents

1	Introduction	3
1.1	Presentation	3
1.2	Motivations : understanding and defining security	3
1.2.1	Some typical problems of day-to-day life	3
1.2.2	Detailing some concrete examples	4
2	At the heart of cryptography	4
2.1	Information hiding	4
2.2	Two kinds of encryption schemes	4
2.3	Historically speaking	5
2.4	On substitution schemes	6
2.4.1	Monoalphabetic substitution	6
2.4.2	The main weakness of substitution	7
2.4.3	Homophonic substitution ciphers	7
2.4.4	Polyalphabetic substitution and Vigenère cipher	8
2.5	One-time pads : Vernam cipher	9
2.6	Transposition ciphers and composition	9
2.7	Today's standards	10
2.7.1	DES, 1993	10
2.7.2	AES, 2002	11
2.8	Some famous schemes	13
2.8.1	Primes, integer factoring and RSA	13
2.8.2	ElGamal Encryption Scheme	13
3	Exchanging keys	14
3.1	First drafts: how to exchange keys securely	14
3.1.1	The main idea	14
3.1.2	In practice: implementation and logical attacks	14
3.1.3	Different ways of attacking a protocol	15
3.2	Needham-Shroeder Protocol	16
3.2.1	The first version of the protocol	16
3.2.2	The man-in-the-middle attack and Lowe's contribution	16

3.2.3	Another flaw	17
3.2.4	Conclusion : proving protocols secure	17

1 Introduction

1.1 Presentation

Over the years, the development of communication tools has led to a tremendous lack of secrecy. From the apprehension of messengers to the opening of e-mails, the need for encoding information that shall not fall into the wrong hands has become a priority in defense policies. Thus, although Julius Caesar was already using cryptographic methods, cryptography and cryptanalysis have nevertheless a promising future.

This course was first given in 2007 to M2R students at University Joseph Fourier in Grenoble. It aims at presenting basic and essential notions, techniques and models used in cryptography and security. It also deals with some topics in verification of cryptographic protocols used on small programs.

Security is linked to several domains, such as mathematics, cryptography, operating system or network study. Choosing a way to deal with security issues often does not only consist in choosing a cryptographic scheme, that is to say, a way of ciphering information. For example, the way information, even ciphered, is dealt with, is essential too: secure transmission can be completely compromised by an uncareful user ! Consequently, only the first part of the course is going to actually give description of ciphering methods, as long as brief historical recalls about cryptography. The interested reader is provided bibliographical details in the end of this first course.

1.2 Motivations : understanding and defining security

1.2.1 Some typical problems of day-to-day life

Our modern ways of communication provide a lot of examples of critical situations involving security issues. Communicating by phone, e-mail, or fax, one requires confidentiality and integrity of exchanged information. Getting connected to a bank via the Internet and performing transactions demands confidentiality too, but also prevention of false transactions, and the impossibility for a user to repudiate a transaction - that is to say once a user has paid, he cannot deny he's done it. Digital payment systems, e-voting systems, or DRM (Digital Rights Management) are other practical examples of the extended need for well-thought-out security policies.

Nevertheless, it is not generally easy to translate this need in practically achievable objectives. Neither is concepting systems satisfying those objectives. Defining standard security properties traditionally required can help in some way. Common properties fall into three categories: *confidentiality* or *secrecy* properties - that is, there must be no improper disclosure of information - *integrity* properties - the information must not be modified improperly - and *availability* properties - this is the management of rights of access to one or another functionality. The thing is that, this is not specific enough, for the main problem raises when trying to define the term "improper".

Some other interesting properties can be listed. Authentication, for example, raises quite big problems when it comes to security matters. Non-repudiation, or accountability,

can be required, or its inverse in some way, plausible deniability. Speaking of privacy, one can also think of anonymity, pseudonymity (this is anonymity plus traceability), or even data protection, in the sense of restricting its use...

1.2.2 Detailing some concrete examples

Let's get into a little details for a couple of examples. First, try and figure out what security properties Internet banking requires. Authentication processes for clients and servers, non-repudiation of transactions, integrity of the accounts, of the other users data, secrecy of customer data, and availability of logging. A *security policy* consist in the conjunction of all these security goals. Take e-voting as a second instance. A secure e-voting system should guarantee integrity of votes, privacy of voting information, and the availability of the system during the voting period (and only then). Furthermore, only registered voters should have the possibility to vote, and each of them must do it at most once...

In practice, it can be very tricky to formulate properly which criteria are to be considered. And, obviously enough, every possible leak must be investigated. Unconvinced readers can try to list the security requirements an international airport should fulfil.

2 At the heart of cryptography

2.1 Information hiding

One can get a little confused when it comes to defining precisely cryptography, cryptology, cryptanalysis... Trying to communicate secretly can be achieved by simply hiding the message - this is *steganography* - while scrambling it is actually *cryptography*. Cryptographic schemes fall into two parts : substitution algorithms, that can in turn be divided in codes (when whole words are replaced by others) and ciphers (involving letters replacement), and transposition algorithms, that modify the word structure with respect to one or a group of permutations - you can think of it as a kind of anagram, details are given a below.

Moreover, *cryptanalysis* consists in finding methods to obtain the meaning of encrypted information. Typically, this involves finding a secret key. In the end, cryptography and cryptanalysis are the two sides of *cryptology*, which is the study of secret writing. Consequently, telling the history of cryptology amounts to recounting the different stages of the unceasing battle that has been raging between cryptographers and cryptanalysts.

2.2 Two kinds of encryption schemes

Encryption is one of the most important cryptographic primitive. To define it properly, one needs three algorithms making up an encryption scheme. The first algorithm is used to generate one or several keys, the second one ciphers messages, and the last one decipher them. A message x , whose meaning can be clearly understood, is called a *plaintext*,

while, once it has been ciphered, it will be referred to as a *ciphertext*. Besides, according to whether you want everybody or just chosen people to be able to send you ciphertexts, you won't choose the same type of schemes.

A *public-key encryption scheme*, more commonly known as an asymmetric encryption scheme, works with a pair of keys, denoted by (pk, sk) . The public key pk can be used by anyone to encrypt a message x , and thus create the encryption of the message x under the key pk : $y = \mathcal{E}_{pk}(x)$. However, to decrypt y and access to the original message x , one needs the secret key sk , whose knowledge is obviously not supposed to be spread. The decryption algorithm allowing to get x back when knowing y is denoted by \mathcal{D}_{sk} . We logically impose on encryption and decryption to be such that $\forall(pk, sk), \mathcal{D}_{sk} \circ \mathcal{E}_{pk} = Id$. Asymmetric encryption schemes are often likened to letter boxes: anyone can deposit a message in the box, and people who have the key can read the letters.

On the contrary, you may want your fellow students to be the only ones who can send you a message or read yours. This time, you need to use a *symmetric encryption scheme*. The key k , used to cipher *and* decipher messages, has then to remain secret to your adversaries, say, students from other universities or teachers. As before, a ciphertext y is denoted by $y = \mathcal{E}_k(x)$, and we require the deciphering of a ciphered message to give us back the original message: $\forall k, \mathcal{D}_k \circ \mathcal{E}_k = Id$. Instead of letter boxes, this time, symmetric encryption schemes can be identified to safes.

2.3 Historically speaking

In the beginning, to communicate secretly, people used mostly steganography. A well-known example is that Ancient Greeks used one of their slaves, by tattooing the message on his skull and waiting for his hair to cover it, to send a secret message to their allies abroad. It was pretty slow, but eventually worked. Nevertheless, the limits of steganography are quickly reached: your adversary only has to find your hiding place and he can read your text.

People then started to scramble texts, in order to complicate their comprehension. The Ancient Greek scytale may have been the first device used to generate a ciphertext. It consists of a cylinder with a strip of leather wound around it on which the message is written horizontally. Once unrolled, the strip reads a meaningless sequence of letters. The recipient has to have a cylinder of the same diameter to be able to read the original message properly. The first substitution ciphering algorithm is also believed to date back to this period. It is usually named after Julius Caesar. The principle is quite simple, it is a monoalphabetic substitution, it's described precisely below.

Several other people deserve to be remembered, for they paved the way for important discoveries. We just cite some of them here. Around 1460, Leon Alberti devised a cipher wheel, to facilitate the use of substitution processes. Enigma, the first mechanic ciphering device was designed between WWI and WWII. Based on the electric connections in rotating rotors, this method was believed to be unbreakable until the Polish mathematician Rejewski found cycles playing a fundamental role in the system. The interested reader can

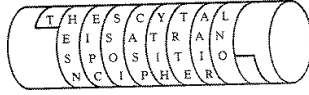


Figure 1: A scytale

find in Singh’s book a pretty detailed history of cryptography. The historian David Kahn has also extensively written about cryptography and its influence on the rest of the world, in particular the role it played during WWII.

2.4 On substitution schemes

Substituting a letter to another is one of the two fundamental ideas used to cipher a message. In fact, it is around 2,000 years old. As any other proceeding, it was soon broken by cryptanalysts, but kept evolving to meet new security requirements.

2.4.1 Monoalphabetic substitution

Let \mathfrak{S} be the set of all permutations (or one-to-one function) on the usual alphabet. Once a permutation $\sigma \in \mathfrak{S}$ is fixed, a substitution algorithm simply proceeds by replacing each letter L of the plaintext by its image by the permutation $\sigma(L)$:

$$m = m_1 \dots m_n \mapsto c = \sigma(m) = \sigma(m_1) \dots \sigma(m_n)$$

To decrypt c , you just have to compute the permutation’s inverse σ^{-1} , and apply it the same way:

$$c = c_1 \dots c_n \mapsto \sigma^{-1}(c) = \sigma^{-1}(c_1) \dots \sigma^{-1}(c_n) = m$$

This method is called monoalphabetic substitution, for the ciphered alphabet is the same during the whole process. Caesar cipher is a special class of monoalphabetic substitutions, since the permutations are chosen from the set of 0 to 25 letter gaps on the right. For instance, try and figure out what KHOOR ZRUOG actually stands for. If you suppose it is an example of Caesar’s ciphering method, you can try all 26 possible ciphered alphabets. But if you’re more of the lazy kind, you can notice that KHOOR contains two letters O, and that not every letter in the English alphabet can be doubled in a meaningful word. Thus, you’ve far less possibilities to try. A little more intuition will allow you to quickly find out ”HELLO WORLD” is the plaintext. A three letter move on the right has been applied to each letter.

The ciphered alphabet can also consist of numbers instead of letters. As an example, 2-25-5 2-25-5 stands for BYE BYE if each letter of the classical alphabet is replaced by its rank.

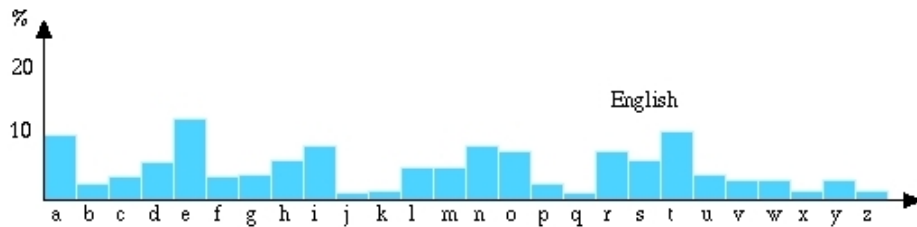


Figure 2: Frequency repartition graph of letters in English

2.4.2 The main weakness of substitution

However, even if the choice of the ciphered alphabet offers $26!$ possibilities, monoalphabetic substitution suffers from a main drawback. Even if blanks between words are suppressed, you can count how many times each letter appears in the ciphertext and draw a graphic representation of these frequencies. By comparing it to the usual frequency repartition graph of letters in English, you can recognize patterns that allow you to restrict possible ciphered alphabets. If you know a Caesar algorithm was applied to the plaintext, it gets even simpler: you just have to find what letter stands for E, A or T (most frequently used letters in English) and it gives you the whole alphabet!

Obviously, frequency analysis presupposes three things: you have to have a long enough sample to practise it, you must know which language the plaintext was written in, and the plaintext must be written "in classical English", for if it's too weird, your frequency analysis won't be relevant. The usual example given to illustrate that is the plaintext 'From Zanzibar to Zambia and Zaire, ozone zones make zebras run zany zigzags', where there clearly is an abnormally high frequency of Z. That said, if you get that message knowing it's full of Z's, frequency analysis helps anyway. The concept of using any little piece of information you have about the plaintext is fundamental. To break Enigma, Rejewski mainly used the simple fact that the key was encrypted twice in the beginning of each message. Turing went even further in extrapolating: he merely supposed the probable occurrence of a special word in the plaintext, and tried to match it at any possible place.

2.4.3 Homophonic substitution ciphers

The priority is now to deprive cryptanalysts of using frequency analysis. A first solution can be found in the use of homophonic substitution ciphers. The basic principle this time is to allow different occurrences of the same letter to be ciphered by different symbols. To each a in the alphabet \mathcal{A} , we associate a set $H(a)$ of possible symbols (or even strings of symbols), such that $a \notin H(a)$. Each time you want to cipher a , you pick a random string from $H(a)$. To decrypt a string c , your adversary's task has become more difficult. He has to figure out which (and how many) symbols belong to $H(a)$. Thus, the key is here $\{H(a), a \in \mathcal{A}\}$.

Let's give an example of this method. Let $\mathcal{A} = \{a, b\}$, $H(a) = \{00, 10\}$ and $H(b) =$

{01, 11}. The plaintext ab can be ciphered four different ways: 0001, 1001, 1001 or 1011. Even though frequency analysis is insufficient, it does not become irrelevant; and major drawbacks of this cipher are a significant data expansion and additive calculus needed to decipher texts.

2.4.4 Polyalphabetic substitution and Vigenère cipher

This time, instead of allowing different symbols to represent the same letter, we change the whole ciphered alphabet after each letter. Vigenère's idea was to complicate Caesar's algorithm by introducing a key-word. Let us take the example of the key-word 'mokona' used to cipher 'tsubasa chronicle'. The first step is to write the key-word above the plaintext cyclicly in order to assign a letter of the password to each letter of the plaintext:

$M O K O N A M O K O N A M O K O$
 $T S U B A S A C H R O N I C L E$

Once this is done, to cipher a letter, you use Caesar cipher, considering that the image of A is the password letter above your letter. Here, we first have to cipher the letter T, using a Caesar ciphered alphabet which ciphers A in M. The first letter of the ciphertext is thus F, then G and E... You may use the figure below to help you find the right result. It is called a Vigenère square. This cipher is quite good, for it substantially complicates frequency analysis, while the only thing the partners exchanging the message have to remember is the password.

More formally, a *polyalphabetic substitution cipher* is a block cipher with block length t

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 3: A square of Vigenère

over alphabet \mathcal{A} , where:

- the key space \mathcal{K} consists of all ordered sets of t permutations over \mathcal{A} , denoted (π_1, \dots, π_t) ;
- the encryption of $m = m_1 \dots m_t$ under key $e = (\pi_1, \dots, \pi_t)$ is $\mathcal{E}_e(m) = \pi_1(m_1) \dots \pi_t(m_t)$;
- the decryption key for e is $d = (\pi_1^{-1}, \dots, \pi_t^{-1})$.

Using these notations, Vigenère ciphers become ciphers using as key the number sequence $e = e_1 \dots e_t$, which defines the substitutions $\pi_i(a) = (a + e_i) \bmod 26$, where we have used the traditional map assigning its rank to each letter of the alphabet.

2.5 One-time pads : Vernam cipher

A fine way to achieve unconditional security is to use one-time pads. It is a cipher defined over $\{0, 1\}$, such that if the plaintext is $m = m_1 \dots m_n$, encrypted under key $k = k_1 \dots k_n$, the corresponding ciphertext is $\mathcal{E}_k(m) = (m_1 \oplus k_1) \dots (m_n \oplus k_n)$, where \oplus stands for the bitwise XOR (exclusive or). To decipher it, you just have to apply XOR again: $\mathcal{D}_k(c) = (c_1 \oplus k_1) \dots (c_n \oplus k_n)$.

As long as each key is only used once, this is indeed a very secure algorithm to cipher information with. As every key sequence is equally likely, plaintexts are uniformly distributed over bitstrings of a given length. Of course, using twice the same key compromises the whole thing: ciphering m and m' using k , you give away $m \oplus m'$ to your adversary for free. And even worse, say your message is 100 times the length of your possible keys, so that you have to cut it into 100 pieces to cipher it. If you choose the same key 100 times, and the adversary knows one sequence of your message (from...to of mails, probable words, etc.), he can obtain the key itself, and your whole message is transparent to him.

Obviously enough, these ciphers, no matter how secure they are, are extremely difficult to put to work, for partners must agree on keys before using it, and keys must be as long as the plaintext! The problem is now different - can we exchange keys securely ? - but essentially as hard as the initial one. Protocols are discussed a little below.

2.6 Transposition ciphers and composition

Substitution is not the only basic idea from which originated ciphers. Transposition is the second fundamental concept used in cryptosystems. The principle is as simple as substitution, this time you mix letters instead of replacing them one by another. Formally, for a given length n , let \mathfrak{S}_n be the set of permutations on $\{1, \dots, n\}$, and let m be a plaintext of length n . Then, for each key permutation $\sigma \in \mathfrak{S}_n$, the encryption algorithm is $\mathcal{E}_\sigma(m) = m_{\sigma(1)} \dots m_{\sigma(n)}$. The set of all such transformations is called a *transposition cipher*. Decryption is obtained by applying $\mathcal{E}_{\sigma^{-1}}$.

An example of an old transposition cipher was given in section 2.3, with the scytale and its leatherstrip. The thing is, frequency analysis can easily reveal whether a transposition

cipher was used on the message. Then, your adversary can decrypt your message by using frequency analysis on diphthongs or probable words. You could think of applying a second time a different transposition to increase the security of your cipher, but to the adversary, things remain the same no matter how many transpositions you apply, it is always like if only you transposed the plaintext once. This is also true for substitution ciphers.

However, composing ciphers is an interesting idea if you compose different kinds of ciphers. The ciphering algorithm is therefore going to become so complicated to calculate by hand that you might need an automated device to help you. This is essentially the idea behind the Enigma machine the Germans used during WWII: there were three rotors placed in one of the 6 possible orders, that had electrical connections within to substitute a letter to another. But once you typed one letter, the first rotor moved one step forward, while the second moved two and the third three steps forward... This device was long believed to be unbreakable, but, as stated before, simple algebraic considerations about cycles and human misuse were enough for Rejewski and his fellow cryptanalysts to break the cipher.

2.7 Today's standards

The National Security Agency (NSA), the American equivalent of a bureau des chiffres, imposed after WWII a standardisation of ciphers. Amongst many proposals, two cryptosystems were chosen, named Data Encryption Standard (DES) and AES (Advanced Encryption Standard). Let's look a little inside those, before considering them as big black boxes operating magically...

2.7.1 DES, 1993

DES is a block cipher ciphering 64 bits at a time, using 56 bit keys expressed as 64 bit numbers (8 bits are added by parity checking). It is based on Feistel cipher, which is applied 16 times with different keys. The diagram below represents what happens during one round. The whole algorithm proceeds as follows: it first performs a permutation σ on the plaintext, then splits the result in two parts L_0 and R_0 , the key scheduler forwards the first key K_1 and everything is given to the first Feistel cipherbox, which results in a pair of messages L_1 and R_1 , and a new key K_2 , and so on... sixteen times. In the end, the initial permutation's inverse is applied before returning the final answer. No one except NSA actually knows why 16 blocks are needed, except that the original proposal used 5 boxes and was broken.

DES is believed secure, but its security properties are not clear. No reduction or security proof is known, and the main attack consists of an exhaustive search that takes 7 hours to a 1 million dollars computer to achieve, and 7 days to an average 10 thousand dollars one. No one has ever submitted a mathematical attack to DES, or he or she was then hired by NSA and silenced... The only currently known objection to DES is that

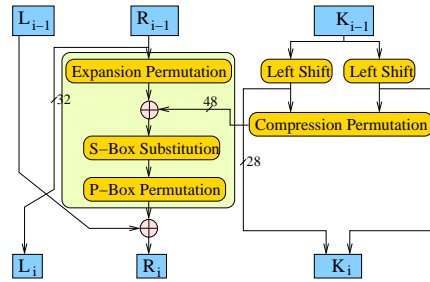


Figure 4: In a Feistel cipherbox

cryptanalysts can reduce the key space from 2^{56} to 2^{43} possibilities. Apart from that, it is a matter of faith.

2.7.2 AES, 2002

AES was the second block cipher to be approved by US government. It is a very popular cryptosystem, which was designed by two Belgian cryptographers. It performs ciphering on 128 bit blocks with keys of length 128, 192 or 256 bits. As DES, it does a certain number of rounds, each using substitutions and transpositions, along with a key scheduler. First, it applies a key expansion algorithm using Rijndael's key scheduler, and applies `AddRoundKey`, that is, each byte of the state is combined with the round key, and each round key is derived from the cipher key using the scheduler. Then, one round consists of:

- `SubBytes`: a non-linear substitution step during which each byte is replaced with another according to a lookup table,
- `ShiftRows`: a transposition step during which each row of the state is shifted cyclically a certain number of steps,
- `MixColumns`: a mixing operation which operates on the columns of the state, combining the four bytes in each column,
- `AddRoundKey` (described above).

Eventually, the final round spares `MixColumns` and forwards the final answer. The details are, like for DES, not particularly clearly accounted for, and the only widely known attack consists of attacking implementations that inadvertently leak information about the key. Consequently, even if unproved, AES security is strongly believed to be true.

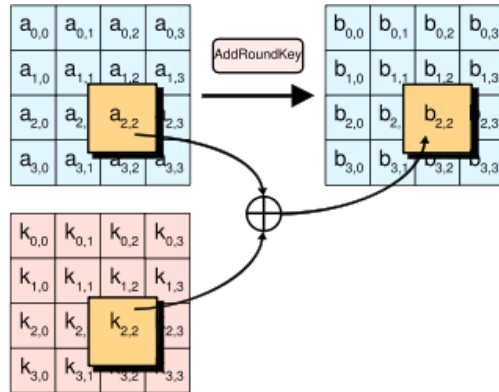


Figure 5: AddRoundKey

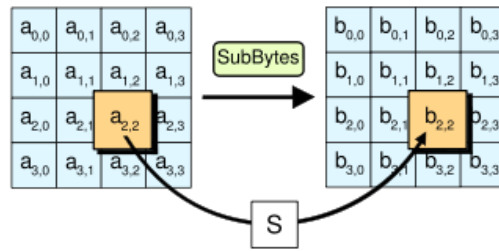


Figure 6: SubBytes

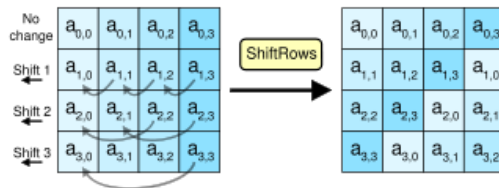


Figure 7: ShiftRows

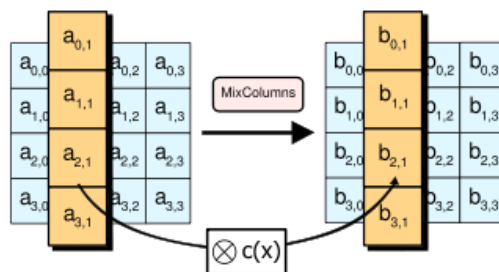


Figure 8: MixColumns

2.8 Some famous schemes

2.8.1 Primes, integer factoring and RSA

Factoring large integers is difficult, whereas computing the product of a pair of large primes is quadratic in the size of your primes representation. In this assertion lies the security of RSA. RSA stands for Rivest, Shamir and Adleman, the three mathematicians that designed it. It is an asymmetric encryption schemes which proceeds as follows:

- Let p, q be two large primes and n their product. The space of possible messages is $\{0, \dots, n-1\}$, so let M be one of these integers.
- Let $e \in \{0, \dots, n-1\}$ such that $e \wedge (p-1) = 1$ and $e \wedge (q-1) = 1$. If ϕ denotes as usual Euler's function, $\phi(n) = (p-1)(q-1)$, $e \wedge \phi(n) = 1$ so e is invertible mod $\phi(n)$: $\exists d/ed \equiv 1 \pmod{\phi(n)}$. The ciphered message is thus $C = M^e$.
- To decipher it, one needs to know d . We have $C^d \equiv M^{ed} \pmod{\phi(n)}$, and $ed = 1 + k\phi(n)$, for some integer k , so $C \equiv M^{k\phi(n)}.M \pmod{n} \equiv (M^{\phi(n)})^k.M \pmod{n}$. But $M^{\phi(n)} \equiv M \pmod{n}$, because $M^{p-1} \equiv 1 \pmod{p}$ and $M^{q-1} \equiv 1 \pmod{q}$ by Fermat's first theorem, and the Chinese remainder theorem gives the conclusion.
- The public key is (n, e) , and the secret key is (n, d) ; the security properties of RSA lie in the intractability of the discrete logarithm calculus.

2.8.2 ElGamal Encryption Scheme

ElGamal encryption scheme is an asymmetric scheme which is based on an hypothesis derived from Diffie-Hellman agreement. The key generator works as follows:

- Alice samples a cyclic group G of order q and group generator g .
- She picks $a \in (\mathbb{Z}/(q-1)\mathbb{Z})$ at random.
- She computes $h = g^a$, and sends G, q, g and h , the public key. She keeps the value of a secret.
- Bob knows the public key h , and wants to send Alice a message m . He picks $b \in (\mathbb{Z}/(q-1)\mathbb{Z})$ at random and computes $(u = g^b, v = m.h^b)$ and sends it to Alice.
- Alice computes $m = v.(u^a)^{-1}$, and she can read the message.

Obviously, using twice the same value of b to cipher m_1 and m_2 allows any eavesdropper to compute $m_1.m_2^{-1}$. Moreover, an important drawback of this cipher is that the ciphertext is twice as long as the plaintext... Besides, the security of ElGamal scheme depends on the properties of the underlying group G as well as any padding scheme used on the messages. If the computational Diffie-Hellman assumption holds the underlying cyclic group G , then the encryption function is one-way - that is, roughly, one cannot infer anything about

a function's argument x knowing only $f(x)$. If the decisional Diffie-Hellman assumption (DDH) holds in G , then ElGamal achieves semantic security... The security of ElGamal has been extensively studied, and an interested reader will find many detailed references on the Internet.

3 Exchanging keys

No matter how secure the ciphers you use can be, you and your partners still have to exchange keys to be able to understand each other. This is the main drawback of secure encryption schemes. Obviously, you cannot send your partner the key through the usual unsecure channels, or you would make the whole ciphering process useless. You may use postmail, if sending the message is not an emergency. And if you are sure adversaries cannot read your mail. The exchange of keys is the least secure stage of the whole communication process. To address this problem, a lot work has been done during the second half of the twentieth century, mainly thanks to two faithful men nobody wanted to listen to: Withfield Diffie and Martin Hellman. This section of the course won't recount the whole story of key-exchange protocols, for it would be too long and it is not the point here. We just want to illustrate the problems raising when studying protocols security and trying to find logical attacks. The protocol we choose to study in details below was originally designed by Needham and Schroeder.

3.1 First drafts: how to exchange keys securely

3.1.1 The main idea

Let us think simply in terms of informal exchanges. Alice and Bob want to exchange a message securely. If Alice sends Bob a locked box, he won't be able to open it. If she sends him the key, she can't be sure he will be the one to use it on the box. The trick, then, is to make Bob lock the box with his own padlock and send it back to Alice. When she gets it back, Alice can't open it, but she doesn't need to: she knows what is in it. She just has to remove her padlock and send it back to Bob. Then, receiving a box which padlock he has a key to, Bob can open it and read the message Alice initially put in it. This way, Alice and Bob have exchanged a message entirely safely, provided their padlocks were secure. From now on, we assume the encryption schemes we use are *perfect*, that is, an intruder cannot read a message if it is ciphered with a key he does not know.

3.1.2 In practice: implementation and logical attacks

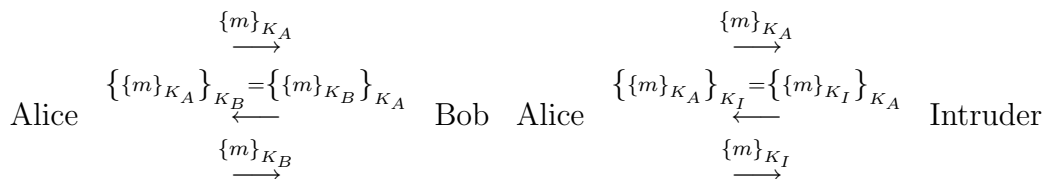
Having the first idea wasn't that easy, and one can think we are nearly done once we thought of our little trick. Unfortunately, this is not true. Implementing the simplest

protocols can raise unsolvable problems, and the tiniest flaws in the implementation can ruin the security of a protocol. Consequently, solutions must be chosen carefully.

We place our study in the symbolic model. This model originates in the work of Dolev and Yao in the early 80's. It is described in the second lecture. Here, the reader just needs to know we consider messages m as algebraic terms (and not bitstrings), on which we can perform two operations: ciphering ($\{m\}_K$) and concatenation ($\langle m_A, m_B \rangle$). Finally, we recall we work under the perfect encryption hypothesis.

In the exchange protocol described above, Alice must be able to remove her padlock when she receives the box from Bob. In practice, this means the encryption must be commutative: $\{\{m\}_{K_A}\}_{K_B} = \{\{m\}_{K_B}\}_{K_A}$. Otherwise, this is like assuming Alice receives a box locked by Bob's padlock, which contains the box she first sent. And our trick is ruined. This is a high request to an encryption scheme, but fortunately, we do know some instances filling in this condition. RSA is one of them.

Another problem to address is authentication. Look at the following figure. The problem is that, the way we implemented our protocol, Alice cannot be sure the padlock is Bob's. If an intruder kills Bob and receives the box in his place, nothing in our protocol can alert Alice. The other way round, an intruder can start a protocol with Bob without him being aware he isn't talking to Alice...



3.1.3 Different ways of attacking a protocol

We have just seen one way an intruder can interfere with the protocol without honest participants noticing. There are several other possible attacks a dishonest instance can perform. Here is a non-exhaustive list:

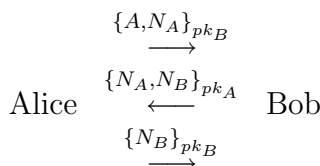
- man-in-the-middle attack : an intruder can intercept messages sent between honest participants and replace them by what he wants (an example of this attack is studied below);
- replay attack: the dishonest player records your transmissions and re-introduces them later to mislead one of the honest instances;
- reflection attack: an information is sent back to the original transmitter. This can for example lead to an honest player deciphering information he's just ciphered;
- oracle attack: this is when an intruder simulates honest answers to use the ciphering and deciphering possibilities of honest instances, thus using honest participants as deciphering or ciphering oracles;

- type flaw: the dishonest participant can replace messages or parts of them by another type of information (e.g. replacing a key by a name).

3.2 Needham-Shroeder Protocol

3.2.1 The first version of the protocol

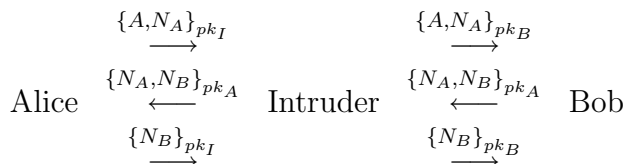
In 1978, Needham and Schroeder came up with a quite relevant protocol that could be used between two partners to exchange a key they would keep during the rest of the session. Let us first define the concept of cryptographic nonce: a nonce is simply a number or bit string used only once. It is often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks. From now on, we denote nonces N , with in subscript the name of the player that generated it. Besides, pk_{name} denotes a public key, subscribed by the owner of the matching secret key. Now, we can write the protocol.



Here, we see this implementation apparently prevents an intruder to play one's role without the other honest participant being aware of it. Indeed, the attack displayed previously is ruined by the presence of the nonces, that allow a player to check whether his or her nonce was deciphered properly. Recall that we assumed ciphers were perfect, so that one can decipher a message only if he possesses the matching secret key.

3.2.2 The man-in-the-middle attack and Lowe's contribution

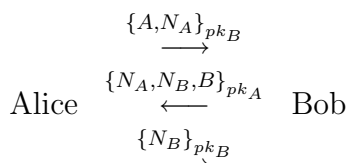
However, nonces turned out insufficient to ensure a proper authentication of a partner. Nevertheless, it took 17 years to Lowe to discover, thanks to an automatic verification tool, that there existed an efficient attack. This latter was named man-in-the-middle attack, for the adversary has to stand between both partners, intercept their transmissions and mix them up. Below, you can see how the intruder can actually make Bob believe he shares a key with Alice.



When we were studying encryption schemes, we saw that people are used to relying on them until they are publicly proven insecure. It obviously seems dangerous, since an

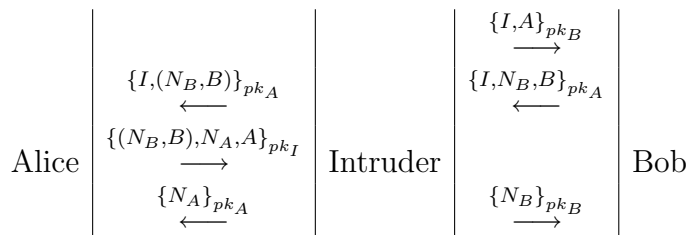
adversary won't let you know the flaws in your primitives if he finds out about them. The same thing seems to happen with protocols, which accounts for cryptologists now trying to design automated verification tools to prove security properties of protocols indubitably. These tools are supposed to search for attacks in an exhaustive way: if they can't find any attack, it is not that they *didn't think* of the right one!

Anyway, once he found out about the man-in-the-middle attack, Lowe corrected the protocol so that it would be impossible to perform. He thought that, if Bob added his name to the plaintext he ciphers, the intruder wouldn't be able to decipher it and to replace it by I , which, under the perfect encryption hypothesis, is true. Needham-Schroeder-Lowe protocol is therefore the following:



3.2.3 Another flaw

Unfortunately, this correction turned out insufficient! Indeed, there exists another kind of attacks to the corrected protocol, using a type flaw in the design of the protocol. First, the adversary sends Bob $\{I, A\}_{pk_B}$, making him believe I was Alice's nonce. The intruder then receives from Bob $\{I, N_B, B\}_{pk_A}$, and forwards it to Alice, making her believe (N_B, B) is his nonce. Here is the type flaw. Alice, playing fairly, answers $\{(N_B, B), N_A, A\}_{pk_I}$, and she has thus deciphered (N_B, B) for free...



Type systems do not allow random nonces to be used in place of non-random terms such as names. Consequently, typing transmitted information is a useful way of getting aware of transmissions being tampered with. Type systems are an active domain of research nowadays, for they would provide an efficient tool to be used in automated verification programs.

3.2.4 Conclusion : proving protocols secure

Until recently, the level of security of a cryptographic scheme, that is to say, the way the information is encoded, was used to being assessed in a practical way. Indeed, a scheme was considered secure until an efficient attack was discovered that broke it completely. This method is obviously far from being rigorous. How can one rely on a scheme whose secrecy

is only guaranteed by the fact that no efficient attack has ever been exhibited ? What if an attack had been found by the precise adversaries that shall not learn this information ? Obviously, these adversaries would not have told the world they could break a scheme, whereas it would potentially allow them to learn interesting information... There lies the need for more reliable security, or rather for more solid ways to *prove* the level of security of a scheme. These are the main reasons that gave birth to *provable cryptography*.

The security of schemes often rely on mathematical problems that can be reduced to other problems known as difficult to solve, according to the complexity theory. For instance, at heart of many public-key encryption schemes is a one-way function, that is, a function that is easy to evaluate at a given point, but that computationally cannot be inverted. The complexity theory offers powerful mathematical tools to construct schemes. Nevertheless, it seems obvious that, even if a function is proven to be one-way, it can be used so that the scheme turns out to be utterly insecure. Indeed, almost all effective attacks against popular cryptosystems have succeeded by finding weakness in the protocol, but not by inverting one-way functions.

Proving security can seem to be hopeless after reading it took 17 years to find an attack to a protocol that was believed and proven secure... Nevertheless, security will always depend on the hypothesis under which you place your reasoning. That is why satisfying and accurate models for properties and protocols are crucial, and why automated or semi-automated verification tools are needed.

Moreover, no matter how precise models can get, one must not forget that in reality, the size of messages is not bounded, nonces are arbitrary numbers, channels are insecure, intruders have unlimited capabilities, or that an unbounded number of instances can play the protocol, whose number of executions is not bounded neither! Consequently, cryptologists still have quite a lot of work to get done, even if designing cryptosystems is not generally their main priority anymore.

References

- [SCH] Bruce Schneier, *Applied Cryptography*, John Wiley and Sons, 1996, ISBN 0-471-11709-9.
- [BIS] Matthew Bishop, *Computer Security: Art and Science*, Addison and Wesley, Dec. 2002.
- [STI] Douglas Stinson, *Cryptography: Theory and Practice*, by CRC Press, Inc ,March 1995.
- [GOL] Oded Goldreich, *The Foundations of Cryptography*, 2 vol., (Basic Tools) ISBN 0-521-79172-3 Cambridge University Press, June 2001; and (Basic Applications) ISBN 0-521-83084-2, Cambridge University Press, May 2004.
- [MOV] A. J. Menezes, P.C. van Oorschot and S.A. Vanstone, *The handbook of Applied Cryptography*, online : www.cacr.math.uwaterloo.ca/hac/index.html

[SIN] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, (there are many editions of this book: Fourth Estate Ltd, National Bestseller, etc.).