CRYPTOLOGY : CRYPTOGRAPHY + CRYPTANALYSIS

Cryptology = science of secrecy.
How :
encipher a **plaintext** into a **ciphertext** to protect its **secrecy**.
The recipient **deciphers** the ciphertext to recover the plaintext.
A **cryptanalyst** shouldn't complete a successful **cryptanalysis**.
Attacks [6] :

- **known ciphertext** : access only to the ciphertext
- **known plaintexts/ciphertexts** : known pairs (plaintext,ciphertext) ; search for the key
- **chosen plaintext** : known cipher, chosen cleartexts ; search for the key

Polybius's square

Polybius, Ancient Greece : communication with torches

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	ij	k
3	l	m	n	o	p
4	q	r	s	t	u
5	v	w	x	y	z

`TEXT` changed in `44,15,53,44`. Characteristics

- encoding letters by numbers
- shorten the alphabet's size

encode a character x over alphabet A in y finite word over B.
Polybius square : $\{a, \dots, z\} \to \{1, \dots, 5\}^2$.

Short history

J. Stern [8] : 3 ages :

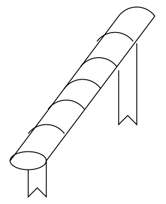
- *craft age* : hieroglyph, bible, ..., renaissance, $\to$ WW2
- *technical age* : complex cipher machines
- *paradoxical age* : PKC

Evolves through maths' history, computing and cryptanalysis :

- manual
- electro-mechanical
- by computer

History – ancient Greece

500 BC : *scytale* of Sparta's generals



Secret key : diameter of the stick

History – Caesar

Change each char by a char 3 positions farther
`A` becomes `d`, `B` becomes `e`...
The plaintext `TOUTE LA GAULE` becomes `wrxwh od jdxoh`.

Goals of cryptology

Increasing number of goals :

- *secrecy* : an enemy shouldn't gain access to information
- *authentication* : provides evidence that the message comes from its claimed sender
- *signature* : same as auth but for a third party
- *minimality* : encipher only what is needed.
- *traceability* : keep the logs for a given time
- *privacy* : the right to be let alone

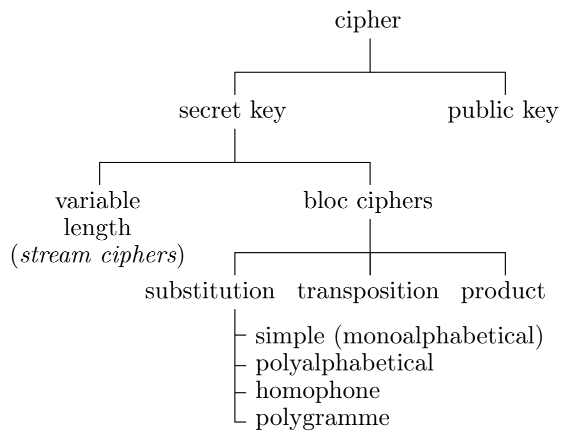
Why enciphering ?

- Yesterday :
  - for strategic purposes
    (the enemy shouldn't be able to read messages)
  - by the church
  - diplomacy
- Today, with our numerical environment
  - confidentiality
  - integrity
  - authentication

The tools

- *Information Theory* : perfect cipher
- *Complexity* : most of the ciphers just ensure computational security
- *Computer science* : all make use of algorithms
- *Mathematics* : number theory, probability, statistics, algebra, algebraic geometry,...

Ciphers Classification



Monoalphabetical ciphers

Monoalphabetical cipher : bijection between letters from $\mathcal{A}_M$ and $\mathcal{A}_C$. If both alphabets are identical : permutation.

Example : Caesar. {a,. . . ,z}≡{A,. . . ,Z} $\equiv \{0, \dots, 25\} = \mathbb{Z}_{26}$
Caesar cipher is **additive**.
Encipher : $\forall x \in \mathbb{Z}_{26}$, $x \mapsto x + 3 \mod 26$
Decipher : $\forall y \in \mathbb{Z}_{26}$, $y \mapsto y - 3 \mod 26$

The + can be changed in $\times$ to get a **multiplicative** cipher (the $\varphi(26)$ acceptables values are s.t. $\mathrm{gcd}(t, 26) = 1 \Leftrightarrow t \nmid 26$).

Affine ciphers combine 26 + ciphers and 12 $\times$:
given s and $t \in \mathbb{N}$, encipher with : $x \mapsto (x + s) \cdot t \mod 26$.
The key is the pair (s, t)

There are 26.12=312 possible affine ciphers. Far from the 26 !=403291461126605635584000000 possible ones.

Symmetrical ciphers

Made of [1] :

- plaintext alphabet : $\mathcal{A}_\mathcal{M}$
- ciphertext alphabet : $\mathcal{A}_\mathcal{C}$
- keys alphabet : $\mathcal{A}_\mathcal{K}$
- encipher ; application $E : \mathcal{A}_\mathcal{K}^\star \times \mathcal{A}_\mathcal{M}^\star \to \mathcal{A}_\mathcal{C}^\star$;
- decipher ; application $D : \mathcal{A}_\mathcal{K}^\star \times \mathcal{A}_\mathcal{C}^\star \to \mathcal{A}_\mathcal{M}^\star$

E and D are such that $\forall K \in \mathcal{A}_\mathcal{K}^\star$, $\forall M \in \mathcal{A}_\mathcal{M}^\star$:

$$D(K, E(K, M)) = M$$

Ciphers defined by keyword

To get all possible monoalphabetical ciphers by :

- a keyword like, for instance `CRYPTANALYSIS` ;
- a key letter like `e`.

Remove multiple occurrences of the same letter in the keyword -here `CRYPTANLSI`- then

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
V	W	X	Z	C	R	Y	P	T	A	N	L	S	I	B	E	D	F	G	H	J	K	M	O	Q	U

Cryptanalysis

Shannon : a small proportion of letters provides more information than the remaining 2/3 of the text.

By applying a frequency analysis on the letters then of bigrams, ... in the ciphertext.

Conclusion

Monoalphabetical ciphers aren't robust against a frequency analysis.

We need ciphers for which the statistical distribution of the letters tend to be a uniform one.

1.st attempt : use a crypto transformation which associates a set of distinct letters in the ciphertext to the plaintext letters.

We get what is called **polyalphabetical ciphers**

Vigenère's cipher (1586)

In a **polyalphabetical cipher**, plaintext characters are transformed by means of a key $K = k_0, \dots, k_{j-1}$ which defines j distinct functions $f_0, \dots, f_{j-1}$ s.t.

$$\begin{array}{ll} \forall i,\ 0 < j \leq n & f_{k_l} : \mathcal{A}_M \mapsto \mathcal{A}_C, \forall l,\ 0 \leq l < j \\ & c_i = f_{k_{i \bmod j}}(m_i) \end{array}$$

Idea : use j distinct monoalphabetical ciphers.

Vigenère's square

```
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ NOPQRSTUVWXYZABCDEFGHIJKLM
BCDEFGHIJKLMNOPQRSTUVWXYZA OPQRSTUVWXYZABCDEFGHIJKLMN
CDEFGHIJKLMNOPQRSTUVWXYZAB PQRSTUVWXYZABCDEFGHIJKLMNO
DEFGHIJKLMNOPQRSTUVWXYZABC QRSTUVWXYZABCDEFGHIJKLMNOP
EFGHIJKLMNOPQRSTUVWXYZABCD RSTUVWXYZABCDEFGHIJKLMNOPQ
FGHIJKLMNOPQRSTUVWXYZABCDE STUVWXYZABCDEFGHIJKLMNOPQR
GHIJKLMNOPQRSTUVWXYZABCDEF TUVWXYZABCDEFGHIJKLMNOPQRS
HIJKLMNOPQRSTUVWXYZABCDEFG UVWXYZABCDEFGHIJKLMNOPQRST
IJKLMNOPQRSTUVWXYZABCDEFGH VWXYZABCDEFGHIJKLMNOPQRSTU
JKLMNOPQRSTUVWXYZABCDEFGHI WXYZABCDEFGHIJKLMNOPQRSTUV
KLMNOPQRSTUVWXYZABCDEFGHIJ XYZABCDEFGHIJKLMNOPQRSTUVW
LMNOPQRSTUVWXYZABCDEFGHIJK YZABCDEFGHIJKLMNOPQRSTUVWX
MNOPQRSTUVWXYZABCDEFGHIJKL ZABCDEFGHIJKLMNOPQRSTUVWXY
```

```
polyalphabetique KSYSSGTUUTZXVKMZ
VENUSVENUSVENUSV
```

Cryptanalysis...

... becomes more difficult : we tend to a uniform distribution.

But, if we re-arrange the ciphertext in a matrix with as many columns as the key length, all the letters in the same column come from the same monoalphabetical cipher.

Cryptanalysis works as follows :
(1) find the key length
(2) apply the previous methods

2 tests to find the key length : Kasiski and Friedman.

Simple array transposition

Given a passphrase, we define a numerical key :

T	R	A	N	S	P	O	S	I	T	I	O	N	S	I	M	P	L	E
18	14	1	8	15	12	10	16	3	19	4	11	9	17	5	7	13	6	2

We encipher, «*le chiffrement est l'opération qui consiste à transformer un texte clair, ou libellé, en un autre texte inintelligible appelé texte chiffré ou chiffré*» [4].

18	14	1	8	15	12	10	16	3	19	4	11	9	17	5	7	13	6	2
l	e	c	h	i	f	f	r	e	m	e	n	t	e	s	t	l	o	p
é	r	a	t	i	o	n	q	u	i	c	o	n	s	i	s	t	e	à
t	r	a	n	s	f	o	r	m	e	r	u	n	t	e	x	t	e	c
l	a	i	r	o	u	l	i	b	e	l	l	é	e	n	u	n	a	u
t	r	e	t	e	x	t	e	i	n	i	n	t	e	l	l	i	g	i
b	l	e	a	p	p	e	l	é	t	e	x	t	e	c	h	i	f	f
r	é	o	u	c	r	y	p	t	o	g	r	a	m	m	e			

Transposition

Implements a permutation of the plaintext letters $\mathcal{A}_C = A_M$.

$$\forall i, \quad 0 \leq i < 0 \quad \begin{array}{l} f : \mathcal{A}_M \to \mathcal{A}_M \\ \eta : \mathbb{Z}_n \to \mathbb{Z}_n \\ c_i = f(m_i) = m_{\eta(i)} \end{array}$$

Vernam cipher (1917)

Is the one-time pad a «perfect» cipher ?

A and B share a true random sequence of n bits : the secret key K.
A enciphers M of n bits in $C = M \oplus K$.
B deciphers C by $M = K \oplus C$.

Example
$M = 0011$, $K = 0101$
$C = 0011 \oplus 0101 = 0110$
$M = K \oplus C$.

Non-reusability : for every new message, we need a new key.

Why a new key ?

... To avoid revealing information on the $\oplus$ of plaintexts.

Eve can sniff $C = \{M\}_K$ and $C' = \{M'\}_K$ and computes :

$$C \oplus C' = (M \oplus K) \oplus (M' \oplus K) = M \oplus M'$$

Given enough ciphertexts, she's able to recover a plaintext by a frequency analysis and with the help of a dictionnary [2].

If we respect the above requirements, Vernam cipher guarantees the condition of perfect secrecy.

Condition (perfect secrecy)

$$Pr(M = m \mid C = c) = Pr(M = m)$$

Intercepting C doesn't reveal any information to the cryptanalyst

Why is it secure ?

Vernam ciphers provides **perfect secrecy**.
We have three classes of information :

- plaintexts M with proba. distribution $Pr(M)/\sum_M Pr(M) = 1$
- ciphertexts C with proba. distribution $Pr(C)/\sum_C Pr(C) = 1$
- keys with proba. distribution $Pr(K)$ s.t. $\sum_K p(K) = 1$

$Pr(M \mid C)$ = proba that M has been sent knowing that C was received (C is the corresponding ciphertext of M). The perfect secrecy condition is defined as

$$Pr(M \mid C) = Pr(M)$$

The interception of the ciphertext does not provide any information to the crypto-analyst.

Conclusion

Perfect secrecy but difficult to achieve

- generate truly random sequences
- huge keylength
- store them and share them with the recipients

example of use : «red phone».

Product and iterated ciphers

Improvement : combine substitutions and transpositions

A cipher is **iterated** if the ciphertext is obtained from repeated applications of a round function to the plaintext
At each round, we combine a round key with the plaintext.

Definition
In an iterated cipher with r rounds, the ciphertext is computed by repeated applications of a ***round function*** *g to the plaintext :*

$$C_i = g(C_{i-1}, K_i) \quad i = 1, \dots, r$$

*C_0 the plaintext, K_i round key and C_r the ciphertext.
Deciphering is achieved by inverting the previous equation. For a fixed K_i, g must be invertible.*

Special case, **Feistel ciphers**.

Feistel ciphers

A **Feistel cipher** with block size $2n$ and r rounds is defined by :

$$g : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^n$$

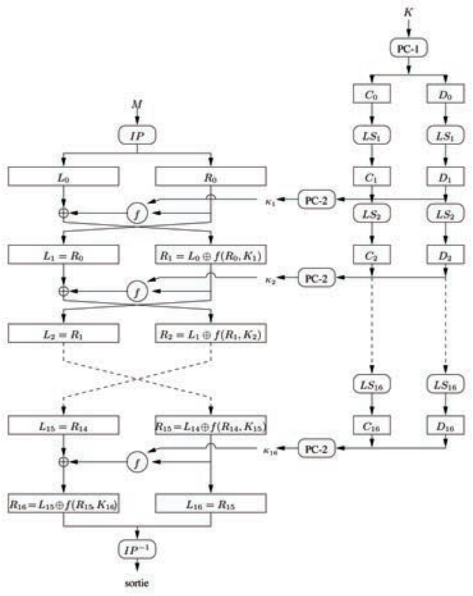
$$X, Y, Z \mapsto (Y, F(Y,Z) \oplus X)$$

g function of $2n \times m$ bits into $2n$ bits and $\oplus$ denoting the n bit XOR
Operation mode
Given a plaintext $P = (P^L, P^R)$ and r round keys $K_1, \ldots, K_r$, the ciphertext (C^L, C^R) is obtained after r rounds.
Let $C_0^L = P^L$ and $C_0^R = P^R$ and we compute for $i = 1, \ldots, r$

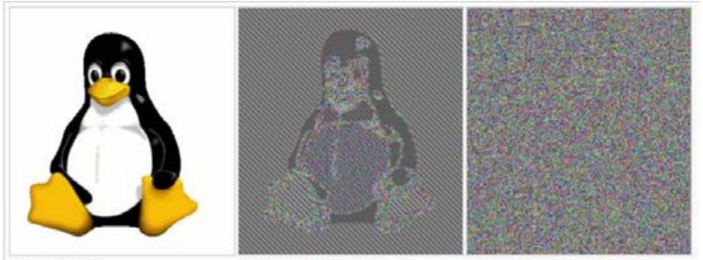
$$(C_i^L, C_i^R) = (C_{i-1}^R, F(C_{i-1}^R, K_i) \oplus C_{i-1}^L)$$

with $C_i = (C_i^L, C_i^R)$ and $C_r^R = C^L$ and $C_r^L = C^R$
The round keys $K_1, \ldots, K_r$, are obtained by a key scheduling algorithm on a master key K.

Block ciphers modes of operation



Modes of operation pictured


Original image

Encrypted using ECB mode

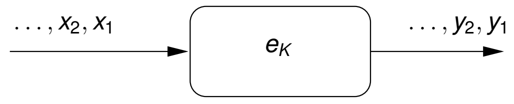
Modes other than ECB result in pseudo-randomness

The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".

http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

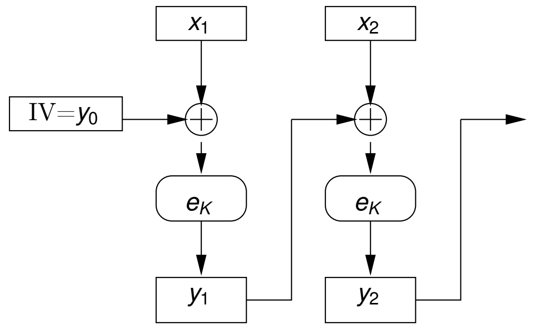
ECB : electronic codebook mode

The one previously used ; given a plaintext, each block x_i is enciphered with the key K, and provides the ciphertext $y_1 y_2 \dots$

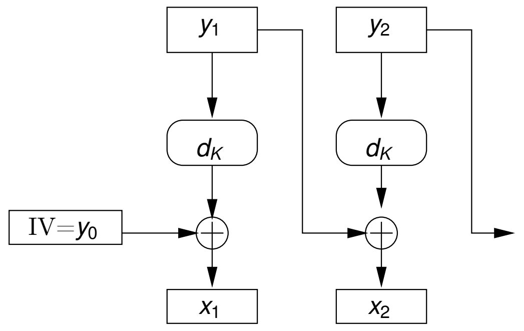


CBC : cipher block chaining mode

Each ciphertext y_i is XORed with next plaintext x_{i+1}



CBC – Deciphering

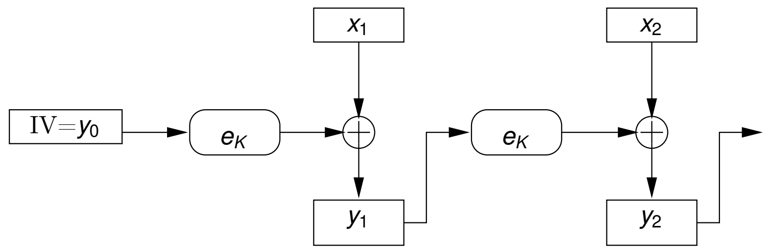


OFB (output feedback mode) and CFB (cipher feedback mode)

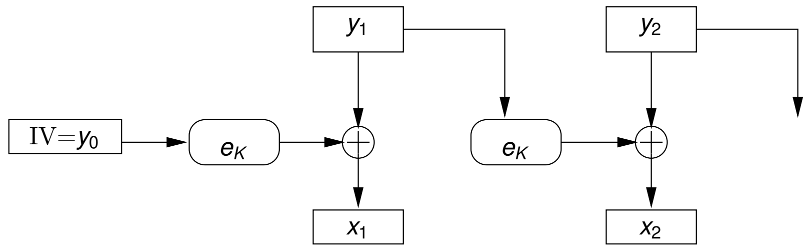
Encipher each plaintext block by successive XORing with keys coming from the application of a secret key cipher :

- **OFB** : sequence of keys comes from the repeated enciphering started on an initial value IV. We let z_0=IV and we compute the sequence $z_1 z_2 \dots$ by $z_i = e_K(z_{i-1})$. The plaintext is then enciphered by $y_i = x_i \oplus z_i$
- **CFB** : We start with y_0=IV and the next key is obtained by enciphering the previous ciphertext $z_i = e_K(y_{i-1})$. Otherwise, everything works like in OFB mode.

CFB enciphering

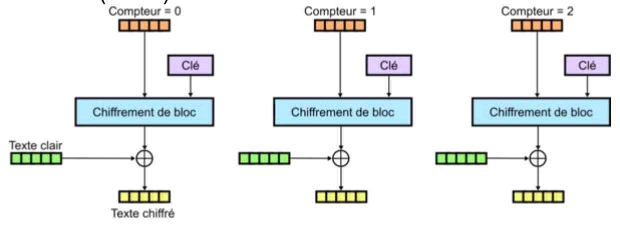


CFB deciphering



CTR Mode

The key stream is obtained by an iterative encryption of a counter value (CTR).



This operation mode is very useful. It allows the design of a stream cipher (though not perfectly secure) and pre-computation of the stream.

MAC-MDC

For Message Authentication Code (Modification Detection Code), or message fingerprint (MAC=MDC+IV$\neq$ 0).

Possible with CBC and CFB.
We start with IV=0. We build the ciphertext $y_1 \dots y_n$ with the key K in CBC mode. MAC is the last block y_n.
Alice sends the message $x_1 \dots x_n$ and the MAC y_n.
Upon reception of $x_1 \dots x_n$, Bob builds $y_1 \dots y_n$ by using the secret key K and verifies that y_n is the same than the received MAC.

Public Keys

Invented recently by Diffie and Hellman [3].

> *We stand today on the brink of a revolution in cryptography.*

Bright idea : asymmetrical ; enciphering $\neq$ deciphering.
Encipher by means of a **public** key.
Decipher by means of a **private** key.
Useful to solve the key distribution problem !
Kerckhoff principle (1883) still useful

> *The security of an algorithm must not depend upon the secrecy of the algorithm but only upon the secrecy of the key.*

http://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

What kind of security ?

Relies on *computational security*.

It means that the cryptanalyst must deploy more computational efforts to recover the plaintext than its life expectancy.

This gives challenges for breaking RSA keys :

- of 140 digits (463 bits in 1999) 2000 mips year
- of 155 digits (512 bits in 1999) 8000 mips year
- of 232 digits (768 bits in 2010)

http://infoscience.epfl.ch/record/173017/files/hetero.pdf
https://en.wikipedia.org/wiki/RSA_numbers

One-way function

Let M and C be two sets and $f : M \to C$ and $f(M)$ is the image of M by f. f is **one-way** if

1. $\forall x \in M$, the computation of $f(x)$ is easy (f poly-time computable) and
2. it is hard to find, for most of the $y \in f(M)$ an $x \in M$ such that $f(x) = y$ (this problem must be difficult [5, 7, 1]).

With only point 2., the deciphering problem is as hard as the cyptanalysis problem.

We need to add another notion for allowing decipherment and render the cryptanalyst's life as hard as possible.

$\to$ Trapdoor.

Trapdoor one-way function

$f : M \to C$ is a **trapdoor** function if it is one-way. Computing in the reverse direction is easy provided we have a private information, the trapdoor, which allows constructing g s.t. $g \circ f = Id$.

It is easy to compute the image by f but computationally hard to invert f without knowing g.

Constructing pairs (f, g) must be easy.
The publication of f should not reveal anything on g.

Idea : use two $\neq$ algorithms, f to encipher and g to decipher.

1.st PKC

1978 : RSA ; Rivest Shamir et Adleman were

- seeking a contradiction in the idea of public key
- successful to find the contrary and obtained the Turing award in 2002 !

http://amturing.acm.org/lectures.cfm

Some maths

Euler totient function of $n \in \mathbb{N}$: $\varphi(n)$: counts how many integers from $[\![1, n]\!]$ are prime with n. $\varphi(1) = 1$ and if p is prime, $\varphi(p) = p - 1$.

$$\varphi(n) = \text{card}\{j \in \{1, \dots, n\} : \gcd(j, n) = 1\}$$

Computation : factor n in $n = \prod_{p|n, p \text{ prime}} p^{\alpha_p}$ then, $\varphi(n) = \prod_{p|n, p \text{ prime}} (p^{\alpha_p} - p^{\alpha_p - 1}) = n \prod_{p|n} (1 - \frac{1}{p})$.
Example : $\varphi(12) = (4 - 2)(3 - 1) = 12(1 - \frac{1}{2})(1 - \frac{1}{3}) = 4$

Theorem (Fermat-Euler)

$$m^{\varphi(n)} \equiv 1 \mod n \text{ if } \gcd(m, n) = 1$$

Rivest, Shamir, Adleman (1978)

Relies on the hardness to factor an integer **and** on the hardness of deciding whether an integer is a prime.
For instance, is 1829 prime ?

No : given 31 and 59, their product equals 1829, but finding the factors is hard since we do not know either how many factors we need.

Or, is 7919 composite ?

No, but the primality certificate is hard to exhibit.

Compute $a^b \mod n$

$\langle b_k, b_{k-1}, \dots b_0 \rangle$ binary representation of b : $b = \sum_{i=0}^{k} b_i 2^i$.

```
Modular Exponentiation $(a, b, n)$
$c, d \leftarrow 0, 1$ ;
Let $\langle b_k, b_{k-1}, \dots b_0 \rangle$ the binary representation of $b$
For $i \leftarrow k$ to 0 step -1 do
    $d \leftarrow (d.d) \mod n$;
    if $b_i = 1$ then
        $d \leftarrow (d.a) \mod n$;
return $d$
```

```python
def expMod(a,b,n):
d = 1
for i in bin(b)[2:] :
    d = (d*d) % n
    if i == '1':  d = (d*a) % n
return(d)
```

By hand

$17^{73} \mod 100$. $73 = \langle 1001001 \rangle$

i	b_i	17^{2^i}	$17^{2^i} \mod 100$	value
0	1	17	$17 \mod 100$	$\boxed{17}$
1	0	17^2	$289 \mod 100$	89
2	0	23^2	$7921 \mod 100$	21
3	1	130^2	$441 \mod 100$	$\boxed{41}$
4	0	9^2	$1681 \mod 100$	81
5	0	81^2	$6561 \mod 100$	61
6	1	44^2	$3721 \mod 100$	$\boxed{21}$

and $17^{73} \mod 100 = 17.17^{2^3}.17^{2^6} = 17.41.21 \mod 100 = 37$.

Attack on the parameters

Cycles : Eve observes $c = m^e \mod n$; he tries to find out ν st.

$$c^{e^\nu} \equiv c \mod n \Leftrightarrow e^\nu \equiv 1 \mod \varphi(n)$$

Allowing to find $m \equiv c^{e^{\nu-1}} \mod n$
Since $c^{e^\nu} \equiv c \mod n \Leftrightarrow c^{e^\nu - 1} \equiv 1 \mod n$ and, by Euler-Fermat, one gets $e^\nu - 1 \equiv 0 \mod \varphi(n) \Leftrightarrow e^\nu \equiv 1 \mod \varphi(n)$. Since $c = m^e \mod n$ and $de \equiv 1 \mod \varphi(n)$, we can take the value $d = e^{\nu-1}$ to decipher..
Example : Alice publishes her public parameters e et n, 17 and 143. Eve sniffs $c = 19$ a message to Alice and computes :

i	2	3	4
c^{e^i}	84	28	19

Eve just has to read m for $i = 3$, thus 28.

RSA cipher

1. choose p, q primes relatively large approx. 10^{100}
2. compute $n = pq$ and publish n
3. compute $\varphi(n) = (p-1)(q-1)$
4. publish e st $\gcd(e, \varphi(n)) = 1$ (PK, $\underline{e}$*ncipher*)
5. compute d st $d.e \equiv 1 \mod \varphi(n)$ (private key, $\underline{d}$*ecipher*)

Encipher : $E : M \mapsto M^e \mod n$.
Decipher : $D : C \mapsto C^d \mod n$ (d is the trapdoor).
Implementations : software, hardware or mixed.
On dedicated hardware, RSA is 1000 times slower than DES.

Attack when $\varphi(n)$ is known

Given $(n, \varphi(n))$ allows to find the factorization of n [5].
We let : $\begin{cases} n = pq \\ \varphi(n) = (p-1)(q-1) \end{cases}$ and $q = \frac{n}{p}$:

$$\varphi(n) - (p-1)\left(\frac{n}{p} - 1\right) = 0 \Leftrightarrow p^2 + p\left(\varphi(n) - n - 1\right) + n = 0$$

equation of order two with solutions p and q.
Thus, computing $\varphi(n)$ is as hard as factoring n.

Example

$n = p.q = 133$ *and* $\varphi(n) = 108$. $\varphi(n) - (p-1)\left(\frac{n}{p} - 1\right) = 0$
$\Leftrightarrow p^2 + p\left(\varphi(n) - n - 1\right) + n = p^2 + p(108 - 133 - 1) + 133 = 0 \Leftrightarrow p^2 - 26.p + 133 = 0$ *with*
$\Delta = (-26)^2 - (4.133) = 144 = 12^2$ *and of solutions*
$p = \frac{26 \pm 12}{2} = \{19, 7\}$.

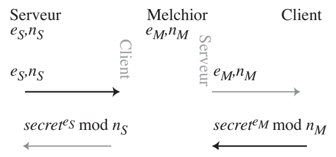
Sieve of Eratosthenes

Divide n by all odd numbers between 3 and $\lfloor\sqrt{n}\rfloor$.
Efficient for $n < 10^{12}$ and known since ancient times.
Sieve of Eratosthenes runs in time $O(\sqrt{n})$.
It's not polynomial ! The time-complexity is not polynomial in the length of the input. It is **pseudo polynomial**.
In addition, in the case of RSA, the modulus n has no small prime factors.

Man in the middle

In the transmission of the public keys :

- Bob (client) asks Alice (server) for her public parameters
- Alice sends e_S, n_S to Bob
- Melchior intercepts e_S, n_S ; replaces by its values e_M, n_M
- Bob enciphers by using e_M, n_M and sends c
- Melchior intercepts c and deciphers it into *secret*
- Melchior enciphers *secret* with Alice's parameters e_S, n_S and transmits to Alice. . .



Bob should have checked that the data were coming from Alice (lack of authentication).

Security

RSA is as secure as factoring n is hard.
Time complexity of some good factoring algorithms :

quadratic sieve	$O(e^{((1+o(1))\sqrt{\log n \log\log n})})$
elliptic curves	$O(e^{((1+o(1))\sqrt{2\log p \log\log p})})$
algebraic sieve	$O(e^{((1{,}92+o(1))(\log n)^{1/3}(\log\log n)^{2/3})})$

(p : smallest prime factor of n).

Another hard problem

The **discrete log** problem.
Find the discrete log of y in basis g :

Instance : g, y elements of a finite group G.

Question : find x st $g^x \equiv y$ in G
or, for a big prime p, g a generator of $G = \mathbb{Z}_p^\star$, $g^x \equiv y \mod p$ and $x = \log_g(y) \mod p-1$.

Example

Let $G = \mathbb{Z}_7^\star$ a cyclic group. For the discrete logarithm in basis 2, only $1, 2$ and 4 have a discrete log. In basis g=3, we have :

number y	1	2	3	4	5	6
logarithm	6	2	1	4	5	3

For instance for number = 1 and log = 6. This means that $\log_3 1 = 6$, which can be checked with $3^6 \mod 7 = 1$.

```python
from sympy import *
g,n,r,y=3,113,11,57
B=[(i,pow(g,i,n)) for i in range(r)]
L=[(j,y*gcdex(pow(g,j*r,n),n)[0]\%n) for j in range(
r+1)]
B=sorted(B, key=lambda x:x[1])
L=sorted(L, key=lambda x:x[1])
print(B,L)
```

```
[(0, 1), (1, 3), (8, 7), (2, 9), (5, 17), (9, 21), (3, 27), (7, 40), (6, 51), (10, 63), (4, 81)]
[(8, 2), (9, 3), (6, 26), (1, 29), (11, 35), (3, 37), (7, 39), (5, 55), (0, 57), (10, 61), (2, 100), (4, 112)]
```

```python
1+r*9
```

```
100
```

Computing the discrete log

Becomes hard when the cardinal of G grows.
Algo for computing the discrete log : Shanks applies to every finite group G. Its time complexity is $O(\sqrt{|G|}\log|G|)$ and its space complexity is $O(\sqrt{|G|})$.
Idea : construct two lists of the powers of g :

- baby steps : $\{g^i : i = 0..\lceil\sqrt{n}\rceil - 1\}$ with $n = |G|$
- giant steps $\left\{y\left(g^{-\lceil\sqrt{n}\rceil j}\right) : j = 0..\lceil\sqrt{n}\rceil\right\}$.

Then find a common term to the two lists. Then,

$$g^{i_0} = y(g^{-j_0\lceil\sqrt{n}\rceil}) \text{ and } m = i_0 + j_0\lceil\sqrt{n}\rceil$$

Example

In $\mathbb{Z}_{113}^\times =< 3 >$ of order $n = 112$; $\sqrt{n} = r = 11$. We search the discrete log of $y = 57$ in basis $g = 3$:
Unordered list of baby steps by (exponent, value) :

$$B = \{(0,1), (\mathbf{1}, \mathbf{3}), (2,9), (3,27), (4,81), (5,17), (6,51), (7,40), (8,7), (9,21), (10,63)\}$$

Unordered list of giant steps by (exponent, value)

$$L = \{(0,57), (1,29), (2,100), (3,37), (4,112), (5,55), (6,26), (7,39), (8,2), (\mathbf{9}, \mathbf{3}), (10,61), (11,35)\}$$

3 is common to both lists. It has been generated for $i_0 = 1$ in the list B and for $j_0 = 9$ in the list L.
The value of the discrete log is $x = i_0 + r.j_0 = 100$. Verification : we compute $g^x \mod 113 = 57$.

Other objectives of PKC

- **secrecy**
- **authentication :** proof of origin authenticity
- **identification :** electronic proof of its own identity
- **integrity :** guarantee that there was no modification
- **non repudiation :** A service that provides proof of the integrity and origin of data.

Other cryptographic techniques are required

- **signature :** the way to associate the sender to a message
- **certificate :** guarantees the relation (identity, PK)
- **trusted third party :** authority who delivers certificates
- **timestamps :** append timestamps to grant uniqueness of the message.

Requirements for **sig**(M)

- easy to compute by the sender for every message M
- the recipient must be able to check the signature
- a third party must be able to check the signature
- the signature must be hard to forge
- the sender should not be able to say that his signature was forged

Signatures

Notion introduced in the Diffie and Hellman seminal paper [3].

Goal of the signatures : prove the sender's **identity** and provide **integrity** of the message. The signature depends upon the sender's identity and on the message contents.

Must counter two kinds of frauds

- message modification
- change the origin of the message (sender's identity)

General mechanism for signatures

- a private algorithm for signing denoted **sig** which, given a fixed key SK, returns a signature S for the plaintext M ;

$$\mathrm{sig}_{SK}(M) = S$$

- a verification algorithm **ver** which, given a fixed key PK and for every pair plaintext/signature (M, S) checks if the signature corresponds to the plaintext.

$$\mathrm{ver}_{PK}(M, S) = \begin{cases} \text{true if } S = \mathrm{sig}_{SK}(M) \\ \text{false if } S \neq \mathrm{sig}_{SK}(M) \end{cases}$$

Signing with RSA

Bob wants to send a signed message M to Alice. They have their respective RSA parameters :

	Private	Public
Alice	d_A	n_A, e_A
Bob	d_B	n_B, e_B

Signing algorithm :

$$\text{sig}_{SK}(M) = M^{d_B} \mod n_B = S$$

Verification algorithm :

$$\text{ver}_{PK}(M, S) = \text{true} \Leftrightarrow S^{e_B} \mod n_B \equiv M$$

El Gamal Signature

Let p be a prime for which the discrete log problem is hard in $\mathbb{Z}_p^\star$ and let α be a generator of $\mathbb{Z}_p^\star$.
The message $M \in \mathbb{Z}_p^\star$ and its signature is made of the pair $(M, S) \in \mathbb{Z}_p^\star \times (\mathbb{Z}_p^\star \times \mathbb{Z}_{p-1})$. The set of keys is

$$K = \{(p, \alpha, a, \beta) : \beta = \alpha^a \mod p\}$$

Private	Public
a	p, α, β

Randomly choose $k \in \mathbb{Z}_{p-1}^\star$; keep it secret ; k is st $\gcd(k, p-1) = 1$.
Signing algorithm :

$$\text{sig}_K(M, k) = (\gamma, \delta)$$

for $\gamma = \alpha^k \mod p$ $\qquad \delta/a\gamma + k\delta \equiv M \mod (p-1)$

RSA allows secrecy + authentication

How can Bob send an authenticated secret message to Alice ?

	Private	Public
Alice	$D_A(C) = C^{d_A} \mod n_A$	$E_A(M) = M^{e_A} \mod n_A$
Bob	$D_B(C) = C^{d_B} \mod n_B$	$E_B(M) = M^{e_B} \mod n_B$

Bob sends

$$C = E_A(D_B(M))$$

which is deciphered by Alice :

$$E_B(D_A(C))$$

provided that $M < n_B < n_A$.

Example

Let $p = 467$ and $a = 127$. We check that $\gcd(a, p-1) = 1$. Let $\alpha = 2$ be a generator of $\mathbb{Z}_p^\times$. We compute

$$\beta = \alpha^a \mod p = 2^{127} \mod 467 = 132$$

If Bob wants to sign the message $M = 100$ for the random value $k = 213$ which verifies $\gcd(k, p-1) = 1$, he computes the multiplicative inverse $k^{-1} \mod p-1$ by the Extended Euclidean algo which gives $k^{-1} = 431$. Then,

$$\gamma = \alpha^k \mod p = 2^{213} \mod 467 = 29$$

and

$$\delta = (M - a\gamma)k^{-1} \mod (p-1) = (100 - 127.29).431 \mod 466 = 51$$

Verification

Given $M, \gamma \in \mathbb{Z}_p^\star$ and $\delta \in \mathbb{Z}_{p-1}$, we define

$$\text{ver}_K(M, \gamma, \delta) = \text{true} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^M \mod p$$

If the signature is correct, the verification algorithm validates the signature since :

$$\beta^\gamma \gamma^\delta \equiv \alpha^{a\gamma} \alpha^{k\delta} \mod p \equiv \alpha^M \mod p$$

since $a\gamma + k\delta \equiv M \mod (p-1)$.

Exemple : We verify the signature $(100, 29, 51)$:

$\text{ver}_K(M, \gamma, \delta) = \text{true} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^M (p) \Leftrightarrow 132^{29} 29^{51} \equiv 2^{100} (p) \equiv 189$

which is correct

G. Brassard.
Cryptologie contemporaine.
Logique, mathématiques, informatique. Masson, 1993.

E Dawson and L Nielsen.
Automated cryptanalysis of xor plaintext strings.
Cryptologia, XX(2) :165–181, May 1996.

W. Diffie and M.E. Hellman.
New directions in cryptography.
IEEE Trans. on Inform. Theory, 22(6) :644–654, 1976.

D. Kahn.
La guerre des codes secrets.
InterEditions, 1980.

N. Koblitz.
A course in number theory and cryptography.
Graduate texts in mathematics. Springer Verlag, 1987.

R.L. Rivest.
Cryptography.
In *Handbook of Theoretical Computer Science*, volume A, chapter 13. Elsevier, 1990.

A. Salomaa.
Public Key Cryptography.
EATCS monographs. Springer Verlag, 1990.

J. Stern.
La science du secret.
Odile Jacob, 1998.