RECAP

- **PUBLIC-KEY CRYPTOSYSTEM**

$\forall x$ (user): $P_x(M)$, $S_x(M) = P_x^{-1}(M)$ $\quad M \in \mathcal{D}$

  - 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 $\quad (X \overset{?}{=} P_A(Y))$

  - COMBINED PROTOCOLS

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

- **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 $\qquad$ DIFFICULT

RSA: DETAILS

Each participant X:

1. Picks two random large primes p, q of a prespecified size (the larger, the more secure) (currently: p, q of ~ 1024 bits)

   security level

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

   $\boxed{\text{NOTE: Given only } n,\ p \text{ and } q \text{ are difficult to compute}}$

3. Computes

$$\varphi(n) = n \cdot \prod_{\substack{p' \mid n \\ p' \text{ prime}}} \left(1 - \frac{1}{p'}\right) =$$

$$= pq\left(1-\frac{1}{p}\right)\left(1-\frac{1}{q}\right) =$$

$$= (p-1)(q-1) =$$

$$= pq - (p+q) + 1 = n - (p+q) + 1$$

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

6. Computes
$$d = e^{-1} \text{ in } \mathbb{Z}^*_{\varphi(n)}$$
(d exists since $\gcd(e, \varphi(n)) = 1$)

7. Sets $P_X = (e, n)$, $S_X = (d, n)$ 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
```

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

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

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

$$T(K) = T(K-1) + \underbrace{O}_{\substack{\text{2 products} \\ \text{1 mod}}}(|\langle n \rangle|^2)$$

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

$$\text{if } K = O(|\langle n \rangle|)$$

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

$$\text{if } K = O(1)$$

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

Nonrecursive implementation (will be used for PRIMALITY).

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

Since $x = \sum_{j=0}^{k} x_j 2^j$, 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:
$\mathcal{O}(k) = \mathcal{O}(\log x) = \mathcal{O}(|\langle x\rangle|)$

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

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

Since each mod costs $\mathcal{O}(|\langle n\rangle|^2)$ 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:

$\forall M \in \mathbb{Z}_n : S_X(P_X(M)) = P_X(S_X(M)) = M$

But

$S_X(P_X(M)) = \left((M^e \bmod n)^d\right) \bmod n \overset{\text{m2}}{=}$

$= M^{ed} \bmod n = P_X(S_X(M))$

Recall that

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

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

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

PROOF IDEA

To prove that $M \equiv M^{ed} \bmod n$, we will prove that $M \equiv M^{ed} \bmod p$ and
$M \equiv M^{ed} \bmod q$

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

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

$\overset{m2}{=} \Big((M \bmod p)\big(((M^{p-1} \bmod p)^{h(q-1)} \bmod p\big) \bmod p\Big)$

Two cases:

1. $M \bmod p = 0 \Rightarrow M^{ed} \bmod p = 0$
$\Rightarrow M \equiv M^{ed} \bmod p$

2. $M \bmod p \neq 0$. By Fermat's little theorem:
$M^{p-1} \bmod p = 1 \Rightarrow$
$\big((M^{p-1} \bmod p)^{h(q-1)} \bmod p\big) = 1 \Rightarrow$
$M^{ed} \bmod p = M \bmod p$, hence
$M \equiv M^{ed} \bmod p$

Using the same line of reasoning we can prove that

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

Thus $M = M^{ed} \bmod n$, 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 $(n, \varphi(n))$ are known with $n = p \cdot q$, then p and q can be computed in poly time.

PROOF

We know that

$$\varphi(n) = (p-1)(q-1) = pq - (p+q) + 1 =$$

$$= n - (p+q) + 1 \Rightarrow p+q = n - \varphi(n) + 1$$

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

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

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

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

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

$K = n-\varphi(n)+1$ (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 $a, b \in \mathbb{Z}_n$ determine $x \in \mathbb{N}$: $a^x \equiv b \bmod n$

(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 $P_x(M) = M^e \bmod n$ and

$S_x(M) = M^d \bmod n$

then

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

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

compute $a = M^e \bmod n$

and set $b = M$

Let x : $a^x \equiv b \bmod n$

$$\Updownarrow$$

$$M^{ex} = M$$

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