Lecture 6: The RSA Cryptosystem and Factoring Integers

-Cryptographic Algorithms and Protocols

Huang, Xiujie (黄秀姐)

Office: Nanhai Building, #411

E-mail: t_xiujie@jnu.edu.cn

Dept. Computer Science

Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Attacks on RSA**
- **6. The Rabin Cryptosystem**
  - **Turing Reduction**
- **7. Semantic Security of RSA**

Why PKC?

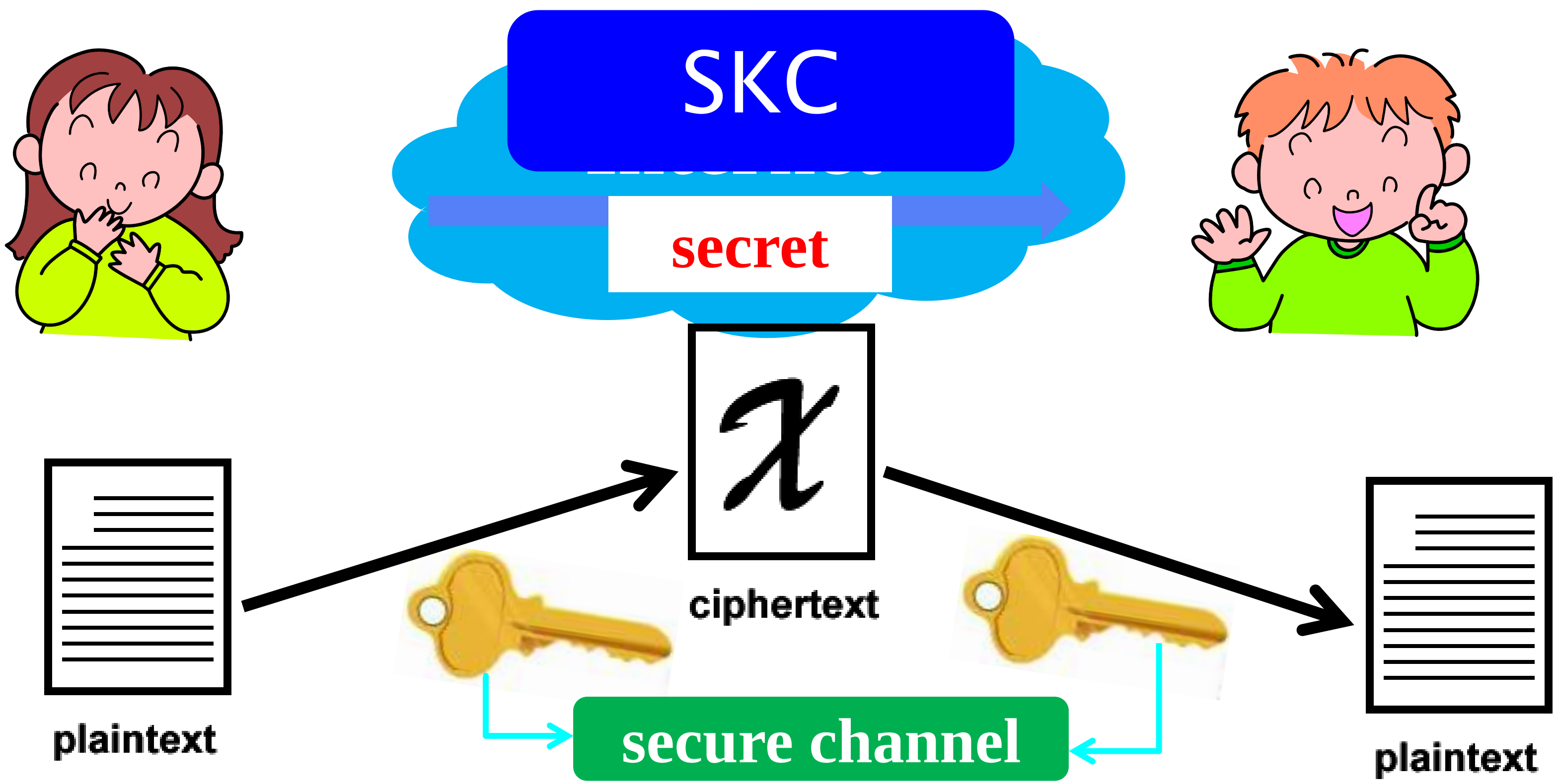
Question:

Alice wants to send Bob a **secret** **through Internet** **against Hacker**.

Why PKC?

Solution 1: SKC

Drawback: **The key should be transmitted in a secure channel and kept in a secure place.**

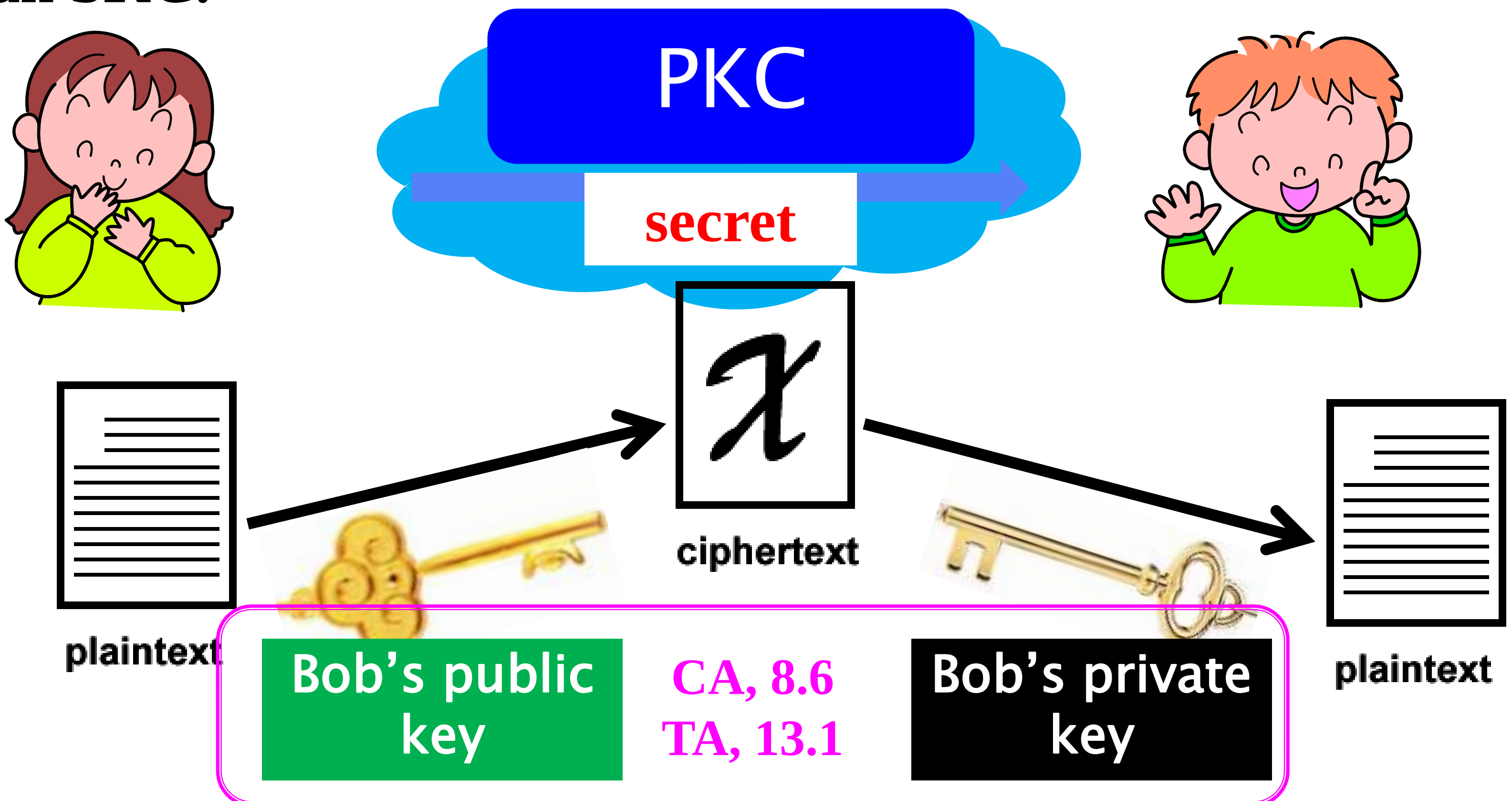


Why PKC?

Solution 2: PKC

Advantage: The encryption key is public.

Disadvantage: The computation complexity is usually higher than SKC.



Why PKC?

- **The Hybrid Cryptography: an intuitive combination of SKC and PKC**
  - **Use SKC to encrypt “long” message**
  - **Use PKC to encrypt the secret key of SKC.**

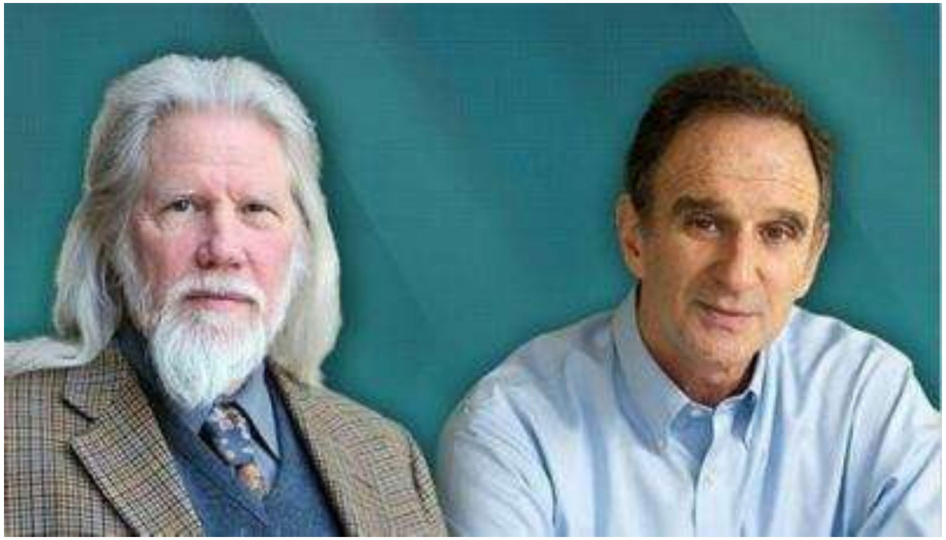
    (Key management is one important application of PKC.)

Brief History of PKC

- **1970, James Ellis,** **"The possibility of non-secret encryption" (PKC)**
- **1973, Clifford Cocks,** **"A note on non-secret encryption" (RSA)**

released in 1997

- **1976, Diffie and Hellman, the idea of PKC**



Bailey Whitfield Diffie

Martin Edward Hellman

Brief History of PKC

- **1970, James Ellis,** **"The possibility of non-secret encryption" (PKC)**
- **1973, Clifford Cocks,** **"A note on non-secret encryption" (RSA)**

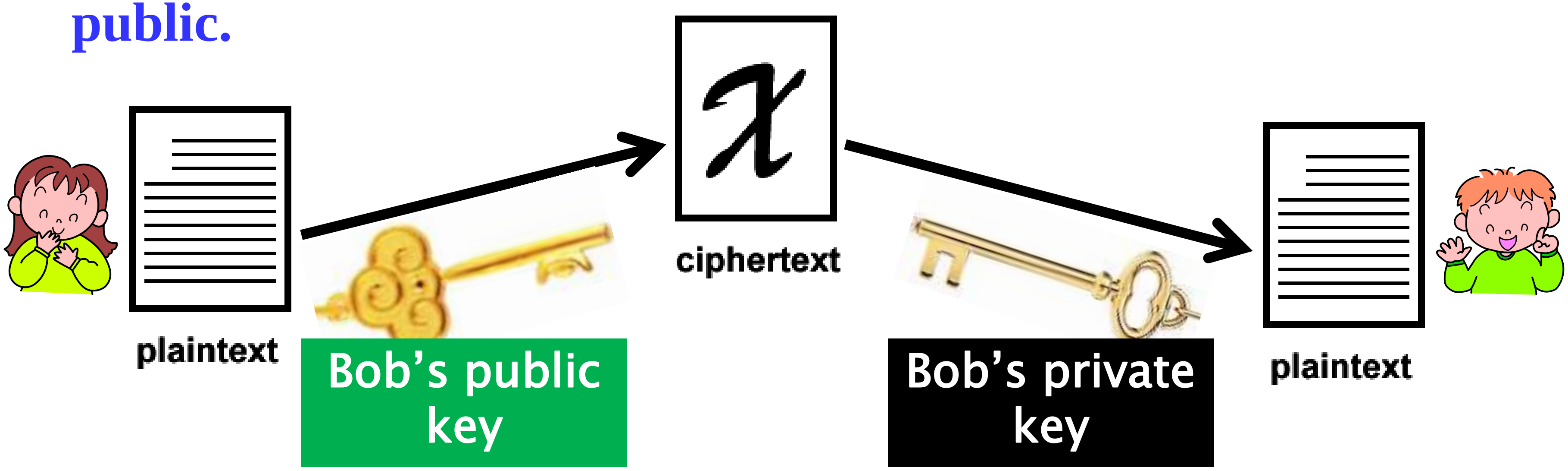
released in 1997

- **1976, Diffie and Hellman, the idea of PKC**
- **1977, Rivest, Shamir and Adleman, RSA**
- **1985, ElGamal, the ElGamal Cryptosystem**
- **Lattice-based Cryptography**
- **Coded-based Cryptography**
- **Identity-based Cryptography**
- **……**

Ch 9 & 13

Basics of PKC?

- Encryption is easy to compute; decryption is unknown to anyone other than Bob.
  - A trapdoor one-way function $y = f_K(x)$: one-way, trapdoor (as K)
- The Computational Security of PKC is studied, since PKC can never provide unconditional security as the encryption rule is public.



Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
  - **Euclidean Algorithms**
  - **The Chinese Remainder Theorem**
  - **Group Theory II**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Attacks on RSA**
- **6. The Rabin Cryptosystem**
- **7. Semantic Security of RSA**

2.1 The Euclidean Algorithm (欧几里得算法)

- **The set of residues modulo *n*:** $\mathbb{Z}_n$

- **The set of residues modulo *n* that are relatively prime to *n*:** $\mathbb{Z}_n^*$

$$|\mathbb{Z}_n^*| = \phi(n)$$

- **To find the <u>multiplicative inverse</u> of** $b \in \mathbb{Z}_n^*$
  - **The Extended Euclidean Algorithm**

1) The basic Euclidean Algorithm

- **To compute the GCD *r* of *a* and *b***

Algorithm 6.1: EUCLIDEAN ALGORITHM(a, b)

$r_0 \leftarrow a$
$r_1 \leftarrow b$
$m \leftarrow 1$
while $r_m \neq 0$
do $\begin{cases} q_m \leftarrow \left\lfloor \frac{r_{m-1}}{r_m} \right\rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m + 1 \end{cases}$
$m \leftarrow m - 1$
return $(q_1, \ldots, q_m; r_m)$
comment: $\boxed{r_m = \gcd(a, b)}$

Long division method

$$\begin{array}{rcll} a \rightarrow r_0 & = & q_1 r_1 + r_2, & 0 < r_2 < r_1 \\ b \rightarrow r_1 & = & q_2 r_2 + r_3, & 0 < r_3 < r_2 \\ \vdots & \vdots & \vdots & \vdots \\ r_{m-2} & = & q_{m-1} r_{m-1} + r_m, & 0 < r_m < r_{m-1} \\ r_{m-1} & = & q_m r_m. & \end{array}$$

$$r = \gcd(a, b) = \gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m$$

2) The Extended Euclidean Algorithm

From Algorithm 6.1:

$$\begin{array}{rcl} a \rightarrow r_0 & = & q_1 r_1 + r_2, \\ b \rightarrow r_1 & = & q_2 r_2 + r_3, \\ \vdots & \vdots & \vdots \\ r_{m-2} & = & q_{m-1} r_{m-1} + r_m \\ r_{m-1} & = & q_m r_m. \end{array}$$

$$\begin{aligned} r_2 &= r_0 - q_1 r_1 \\ r_3 &= r_1 - q_2 r_2 \\ &= r_1 - q_2 (r_0 - q_1 r_1) \\ &= -q_2 r_0 + (1 + q_2 q_1) r_1 \\ &\ldots\ldots \end{aligned}$$

THEOREM 6.1 *For $0 \le j \le m$, we have that $\boxed{r_j = s_j r_0 + t_j r_1}$, where the r_j's are defined as in Algorithm 6.1, and the s_j's and t_j's are defined in the above recurrence.*

$$s_j = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{if } j = 1 \\ s_{j-2} - q_{j-1} s_{j-1} & \text{if } j \ge 2 \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = 0 \\ 1 & \text{if } j = 1 \\ t_{j-2} - q_{j-1} t_{j-1} & \text{if } j \ge 2 \end{cases}$$

$$\boxed{r_m = s_m r_0 + t_m r_1 \leftrightarrow r = sa + tb}$$

2) The Extended Euclidean Algorithm

- **To obtain the GCD *r* of *a* and *b* with *r* = *sa* + *tb***

Algorithm 6.2: EXTENDED EUCLIDEAN ALGORITHM(a, b)

Initialization:

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
$r \leftarrow a_0 - qb_0$

while $r > 0$

do $\begin{cases} temp \leftarrow t_0 - qt \\ t_0 \leftarrow t \\ t \leftarrow temp \\ temp \leftarrow s_0 - qs \\ s_0 \leftarrow s \\ s \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$

$r \leftarrow b_0$
return $\boxed{(r, s, t)}$
comment: $\boxed{r = \gcd(a, b) \text{ and } sa + tb = r}$

If *r* = 1, then b^{-1} mod *a* = *t*.

3) The Multiplicative Inverse Algorithm

- **To compute the multiplicative inverse of *b***

Algorithm 6.3: MULTIPLICATIVE INVERSE(a, b)

Initialization

$$\begin{cases} a_0 \leftarrow a \\ b_0 \leftarrow b \\ t_0 \leftarrow 0 \\ t \leftarrow 1 \\ q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor \\ r \leftarrow a_0 - q b_0 \end{cases}$$

while $r > 0$

$$\textbf{do} \begin{cases} temp \leftarrow (t_0 - qt) \bmod a \\ t_0 \leftarrow t \\ t \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor \\ r \leftarrow a_0 - q b_0 \end{cases}$$

if $b_0 \neq 1$

  then b has no inverse modulo a

  else return (t)

If $b_0 = 1$, then b^{-1} mod $a = t$.

Col: $b \in \mathbb{Z}_n^*$ **, i.e., gcd(n, b) = 1,**

then Multiplicative Inverse(n, b) = t = b^{-1} mod n.

2.2 The Chinese Remainder Theorem (中国剩余定理)

在《孙子算经》中有这样一个问题：

“今有物不知其数，
三三数之剩二，
五五数之剩三，
七七数之剩二，
问物几何？”

这个问题称为“孙子问题”

$$x = 3q_1 + 2$$
$$x = 5q_2 + 3$$
$$x = 7q_3 + 2$$

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

$$x = 23,\ 128,\ 233, \cdots$$

2.2 The Chinese Remainder Theorem

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x \equiv a_r \pmod{m_r}$$

gcd(m_i, m_j) = 1 if $i \neq j$

Define a function:

$$\chi : \mathbb{Z}_M \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_r}$$

$$\chi(x) = (x \bmod m_1, \ldots, x \bmod m_r)$$

$$M = m_1 m_2 \ldots m_r$$

Example 6.2 Suppose $r = 2$, $m_1 = 5$ and $m_2 = 3$, so $M = 15$. Then the function χ has the following values:

$$\begin{array}{rclrclrcl}
\chi(0) & = & (0,0) & \chi(1) & = & (1,1) & \chi(2) & = & (2,2) \\
\chi(3) & = & (3,0) & \chi(4) & = & (4,1) & \chi(5) & = & (0,2) \\
\chi(6) & = & (1,0) & \chi(7) & = & (2,1) & \chi(8) & = & (3,2) \\
\chi(9) & = & (4,0) & \chi(10) & = & (0,1) & \chi(11) & = & (1,2) \\
\chi(12) & = & (2,0) & \chi(13) & = & (3,1) & \chi(14) & = & (4,2).
\end{array}$$

The function χ is a bijection

2.2 The Chinese Remainder Theorem

gcd(m_i, m_j) = 1 if $i \neq j$

THEOREM 6.3 (Chinese remainder theorem) *Suppose $m_1, \ldots, m_r$ are pairwise relatively prime positive integers, and suppose $a_1, \ldots, a_r$ are integers. Then the system of r congruences $x \equiv a_i \pmod{m_i}$ $(1 \leq i \leq r)$ has a unique solution modulo $M = m_1 \times \cdots \times m_r$, which is given by*

$$\boxed{x = \sum_{i=1}^{r} a_i M_i y_i \bmod M,}$$

where $M_i = M/m_i$ and $y_i = M_i^{-1} \bmod m_i$, for $1 \leq i \leq r$.

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x \equiv a_r \pmod{m_r}$$

Define functions:

$$\chi : \mathbb{Z}_M \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_r}$$

$$\chi(x) = (x \bmod m_1, \ldots, x \bmod m_r)$$

***M*=*m*$_1$*m*$_2$…*m*$_r$**

$$\rho : \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_r} \to \mathbb{Z}_M$$

$$\rho(a_1, \ldots, a_r) = \sum_{i=1}^{r} a_i M_i y_i \bmod M$$

$$\boxed{\rho = \chi^{-1}}$$

2.2 The Chinese Remainder Theorem

Example 6.3 Suppose $r = 3$, $m_1 = 7$, $m_2 = 11$, and $m_3 = 13$. Then $M = 1001$. We compute $M_1 = 143$, $M_2 = 91$, and $M_3 = 77$, and then $y_1 = 5$, $y_2 = 4$, and $y_3 = 12$. Then the function $\chi^{-1} : \mathbb{Z}_7 \times \mathbb{Z}_{11} \times \mathbb{Z}_{13} \to \mathbb{Z}_{1001}$ is the following:

$$\chi^{-1}(a_1, a_2, a_3) = (715a_1 + 364a_2 + 924a_3) \bmod 1001.$$

For example, if $x \equiv 5 \pmod 7$, $x \equiv 3 \pmod{11}$ and $x \equiv 10 \pmod{13}$, then this formula tells us that

$$\begin{aligned} x &= (715 \times 5 + 364 \times 3 + 924 \times 10) \bmod 1001 \\ &= 13907 \bmod 1001 \\ &= 894. \end{aligned}$$

This can be verified by reducing 894 modulo 7, 11 and 13.

2.3 Group Theory II

- **Let G be a finite multiplicative group (乘法群)**
- **Order of the group G, denoted by $|G|$: the number n of elements in G**
- **Order of an element g in G: the smallest positive integer m such that $g^m = 1$**

Examples: $Z_4^*=\{1,3\}$, ord(1)=1, ord(3)=2; (Z_4 is not a multiplicative group)

$Z_5^*=\{1,2,3,4\}$, ord(1)=1, ord(2)=4, ord(3)=4, ord(4)=2.

THEOREM 6.4 (Lagrange) *Suppose G is a multiplicative group of order n, and $g \in G$. Then the order of g divides n.* $\Rightarrow$ $\boxed{m|n}$

COROLLARY 6.5 *If $b \in \mathbb{Z}_n{}^*$, then $b^{\phi(n)} \equiv 1 \pmod{n}$.* $\Leftarrow$ $\boxed{|\mathbb{Z}_n^*| = \phi(n)}$

COROLLARY 6.6 (Fermat) *Suppose p is prime and $b \in \mathbb{Z}_p$. Then $b^p \equiv b \pmod{p}$.*

2.3 Group Theory II

- **Cyclic Group *G*: if there exists an element *g* in *G* such that the order of *g* is $|G|$. Such *g* is called a primitive element, and G can be represented as**

$$G = \{g^1, g^2, \ldots, g^{|G|-1}, g^{|G|}=1\} = \langle g \rangle$$

THEOREM 6.7 *If p is prime, then $\mathbb{Z}_p{}^*$ is a cyclic group.*

Examples: $Z_5{}^*=\{1,2,3,4\}$, ord(1)=1, ord(2)=4, ord(3)=4, ord(4)=2;

$Z_5{}^* = \langle 2 \rangle = \langle 3 \rangle$: $3^1 \pmod 5 = 3$, $3^2 \pmod 5 = 4$, $3^3 \pmod 5 = 2$, $3^4 \pmod 5 = 1$

A quick method to verify whether an element is a primitive element:

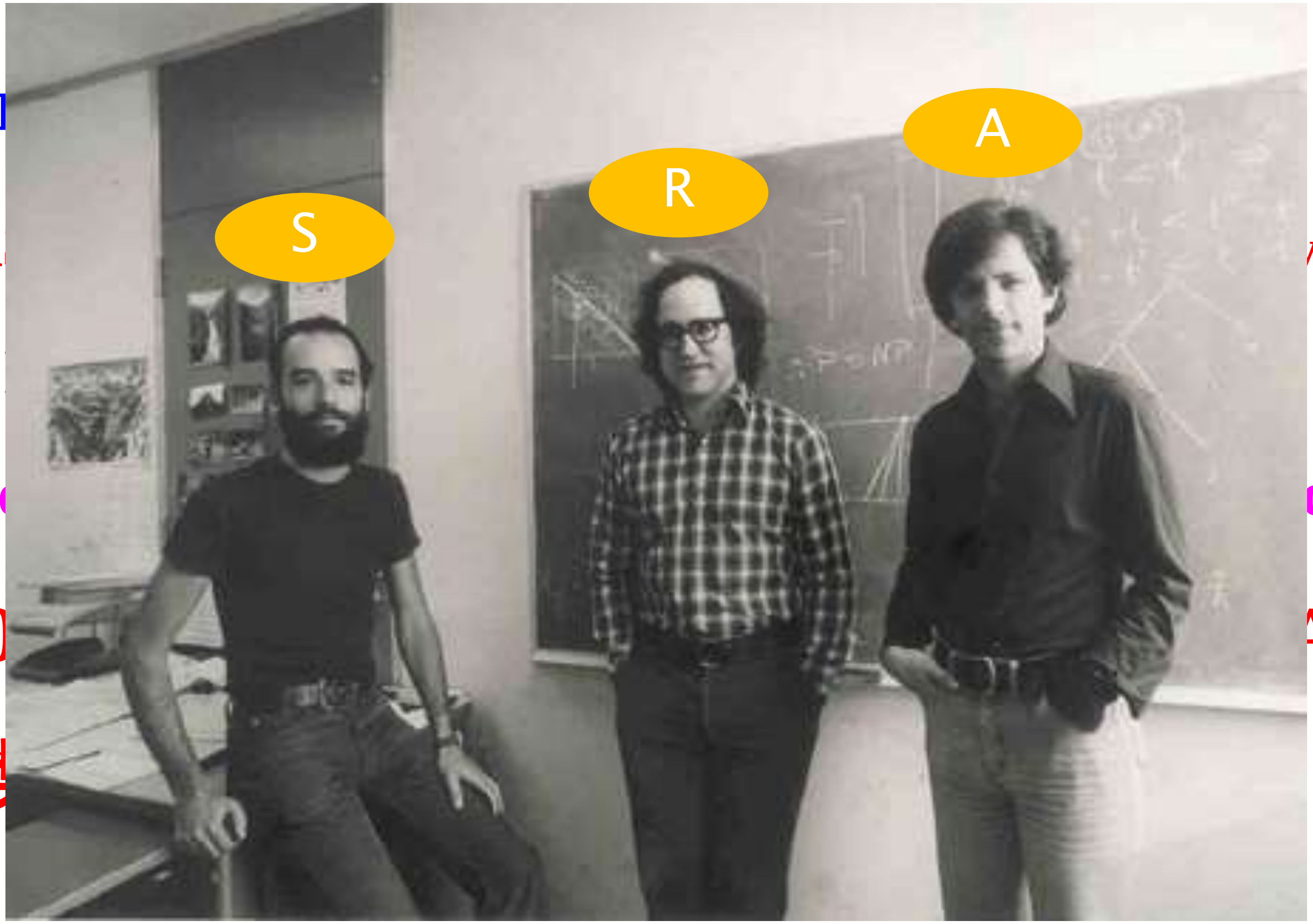
THEOREM 6.8 *Suppose that $p > 2$ is prime and $\alpha \in \mathbb{Z}_p{}^*$. Then α is a primitive element modulo p if and only if $\alpha^{(p-1)/q} \not\equiv 1 \pmod p$ for all primes q such that $q \mid (p-1)$.*

Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
  - **Euclidean Algorithms**
  - **The Chinese Remainder Theorem**
  - **Group Theory II**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Attacks on RSA**
- **6. The Rabin Cryptosystem**
- **7. Semantic Security of RSA**

Introduction to RSA

- **Background of RSA cryptosystem**
  - **1977, Rivest, Shamir and Adleman, MIT**



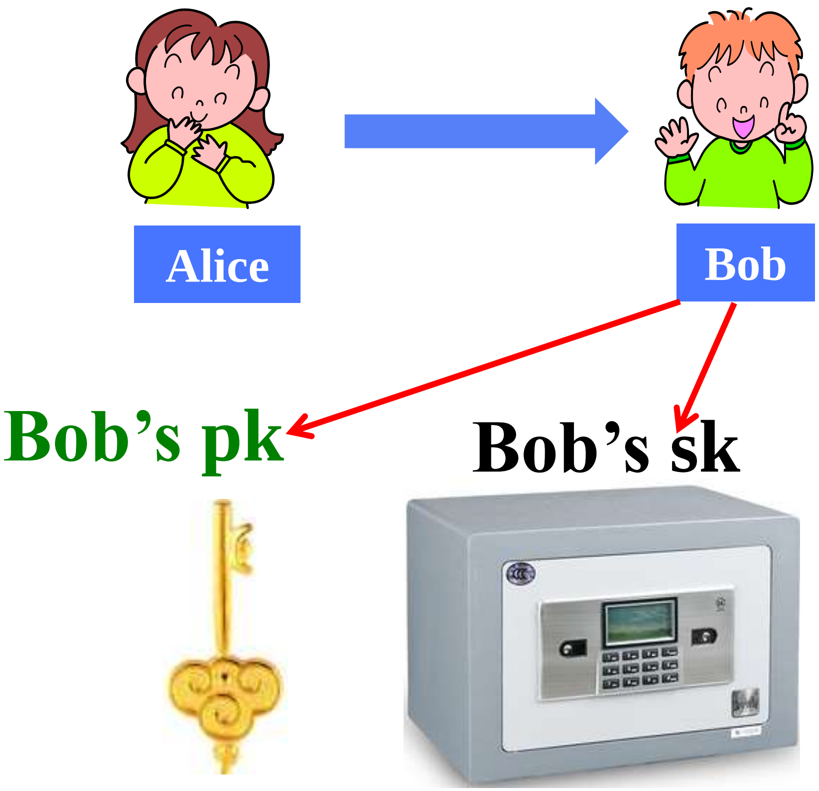
- **Co**
  - **R** ... **ystem)**
    - ➢
  - **C** ... **overnment**
- **200** ... **Award (图**

灵

Description of PKC Cryptosystem

◆ **Three algorithms of the PKC cryptosystem:**

1) **Parameter and key generation Alg.:**
   - **KeyGen(θ)= (pk, sk)**
2) **Encryption Alg.:**
   - **Enc(m, pk) = c**
3) **Decryption Alg.:**
   - **Dec(c, sk) = m'**



RSA-Parameter Generation Alg

Alg. 1: KeyGen(θ)= (pk, sk) (Algorithm 6.4)

① **Generate two large primes p and q, such that $p \neq q$**

② **Compute $n = pq$, and $\phi(n) = (p-1)(q-1)$**

③ **Choose a number b ($1 < b < \phi(n)$) such that $\gcd(b, \phi(n))=1$**

④ **Compute $a \equiv b^{-1} \pmod{\phi(n)}$**

Euclidean algorithm

⑤ **Output:**

pk = (n, b) is the public key

***n* should be large**

sk = (p, q, a) is the secret key

RSA – Encryption & Decryption Algs

Alg. 2: Enc(m, pk) = c (Cryptosystem 6.1)

- **Compute:** Enc(m, pk) = $m^b \bmod n = c$

pk=(n, b) is the public key

Correctness: $m = m'$?

Alg. 3: Dec(c, sk) = m'

- **Compute:** Dec(c, sk) = $c^a \bmod n = m'$

sk=(p, q, a) is the secret key

RSA – Correctness

Correctness: $m = m'$.

- **Sketch of the proof:**
  - $m' = \mathrm{Dec}(c, sk) = c^a \mod n$

$$\equiv [(m^b)^a] \pmod n \quad (\because c \equiv m^b \pmod n)$$

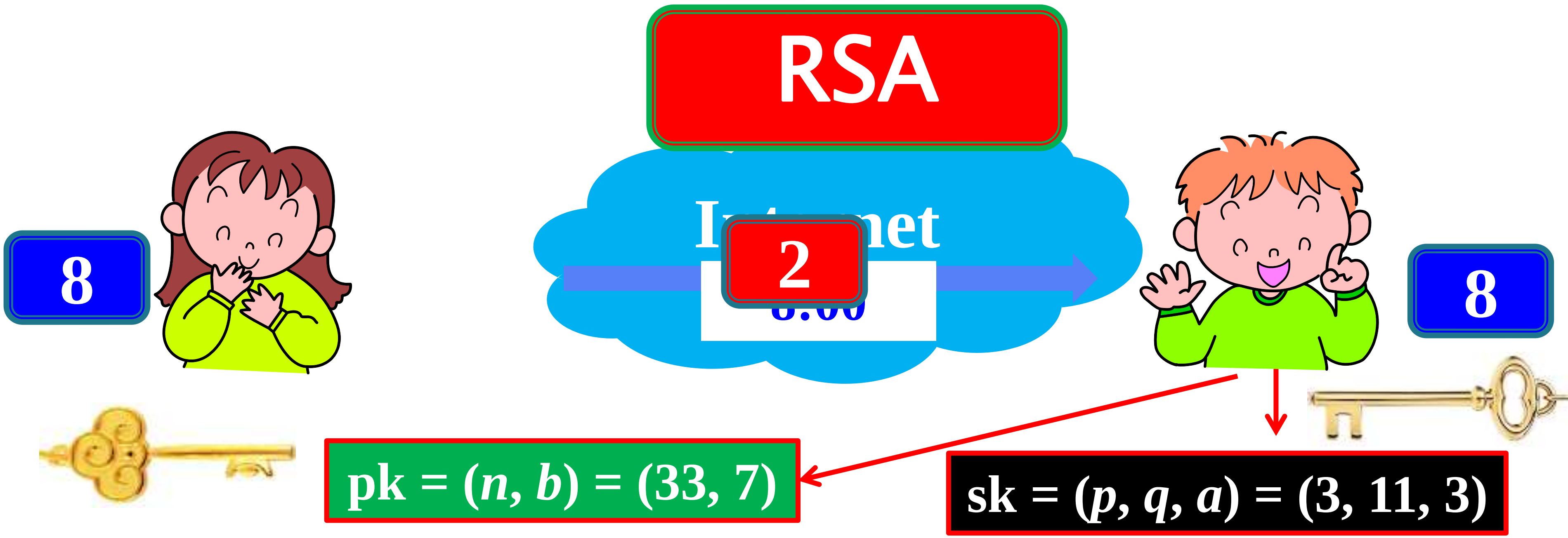
$$\equiv (m^{ab}) \pmod n$$

$\because a \equiv b^{-1} \pmod{\phi(n)} \Leftrightarrow \phi(n) \mid (ab - 1)$

$\therefore$ By Euler-Fermat Theorems, we have (see P247 and exercise 6.11)

$$(m^{ab}) \pmod n \equiv m.$$

RSA-An Example



1) **Encryption:** $\mathrm{Enc}(8, pk) = 8^b \bmod n = 8^7 \bmod 33 = 2$

2) **Decryption:** $\mathrm{Dec}(2, sk) = 2^a \bmod n = 2^3 \bmod 33 = 8$

Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Attacks on RSA**
- **6. The Rabin Cryptosystem**
- **7. Semantic Security of RSA**

Implementing RSA

RSA Set-up of Key Generation:

1) **Generate** two large odd primes p and q such that $p \neq q$
2) **Compute** $n = pq$ and $\phi(n) = (p-1)(q-1)$
3) **Choose** a random number b $(1 < b < \phi(n))$ such that $\gcd(b, \phi(n))=1$
4) **Compute** $a \equiv b^{-1} \pmod{\phi(n)}$
5) **Output:** $pk = (n, b)$, $sk = (p, q, a)$

Euclidean Algorithms

RSA Encryption and Decryption:

1) **Encryption rule:** $e_K(m) = m^b \bmod n$
2) **Decryption rule:** $d_K(c) = c^a \bmod n$

Square and Multiply Algorithm

The Square and Multiply Algorithm

- **To compute $z = x^c \bmod n$:**

$$c = \sum_{i=0}^{\ell-1} c_i 2^i, \quad \text{where } c_i = 0 \text{ or } 1,\ 0 < i \leq \ell - 1.$$

Algorithm 6.5: SQUARE-AND-MULTIPLY(x, c, n)

$z \leftarrow 1$
for $i \leftarrow \ell - 1$ **downto** 0
do $\begin{cases} z \leftarrow z^2 \bmod n \\ \textbf{if } c_i = 1 \\ \quad \textbf{then } z \leftarrow (z \times x) \bmod n \end{cases}$
return (z)

O $((\log(n))^3)$

$z = x^c = x^{\sum_{i=0}^{\ell-1} c_i 2^i}$

$9726^{3533} \bmod 11413$

3533 is 110111001101

$c \leftrightarrow b_{11} b_{10} \ldots b_1 b_0$

i	b_i	z
11	1	$1^2 \times 9726 = 9726$
10	1	$9726^2 \times 9726 = 2659$
9	0	$2659^2 = 5634$
8	1	$5634^2 \times 9726 = 9167$
7	1	$9167^2 \times 9726 = 4958$
6	1	$4958^2 \times 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \times 9726 = 10185$
2	1	$10185^2 \times 9726 = 105$
1	0	$105^2 = 11025$
0	1	$11025^2 \times 9726 = 5761$

Computational Complexity

Now we turn to modular arithmetic, i.e., operations in $\mathbb{Z}_n$. Suppose that n is a k-bit integer, and $0 \leq m_1, m_2 \leq n-1$. Also, let c be a positive integer. We have the following:

- Computing $(m_1 + m_2) \bmod n$ can be done in time $O(k)$.
- Computing $(m_1 - m_2) \bmod n$ can be done in time $O(k)$.
- Computing $(m_1 m_2) \bmod n$ can be done in time $O(k^2)$.
- Computing $(m_1)^{-1} \bmod n$ can be done in time $O(k^3)$ (provided that this inverse exists). **→O $((\log(n))^2)$**
- Computing $(m_1)^c \bmod n$ can be done in time $O((\log c) \times k^2)$.

Algorithm 6.3 Multiplicative Inverse (n, m_1)

Algorithm 6.5 Square-and-Multiply (m_1, c, n)

See 《Introduction to Algorithms》 or go to the link: https://richardyan.site/Number-Theory-and-Group-based-Cryptography-01-Time-Complexity-of-Arithmetic/

Implementing RSA

RSA Set-up of Key Generation:

Miller-Rabin Algorithm

1) Generate two large odd primes p and q such that $p \neq q$
2) Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$ $O((\log(n))^2)$
3) Choose a random number b $(1 < b < \phi(n))$ such that $\gcd(b, \phi(n))=1$
4) Compute $a \equiv b^{-1} \pmod{\phi(n)}$ $O((\log(n))^2)$ **Euclidean Algorithms**
5) Output: pk = (n, b), sk = (p, q, a)

RSA Encryption and Decryption:

$O((\log(n))^3)$

1) Encryption rule: $e_K(m) = m^b \bmod n$
2) Decryption rule: $d_K(c) = c^a \bmod n$

Square and Multiply Algorithm

Primality Testing

- **Generate large random primes**
  - **Generate large random number + Primality Testing**

- **Primality Testing:**
  - **polynomial-time deterministic algorithm** (2002, Agrawal, Kayal and Saxena, theoretic proof) **: a breakthrough**
  - **Randomized polynomial-time Monte-Carlo algorithms** (in practice)**:**
    - ✓ Solovay-Strassen Algorithm **$O((\log(n))^3)$**
    - ✓ **Miller-Rabin Algorithm** **$O((\log(n))^3)$**

- **Testing Size:** **$\pi(N)$ = no. of primes in [1, N]**
  - **Prime number theorem, $\pi(N) \approx N/\ln N$; prob.(p is prime) $\approx 1/\ln N$.**

Primality Testing – Monte Carlo algorithms

- **Decision Problem:** **a question is to be answered "Yes" or "No"**
- **A (randomized) Monte Carlo algorithm:**
  - **always gives an answer, but the answer may be incorrect**
  - **different from the Las Vegas algorithm**
  - **yes-biased MC-Alg**
  - **no-biased MC-Alg**

Definition 6.1: A *yes-biased Monte Carlo algorithm* is a randomized algorithm for a decision problem in which a "yes" answer is (always) correct, but a "no" answer may be incorrect. A *no-biased Monte Carlo algorithm* is defined in the obvious way. We say that a yes-biased Monte Carlo algorithm has *error probability* equal to ϵ if, for any instance in which the answer is "yes," the algorithm will give the (incorrect) answer "no" with probability at most ϵ. (This probability is computed over all possible random choices made by the algorithm when it is run with a given input.)

Primality Testing – Composites Probl.

- **Decision Problem:** **a question is to be answered "Yes" or "No"**
- **A decision problem of Composites:**

Problem 6.1: Composites

Instance: A positive integer $n \geq 2$.
Question: Is n composite?

A yes-biased Monte Carlo Alg. with error prob. (at most) ¼.

Algorithm 6.6: SOLOVAY-STRASSEN(n)

choose a random integer a such that $1 \leq a \leq n-1$
$x \leftarrow \left(\frac{a}{n}\right)$ **Legendre symbol in Definition 6.3**
if $x = 0$
  then return ("n is composite")
$y \leftarrow a^{(n-1)/2} \pmod{n}$
if $x \equiv y \pmod{n}$
  then return ("n is prime")
  else return ("n is composite")

Algorithm 6.7: MILLER-RABIN(n)

write $n - 1 = 2^k m$, where m is odd
choose a random integer a, $1 \leq a \leq n-1$
$b \leftarrow a^m \bmod n$
if $b \equiv 1 \pmod{n}$
  then return ("n is prime") **O ((log(n)) ³)**
for $i \leftarrow 0$ **to** $k - 1$
  do $\left\{\begin{array}{l}\textbf{if } b \equiv -1 \pmod{n} \\ \quad \textbf{then return } (\text{“}n\text{ is prime”}) \\ \quad \textbf{else } b \leftarrow b^2 \bmod n\end{array}\right.$
return ("n is composite")

Complexity of Implementing RSA

Computational Complexity: <u>Polynomial time</u>

RSA Set-up of Key Generation:

Algorithm 6.7 Miller-Rabin (p)/ (q)

1) **Generate** two large odd primes p and q such that $p \neq q$ $O((\log n)^3)$
2) **Compute** $n = pq$ and $\phi(n) = (p-1)(q-1)$ $O((\log n)^2)$

Computational complexity of RSA is <u>higher</u> than DES and AES

RSA Encryption and Decryption:

1) **Encryption rule:** $e_K(m) = m^b \bmod n$ $O((\log n)^3)$
2) **Decryption rule:** $d_K(c) = c^a \bmod n$ $O((\log n)^3)$

Algorithm 6.5 Square-and-Multiply (m,b,n) /(c,a,n)

Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Security Discussion and Attacks on RSA (total break)**
  - **Fatoring *n* and related attack**
- **6. The Rabin Cryptosystem**
- **7. Semantic Security of RSA**

Security Discussions

- **Belief:**
  - **Encryption $c = m^b \bmod n$ is a <u>one-way function</u>**
  - **The trapdoor is the knowledge of the factorization $n=pq$.**

pk = (n, b) is the <u>public</u> key

***n* should be large**

sk = (p, q, a) is the <u>secret</u> key

Security of RSA is based on the difficulty of factorizing the large integer $n=pq$

- **the difficulty of break RSA $\leq$ the difficulty of factoring n.**

Algorithms of Factoring n (Skipped)

- **Factoring n: $n=pq$**
  - **Algorithms in Practice**
    - **The Quadratic Sieve Alg.**
    - **The Elliptic Curve Factoring Alg.**
    - **The Number Field Sieve**
  - **Precursor Algorithms**
    - **The Pollard Rho Alg.**
    - **The Pollard p-1 Alg.**
    - **Dixon's Random Squares Alg.**

Factoring n (1 / 3)

- **exhaustive search attack**: **by one computer with 1million executions per second**

Length of Key n (bits)	Time of success attack (years)
116	400
129	5000
512	30000
768	200 000 000
1024	300 000 000 000
2048	300 000 000 000 000 000 000

Factoring n (2/3)

- **MPQS** (Multiple polynomial quadratic sieve): by a desktop computer (Processor: Intel Dual-Core i7-4500U 1.80GHz)

Length of Key n (bits)	Time of Success Attack
128	Less than 2 seconds
192	16 seconds
256	35 minutes
260	1 hour

- **YAFU:**
- ……

Factoring *n* (3/3)

- **Insecure *n*:**
  - **1999, RSA-155(155 digits, 512bits), about 7 months;**
  - **2009, RSA-155(155 digits, 512bits), 73days in one desktop;**
  - **2009~2010, RSA-232(232 digits, 768 bits), distributed system of hundreds of computers, 2 years**
  - **2018, August, RSA-230(230 digits),**
  - **2019, December, RSA-240(240 digits, 795bits)**

RSA Challenge:

1. http://www.rsasecurity.com/rsalabs/challenges/
2. https://link.springer.com/referenceworkentry/10.1007%2F0-387-23483-7_362
3. http://unsolvedproblems.org/index_files/RSA.htm

Secure *n*

- **<u>Currently Secure *n*</u>:**

  1024-bit, 2048-bit, 3072-bit, 4096-bit

- **<u>Challenges:</u> A Quantum Computer would be able to factor large *n* in polynomial time and then break RSA**

Computing $\phi(n)$

- **Computing the Euler Function $\phi(n)$ to attack RSA**
  - We first observe that computing $\phi(n)$ is no easier than factoring n.

$$n = pq$$
$$\phi(n) = (p-1)(q-1)$$

$$\xrightarrow{q = n/p}$$

$$p^2 - (n - \phi(n) + 1)p + n = 0$$

The two roots will be *p* and *q*, where *n*=*pq*.

pk = (*n*, *b*) is the public key

sk = (*p*, *q*, *a*) is the secret key

Computing a

- **Computing the Decryption Exponent a to attack RSA:**
  - **Computing a is no easier tha**
    - **If the decryption exponent a is k** **<u>polynomial time</u> by means of a r**
      - **Algorithm 6.10 (P226)**
  - **Once a is revealed, a new mo**
  - **Wiener's Low Decryption Ex** **cases** $3a < n^{1/4}$ and $q < p < 2q$
    - **Algorithm 6.11**

sk = (p,

Algorithm 6.10: RSA-FACTOR(n, a, b)

comment: we are assuming that $ab \equiv 1 \pmod{\phi(n)}$
write $ab - 1 = 2^s r$, r odd
choose w at random such that $1 \leq w \leq n - 1$
$x \leftarrow \gcd(w, n)$
if $1 < x < n$
  then return (x)
comment: x is a factor of n
$v \leftarrow w^r \bmod n$
if $v \equiv 1 \pmod{n}$
  then return ("failure")
while $v \not\equiv 1 \pmod{n}$
  $- v$
  $v^2 \bmod n$
if $v_0 \equiv -1 \pmod{n}$
  then return ("failure")
  else $\begin{cases} x \leftarrow \gcd(v_0 + 1, n) \\ \textbf{return } (x) \end{cases}$
comment: x is a factor of n

Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Security Discussions and Attacks on RSA**
- **6. The Rabin Cryptosystem**
  - **Turing Reduction**
- **7. Semantic Security of RSA**

The Rabin Cryptosystem

Cryptosystem 6.2: *Rabin Cryptosystem*

Let $n = pq$, where p and q are primes and $p, q \equiv 3 \pmod 4$. Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n{}^*$, and define

$$\mathcal{K} = \{(n, p, q)\}.$$

For $K = (n, p, q)$, define

$$e_K(x) = x^2 \bmod n$$

and

$$d_K(y) = \sqrt{y} \bmod n.$$

The value n is the public key, while p and q are the private key.

A drawback in decryption

- **It is a provably secure cryptosystem:**
  - **computationally secure against a chosen-plaintext attack**
  - **If the problem of factoring is computationally infeasible, then the Rabin Cryptosystem is secure.**

Turing Reduction

- **Turing Reduction from G to H:** $\mathbf{G} \propto_T \mathbf{H}$

Definition 6.5: Suppose that **G** and **H** are problems. A *Turing reduction* from **G** to **H** is an algorithm SOLVEG with the following properties:

1. SOLVEG assumes the existence of an arbitrary algorithm SOLVEH that solves the problem **H**.
2. SOLVEG can call the algorithm SOLVEH and make use of any values it outputs, but SOLVEG cannot make any assumption about the actual computations performed by SOLVEH (in other words, SOLVEH is an oracle that is treated as a "black box").
3. SOLVEG is a polynomial-time algorithm, when each call to the oracle is regarded as taking $O(1)$ time. (Note that the complexity of SOLVEG takes into account all the computations that are done "outside" the oracle.)
4. SOLVEG correctly solves the problem **G**.

If there is a Turing reduction from **G** to **H**, we denote this by writing $\mathbf{G} \propto_T \mathbf{H}$.

If there exists a polynomial-time algorithm to solve **H**, then there exists a polynomial-time algorithm to solve **G**.

Security of the Rabin Cryptosystem

- **Turing Reduction**

Factoring $\propto_T$ Rabin decryption.

(1/2, 1)-algorithm---

Algorithm 6.12: RABIN ORACLE FACTORING(n)

external RABIN DECRYPT
choose a random integer $r \in \mathbb{Z}_n^*$
$y \leftarrow r^2 \bmod n$
$x \leftarrow$ RABIN DECRYPT(y)
if $x \equiv \pm r \pmod n$
  then return ("failure")
  else $\begin{cases} p \leftarrow \gcd(x+r, n) \\ q \leftarrow n/p \\ \textbf{return } (\text{"}n = p \times q\text{"}) \end{cases}$

By Theorem 6.13
$x \equiv \pm r \pmod n$
or $x \equiv \pm \omega r \pmod n$

The Rabin Cryptosystem is provably secure against a chosen-plaintext attack.

The Rabin Cryptosystem is <u>completely insecure</u> against a <u>chosen-ciphertext</u> attack.

chosen plaintext attack
The opponent has obtained temporary access to the encryption machinery. Hence he can choose a plaintext string, **x**, and construct the corresponding ciphertext string, **y**.

chosen ciphertext attack
The opponent has obtained temporary access to the decryption machinery. Hence he can choose a ciphertext string, **y**, and construct the corresponding plaintext string, **x**.

Outline

- **1. Introduction to Public-Key Cryptography**
  - **SKC v.s. PKC**
- **2. Mathematical Backgrounds III**
- **3. The RSA Cryptosystem**
- **4. Implementing RSA and Complexity**
- **5. Attacks on RSA**
- **6. The Rabin Cryptosystem**
  - **Turing Reduction**
- **7. Semantic Security**

Different Attack Goals

- **Total Break:** **to know the private key or the secret key**
- **Partial Break:** **be able to decrypt a previously unseen ciphertext without the key, or to determine some specific information about the plaintext given the ciphertext, with non-negligible probability**
- **Distinguishability of Ciphertexts:** **be able to distinguish between encryptions of two given plaintexts, or between an encryption of a given plaintext and a random string, with probability exceeding 1/2**

Partial break of RSA

- **Given *y*, where $y = x^b \bmod n$ and that $\gcd(b, \phi(n))=1$, it must be the case that *b* is odd.**

given $y = e_K(x)$, compute $parity(y)$, where $parity(y)$ denotes the low-order bit of x (i.e., $parity(y) = 0$ if x is even and $parity(y) = 1$ if x is odd).

given $y = e_K(x)$, compute $half(y)$, where $half(y) = 0$ if $0 \leq x < n/2$ and $half(y) = 1$ if $n/2 < x \leq n-1$.

- **RSA does not leak these types of information provided that RSA encryption is secure.**
  - **Turing reduction from RSA decryption to *half*(*y*)**

Semantic Security

- **A cryptosystem is said to achieve *semantic security*** **if the adversary cannot (in polynomial time) distinguish ciphertexts.**

Problem 6.3: Ciphertext Distinguishability

Instance: An encryption function $f : X \to X$; two plaintexts $x_1, x_2 \in X$; and a ciphertext $y = f(x_i)$, where $i \in \{1, 2\}$.
Question: Is $i = 1$?

Semantic Security of PKC (skipped)

Cryptosystem 6.3: *Semantically Secure Public-key Cryptosystem*

Let m, k be positive integers; let $\mathcal{F}$ be a family of trapdoor one-way permutations such that $f : \{0,1\}^k \rightarrow \{0,1\}^k$ for all $f \in \mathcal{F}$; and let $G : \{0,1\}^k \rightarrow \{0,1\}^m$ be a random oracle. Let $\mathcal{P} = \{0,1\}^m$ and $\mathcal{C} = \{0,1\}^k \times \{0,1\}^m$, and define

$$\mathcal{K} = \{(f, f^{-1}, G) : f \in \mathcal{F}\}.$$

For $K = (f, f^{-1}, G)$, let $r \in \{0,1\}^k$ be chosen randomly, and define

$$e_K(x) = (y_1, y_2) = (f(r), G(r) \oplus x),$$

where $y_1 \in \{0,1\}^k$, $x, y_2 \in \{0,1\}^m$. Further, define

$$d_K(y_1, y_2) = G(f^{-1}(y_1)) \oplus y_2$$

($y_1 \in \{0,1\}^k$, $y_2 \in \{0,1\}^m$). The functions f and G are the public key; the function f^{-1} is the private key.

- **Cryptosystem 6.3 is semantically secure in the random oracle model.**
- **RSA encryption is secure if the length of *n* is at least 1024 bits.**

Summary

- 1. Introduction to Public-Key Cryptography
  - SKC v.s. PKC
- 2. Mathematical Backgrounds III
  - Euclidean Algorithms
  - The Chinese Remainder Theorem
  - Group Theory II
- 3. The RSA Cryptosystem
- 4. Implementing RSA and Complexity
- 5. Security Discussions and Attacks on RSA
- 6. The Rabin Cryptosystem
  - Turing Reduction
- 7. Semantic Security of RSA

Homework

Problem Set 5: Exercises 6.3, 6.4, 6.5, 6.7, 6.11, 6.13(optional), 6.15, 6.16.

Thank you!

Questions?