Université Grenoble Alpes

6. Ciphering and applications (Keys, cryptography)

6.1 Introduction

6.2 Symmetric cryptosystems

DES

6.3 Asymmetric cryptosystems

RSA

6.4 Comparison of encryption algorithms

6.5 Other applications of cryptography

6.6 Public Key Infrastructure

Université Grenoble Alpes

6.1 Introduction

6.1 Mission of cryptography

- To guarantee as well as possible
  - Confidentiality
  - Authenticity
  - Integrity

of data (or information) exchanged (or stored)

6.1 Cryptology and cryptography

- Cryptology
  - Science of enciphering
- Cryptography
  - Art to write the messages in an enciphered form
- Cryptanalyse
  - Attempt to decipher enciphered messages
- Modern cryptology
  - Based on digits
  - Use the results of arithmetic (some of these results are really old!!!)

6.1 Description of an enciphering system

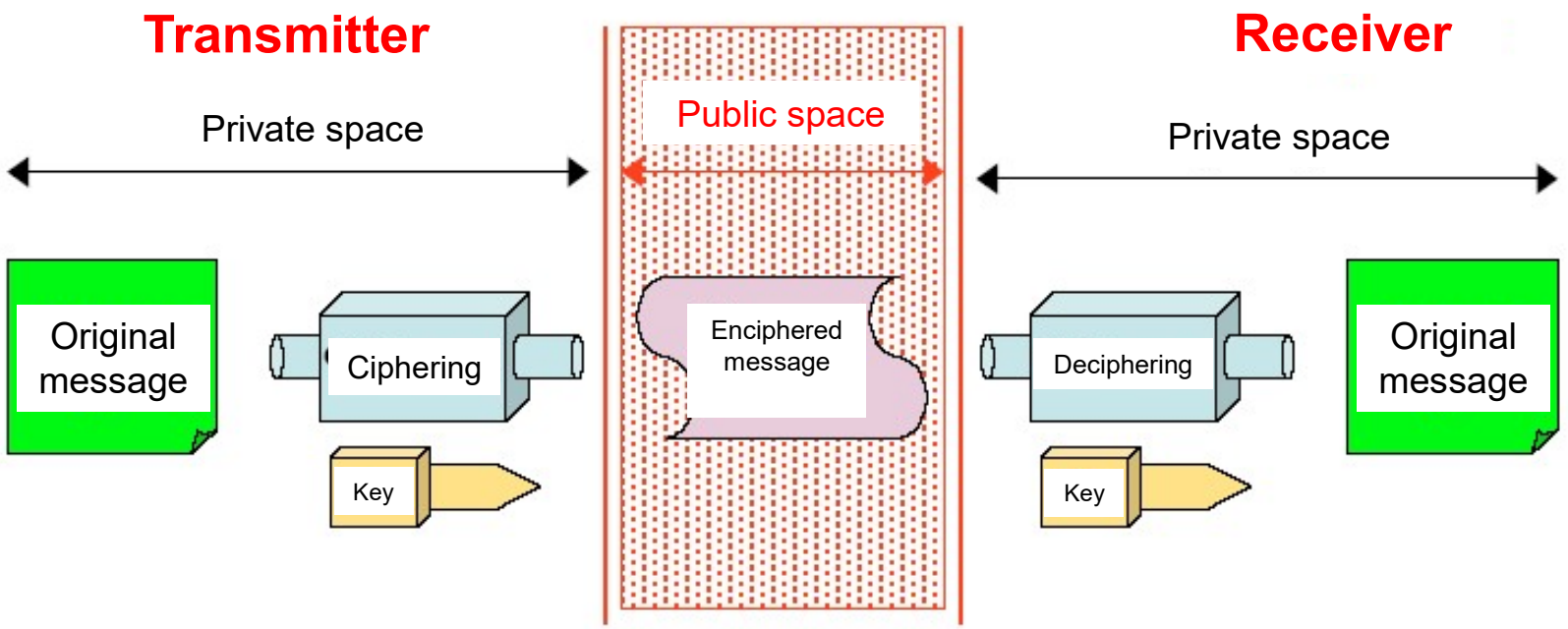
- Choice
  - Algorithm
  - One or more security keys
  - Transmission Media

6.1 Panorama of modern cryptography

Cryptography with secret key	Cryptography with public key	quantum cryptography
•Public algorithm or not •Pb. trans. of key •Attacks •unknown enciphered text •Plaintext known •Chosen enciphered text •Unspecified mode of calculation	•Public algorithm •Public keys •Attacks •selected quantified text •Mode of calculation •Function with single direction •Function “trap door”	•Public algorithm Keys: quantum channel •Impossible Espionage (spying) •Require correction of the errors

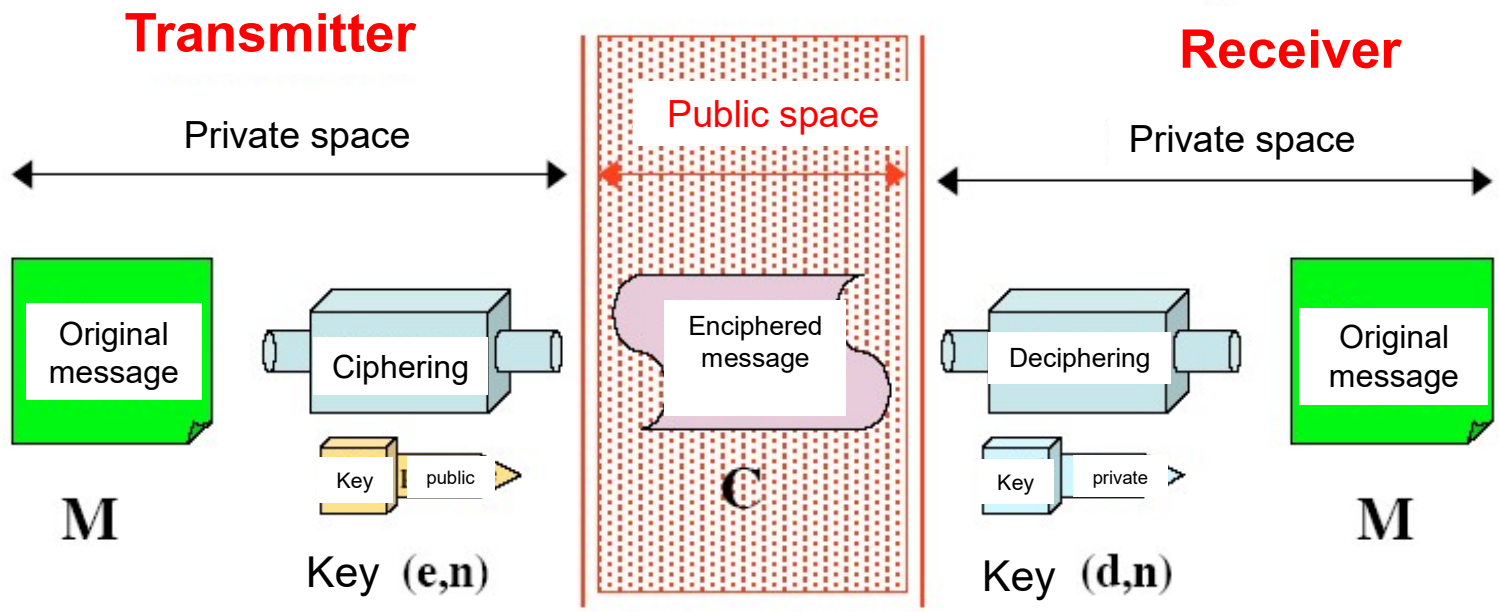
In RESEARCH

6.1 Symmetric encryption



- ✓ The same key is used for enciphering and deciphering
- ✓ Problem: how to transfer the key

6.1 Asymmetric encryption



- Enciphering is achieved thanks to the public key
- Warrant that the owner of the private key ONLY can decipher the message

6.1 Size of the key

- Key enciphered on n bits => 2^n values
- The longest is the key
  - The most important is the number of possible keys
  - more time is necessary to compute and find the result
- A 40 bit-key (10^{12} different possibilities) => it has become now rather simple to break them
- Significant information => prefer a 128 bit-key (10^{38} possibilities) or a 256 bit-one
- Note: it is easier to find the key from a user or from the storage system than to find it thanks to deciphering

6.1 Robustness of the enciphering system

- Power of the algorithm (non-secret algorithm)
- Size of the key used
- Capacity to keep the secret keys in a protected way
- A system of enciphering is known as reliable, robust, sure, protected if it remains inviolable independently of the computing power or time available to an attacker
- It is known as operationally protected (*computational secure*) if its security depends on a series of realizable operations in theory, but unrealizable practically (too long processing times…)
- It is necessary to frequently change the enciphering key

Université Grenoble Alpes

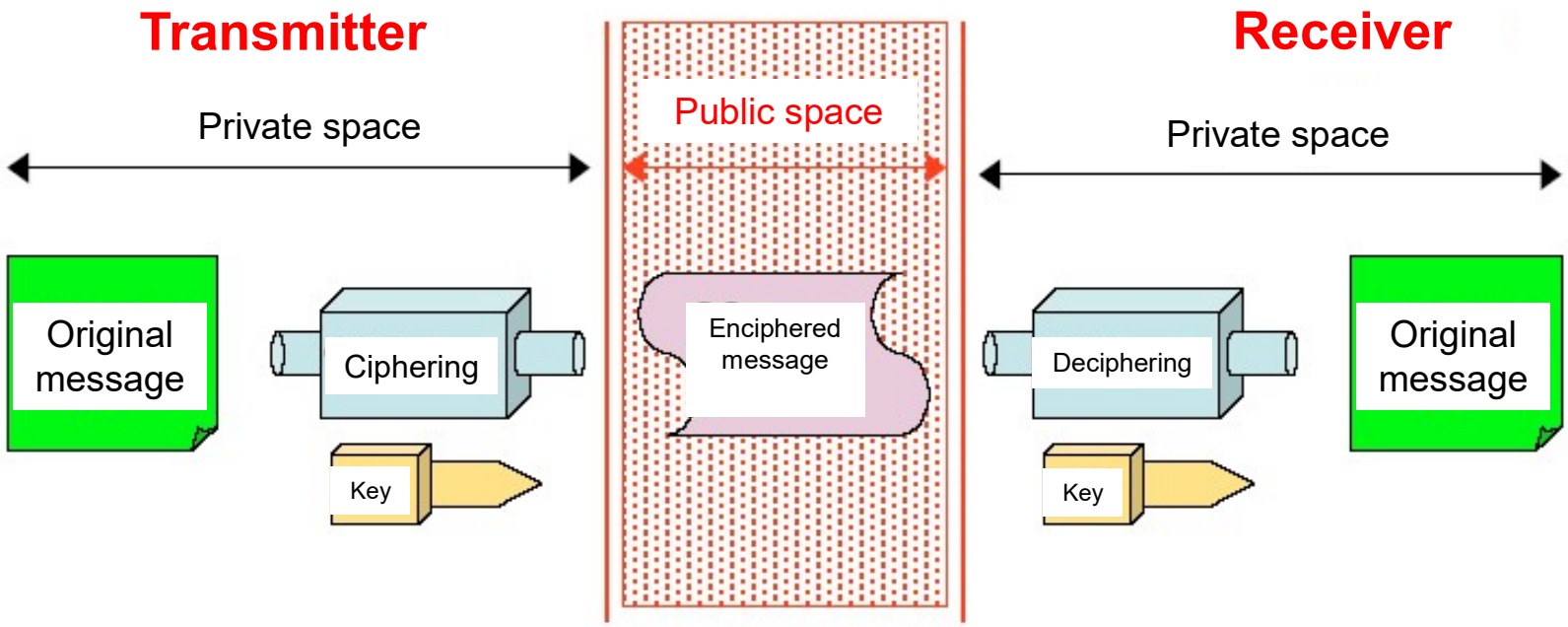
6.2 Symmetric cryptosystems

6.2.1 Description
6.2.2 Examples
6.2.3 DES (Data Encryption Standard)
6.2.4 Other symmetric algorithms

6.2.1 Symmetric cryptosystems

- At the origin (once upon a time…), algorithms of enciphering (long continuations of operations on characters) => secretly preserved
- Today
  - Systems such as DES (Data Encryption Standard)
    - public algorithms enciphering and deciphering (known and fast)
    - Robust algorithms (it is almost impossible to break the algorithm) => only possibility: exhaustive research…

6.2.1 Symmetric cryptosystems



- The same key is used for enciphering and deciphering
- Problem: how to transfer the key

6.2.1 Symmetric cryptosystems

- Set of keys: K
- Algorithm for enciphering: E
- Algorithm for deciphering: D
- ∀ (whatever) a k key and the t message:

$$D(E(t,k),k)=t$$

6.2.1 Vulnerability of symmetric cryptosystems

- 4 possibilities of attacks of a secret key-cryptosystem
  - Attack “gloutonne”: to test all the keys (if the algorithm E is known from the spy)
  - Attack with quantified text: consists in discovering whole or part of the key starting from encrypted messages
  - Attack with quantified and plaintexts: by an illegal mean, the spy has quantified texts for which he knows the plaintext corresponding
  - Attack with text clearly selected: pernicious, consists in choosing the plaintexts to obtain in return the corresponding encrypted texts
- Robust algorithm = time of forcing of an order of magnitude quite higher than that of the human activities!!

6.2.2 Enciphering examples: substitutions (Caesar translation algorithm)

- Replacement of a letter by another
- Robustness?
  - Identical frequency contents
  - Can be easily “broken” easily starting from a message of 28 letters…
- Example: “right” shift of two letters
  - *Bonjour* => Dqplqwt
- Another example: shift of a letter towards the left
  - IBM => HAL (“2001: A Space Odyssey”)

6.2.2 Enciphering examples : poly-alphabetical codes (1/3)

- To code a character in several different ways according to its position
- One chooses a key which is used as an entrance point in a poly-alphabetical grid
- Each character of the key indicates an alphabet (K) grid, well defined
- To code a character from the plaintext, we should use the grid

6.2.2 Enciphering examples : poly-alphabetical codes (2/3)

- Let's consider an alphabet {A, B, C, D}

text t / key k	A B C D
A	C D B A
B	D C A B
C	C A B D
D	B D A C

plaintext: ABCB ACCB AACB B

Key: DBBC BAAC DDBB C

Encrypted text: BCAA DBBA BBAC A

- Require very large size keys not to be very vulnerable …

6.2.2 Enciphering examples : poly-alphabetical codes (3/3)

- Encrypt
- ACDBA with the key: BDBA

6.2.2 Enciphering examples : poly-alphabetical codes (3/3)

- Encrypt
- ACDBA with the key: BDBA

- Result:
- DABDD

6.2.2 Enciphering examples : Operations at the bit level

- Change of scale: character => bit
- Use of the “digital” techniques, mathematical jamming (“*brouillage*”) techniques
  - Permutations
  - Transpositions
  - Substitutions of forms (shapes, “*motifs*”)
- Use of the exclusive OR (or XOR) Boolean function => bijective operation + equals to its opposite

6.2.2 Enciphering examples: Operations at the bit level Distance permutations

- d1=1, d2 = 01, d3 = 001, d4= 0001…
- Distance permutation (di, dj)
- Example: TS
- Form substitution
- Example (d1, d2, d3, d4) substituted by (d2d3, d3d1, d1d4, d1d3) => increases the size of the data
- TS
- 54 53
- 0101 0100, 0101 0011
- d2 d2 d2 d4 d2 d3 d1
- d3d1 d3d1 d3d1 d1d3 d3d1 d1d4 d2d3
- 0011 0011 0011 1001 0011 10001 01001
- What is the size of the original message? The encrypted one?

- Then to decipher…

6.2.2 Example

- Encipher *BON* by substituting (d1, d2, d3, d4, d5, d6) by (d2d3, d3d1, d1d4, d1d3, d2d4, d5d6) with d1=1, d2 = 01, d3 = 001, d4= 0001…
- BON (each character is encoded as a 8-bit hexadecimal ASCII code)
- What is the size of the original message? The encrypted one?
- 42, 4F, 4E

6.2.2 Example

- Substitution (d1, d2, d3, d4, d5, d6) by (d2d3, d3d1, d1d4, d1d3, d2d4, d5d6)
- 42, 4F, 4E
- 0100 0010 0100 1111 0100 1110
- Encoding
- d2 d5 d3 d3 d1d1d1 d2 d3d1 d1, ! 0 is not taken into account…
- Encryption
- d3d1 d2d4 d1d4 d1d4 d2d3d2d3d2d3 d3d1d1d4 d2d3d2d3
- 0011 010001 10001 10001 010010100101001 001110001 0100101001
- What is the size of the original message? The encrypted one?
- 24 bits (3 bytes) for the original message, 54 bits for the encrypted one

6.2.2 Inversion of bits according to a random suite

- Purpose: Transforming each byte of a file F by reversing certain bits by operations of binary negation
- Let's consider a pseudo-random numbers suite (a_n)
- For each byte, the bits to be reversed are obtained by calculating the modulo 8 of the terms of the suite (a_n). The suites of modulo 8-numbers is called (b_n)
- If $b_{n+1} \leq b_n$ then one passes to the following byte => the nbr of bits which are reversed in a byte is random…

6.2.2 Inversion of bits according to a random suite : example 1

- (a_n) = (2, 14, 7, 11, 74, 25, 32, 37, 152, 99, 7) => (b_n) = (2, 6, 7, 3, 2, 1, 0, 5, 0, 3, 7)
- F= 01001010 10010101 00101001 00010100 11010110 11110001
- And
- F'= 01**1**0101**01** 100**0**0101 00**0**01001 0**1**010100 **0**1010**0**10 **0**11**0**000**0**

Bit 2 Bit 6 Bit 3 Bit 2…

6.2.2 Inversion of bits according to a random suite : example 2

- *BON* with the random suite (a_n) = (3, 4, 11, 27, 32, 25, 12, 153, 77, 7) modulo 8
- 42, 4F, 4E

6.2.2 Inversion of bits according to a random suite : example 2

- *BON* with the random suite (a_n) = (3, 4, 11, 27, 32, 25, 12, 153, 77, 7) modulo 8
- 42, 4F, 4E
- 01000010 01001111 01001110
- (b_n) = (3,4,3,3,0,1,4,1,5,7)
- 01011010 01011111 01011110

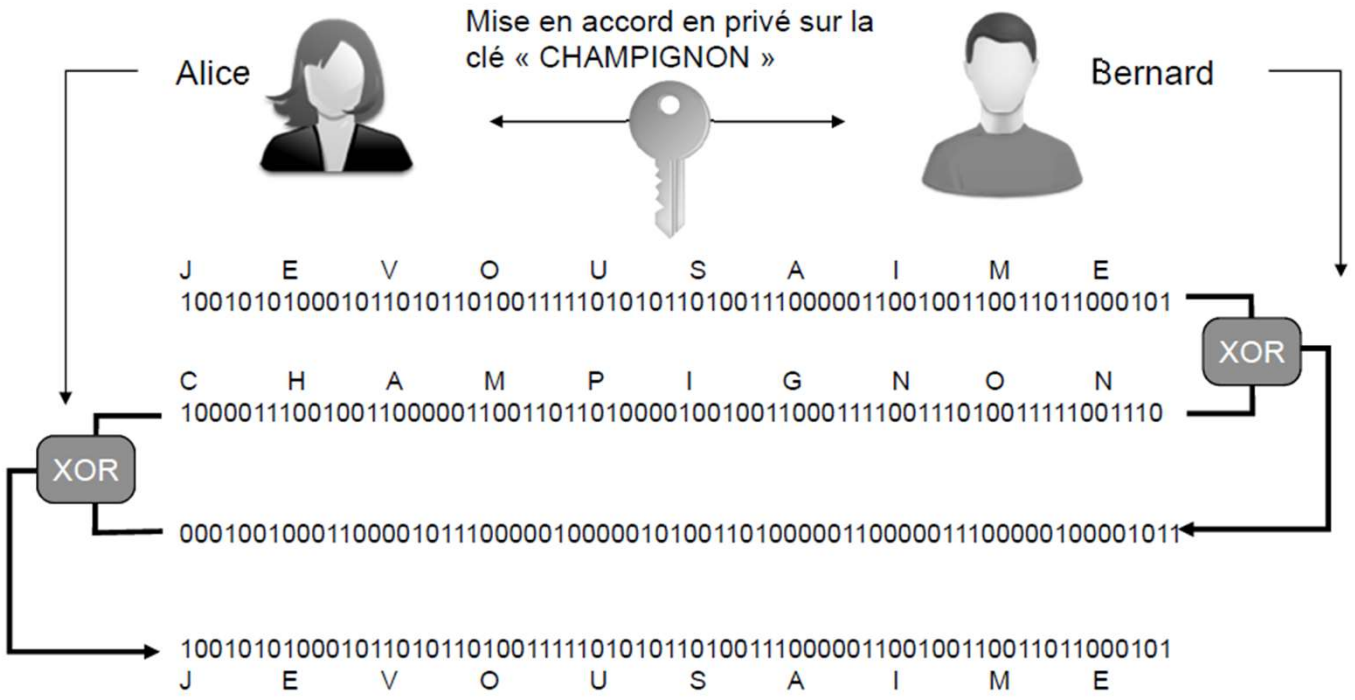
6.2.2 Inversion of bits according to a random suite : example 3

- *BON* with the random suite (a_n) = (3, 4, 11, 27, 32, 25, 12, 153, 77, 7) **modulo 9**
- BON
- 42, 4F, 4E

6.2.2 Inversion of bits according to a random suite : example 3

- *BON* with the random suite (a_n) = (3, 4, 11, 27, 32, 25, 12, 153, 77, 7) **modulo 9**
- BON
- 42, 4F, 4E
- $(b_n) = (a_n)$modulo 9 = (3, 4, 2, 0, 5, 7, 3, 0, 5, 7)
- 0100 0010 0 100 1111 01 00 1110
- 0101 1010 0 101 1111 01 10 1111

Asymmetric encryption: Example XOR function



XOR	0	1
0	0	1
1	1	0

Université Grenoble Alpes

6.2.3 DES (Data Encryption Standard)

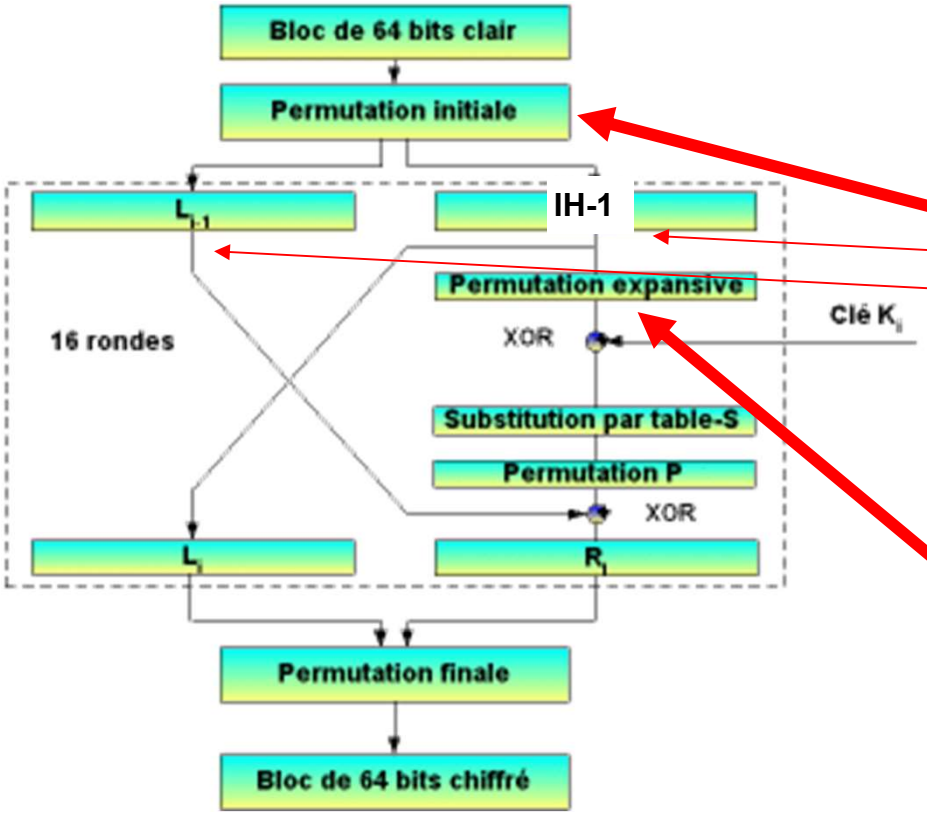
6.2.3 The standard algorithm for enciphering: IBM DES (Data Encryption Standard)

- Created 1977
- Algorithm for blocks encryption
- At first for classified or secret documents
- Today software and smart cards industry
- Enciphering and deciphering speed (rapidity)
  - Can be developed in less than 200 lines
  - Very fast on dedicated electronic charts
    - Smart cards
    - Electronic systems of telecommunications
- Implementation on Unix, Windows and MacOs available on Internet (chalmers.se/pub for example)

6.2.3 DES

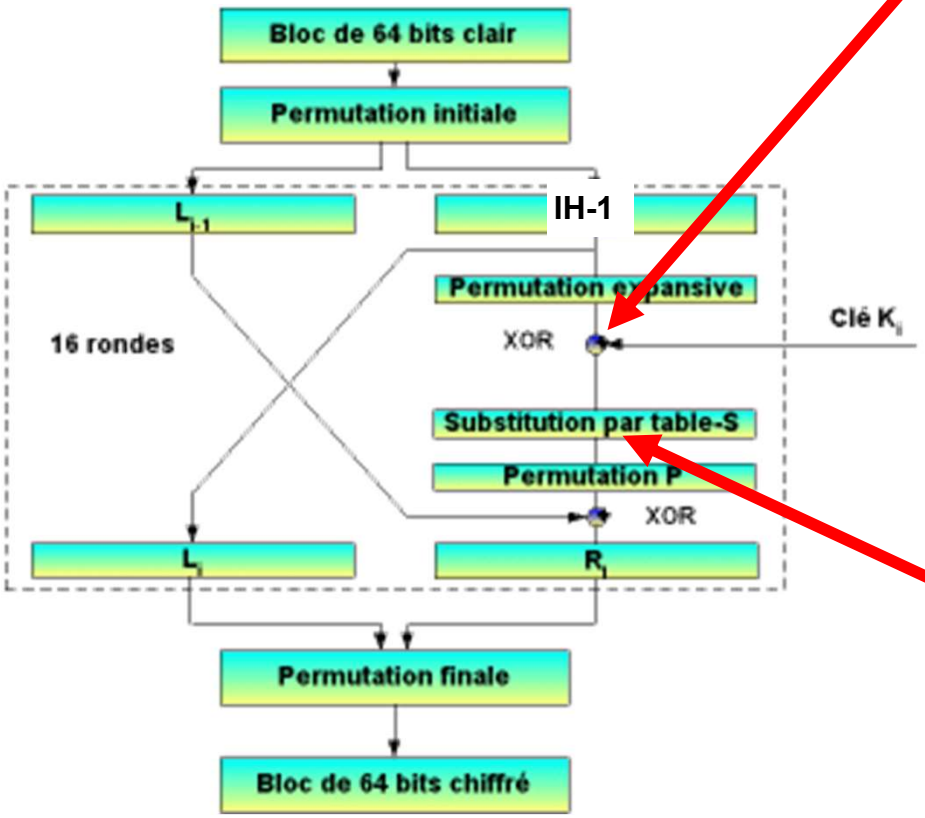
- Principle
  - product whose idea comes from Shannon (inventor of the information theory)
  - Mixing of permutations and substitutions
- Code with 64 bits-blocks (but also 128 or 256)
  - transformation of a block is composed of 16 iterations of a process of coding which is known as ITER ()
  - Based on a private K key (64, 128 or 256 bits)

6.2.3 DES



- 1st step: Initial permutation
  - Each 64 bits-block of undergoes a permutation then is divided into two blocks (L_0 et R_0) of 32 bits
- *Beginning of the first iteration*
- 2nd step: expansive permutation
  - The 32 bits of R_0 enter a table of selection of bits, they are mixed and repeated. **48** bits are obtained

6.2.3 DES



- 3rd step: Calculation of the Ki key
  - Calculation of the K_1 key (48 bits) from the origin key K
  - Transformation of the preceding block with a XOR with the K_1 key
- 4th step: substitution by S table
  - The preceding result is broken up into 8 R_{0i} blocks of 6 bits each (b_1, b_2, b_3, b_4, b_5, b_6)
  - This decomposition makes it possible to calculate a position in a table of selection (S) composed of 8 blocks of 16 columns and 4 rows
    - The number (b_1, b_6) represents the number of the row
    - The number ($b_2b_3b_4b_5$) represents the number of the column
  - We replace the R_{0i} block with the 4 bits-block found in the table => total of 32 bits for the 8 blocks

Université Grenoble Alpes

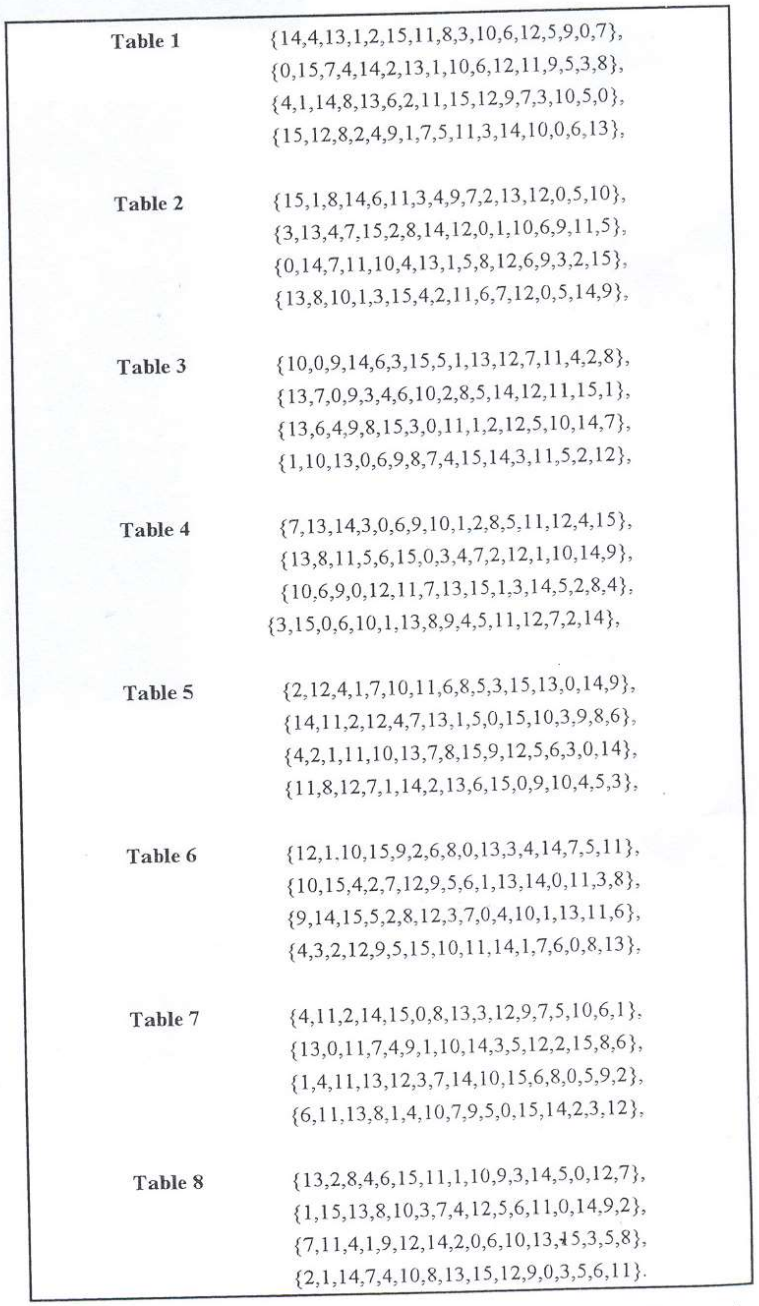
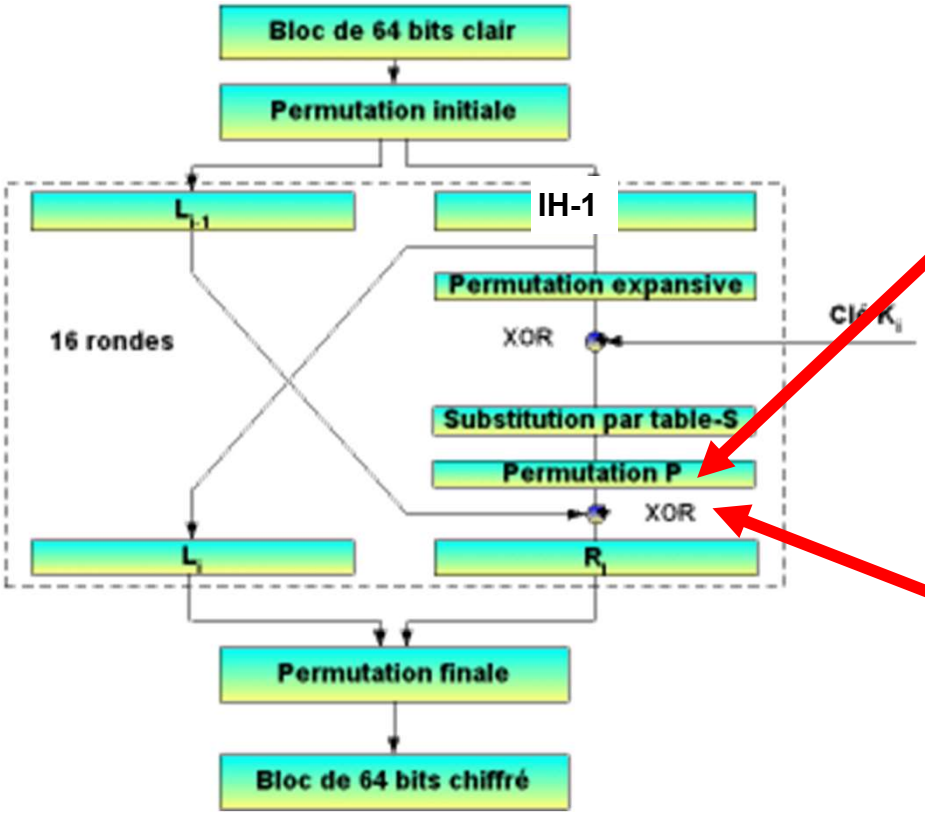


Table 1	{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7}, {0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8}, {4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0}, {15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13},
Table 2	{15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10}, {3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5}, {0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15}, {13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9},
Table 3	{10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8}, {13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1}, {13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7}, {1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12},
Table 4	{7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15}, {13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9}, {10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4}, {3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14},
Table 5	{2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9}, {14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6}, {4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14}, {11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3},
Table 6	{12,1.10,15,9,2,6,8,0,13,3,4,14,7,5,11}, {10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8}, {9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6}, {4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13},
Table 7	{4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1}, {13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6}, {1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2}, {6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12},
Table 8	{13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7}, {1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2}, {7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8}, {2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}.

Figure 10.2. *Table de substitution non-linéaire utilisée à l'étape 4.*

6.2.3 DES: nonlinear substitution table used for the 4th step

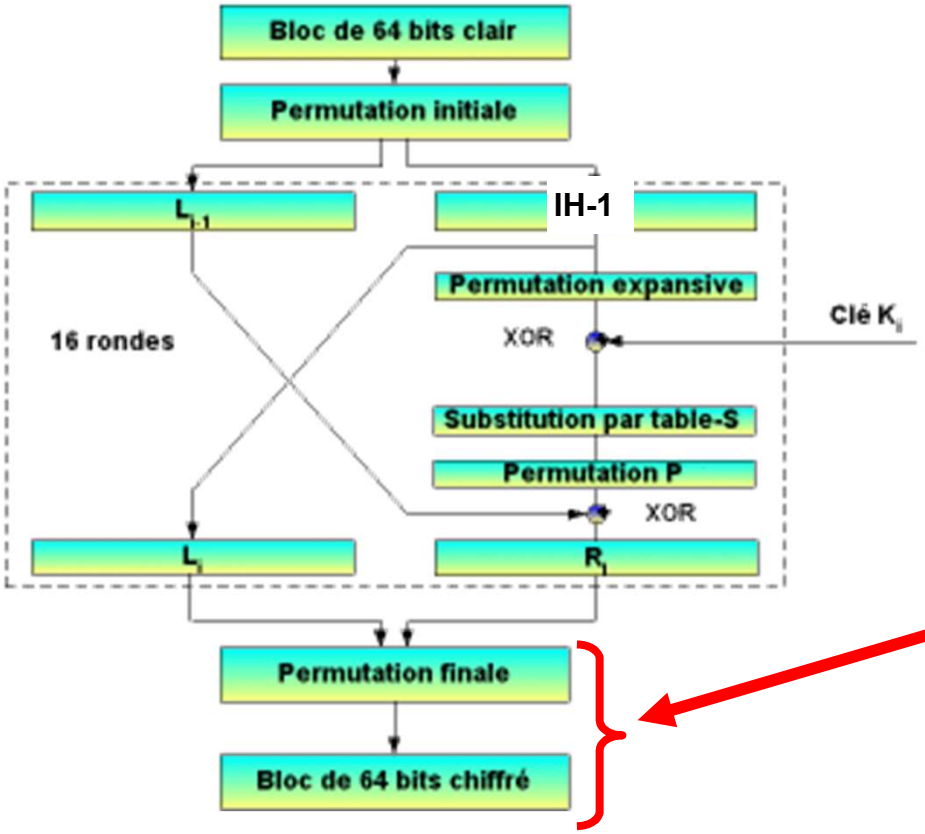
6.2.3 DES



- 5th stage: New permutation
  - The 32 bits are permuted in the following way:

    $b_{16}, b_7, b_{20}, b_{21}, b_{29}, b_{12}, b_{28}, b_{17}, b_1, b_{15}, b_{23}, b_{26}, b_5, b_{18}, b_{31}, b_{10}, b_2, b_8, b_{24}, b_{14}, b_{32}, b_{27}, b_3, b_9, b_{19}, b_{13}, b_{30}, b_6, b_{22}, b_{11}, b_4, b_{25}$
- 6th stage: exclusive OR
  - The result of step 5 is subjected to a XOR with L0 to form R1
  - L1=R0 is posed
- *End of the first iteration*

6.2.3 DES



- The procedure is then repeated 14 times (round 2 to 15) using the K_i key
- The 16th iteration ends with a final permutation

6.2.3 Calculation of the K_i key starting from the key of origin K

- The security of thus rests on the key K
- This key is an alphanumeric chain of 64 bits (8 bytes)
- These 8 bytes undergo a P_1 permutation where the first 8 bits (parity) are eliminated
- then form two blocks
  - $L_0 = (b_{57}, b_{49}, b_{41}, b_{33}, \ldots., b_k, b_{k-8}, \ldots)$
  - $R_0 = (b_{63},b_{55},b_{47},b_{39},\ldots,b_k, b_{k-8},\ldots)$
- The key K_1 is obtained by shifting L_0 and R_0 of a bit towards the left => one obtains the blocks L_1 and R_1
  - The blocks L_1 and R_1 undergo a P_2 permutation which does not turn more than 48 bits => those form the K_1 key
- This calculation spreads to generate 15 other keys K_i starting from the blocks L_{i-1} and R_{i-1}. Attention the number of bits of decay of L_i and R_i varies as a function of i in the following way:

{1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1}

6.2.3 Decoding of DES

- DES is a symmetric process, the operations of coding are thus equal to their reverse => decoding uses the same operations as encoding, by using the same keys k_i, but beginning with k_{16} instead of k_1

6.2.3 Security of DES

- Non linearity of the functions of substitution (4th step), they are designed to resist a cryptanalyse. N.B a small change in these tables can ruin the security of DES
- The procedures used cause that the cryptanalyse fails in an attempt to decrypt thanks to a frequency analysis of the encrypted texts
- 16 iterations per block scramble the plaintext and propagate the jamming quasi uniformly
- The number of keys implies the practical impossibility, even with large samples, to find the key K starting from the encrypted text (time estimated at 500 years at the end of the years 1990).
- The possibility of using longer keys, with 2 times more bits (128 bits), the test of all the possibility would take 268.000.000 times more time…
- But DES security can be easily corrupted from 1997 using an exhaustive research (size of the key)

6.2.4 Other symmetric algorithms

- 3DES (triple DES, 168 bit-key)
  - It consists in using three times the DES algorithm with three keys k_1, k_2 and k_3 : $m'=DES_{k1}(DES_{k2}(DES_{k3}(m)))$
  - Alternative with 2 keys and by using twice the algo of encrypting and once the algo of decoding: $m'=DES_{k1}(DES^{-1}_{k2}(DES_{k1}(m)))$, this alternative is considered more secure
- DESX (DES XORed), GDES (Generalized DES), RDES (Randomized DES): comes from DES, using larger keys.
- AES (Advanced Encryption Standard) published in 1998, recommended from 2002
  - developed to replace DES and offer a better security
  - Use 128, 192 or 256 bit-keys
  - N.B. At the end of 2003, the American department of defense approved its authorization
  - Used in IPSec (secured IP) and IKE (Internet Key Exchange)
  - Considered as “unbreakable” and the surest system nowadays

AES (some aspects)

- DES obsolete at the end of the 90s:
  - key size too small,
  - execution time too long (accentuated with triple-DES),
  - existence of back doors from NSA (?)
- 1997: competition for AES by NIST (National Institute of Standards and Technology, USA)
  - Can use 128, 192 or 256 bit keys
  - Block size of at least 128 bits
  - Fast execution on as many platforms as possible
- Rijndael algorithm designed by Daemen and Rijmen, from UC Louvain (BE) in 2000
- Cryptanalysts are constantly working on attack algorithms to detect vulnerabilities. For example, of the 10 rounds of AES-128, if it is easy to break one round of the AES, there are no significant results today (2016) on more than 6 rounds

6.2.4 Other symmetric algorithms

- RC (Rivest Cipher) from RC2 to RC6
  - Algorithms with symmetric keys diffused by RSA Security Inc. (www.rsasecurity.com)
  - Use a variable key length (up to 2048 bits)
  - Used to make confidential the applicative flows
- IDEA (International Data Encryption Algorithm)
  - Key of 128 bits keys to code 64 bits-blocks of data
  - Used by the protected protocol of mail PGP (*Pretty Good Privacy*) http://sebsauvage.net/logiciels/pgp.html
- Twofish
  - Key size up to 256 bits
- Blowfish
  - Symmetric encryption algorithm developed in 1993, largely replaced by AES

6.2.4 Conclusion on symmetric systems

- Sure
- Fast

but

- Need to exchange the key
  - Problem of security during the transmission of the key
- Problems of the management of keys

Université Grenoble Alpes

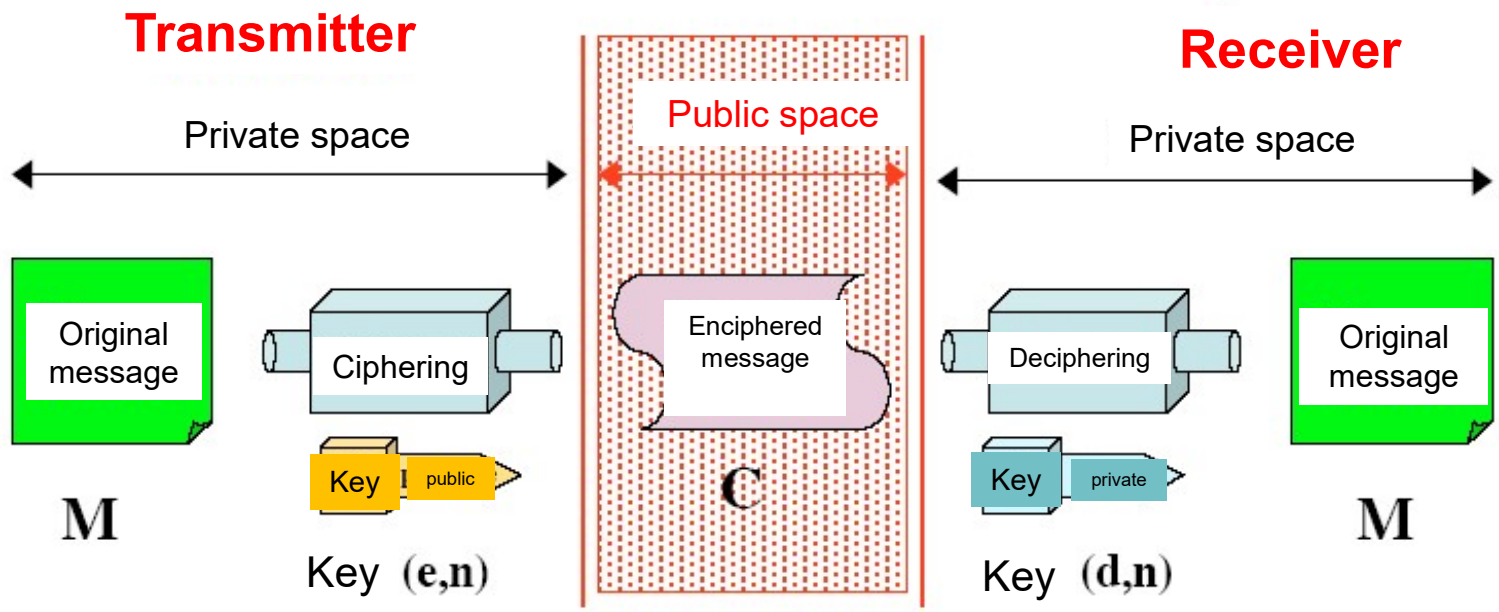
6.3 Asymmetric cryptosystems

6.3.1 Principles
6.3.2 RSA (Rivest, Shamir and Adleman)
6.3.3 Other public keys-algorithms

6.3.1 Asymmetric ciphering

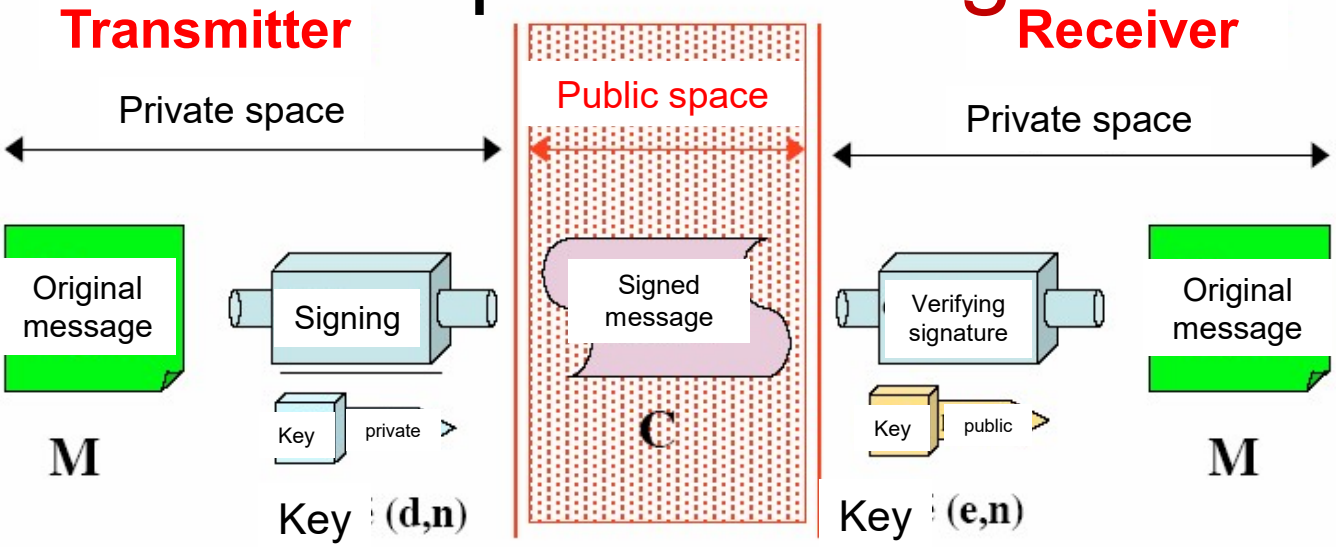
- Developed to resolve the problem of the distribution of the keys of the symmetric systems
- Use of a single couple of two complementary keys, calculated one compared to the other (public key and private key)
- the public key only can be known of all, the private key must be confidential and be treated like a secrecy
- Ex: one must know the public key of a recipient to send statistical data to him
  - ⇒ He will decipher them at reception with his private key, which he is alone to know
- Execution time of these algorithms produces important processing times processor

6.3.1 Asymmetric cryptography: Protection of the confidentiality



- ✓ Enciphering is achieved thank to the public key
- ✓ Warrant that the owner of the private key ONLY can decipher the message

6.3.1. Asymmetric cryptography: Principle of the signature



-to sign a message, the private key is used (the hash is encrypted with the private key)

-Garanty of authentication, but no confidentiality

- deciphering with the public key is a proof that the private key only was used for the signature

Combination

- Sender side
  - 1. Encryption using the public key of the receiver
  - 2. Signature using the private key of the sender
- Transmission
- Receiver side
  - 1. Verification of the signature using the public key of the sender
  - 2. Decryption using the private key of the receiver

6.3.1 Asymmetric systems: principles

- To allow the exchange of encrypted information without meeting to exchange the keys
- security is based on the fact it is practically impossible to solve a data-processing problem which is “difficult”, for which the search for a solution amounts to thousands, or even billion years.

6.3.1 Asymmetric systems: some concepts (1/2)

- Functions with single direction
  - According to the theory of the calculability and algorithmic complexity, a function has a single direction if:
    - The function f is calculable quickly
    - Its reverse f^{-1} needs a very long time to be calculated
  - Example of function with a single direction:

$a^p \bmod n$ => called exponential of the variable ***p***

Base a fixed

n is the product of two “large” prime numbers

the reverse function (called discrete logarithm) is calculable “with difficulty”

6.3.1 Asymmetric systems: some concepts (2/2)

- Trap functions or secret breach function
  - Function with one single direction except for any person knowing a secrecy, or a breach, allowing to calculate an fast inversion algorithm
  - Example of secret breach function:

$a^p \bmod n$ => called modular exponentiation of the variable ***a***

Power p fixed, n product of two prime numbers

The existence of a reciprocal algorithm which allows the calculation of the p^{th} roots modulo n of a is proved, but this algorithm is not known

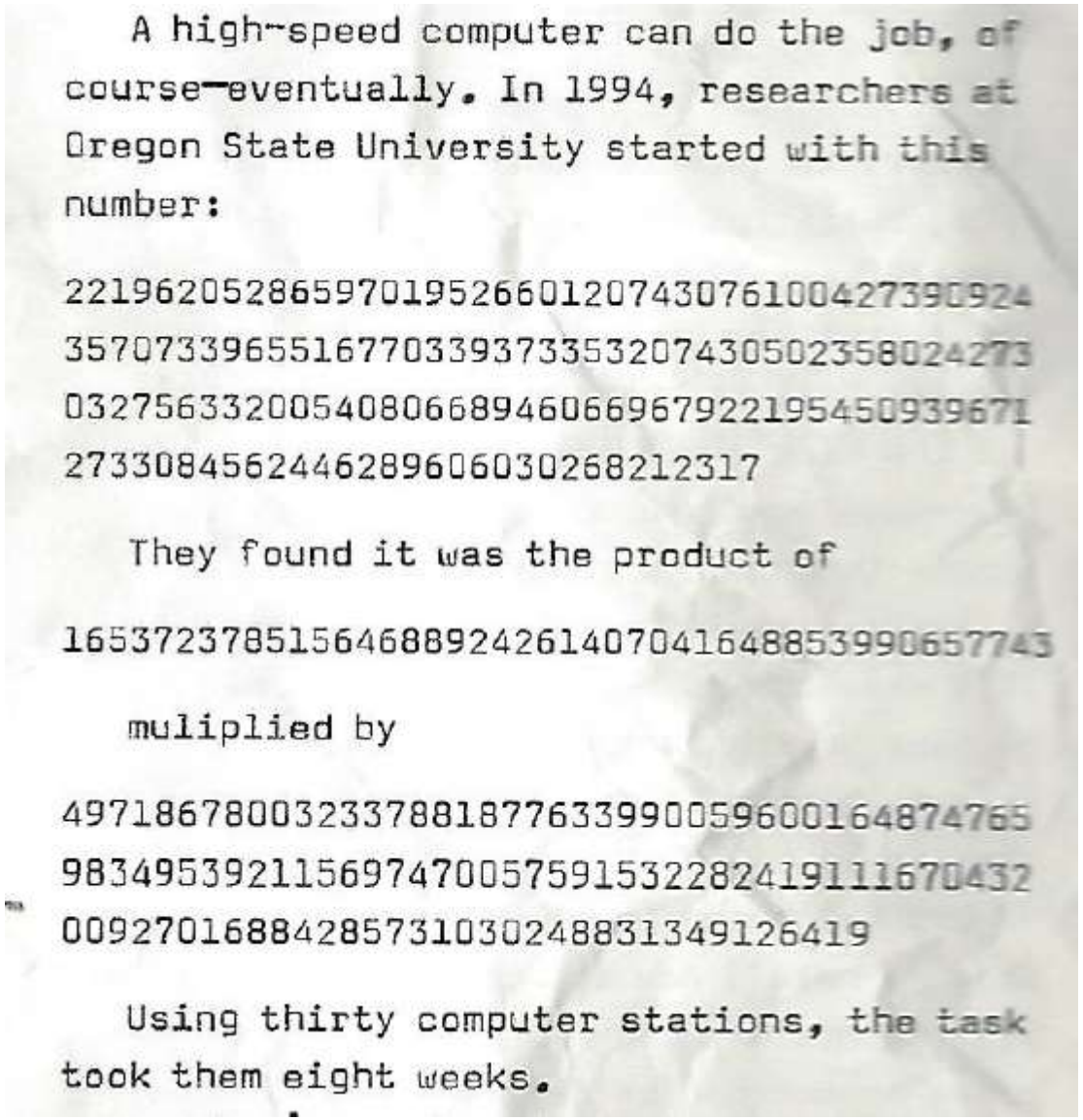
On the other hand, if one knows the factorization of n (the breach), it is easy to reverse the modular exponentiation

6.3.1 Definition of asymmetric crypto-systems (according to **Whitfield DIFFIE and Martin HELLMAN**)

- It exists a public algorithm for coding: E (encoder)
- A secret algorithm for decoding: D (decoder)
- E and D are a function of the key K used
- m=D(E(m))
- D and E are calculable immediately for any person knowing the key K
- The knowledge of E should not make it possible to know D, if not the security of encoding is not guaranteed
  - The process is public and depends on the key =>the decoding algorithm D remains secret and impossible (in reasonable time) to calculate starting from E => E should be a secret breach function, the breach being the algorithm D

Factorisation (Gary Blackwood « Mysterious messages, a history of codes and ciphers », 2009

- Factoring is not easy
- Ex: 11 * 13 = 143 ; You have to determine what two prime numbers (or factors) can be multiplied to make that numbers. A small number like 143 can be factorised easily by trail and errors.



A high-speed computer can do the job, of course—eventually. In 1994, researchers at Oregon State University started with this number:

2219620528659701952660120743076100427390924
3570733965516770339373353207430502358024273
0327563320054080668946066967922195450939671
27330845624462896060302682123 17

They found it was the product of

1653723785156468892426140704164885399065774 3

muliplied by

4971867800323378818776339900596001648747 65
9834953921156974700575915322824191116704 32
0092701688428573103024883134912641 9

Using thirty computer stations, the task took them eight weeks.

- **Largest Known Prime Number:**
- **$2^{82\ 589\ 933}-1$**
- **Found in December 2018, composed of** 24 862 048 digits
- 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223…

Université Grenoble Alpes

6.3.2 RSA

6.3.2 RSA ciphering Protocol

- Proposed in 1977 by the cryptologists Rivest, Shamir and Adleman
- Based on the modular exponentiation (trap function)
- Main applications
  - Sending of confidential messages to a person
  - Authentication by any person of the message sent by an individual
  - Authentication by password (smart cards, bank cards)
- security based on the impossibility of carrying out the factorization of a large number of a few hundreds of digits in a reasonable time
  - The user selects two large prime numbers p and q, then multiplies them to obtain n=p.q (integer modulating the RSA protocol)

6.3.2 RSA (encryption, confidentiality)

- The algorithm is remarkable by its simplicity. It is based on the prime numbers.
- To encrypt a message:

  c = m^e mod n

- To decrypt: **m = c^d mod n**
  - **m** = clear message
    c = encrypted message
    (e, n) constitutes the public key
    (d, n) constitutes the private key
    n is the result of the multiplication of 2 prime numbers
    ^ is the power function (a^b: a power b)
    mod is the operation of modulo (rest of the *integer division*)

6.3.2 RSA (digital signature to protect authentication and integrity ⇔ signature)

- The algorithm is remarkable by its simplicity. It is based on the prime numbers.
- To sign a message:

s = m^d mod n

- To verify the signature: **m = s^e mod n**
  - **m** = clear message
    s = signed message
    (e, n) constitutes the public key
    (d, n) constitutes the private key
    n is the result of the multiplication of 2 prime numbers
    ^ is the power function (a^b: a power b)
    mod is the operation of modulo (rest of the *integer division*)

6.3.2 RSA: Creation of a pair of keys

- It is simple, but the **e**, **d** and **n** should be chosen with care! And the calculation of these three numbers is delicate.
- Methodology:
  - The user selects two large prime numbers p and q, and multiplies them to obtain n=p.q (integer modulating the RSA protocol), We should choose p and q with equivalent sizes.

    It is advised that n is higher or equal to 512 bits
  - Take a number **e** which does not have any factor in common with **(p-1) (q-1)**.
  - Calculate **d** such as **ed mod (p-1)(q-1) = 1** (We need to choose e randomly in order that e doesn't have any common factors with r=(p-1) (q-1))
- The couple **(e, n)** constitutes the public key.
- **(d, n)** is the private key.
- Various other rules are to be respected for the use of these prime numbers so that the algorithm cannot be "broken"

6.3.2 RSA: Operative rules

- We use the decimal codes of the characters to code (for example ASCII)
  - N.B. If we take other codes (ex: for instance hexadecimal), it is necessary to treat all calculations in the corresponding base
- We cut out the message in blocks which is composed of less figures than **n** (we add zeroes if it is needed to obtain the last block)

6.3.2 RSA: Example (1/4)

- https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
- Let us start by creating our pair of keys:
  - Let us take 2 prime numbers randomly: **p = 127**, **q = 167**
  - Let us calculate **n = pq = 127 * 167 = 21209**
- We must choose **e** randomly such as **e** does not have any factor in common with r=**(p-1) (q-1)**:
  - **r=(p-1) (q-1) = (127-1) (167-1) = 20916**
- We need to find two numbers **e** and **d** whose product is a number equal to **1 mod r**. Below appears a list of some numbers which equal **1 mod r**.
- We must choose e randomly such as e does not have any factor in common with r=(p-1) (q-1):
- Candidates numbers (k.r+1), k app N* donc nbr mod r = 1 : 20917 41833 62749 83665 104581 125497 146413 167329
- 188245 209161 230077 250993 271909 292825 313741 334657
- 355573 376489 397405 418321 439237 460153 481069 501985
- 522901 543817 564733 585649 606565 627481

6.3.2 RSA: Example (2/4)

- Step 2. Find K a number equal to 1 mod r which can be factored:
  - Let's choose for instance K=20917
- Find the factors of K (see https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html)
  - 13*1609
- Let's take **e = 13**
- Let's choose **d** such as **13*d mod 20916 = 1**
  - We find **d = 1609**
- These are the keys:
  - The public key is **(e, n) = (13,21209)** (=enciphering key)
  - The private key is **(d, n) = (1609,21209)** (=deciphering key)

6.3.2 RSA: Example (3/4)

- https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
- Let's encipher the message "**HELLO**". Let's take first the ASCII code (into decimal) of each character and one puts them end to end:
  - **m = 72-69-76-76-79**
- Then, it is necessary to cut out the message in blocks which is composed of less digits than **n**. **n** is composed of 4 digits, one thus will cut out our message in blocks of 3 digits:
  - 726 976 767 900
    (let's complete with zeros)
- Then one encrypt each one of these blocks:
  - 726^13 mod **21209** = 11600
    976^13 mod **21209** = 5705
    767^13 mod **21209** = 16590
    900^13 mod **21209** = 3565
- The encrypted message is **11600.5705.16590.3565**. One can decipher it with d:
  - 11600^1609 mod **21209** = 726
    5705^1609 mod **21209** = 976
    16590^1609 mod **21209** = 767
    3565^1609 mod **21209** = 900
- I.e. the digit suite: **726976767900**.
  We find the clear message: **72 69 76 76 79**: "**HELLO**".

6.3.2 RSA: Example (4/4)

- https://www.cs.drexel.edu/~jpopyack/IntroCS/HW/RSAWorksheet.html
- Let's encipher the message "**HELLO**". Let's take first the ASCII code (into decimal) of each character and one puts them end to end:
  - **m = 72-69-76-76-79**
- Then, it is necessary to cut out the message in blocks which is composed of less digits than **n**. **n** is composed of 4 digits, one thus will cut out our message in blocks of 3 digits:
  - 726 976 767 900
    (let's complete with zeros)
- Then one encrypt each one of these blocks:
  - 726^13 mod **21209** = 11600
    976^13 mod **21209** = 5705
    767^13 mod **21209** = 16590
    900^13 mod **21209** = 3565
- The encrypted message is **11600.5705.16590.3565**. One can decipher it with d:
  - 11600^1609 mod **21209** = 726 (if 1 bit is corrupted 11601^1609 mod 21209 = 6051)
    5705^1609 mod **21209** = 976
    16590^1609 mod **21209** = 767
    3565^1609 mod **21209** = 900
- I.e. the digit suite: **726976767900**.
  We find the clear message: **72 69 76 76 79**: "**HELLO**".

6.3.2 Remarks on the RSA

- In practice, it is not so simple to program: Large prime numbers should be found (that can take very a long time to calculate)
- It is necessary to obtain prime numbers **p** and **q** which should be *really* random
- We should not use blocks as small as in the example before: it is necessary to be able to calculate powers and modulos on very large numbers
- In fact, one never uses the asymmetric algorithms to encrypt all the data, because it will take a too long time to calculate: one encrypts the data with a simple symmetric algorithm from which the key is drawn randomly, and it is this key which is exchanged through an asymmetric algorithm like the RSA

6.3.3 Other asymmetric algorithms

- gpg (GNU Privacy Guard)
- **ECC** (Elliptic Curve Cryptosystems, Encryption by Elliptic Curve). This system is based on a parametric curve which passes through a certain number of points with integer co-ordinates. It is not very developed yet, but it is promising
- **Diffie-Hellman** (Used in IKE (Internet Key Exchange) negotiations), more and more preferred than RSA. (Diffie-Hellman had quickly been adopted by the open-source community when RSA was not in the public domain yet)
- **El Gamal**: based on the calculation of discrete logarithms

Université Grenoble Alpes

6.4 Comparisons of the ciphering algorithms

Comparison

- Asymmetric ciphering more useful
  - No problem with the key transfer
  - Allow the message signature
  - Possibility to manage Public Key Infrastructure (PKI)
- Symmetric encryption is faster (La Recherche, June 2018)
  - AES allows enciphering many gigabytes per second on a recent processor
  - Asymmetric cryptography standards reach less than one megabyte per second (1000 to 10000 slower !)
- Generally the two strategies are combined
  - Interest: to use a protocol with public key to transmit the DES key => hybrid cryptography

Hybrid cryptography, generation of a sharing key Diffie-Helman strategy

- Two users will design a common key which will be useful for them only

ASYMMETRIC ASPECT

- They choose n the multiple of 2 prime numbers p and q and an integer a (a and n can be known (not confidential))
- Then each one chooses an integer X belonging to [1, n-1] and calculates the integer $Y=a^X \bmod n$
- We obtain two couples (X_1,Y_1) and (X_2, Y_2) where the values Y_1 and Y_2 will be published

HYBRID ASPECT

- Each one of them can then calculate the key $c=a^{X1X2} \bmod n$ because $c=(Y1^{X2} \bmod n)=(Y2^{X1} \bmod n)$
- R: Each one knows its own X only
- Security comes from the fact that it is impossible in a reasonable time to obtain the key C by the calculation of a discrete logarithm (unfeasible in a reasonable time taking into account the size of p and q)

SYMMETRIC ASPECT

- Users can now exchange encrypted data using a symmetric system with the common key c

Application

- User 1
- User 2

n

a

Application

- User 1
- X1 : private key

- User 2
- X2 : private key

n

a

Application

- User 1
- X1 : private key
- $Y1=a^{X1} \bmod n$: public key

n

a

- User 2
- X2 : private key
- $Y2=a^{X2} \bmod n$: public key

Application

- User 1
- X1 : private key
- $Y1=a^{X1} \bmod n$: public key
- Send Y1 to user 2
- Receive Y2

n

a

- User 2
- X2 : private key
- $Y2=a^{X2} \bmod n$: public key
- Send Y2 to user 1
- Receive Y1

Application

- User 1
- X1 : private key
- $Y1=a^{X1} \bmod n$: public key
- Send Y1 to user 2
- Receive Y2
- $c=(Y2^{X1} \bmod n)$

n

a

- User 2
- X2 : private key
- $Y2=a^{X2} \bmod n$: public key
- Send Y2 to user 1
- Receive Y1
- $c=(Y1^{X2} \bmod n)$

Application

- User 1
- X1 : private key
- $Y1=a^{X1} \bmod n$: public key
- Send Y1 to user 2
- Receive Y2
- $c=(Y2^{X1} \bmod n)$
- Key « c » in order to use the symetric system

n

a

- User 2
- X2 : private key
- $Y2=a^{X2} \bmod n$: public key
- Send Y2 to user 1
- Receive Y1
- $c=(Y1^{X2} \bmod n)$
- Key « c » » in order to use the symetric system

Private and public keys

- The public key is composed of two large prime numbers p and q (several hundreds of bits).
- The public key n is given by n =p*q.
- As n est very large, it is impossible to find all the possible factorisations.
- The knowledge of n does not allow to deduce the values of p and q.

Exercise

- Generate a shared key with your neighbor
- (ex:
  - a=3 et n=14 (public values (known)) n=2*7
  - X_1 = 4 (secret value known only by the participant on the left)
  - X_2 = 3 (secret value known only by the participant on the right)

Some considerations on breaking a 768-bit RSA key

- From an Inria document, 2010.
- Key used for bank cards
- To break the key, find the prime numbers which compose the key: it is a number composed of 232 figures (2^{768})…
- Need efficient algorithm
- Need large calculation capacities: use of Grid'5000 => 1544 computers with more than 5000 cores.
- Collaboration with CH, JP, NL, DE : on average 1700 cores used during one year of calculation…
- One week by using the supercomputer *Jaguar* (from *Oak Ridge National Laboratory*) if available (not such computers in Europe…)
- The purpose was to show if it is possible to break using grid of « classical » computers
- Next step: to break a 1024 bit-key => it should be possible around 2020
- Advise from ANSSI (2010):
  - Use at least 1536 bit-keys for applications until 2010
  - Use at least 2048 bit-keys for application beyond 2010

Other considerations regarding calculation time (2016)

- 2^{40}~10^{12} : operations possible on a personal computer
- 2^{56}~10^{16} : operations possible for a company or research lab
- 2^{64}~10^{19} operations possibly possible for NSA (US), GCHQ (GB), DGSI (FR)
- 2^{80}~10^{24} operations considered somewhat unfeasible
- 2^{128}~2.10^{38} operations impossible with current technologies
- 2^{256}~6.10^{76} almost physically impossible: a computer, perfect in the physical sense, would need the same level of energy as the sun for several years ...

These orders of magnitude will not be the same in 20 years (maybe about 2^{80} will be possible for a company ...)

Other considerations considering the size of the keys

- ”From "Randomness, the keystone of computer security": LA RECHERCHE, July-August 2019, D. Vergnaud
- Is perfect security just an illusion? In theory, no.
- Disposable mask encryption (invented by the American Gilbert Vernam in the early 20th century) shows that if you have a key as long as the message, drawn in a random and uniform way, and used only once, it is possible to achieve perfect security, i.e. the **view of a ciphertext does not reveal any information about the plain text to the attacker.** Since the key must be exchanged and never reused, the practical difficulties are immense for large-scale use of this method of cryptography.
- The unpredictability of keys remains essential to ensure the security of these modern algorithms.

Université Grenoble Alpes

6.5 Other applications of cryptography

6.5.1 Hashing

6.5.2 Signature

6.5.3 Hash for passwords

6.5.4 Certificates

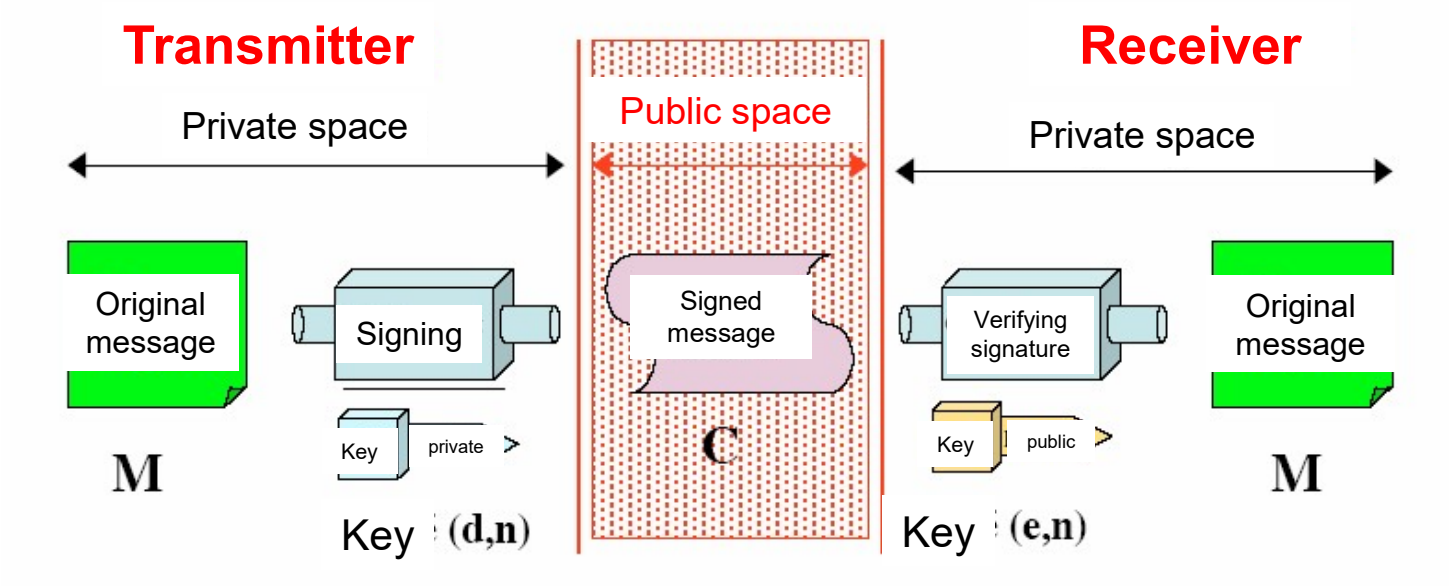
6.5.1 Hashing Functions

- How it works?
  - A hash is calculated by applying a mathematical algorithm to a set of data
  - The value obtained is generally shorter than the set of data
  - It is almost not possible to have two sets of data with the same hash (but it can be!)
  - It is impossible to obtain the data from the hash (non reversible function)
- Purpose
  - To guarantee the Authentication of the parts
  - To guarantee the integrity of the data
- Names
    - Hash (*fr. haché,* ***somme de contrôle***)
    - Fingerprint (*fr. empreinte*)
    - Digest (*fr. condensé*)

6.5.1 Algorithms for Hashing

- MD5 (message digest 5)
  - 1994 (by Ron Rivest), RFC 1321
  - irreversible print of 128 bits
  - allows to check the integrity of the message
  - http://www.isi.edu/in-notes/rfc1321.txt
- SHA-1 (Secure Hash Algorithm)
  - SHA-1 appeared in 1995
  - irreversible print of 160 bits
- SHA-2 (Last version of the standard: 2012)
  - Two functions: les fonctions, SHA-256 et SHA-512 (size of the hash), also truncated version of SHA 512: SHA-512/256 et SHA-512/224

6.5.2. Principles of signature (without hash)

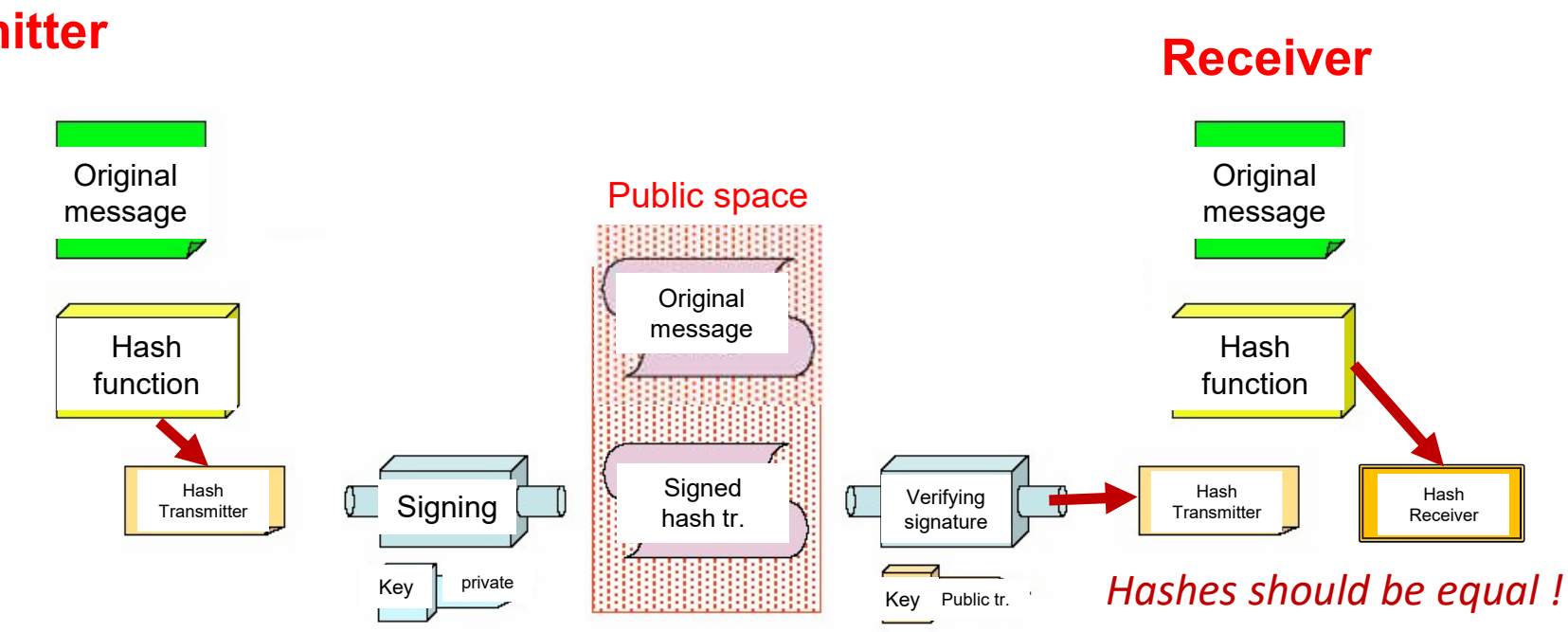


-to sign a message, the private key is used (the hash is encrypted with the private key)

-Garanty of authentication, but no confidentiality

- deciphering with the public key is a proof that the private key only was used for the signature

6.5.2. Signature (with hash)



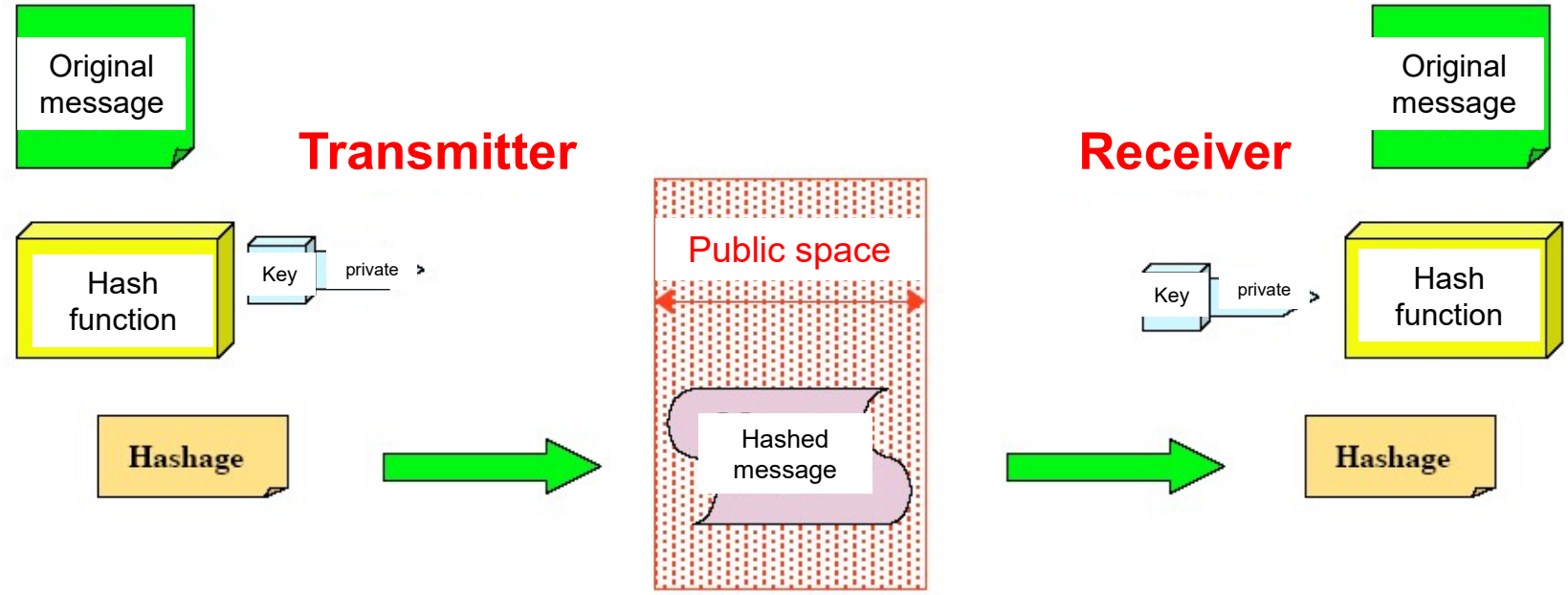
-It is not necessary to encrypt a complete message in order to sign it, it is enough to sign its hash only

-The robustness of the hashing procedure warrants that this is this document which has been signed

6.5.2 Authentication of the messages (1/2)

- If a symmetric ciphering is used to encipher the hash
  - It is an authentication, not a signature
  - Because the key needed to check the signature allows also to create it
  - If the key is known only by the two partners, it allows really to authentify the sender of a message.
- Ex: HMAC-SHA, HMAC-MD5

6.5.2 Authentication of the messages (2/2)



-For authentication, we do not encipher the hash, but use the key for calculation of the hash

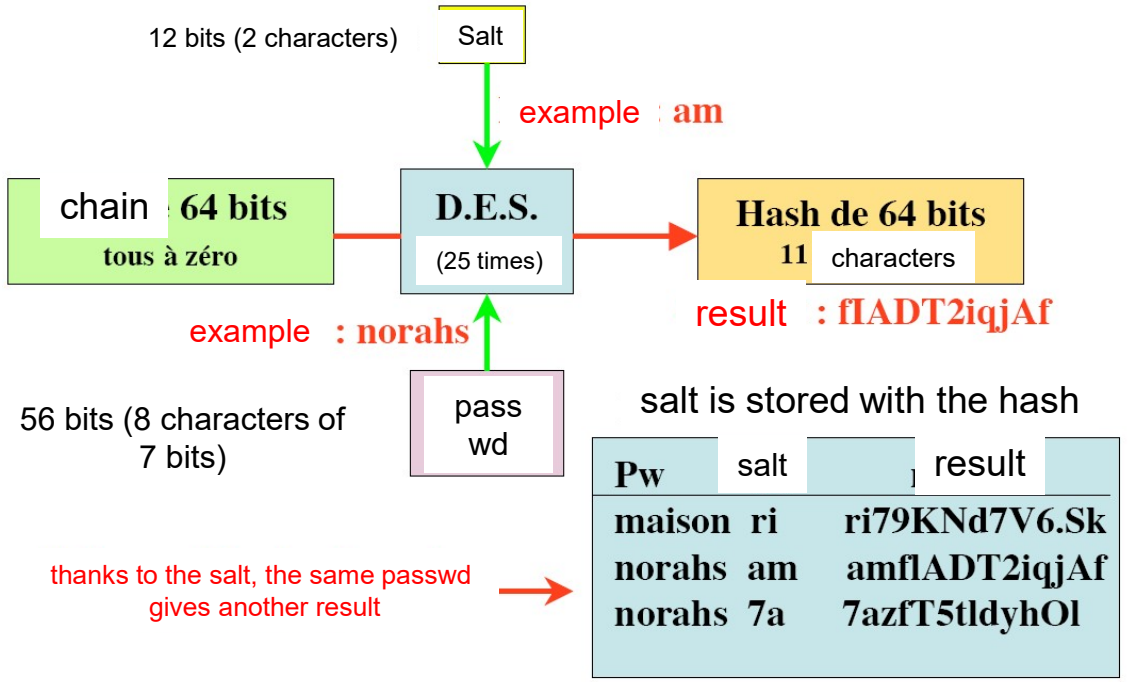
-Such a hash is called M.A.C. (Message Authentication Code)

6.5.3 Application of hashing: storage of passwords

- On a secure system, passwords are stored in an encrypted way (hash)
- By comparing the hash stored with the hash received (during the login) => the hashes have been created or not by the same passwords

6.5.3 Hash on a Unix system (½)

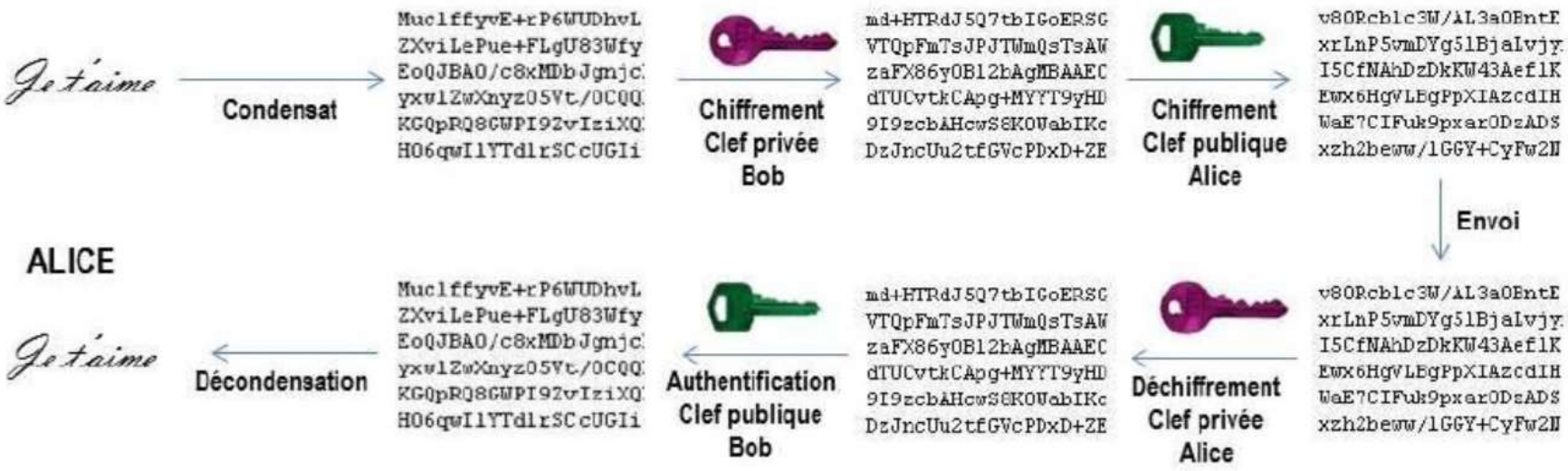
- Hash = encipher 25 times an empty character chain, with the password as a key
- For example, DES uses a 56-bits key, so 7 bits are taken from the passwd and the characters after the 8th are ignored
- A « salt » is added to avoid that two users using the same passwd (it could occurs) obtain the same hashes



6.5.3 Hash on a Unix system (2/2)

- Hash is generated thanks to the password and the « salt ». The generated hash is compared with the stored one
- When using a login through a network, hash and « salt » are obtained from a central server through a ciphered communication
- Storage
  - Before: logins and hash of the passwds in /etc/passwd => free reading access
  - Now: a specific file is used for the hash (can be read only by the administrator) => /etc/shadow
- In order to obtain /etc/shadow
  - Boot the computer with a disk or CD
  - Obtain the administrator passwd (exploit)

Chiffrement asymétrique



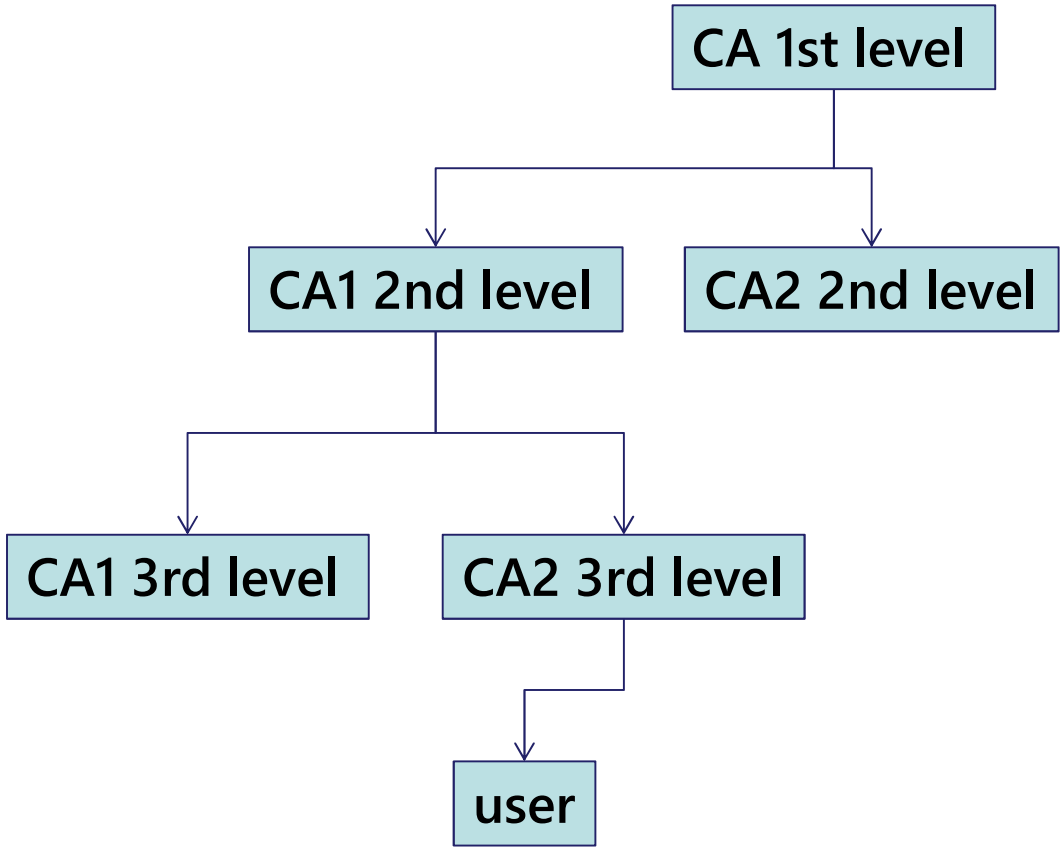
Université Grenoble Alpes

6.5.4 Certificates

6.5.4 Certificates

- Document which proves that a public key belongs to its owner
- It contains at least the following documents:
  - Public key
  - Identity of the entity owning the certificate
    - Name, firstname
    - IP address
    - E-mail address
  - Expiration date (not compulsory)
  - Signature of the certificate by a tierce/third party => this is a trusted partner (known company with good reputation)
  - Ex: https://premium.wpmudev.org/blog/ssl-certificate-authorities-reviewed/

Certification Authorities



6.5.4. Certification Authority (CA)

- CA creates and signs the certificates
- Each participant needs to present himself
  - Physical authentication of the participant
  - CA requests the participant to generate a public key-private key pair
  - CA creates a certificate with the identity of the participant, the CA public key, a expiration date and the signature of the CA
- With its certificate and the CA public key, the new participant can communicate with all the other participants certified by the same CA
- A CA can be private (company) or public
- A CA can ask another CA to certify its public key (hierarchy)
- Certificate format: (Open)PGP or X.509 v2 or v3 standards

6.5.4 Main parameters of a digital certificate according to X509v3 standard

1. Version of the certificate
2. Serial number
3. Algorithm used to sign the certificate
4. Name of the organization which managed the certificate
   - The couple serial number -name of the organization must be unique
5. Time of validity
6. Name of the owner of the certificate
7. Public key of the owner
8. Additional information concerning the owner or the ciphering mechanisms
9. Certificate signature
   - Signature and Algorithm and parameters used for the signature

6.5.4 Validation of the certificate

- To validate the received certificate, the customer must obtain the public key of the organization which created the certificate relating to the algorithm used to sign the certificate (field 3) and must decipher (verify) the signature contained in field 9.
- Using the information also contained in this field, the customer calculates the value of the digest (or *hash*) and compares the value found with that contained in the last field => if the two values correspond, the certificate is authenticated
- Then, the customer makes sure that the period of validity of the certificate is correct

6.5.4. Certificates directory

- To facilitate to access to certificate, CA proposes the use of a directory (LDAP, HTTP)
- To send an e-mail to User2, User1 ask the certificate to the directory, User1 does not need to be accessible (reachable)
- The directory can provide the certificate revoked list (CRL)
- A certificate can be revoked because it has been stolen, lost, or because its owner has moved in another company for instance

6.5.4 Limits of the certificates

- Various certification authorities exist
  - Impossible that there is only one CA (technical and strategic problems)
  - Interoperability of the authorities
  - Mutual recognition
  - Compatibility of the certificates and their validity
  - Limits inherent in the public key infrastructures (PKI)
    - Complexity, cost for an infrastructure deployment and management
    - High level of security necessary to the realization of the services
    - Validity, life duration, revokation of the certificates
  - Real lack of confidence of the users in the certification authorities which are generally outside the company and the services offered
    - Value of the certificates
    - Mechanisms and procedures of Authentication
    - Personal data protection, their identity, their transaction

Université Grenoble Alpes

6.6 Public Key Infrastructure (PKI)

6.6 Public Key Infrastructure (PKI)

- Setting of mechanisms necessary to the realization of asymmetric ciphering systems
  - PKI: Public Key Infrastructure
- Impossible to memorize all the public keys of all the potential correspondents of an Internet site
  - PKI = answers the need for knowing the keys in order to implement a public keys-asymmetric ciphering system

6.6 Function of a Public Key Infrastructure

- Generation of a single couple of keys (public key, private key)
  - Saving of the information necessary for its management
    - Filing (*fr. archivage*) of the keys
    - Procedure of covering in the event of loss by a user or of request for provision by the legal authorities
- Management of the digital certificates
  - Creation
  - Signature
  - Emission
  - Validation
  - Revocation
  - Renewal of the certificates
- Diffusion of the public keys to the resources which would request it and which would be entitled to obtain it
- Certification of the public keys (signature of the digital certificates)

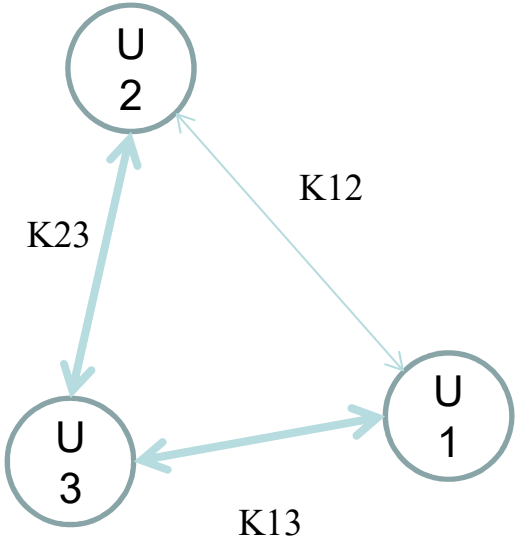
Bibliographical references

- Cours réseaux et télécoms avec exercices corrigés, G. Pujolle, Eyrolles 2006
- SSL VPN, Understanding, evaluating and planning secure, web-based remote access – J. Steinberg & T. Speed, 2005.
- P. H. Oechlin, LASEC/EPFL
- http://sebsauvage.net/comprendre/encryptage/crypto_rsa.html
- S. Ghernaouti-Helie – *Sécurité informatique et réseaux, 4ème édition* – Dunod, 2013
- F. Halsall – Computer networking and the internet – Addison Welseley, 2005 + additional student support at www.pearsoned.co.uk/halsall
- D. Vergnaud – Exercices et problèmes de cryptographie, Dunod, 2015
- CEH, Certified Ethical Hacker, Matt Walker, McGrawHill, 2017
- L. Bloch & al. – Sécurité informatique pour les DSI, RSSI et administrateurs, Eyrolles, 2016.
- Cours C. Bulfone

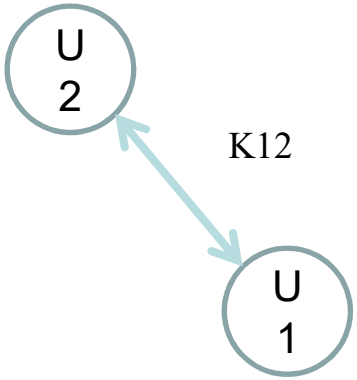
(rappels) le dénombrement

- Permutations
  - Tous les mélanges des objets
  - *Combien de jeux différents avec 32 cartes ?*
  - 32!=32*31*30*..*2*1=2,6.10^{35}
- Arrangements A_n^p=n!/(n-p)!
  - Tirage d'une quantité d'objets dans l'ordre
  - *Combien de fois 3 chevaux dans l'ordre avec 10 chevaux au départ ?*
  - 10!/(10 – 3)! = 10*9*8=720
- Combinaisons C_n^p=n!/(n-p)!p!
  - Tirage d'une quantité sans ordre
  - *Combien de possibilités de cinq numéros parmi 49 ?*
  - 49!/(49 – 5)!5!=(49*..*45)/(5*…*1)=1906884

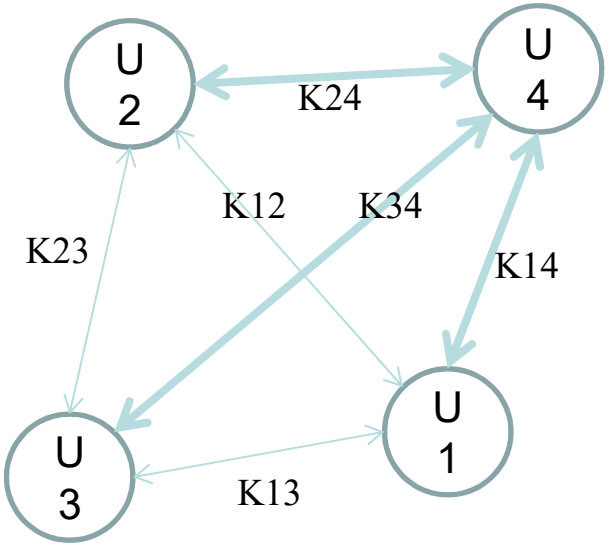
3 users, 3 keys



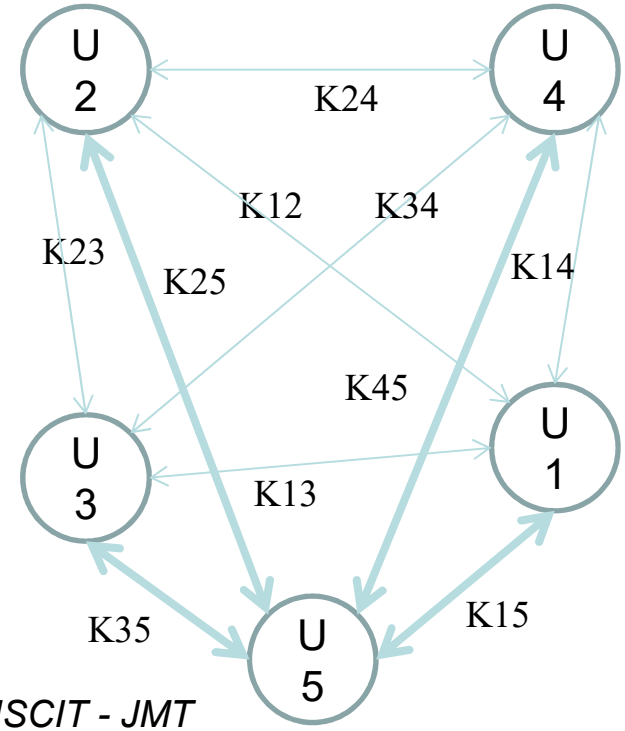
2 users, 1 key



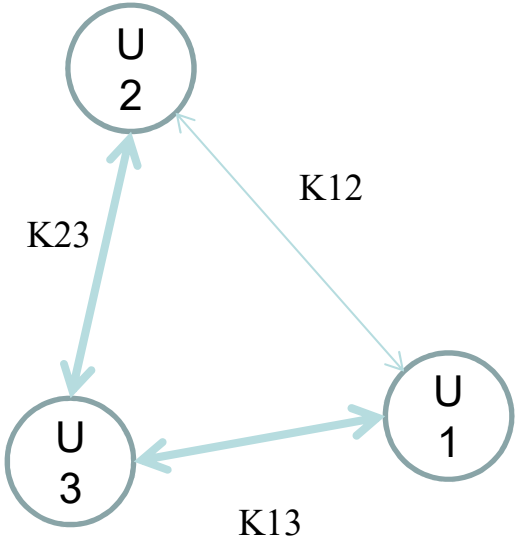
4 users, 6 keys



5 users, 10 keys

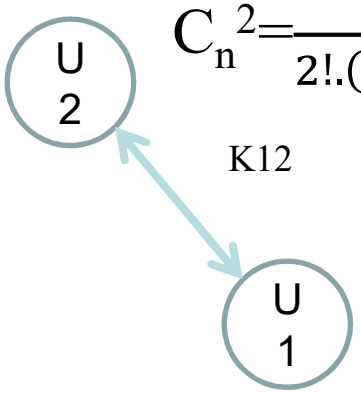


3 users, 3 keys

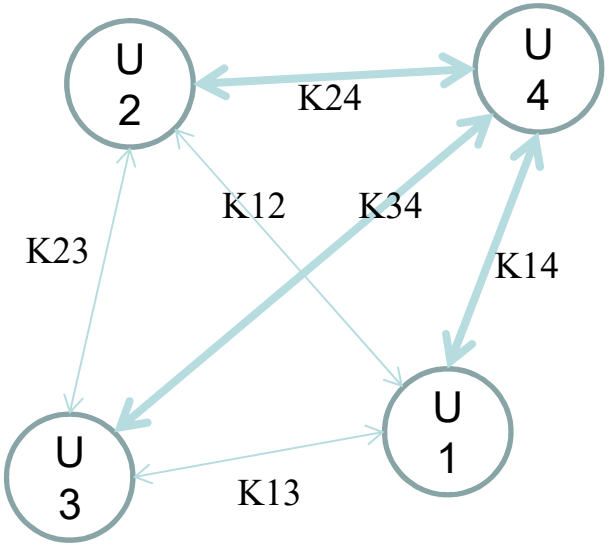


2 users, 1 key

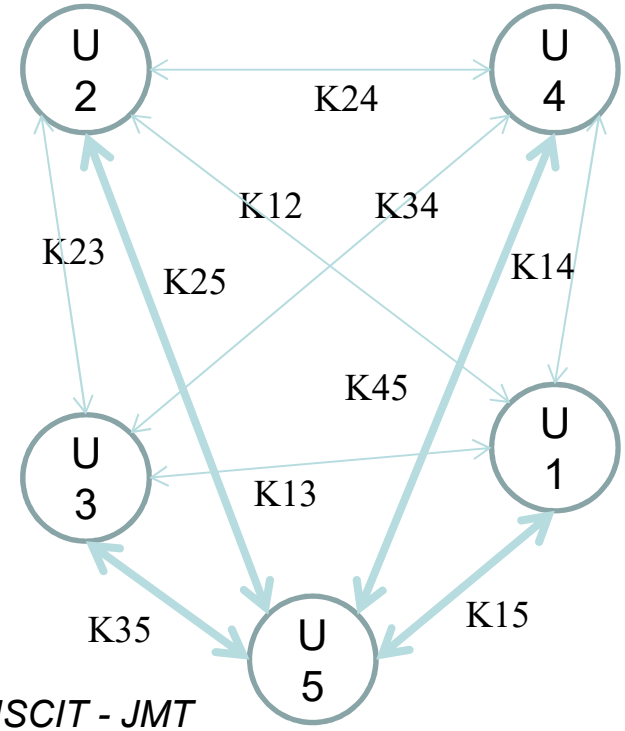
$$C_n^2 = \frac{N!}{2!.(N-2)!} = \frac{N.(N-1)}{2}$$



4 users, 6 keys



5 users, 10 keys



Hardware security modules (HSM)

- For embedded systems for example:
- Cryptographic modules: specific hardware dedicated to cryptography
- May be certified on an international basis: (ex: Common Criteria, FIPS 140)
- Because cryptographic functions (especially asymmetric ones) are intensive regarding processing power and memory comsumption

Hardware security modules (HSM)

- Problem with « software-only » servers is they are general computers that aren't designed for security purposes
  - Many of their parts are « unsupervised »
  - Ex : shared memory
  - Not resistant against physical access
- A HSM is a physical computer that keeps, manages, and processes digital keys for authentication, signing, and verification and provides crypto-processing functions
- Exist in form of sole-appliance (used in networks) or as external attachments that re plug-able on a typical server to be used

Hardware security modules (HSM): characteristics

- Secure « crypto processor »: processor specified on small number of operations designed and dedicated for the crypto-processing
- Secure memory: accessible only through the defined channel (not under any condition through any other channel)
  - Should support fast and repetitive I/O
  - Caching operations should be handled with caution because of sensitivity of data
  - Optionally content of memory can be protected through mechanisms like
    - Virtual Addressing
    - ASLR (Address Space Layout Randomization
    - W^X

Hardware security modules (HSM): characteristics

- Random number generator
  - Expected to be implemented in any HSM
  - Better to implement TRNG (True Random Number Generator) than PRNG (Pseudo-Random NG)
- Tamper-proof module
  - « resistance to tampering (intentional malfunction or sabotage) by either the normal users of a product, package, or system or others with physical access to it » (Wikipedia)
- Inter-component connections
  - HSMs consist of different units, connected to each other and all should be physically guarded by the tamper-proof module

HSM: standards

- FIPS 140 (Federal Information Processing Standards)
  - Standards for cryptographic modules which includes both hardware and software components
- PKCS #11 (Public-Key Cryptography Standards)
  - Maintained by OASIS PKCS #11 Technical Committee
  - Contains a very detailed and technical defined API (called « Cryptoki »)
    - For cryptographic tokens (HSM, cards…)
    - Includes mainly aspects (keys like RSA, certificates like X509)
  - Widely used by CA authority software and hardware

HSM: Usage

- PKI environments
- Banking and payment systems
- More recently
  - DNSSEC