RECAP

- PUBLIC-KEY CRYPTOSYSTEM

$\forall X$ (user): $P_X(M)$, $S_X(M) = P_X^{-1}(M)$ $M \in \mathcal{D}$

  - easily computable given P_X, S_X
  - difficult to compute $S_X(M)$, given $P_X(M)$

- PUBLIC-KEY PROTOCOLS
  - SECRET MESSAGE PASSING

    BOB $\rightarrow$ $Y = P_A(M)$ $\rightarrow$ ALICE $\quad (M = S_A(Y))$
  - AUTHENTICATION

    ALICE $\rightarrow$ $(X, Y) = (M, S_A(M))$ $\rightarrow$ BOB $\quad \left(X \stackrel{?}{=} P_A(Y) \right)$
  - COMBINED PROTOCOLS

    ALICE $\rightarrow$ $Z = P_B(\langle M, S_A(M) \rangle)$ $\rightarrow$ BOB $\begin{cases} \langle x, y \rangle = S_B(Z) \\ x \stackrel{?}{=} P_A(y) \end{cases}$

- ASYMMETRY

DIFFERENT COMPLEXITY OF

LARGE PRIMES: Given m, determine prime $p \geq m$

FACTORING: Given $m = p \cdot q$, $p, q \gg 1$, determine p, q

EASY

DIFFICULT

RSA: DETAILS

Each participant X:

1. Picks two random large primes ($\nearrow$ a random) $\boxed{p, q}$ of a prespecified size (the larger, the more secure, + lento) (currently: p, q of ~ 1024 bits)

   security level

2. Computes $\boxed{n = p \cdot q}$ and sets $\boxed{\mathcal{D} = \mathbb{Z}_n}$

   NOTE: Given only n, p and q are difficult to compute

3. Computes

$$
\boxed{
\begin{aligned}
\varphi(n) &= n \cdot \prod_{\substack{p' \mid n \\ p' \text{ prime}}} \left(1 - \frac{1}{p'}\right) = \\
&= pq\left(1-\frac{1}{p}\right)\left(1-\frac{1}{q}\right) = \\
&= (p-1)(q-1) = \\
&= pq - (p+q) + 1 = n - (p+q) + 1
\end{aligned}
}
$$

4. Selects a "small integer" $\boxed{e}$, with $\boxed{\gcd(e, \varphi(n)) = 1}$ (usually a "small" prime)

6. Computes

$$\boxed{d = e^{-1} \text{ in } \mathbb{Z}^*_{\varphi(n)}}$$

(d exists since $\gcd(e, \varphi(n)) = 1$)

7. Sets $\boxed{P_x = (e, n), \quad S_x = (d, n)}$ and:

$$\boxed{\begin{aligned} \forall M \in \mathbb{Z}_n : P_x(M) &= M^e \bmod n \\ S_x(M) &= M^d \bmod n \end{aligned}}$$

1. Given $P_x = (e, n)$, $S_x = (d, n)$

$P_x(M) = M^e \bmod n$, $S_x(M) = M^d \bmod n$

can be computed efficiently via squaring. Recursive algorithm based on the following substructure property (squaring):

$$M^e \bmod n = \begin{cases} 1 & e = 0 \\ M & e = 1 \\ \left(M^{\lfloor e/2 \rfloor}\right)^2 \bmod n & e \geq 1 \text{ even} \\ \left[\left(M^{\lfloor e/2 \rfloor}\right)^2 \cdot M\right] \bmod n & e \geq 1 \text{ odd} \end{cases}$$

$$e = \left(\frac{e-1}{2}\right) \cdot 2 + 1 \longrightarrow$$

```
MOD_POWER(M, x, n)
 if (x=0) then return 1
 if (x=1) then return M
 temp ← MOD_POWER(M, ⌊x/2⌋, n)
 if even(x)
   then return (temp·temp) mod n
   else return (temp·temp·M) mod n
```

$M \in \mathbb{Z}_n \Rightarrow M = 0, 1, \ldots, n-1$

$\Rightarrow |\langle M \rangle| = |\langle n \rangle| \; (= O(\log n))$

$K = |\langle x \rangle| \Rightarrow |\langle \lfloor \frac{x}{2} \rfloor \rangle| = K - 1$

$T(K) = T(K-1) + \Theta(|\langle n \rangle|^2)$

2 products
1 mod

$T(K) = K \cdot |\langle n \rangle|^2$

if $K = \Theta(|\langle n \rangle|)$

$\Rightarrow \Theta(|\langle n \rangle|^3)$

if $K = O(1)$

$\Rightarrow \Theta(|\langle n \rangle|^2)$

Nonrecursive implementation (will be used for PRIMALITY).

Let $\boxed{(x)_2 = (x_k, x_{k-1}, ---, x_0) \equiv \vec{x}}$

Since $\boxed{x = \sum_{j=0}^{k} x_j 2^j}$, we can obtain x from $\vec{x}$ as follows:

```
BIN2NUM(x⃗)            {conversion}
  k ← x⃗.length − 1
  c ← 0
  for i ← k downto 0 do
    c ← 2·c            {shift c leftward}
    if (x_i = 1) then c ← c+1
                       {insert 1 in i-th pos.}
  return c   {c = x}
```

We can obtain modular exponentiation by a simple modification of BIN2DEC

```
MOD-EXP(M, x⃗, n)
  k ← length(x⃗) − 1
  d ← 1    {c ← 0 : INVARIANT : d = M^c mod n}
  for i ← k downto 0 do
    d ← d·d mod n    {c ← 2c : M^c · M^c mod n = M^2c mod n}
    if (x_i = 1) then d ← (d·M) mod n
                     {c ← c+1 : M^c · M mod n = M^(c+1) mod n}
  return d           {d = M^x mod n}
```

COMPLEXITY

Number of modular operations:

$$\boxed{\mathcal{O}(k) = O(\log x) = O(|\langle x\rangle|)}$$

Public encoding: $|\langle e\rangle| = O(1)$

Secret encoding: $|\langle d\rangle| = O(|\langle n\rangle|)$

Since each mod costs $O(|\langle n\rangle|^2)$
the running time is either
quadratic (public encoding) or
cubic (secret encoding)

$\Rightarrow$ RSA is rather heavy when $|\langle n\rangle|$ grows (currently $|\langle n\rangle|$ around 10^3)

CORRECTNESS OF RSA

We have to prove that:

$$\boxed{\forall M \in \mathbb{Z}_n : S_x(P_x(M)) = P_x(S_x(M)) = M}$$

But

$$\boxed{\begin{aligned} S_x(P_x(M)) &= \left((M^e \bmod n)^d\right) \bmod n \overset{m2}{=} \\ &= M^{ed} \bmod n = P_x(S_x(M)) \end{aligned}}$$

Recall that

$$\boxed{d = e^{-1} \bmod \varphi(n) \Rightarrow ed \equiv 1 \bmod \varphi(n)}$$

Thus $\boxed{\exists\, h \in \mathbb{Z} : \ ed = 1 + h\varphi(n) = 1 + h(p-1)(q-1)}$

and

$$\boxed{M^{ed} = M^{1+h(p-1)(q-1)}}$$

PROOF IDEA

To prove that $\boxed{M \equiv M^{ed} \mod n}$, we will prove that $\boxed{\begin{array}{l} M \equiv M^{ed} \mod p \\ M \equiv M^{ed} \mod q \end{array}}$ and

Then $M \equiv M^{ed} \mod n$ by COROLLARY 2 of the C.R.T.

Consider $\boxed{M^{ed} \mod p = M^{1+h(p-1)(q-1)} \mod p}$

$$\boxed{\overset{m2}{=} \left((M \mod p) \left((M^{p-1} \mod p)^{h(q-1)} \mod p \right) \mod p \right)}$$

Two cases: rivela molto su sistema

1. $\boxed{\begin{array}{l} M \mod p = 0 \Rightarrow M^{ed} \mod p = 0 \\ \Rightarrow M \equiv M^{ed} \mod p \end{array}}$

2. $\boxed{M \mod p \neq 0.}$ By Fermat's little theorem:

$$\boxed{\begin{array}{l} M^{p-1} \mod p = 1 \Rightarrow \\ \left((M^{p-1} \mod p)^{h(q-1)} \mod p \right) = 1 \Rightarrow \\ M^{ed} \mod p = M \mod p, \text{ hence} \\ M \equiv M^{ed} \mod p \end{array}}$$

Using the same line of reasoning we can prove that

$$\boxed{M \equiv M^{ed} \bmod q}$$

Thus $\boxed{M = M^{ed} \bmod n}$, which proves that $P_x(M)$ and $S_x(M)$ are the inverse of each other!

COMPLEXITY CONSIDERATIONS

- If FACTORING $\in$ P then RSA is unsecure (a cryptoanalyst would compute $\varphi(n) = (p-1)(q-1)$ in polynomial time.
- Could we compute $\varphi(n)$ efficiently without being able to factor n? NO!

PROPERTY If $(\boxed{n, \varphi(n)})$ are known with $\boxed{n = p \cdot q}$, then p and q can be computed in poly time.

PROOF

We know that

$$\boxed{\begin{aligned}\varphi(n) &= (p-1)(q-1) = pq - (p+q) + 1 = \\ &= n - (p+q) + 1 \Rightarrow p+q = n - \varphi(n) + 1\end{aligned}}$$

Moreover $(p-q)^2 = p^2+q^2-2pq \pm 2pq$

$$= p^2+q^2+2pq-4pq = (p+q)^2-4n =$$

$$= (n-\varphi(n)+1)^2-4n$$

$$\Longrightarrow p-q = +\sqrt{(n-\varphi(n)+1)^2-4n}$$

Since $n, \varphi(n)$ are known, let

$\boxed{K = n-\varphi(n)+1}$ (K is known).

Then

$$\begin{cases} p+q = K \\ p-q = \sqrt{K^2-4n} \end{cases}$$

known

is a linear system yielding

$$p = \left(K+\sqrt{K^2-4n}\right)/2$$

$$q = \left(K-\sqrt{K^2-4n}\right)/2$$

MORALE Factoring n or computing $\varphi(n)$ are computationally equivalent

- RSA could be crackable even if FACTORING were difficult!

E.g. Computation of the discrete logarithm

given $\boxed{a, b \in \mathbb{Z}_n}$ determine $\boxed{x \in \mathbb{N} : a^x \equiv b \bmod n}$

(x is like the "logarithm" to the base a of b in $\mathbb{Z}_n$)

This is a difficult problem (like factoring, no efficient algorithms)

Cracking RSA through discrete logarithm:

We know that if

$$\boxed{\begin{array}{l} P_x(M) = M^e \bmod n \\ S_x(M) = M^d \bmod n \end{array}}$$

and

then

$$\boxed{P_x(S_x(M)) = M^{ed} \bmod n = M.}$$

For any $\boxed{M \in \mathbb{Z}_n - \{1\}}$,

compute $\boxed{a = M^e \bmod n}$

and set $\boxed{b = M}$

$$\boxed{\begin{array}{l} \text{Let } x: \quad a^x \equiv b \bmod n \\ \qquad\qquad\quad \Updownarrow \\ \qquad\qquad M^{ex} = M \\ \Rightarrow \quad x \equiv d \equiv e^{-1} \bmod \varphi(n) \end{array}}$$