Public Key Cryptography

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`

Public Keys

Invented recently by Diffie and Hellman [2].

> *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`

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 [3, 4, 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.

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.

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, $\varphi(p) = p - 1$.

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

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)

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

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

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, *<u>e</u>ncipher*)
5. compute d st $d.e \equiv 1 \mod \varphi(n)$ (private key, *<u>d</u>ecipher*)

Encipher : $E : M \mapsto M^e \mod n$ provided $M < n$
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.

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	89^2	7921 mod 100	21
3	1	21^2	441 mod 100	$\boxed{41}$
4	0	41^2	1681 mod 100	81
5	0	81^2	6561 mod 100	61
6	1	61^2	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 $c = m^e \mod n$; she 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.

Attack when $\varphi(n)$ is known

Given $(n, \varphi(n))$ allows to find the factorization of n [3].

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
$p = \frac{26 \pm 12}{2} = \{19, 7\}$.*

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).

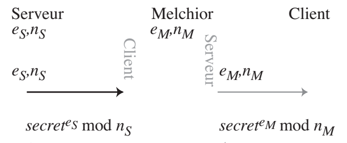
Sieve of Eratosthenes

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).

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 large prime p, g a generator of $G = \mathbb{Z}_p^\star$, $g^x \equiv y \mod p$
and $x = \log_g(y) \mod p-1$.

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

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 $3^6 \mod 7 = 1$.

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 by Diffie and Hellman in [2].

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 :

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

Verification algorithm :

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

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 $\mathrm{gcd}(k, p-1) = 1$.
Signing algorithm :

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

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

RSA allows secrecy and 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 $M < n_B < n_A$.
Why isn't it a signature ?

Example

Let $p = 467$ and $a = 127$. We check that $\mathrm{gcd}(a, p-1) = 1$. 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 $\mathrm{gcd}(k, p-1) = 1$, 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 $a\gamma + k\delta \equiv M \mod (p-1)$.

Example : 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.

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

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

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