



# CS215 DISCRETE MATH

Dr. QI WANG

Department of Computer Science and Engineering

Office: Room413, CoE South Tower

Email: [wangqi@sustech.edu.cn](mailto:wangqi@sustech.edu.cn)

# The Chinese Remainder Theorem

- About 1500 years ago, the Chinese mathematician Sun-Tsu asked:  
“There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; when divided by 7, the remainder is 2. What will be the number of things?”

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

两种方法 ↴  
| CRT  
| back substitution

# The Chinese Remainder Theorem

- About 1500 years ago, the Chinese mathematician Sun-Tsu asked:  
“There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; when divided by 7, the remainder is 2. What will be the number of things?”

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

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

# The Chinese Remainder Theorem

- **Theorem** (*The Chinese Remainder Theorem*) Let  $m_1, m_2, \dots, m_n$  be pairwise relatively prime positive integers greater than 1 and  $a_1, a_2, \dots, a_n$  arbitrary integers. Then the system

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

...

$$x \equiv a_n \pmod{m_n}$$

$m_i$  两两互素

has a unique solution modulo  $m = m_1 m_2 \cdots m_n$ .

# The Chinese Remainder Theorem

构造性证明

公则与两两互素矛盾

Proof Let  $M_k = m/m_k$  for  $k = 1, 2, \dots, n$  and  $m = m_1 m_2 \cdots m_n$ .

Since  $\gcd(m_k, M_k) = 1$  there is an integer  $y_k$ , an inverse of  $M_k$  modulo  $m_k$  such that  $M_k y_k \equiv 1 \pmod{m_k}$ . Let

定义了一步构造 若不互素⇒分解成两两互素  
 $x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_n M_n y_n$   
由 mod m

extended  
Euclidean  
Algorithm

$$\begin{cases} x \equiv 2 \pmod{2} \\ \vdots \\ x \equiv 2 \pmod{3} \\ \vdots \\ x \equiv 2 \pmod{7} \end{cases}$$

It is checked that  $x$  is a solution to the  $n$  congruences.

证明  $x \pmod{m_i} = ?$

$$a_k M_k y_k \pmod{m_i} = \begin{cases} 0 & k \neq i \\ a_i & k = i \end{cases}$$

(for  $M_k y_k \equiv 1 \pmod{m_k}$ )

# The Chinese Remainder Theorem

- **Proof** Let  $M_k = m/m_k$  for  $k = 1, 2, \dots, n$  and  $m = m_1 m_2 \cdots m_n$ . Since  $\gcd(m_k, M_k) = 1$ , there is an integer  $y_k$ , an inverse of  $M_k$  modulo  $m_k$  such that  $M_k y_k \equiv 1 \pmod{m_k}$ . Let

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_n M_n y_n.$$

It is checked that  $x$  is a solution to the  $n$  congruences.

证唯一性

How to prove the uniqueness of the solution modulo  $m$ ?

$$x \equiv a_i \pmod{m_i} \quad (1 \leq i \leq n)$$

$$\text{Goal: } x \equiv y \pmod{m}$$

$$y \equiv a_i \pmod{m_i}$$

$$x \equiv y \pmod{m_i}$$

$$\left\{ \begin{array}{l} x \equiv 2 \pmod{3} \\ x \equiv 2 \pmod{7} \\ \Downarrow \\ x \equiv 2 \pmod{21} \end{array} \right.$$

# The Chinese Remainder Theorem

## Example

$$x \equiv 2 \pmod{3} \quad M_1 = 35$$

$$x \equiv 3 \pmod{5} \quad M_2 = 21$$

$$x \equiv 2 \pmod{7} \quad M_3 = 15$$

都可以先去  
mod 再计算

$$M_1 y_1 \equiv 1 \pmod{3}$$

$$M_2 y_2 \equiv 1 \pmod{5}$$

$$M_3 y_3 \equiv 1 \pmod{7}$$

$$35 y_1 \equiv 1 \pmod{3} \Rightarrow 2y_1 \equiv 1 \pmod{3} \quad y_1 = 2$$

$$21 y_2 \equiv 1 \pmod{5} \Rightarrow y_2 \equiv 1 \pmod{5} \quad y_2 = 1$$

$$15 y_3 \equiv 1 \pmod{7} \Rightarrow y_3 \equiv 1 \pmod{7} \quad y_3 = 1$$

$$\begin{aligned} x &= 2M_1 y_1 + 3M_2 y_2 + 2M_3 y_3 \\ &= 2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1 \end{aligned}$$

# The Chinese Remainder Theorem

## ■ Example

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

Let  $m = 3 \cdot 5 \cdot 7 = 105$ ,  $M_1 = m/3 = 35$ ,  $M_2 = m/5 = 21$ ,  
 $M_3 = m/7 = 15$ .

$$35 \cdot 2 \equiv 1 \pmod{3}$$

$$21 \equiv 1 \pmod{5}$$

$$15 \equiv 1 \pmod{7}$$

# The Chinese Remainder Theorem

## ■ Example

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

Let  $m = 3 \cdot 5 \cdot 7 = 105$ ,  $M_1 = m/3 = 35$ ,  $M_2 = m/5 = 21$ ,  
 $M_3 = m/7 = 15$ .

$$35 \cdot 2 \equiv 1 \pmod{3} \quad y_1 = 2$$

$$21 \equiv 1 \pmod{5} \quad y_2 = 1$$

$$15 \equiv 1 \pmod{7} \quad y_3 = 1$$

# The Chinese Remainder Theorem

## ■ Example

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

Let  $m = 3 \cdot 5 \cdot 7 = 105$ ,  $M_1 = m/3 = 35$ ,  $M_2 = m/5 = 21$ ,  
 $M_3 = m/7 = 15$ .

$$35 \cdot 2 \equiv 1 \pmod{3} \quad y_1 = 2$$

$$21 \equiv 1 \pmod{5} \quad y_2 = 1$$

$$15 \equiv 1 \pmod{7} \quad y_3 = 1$$

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 \equiv 233 \equiv 23 \pmod{105}$$

# The Chinese Remainder Theorem

## ■ Example

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

三人同行七十稀，五树梅花廿一枝，  
七子团圆正月半，除百零五便得知。  
-- 程大位《算法统要》(1593年)

Let  $m = 3 \cdot 5 \cdot 7 = 105$ ,  $M_1 = m/3 = 35$ ,  $M_2 = m/5 = 21$ ,  $M_3 = m/7 = 15$ .

$$35 \cdot 2 \equiv 1 \pmod{3} \quad y_1 = 2$$

$$21 \equiv 1 \pmod{5} \quad y_2 = 1$$

$$15 \equiv 1 \pmod{7} \quad y_3 = 1$$

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 \equiv 233 \equiv 23 \pmod{105}$$

# Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

# Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

## Example

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

# Back Substitution

- We may also solve systems of linear congruences with pairwise relatively prime moduli by *back substitution*.

向后回代

$$x = 3k + 2$$

$$3k + 2 \equiv 3 \pmod{5}$$

$$3k \equiv 1 \pmod{5}$$

$$k \equiv 2 \pmod{5}$$

$$k = 5l + 2$$

中间步骤是前几个方程的解

$$3(5l + 2) + 2 = 15l + 8 \equiv 2 \pmod{7}$$

$$l \equiv 1 \pmod{7}$$

$$l = 7m + 1$$

$$x = 15l + 8 = 15(7m + 1) + 8 = 105m + 23$$

$$x = 15l + 8 \rightarrow x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 8 \pmod{15}$$

$$x \equiv 2 \pmod{21}$$

$$x \equiv 2 \pmod{3}$$

$$x \equiv 2 \pmod{7}$$

# Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let  $p$  be a prime, and let  $x$  be an integer such that  $x \not\equiv 0 \pmod{p}$ .  
Then  $\Leftrightarrow p \nmid x$

$$x^{p-1} \equiv 1 \pmod{p}.$$

# Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let  $p$  be a prime, and let  $x$  be an integer such that  $x \not\equiv 0 \pmod{p}$ . Then

$$x^{p-1} \equiv 1 \pmod{p}.$$

**Example:** Find  $7^{222} \pmod{11}$

$$p=11$$

$$7^0 \equiv 1 \pmod{11}$$

$$7^{10 \times 22 + 2} \equiv (7^0)^{22} \times 7^2 \equiv 49 \pmod{11}$$
$$\equiv 5$$

# Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let  $p$  be a prime, and let  $x$  be an integer such that  $x \not\equiv 0 \pmod{p}$ . Then

$$x^{p-1} \equiv 1 \pmod{p}.$$

**Example:** Find  $7^{222} \pmod{11}$

$$7^{222} = 7^{22 \cdot 10 + 2} = (7^{10})^{22} 7^2 = 1^{22} \cdot 49 \equiv 5 \pmod{11}$$

# Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let  $p$  be a prime, and let  $x$  be an integer such that  $x \not\equiv 0 \pmod{p}$ . Then

$$x^{p-1} \equiv 1 \pmod{p}.$$

**Example:** Find  $7^{222} \pmod{11}$

$$7^{222} = 7^{22 \cdot 10 + 2} = (7^{10})^{22} 7^2 = 1^{22} \cdot 49 \equiv 5 \pmod{11}$$

**Q :** How to prove Fermat's little theorem?

# Fermat's Little Theorem

- **Theorem (Fermat's little theorem)** : Let  $p$  be a prime, and let  $x$  be an integer such that  $x \not\equiv 0 \pmod{p}$ . Then

$$x^{p-1} \equiv 1 \pmod{p}.$$

**Example:** Find  $7^{222} \pmod{11}$

$$7^{222} = 7^{22 \cdot 10 + 2} = (7^{10})^{22} 7^2 = 1^{22} \cdot 49 \equiv 5 \pmod{11}$$

**Q :** How to prove Fermat's little theorem?

$$\{1, 2, \dots, p-1\} = \{x, 2x, \dots, x(p-1) \pmod{p}\}$$

$$A = \{1, 2, \dots, p-1\} = \mathbb{Z}^+ P \setminus \{0\}$$

$$B = \{x, 2x, \dots, (p-1)x \pmod p\}$$

① prove  $A=B$ :  $A \subseteq \mathbb{Z}^+ P$ ,  $B \subseteq \mathbb{Z}^+ P$

只要证两两不相等

$$\text{assume } ix \equiv jx \pmod p$$

$$\Rightarrow p \mid (i-j)x$$

$$\gcd(p, x) = 1 \Rightarrow p \nmid i-j$$

$$i, j \in \{1, 2, \dots, p-1\} \Rightarrow i-j=0$$

两两不相等

②  $1 \cdot 2 \cdots (p-1) \equiv \underline{x \cdot 2x \cdots (p-1)x} \pmod p$

可除去互素的数                  这样有  $x^{p-1}$

$$\Rightarrow x^{p-1} \equiv 1 \pmod p$$

# Euler's Theorem

- Euler's *totient* function:  $\phi(n)$  欧拉函数

the number of positive integers coprime to  $n$  in  $\mathbb{Z}_n$

与 n 互素的个数

# Euler's Theorem

- Euler's *totient* function:  $\phi(n)$

the number of positive integers coprime to  $n$  in  $\mathbb{Z}_n$

$p, q$  都是素数

$$\phi(p) = p - 1 \quad \text{是 } q \text{ 个 } p \text{ 的倍数, } pq - p - q + 1$$

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

$$\phi(p^i) = p^i - p^{i-1}$$

# Euler's Theorem

- Euler's *totient* function:  $\phi(n)$

the number of positive integers coprime to  $n$  in  $\mathbb{Z}_n$

$$\phi(p) = p - 1$$

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

$$\phi(p^i) = p^i - p^{i-1}$$

- **Theorem (Euler's theorem)** : Let  $n$  be a positive integer, and let  $x$  be an integer such that  $\gcd(x, n) = 1$ . Then

$$x^{\phi(n)} \equiv 1 \pmod{n}.$$

弱化:  $n=p$  素数

# Euler's Theorem

- Euler's *totient* function:  $\phi(n)$

the number of positive integers coprime to  $n$  in  $\mathbb{Z}_n$

$$\phi(p) = p - 1$$

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

$$\phi(p^i) = p^i - p^{i-1}$$

- Theorem (Euler's theorem)** : Let  $n$  be a positive integer, and let  $x$  be an integer such that  $\gcd(x, n) = 1$ . Then

$$x^{\phi(n)} \equiv 1 \pmod{n}. \text{ 这个集合大小是 } \phi(n)$$

*Q* : How to prove Euler's theorem?  $Z_n^* = \{ i \in Z_n^* \mid \gcd(i, n) = 1 \}$   
 $|Z_n^*| = \phi(n)$

# Primitive Roots

- A *primitive root* modulo a prime  $p$  is an integer  $r \in \mathbb{Z}_p$  such that every nonzero element of  $\mathbb{Z}_p$  is a power of  $r$ .

# Primitive Roots

本原根

- A *primitive root* modulo a prime  $p$  is an integer  $r \in \mathbb{Z}_p$  such that every nonzero element of  $\mathbb{Z}_p$  is a power of  $r$ .

**Example:** 3 is a primitive root of  $\mathbb{Z}_7$ . 2 is **not** a primitive root of  $\mathbb{Z}_7$ .

# Primitive Roots

- A *primitive root* modulo a prime  $p$  is an integer  $r \in \mathbb{Z}_p$  such that every nonzero element of  $\mathbb{Z}_p$  is a power of  $r$ .

**Example:** 3 is a primitive root of  $\mathbb{Z}_7$ . 2 is **not** a primitive root of  $\mathbb{Z}_7$ .

$$\begin{array}{ll} 3^1 \equiv 3 & 3^5 \equiv 5 \\ 3^2 \equiv 2 & 3^6 \equiv 1 \\ 3^3 \equiv 6 & \quad (\text{mod } 7) \\ 3^4 \equiv 4 & \end{array}$$

其幂次表括  $\mathbb{Z}_p$

**Theorem** \* There is a primitive root modulo  $n$  if and only if  $n = 2, 4, p^e$  or  $2p^e$ , where  $p$  is an odd prime.

Q : proof? The number of primitive roots? \*

# Number Theory and Cryptography

- Division, Primes
- Congruence
- Greatest Common Divisor (GCD)
- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = \color{blue}{d}q + r$$

- Congruence

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

(extended) Euclidean algorithm

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

Find the GCD of 286 and 503.

$\gcd(503, 286)$	$503 = 1 \cdot 286 + 217$	$1 = 10 - 1 \cdot 9$
$= \gcd(286, 217)$	$286 = 1 \cdot 217 + 69$	$1 = 7 \cdot 10 - 1 \cdot 69$
$= \gcd(217, 69)$	$217 = 3 \cdot 69 + 10$	$1 = 7 \cdot 217 - 22 \cdot 69$
$= \gcd(69, 10)$	$69 = 6 \cdot 10 + 9$	$1 = 29 \cdot 217 - 22 \cdot 286$
$= \gcd(10, 9)$	$10 = 1 \cdot 9 + 1$	$1 = 29 \cdot 503 - 51 \cdot 286$
$= 1$	$9 = 9 \cdot 1$	

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

(extended) Euclidean algorithm

find the modular inverse

solve linear congruence  $ax \equiv b \pmod{m}$  ( $\gcd(a, m) = 1$ )

- Euler's Theorem / Fermat's Little Theorem

# Number Theory and Cryptography

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

(extended) Euclidean algorithm

find the modular inverse

solve linear congruence  $ax \equiv b \pmod{m}$  ( $\gcd(a, m) = 1$ )

Chinese Remainder Theorem / back substitution

- Euler's Theorem / Fermat's Little Theorem

# Number Theory Summary

- Division, Primes

$$a = d q + r \quad q = a \text{ div } d \quad r = a \text{ mod } d$$

- Congruence

$$a \equiv b \pmod{m} \text{ if } m \text{ divides } a - b$$

- Greatest Common Divisor (GCD)

(extended) Euclidean algorithm

find the modular inverse

solve linear congruence  $ax \equiv b \pmod{m}$  ( $\gcd(a, m) = 1$ )

Chinese Remainder Theorem / back substitution

- Euler's Theorem / Fermat's Little Theorem

$$x^{\phi(n)} \equiv 1 \pmod{n} \text{ if } \gcd(x, n) = 1$$

$$x^{p-1} \equiv 1 \pmod{p} \text{ if } x \not\equiv 0 \pmod{p}$$

# Modular Arithmetic in CS

- Modular arithmetic and congruencies are used in CS:
  - ◊ Pseudorandom number generators
  - ◊ Hash functions
  - ◊ Cryptography

# Pseudorandom Number Generators

## ■ *Linear congruential method*

We choose four numbers:

- ◊ the modulus  $m$
- ◊ multiplier  $a$
- ◊ increment  $c$
- ◊ seed  $x_0$

# Pseudorandom Number Generators

## ■ *Linear congruential method*

线性同余法

We choose four numbers:

- ◊ the modulus  $m$
- ◊ multiplier  $a$
- ◊ increment  $c$
- ◊ seed  $x_0$

We generate a sequence of numbers  $x_1, x_2, \dots, x_n, \dots$  with  $0 \leq x_i < m$  by using the congruence

$$x_{n+1} = \underbrace{(ax_n + c)}_{\text{mod } m}$$

# Pseudorandom Number Generators

- *Linear congruential method*

$$x_{n+1} = (ax_n + c) \pmod{m}$$

# Pseudorandom Number Generators

## ■ *Linear congruential method*

$$x_{n+1} = (ax_n + c) \pmod{m}$$

**Example:**

密码学上安全性很弱  
观察到几个值就可以还原所有参数

- Assume :  $m=9, a=7, c=4, x_0 = 3$

- $x_1 = 7*3+4 \pmod{9} = 25 \pmod{9} = 7$

- $x_2 = 53 \pmod{9} = 8$

- $x_3 = 60 \pmod{9} = 6$

- $x_4 = 46 \pmod{9} = 1$

- $x_5 = 11 \pmod{9} = 2$

- $x_6 = 18 \pmod{9} = 0$

- ....

能确定过程的一定不随机  
真实的随机数：(一定和物理过程相关)  
抛硬币、杨氏双缝干涉

# Hash Functions

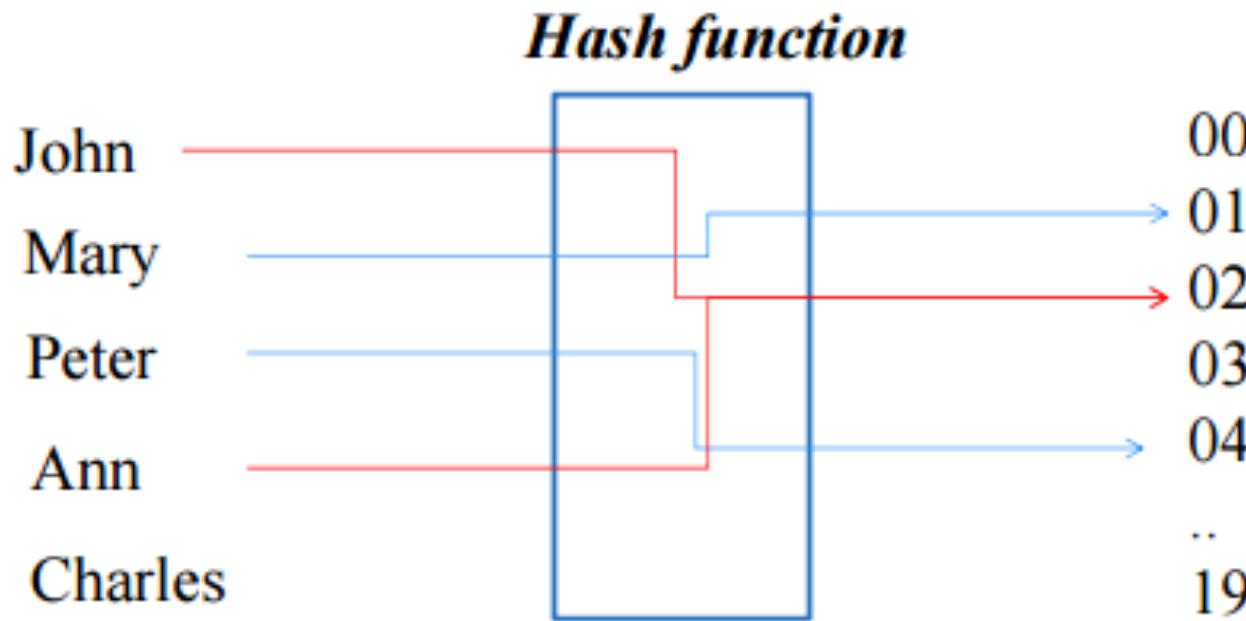
- A *hash function* is an algorithm that maps data of arbitrary length to data of a fixed length. The values returned by a hash function are called *hash values* or *hash codes*.

访存要常数复杂度，基本都是哈希  
(算出来的)

# Hash Functions

- A *hash function* is an algorithm that maps data of arbitrary length to data of a fixed length. The values returned by a hash function are called *hash values* or *hash codes*.

**Example:**



# Hash Functions

- **Problem:** Given a large collection of records, how can we store and find a record quickly?

# Hash Functions

- **Problem:** Given a large collection of records, how can we store and find a record quickly?

**Solution:** Use a hash function, calculate the location of the record based on the record's ID.

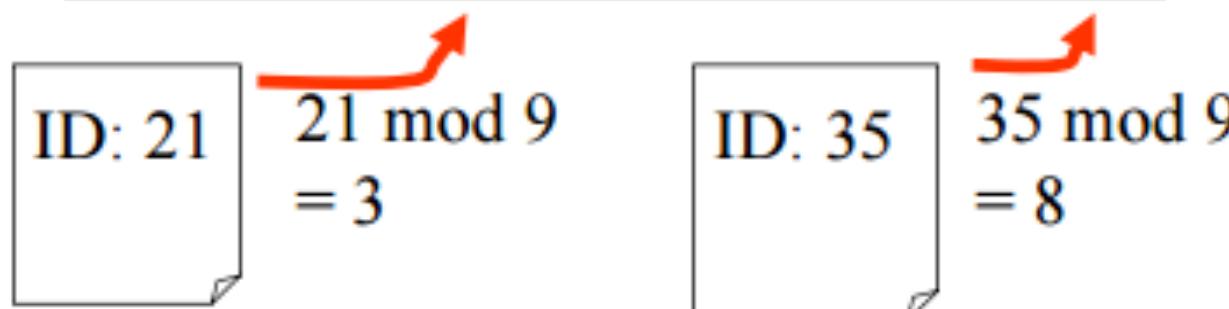
**Example:** A common hash function is

- $h(k) = k \bmod n,$

where  $n$  is the number of available storage locations.

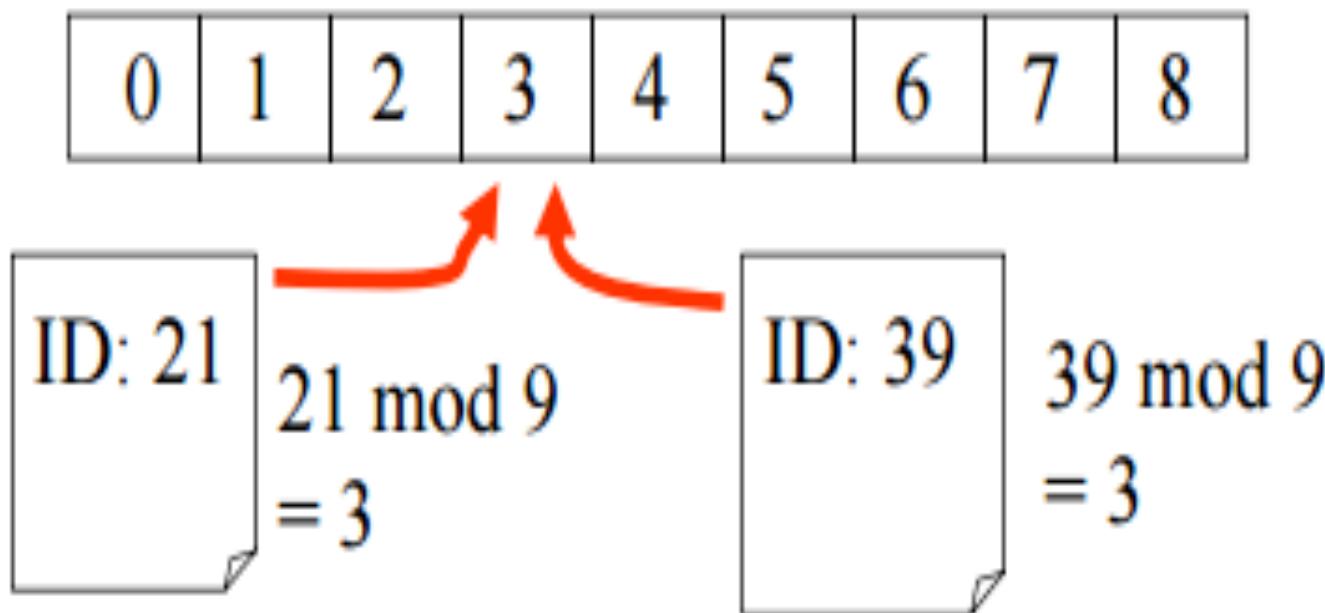
哈希碰撞：不同ID但哈希值相同

0	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---



# Hash Functions

- Two records mapped to the same location



# Hash Functions

- Solution 1: move to the next available location

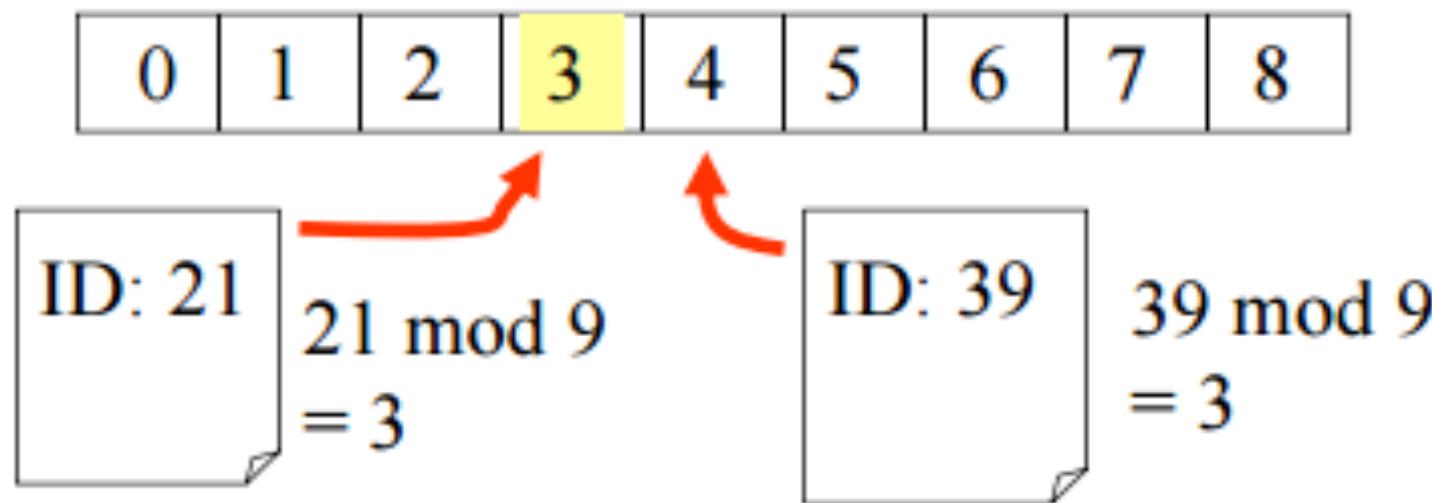
try

$$h_0(k) = k \bmod n$$

$$h_1(k) = (k+1) \bmod n$$

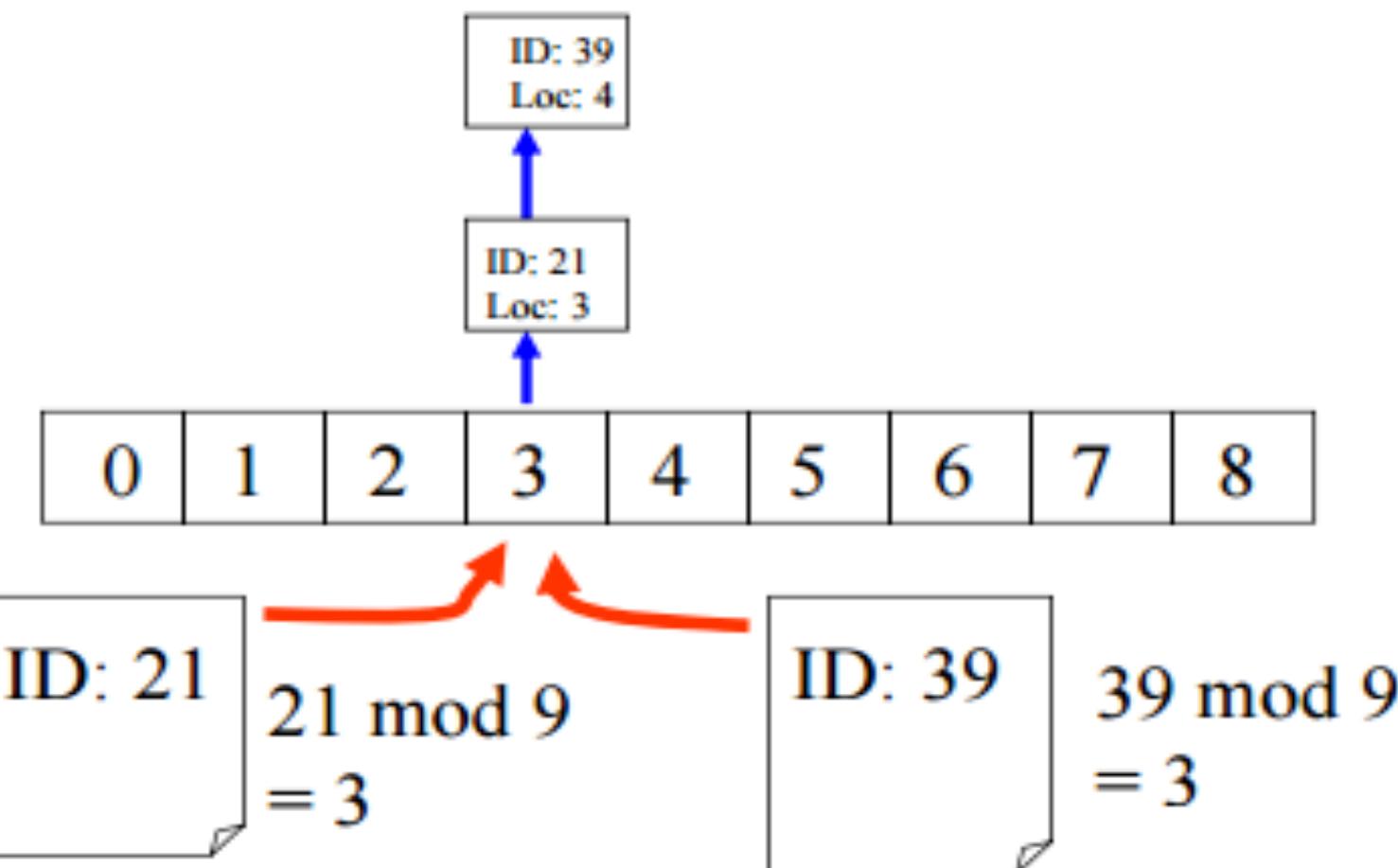
...

$$h_m(k) = (k+m) \bmod n$$



# Hash Functions

- Solution 2: remember the exact location in a secondary structure that is searched sequentially



# Applications of Number Theory in Cryptography

- Introduction
- Symmetric cryptography
- Asymmetric cryptography
- RSA Cryptosystem
- DLP and El Gamal cryptography
- Diffie-Hellman key exchange protocol
- Cryptocurrency, e.g., bitcoin

# Cryptography

- History of almost 4000 years (from 1900 B.C.)

Cryptography = kryptos + graphos

# Cryptography

- History of almost 4000 years (from 1900 B.C.)

Cryptography = kryptos + graphos

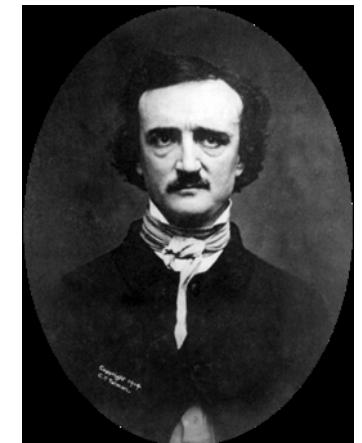
(secret)    (writing)

# Cryptography

- History of almost 4000 years (from 1900 B.C.)

Cryptography = kryptos + graphos  
(secret)    (writing)

The term was first used in *The Gold-Bug*, by Edgar Allan Poe (1809 - 1849).



# Cryptography

- History of almost 4000 years (from 1900 B.C.)

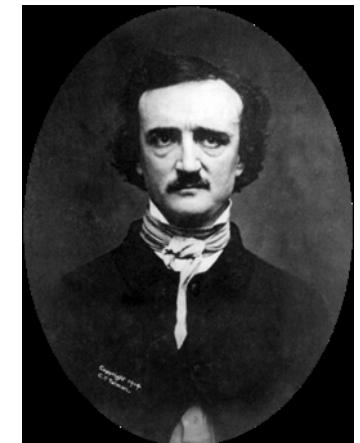
Cryptography = kryptos + graphos  
(secret)    (writing)

The term was first used in *The Gold-Bug*, by Edgar Allan Poe (1809 - 1849).

爱伦坡

“Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve.” – 1841

不对：可以通过人类智慧  
加密方式公开，但密钥保密  
不依赖算法本身，不然维护成本很高



# Cryptography

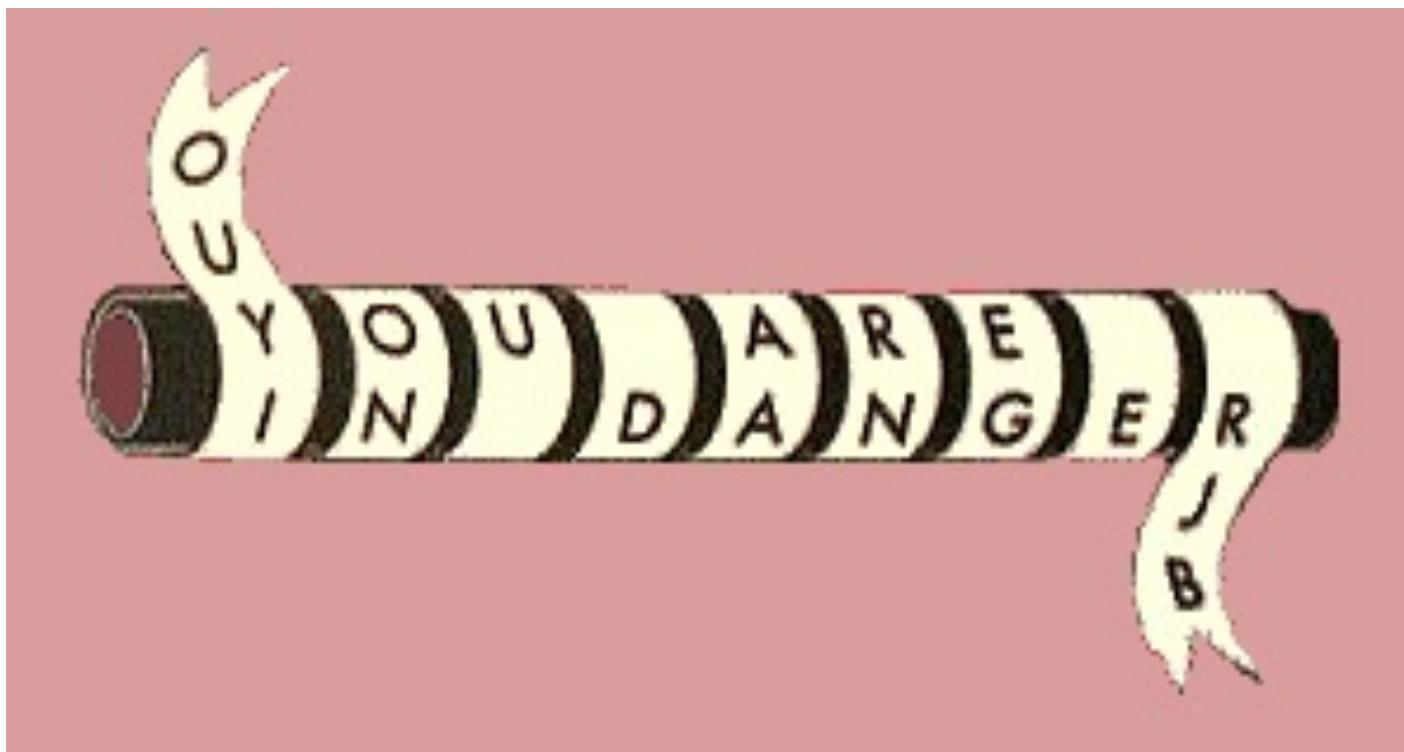
- One-sentence definition:

“Cryptography is the practice and study of techniques for secure communication in the presence of third parties called *adversaries*.<sup>敌手</sup>” – Ronald L. Rivest



# Some Examples

- In 405 BC, the Greek general LYSANDER OF SPARTA was sent a coded message written on the inside of a servant's belt.



# Some Examples

- The Greeks also invented a cipher which changed **letters** to **numbers**. A form of this code was still being used during *World War I*.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

# Some Examples

- Caesar Cipher (after the name of JULIUS CAESAR)



VENI, VIDI, VICI

YHQL YLGL YLFL

# Some Examples

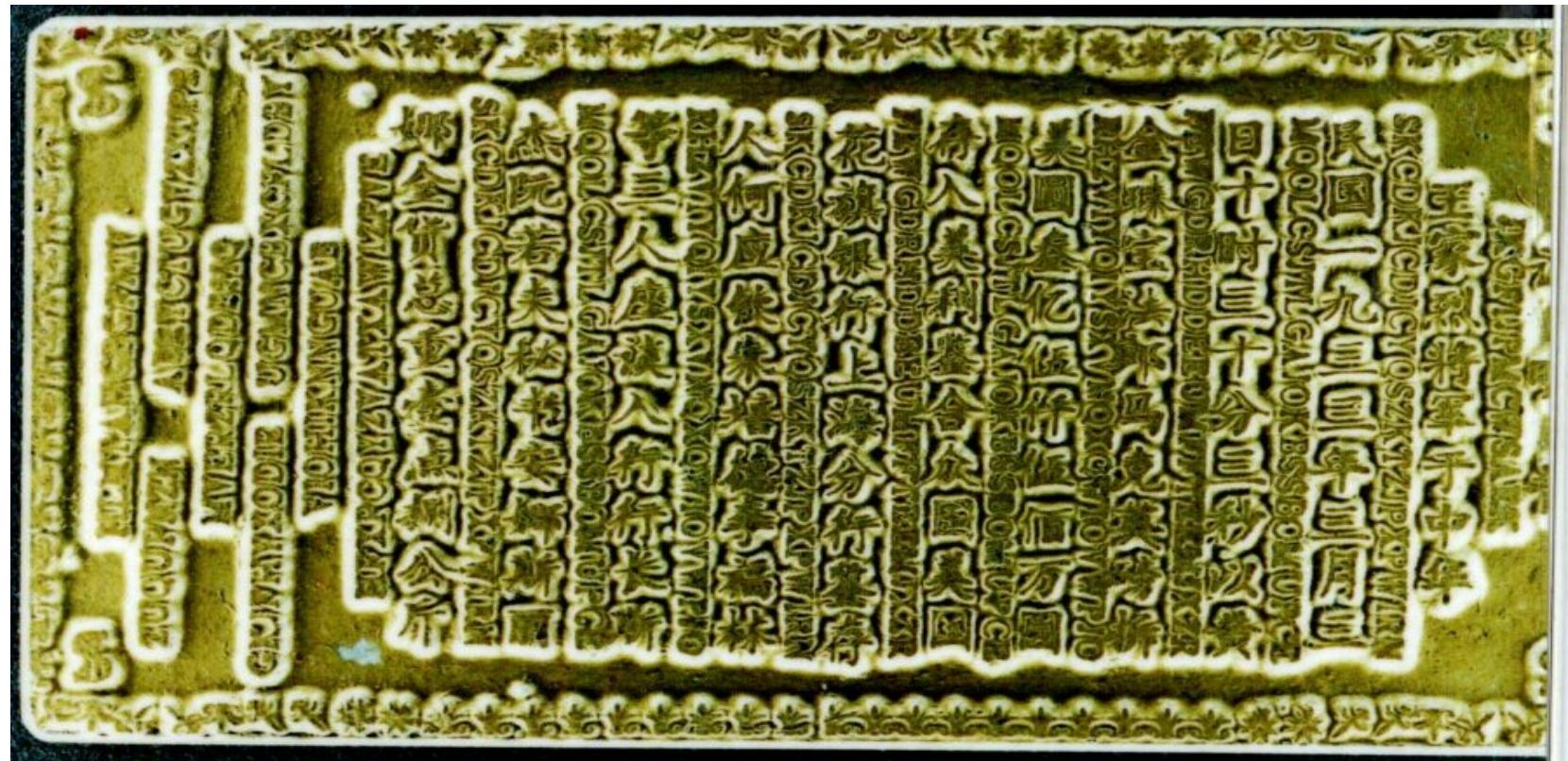
- Morse Code: created by Samuel Morse in 1838

出现频率  
最高

Morse Alphabet	
A • -	O - - -
B - • • •	P • - - - •
C - • - •	Q - - • -
D - • •	R • - •
E •	S • • •
F • • - - •	T -
G - - •	U • • -
H • • • •	V • • • -
I • •	W • - -
J • - - -	X - • • -
K - • -	Y - • - -
L • - • •	Z - - • •
M - -	
N - •	
Full stop (.) • - • - • -	
Break signal or fresh line - - - - -	
Apostrophe ('') • - - - - - •	
Hyphen (-) - • • • -	
Exclamation (!) - - • • - -	
Interrogation (?) • • - - • •	
Underline (_____ ) • • - - • -	
Parenthesis ( ) - • - - • -	
Inverted commas (" ") • - • • - •	

# Some Examples

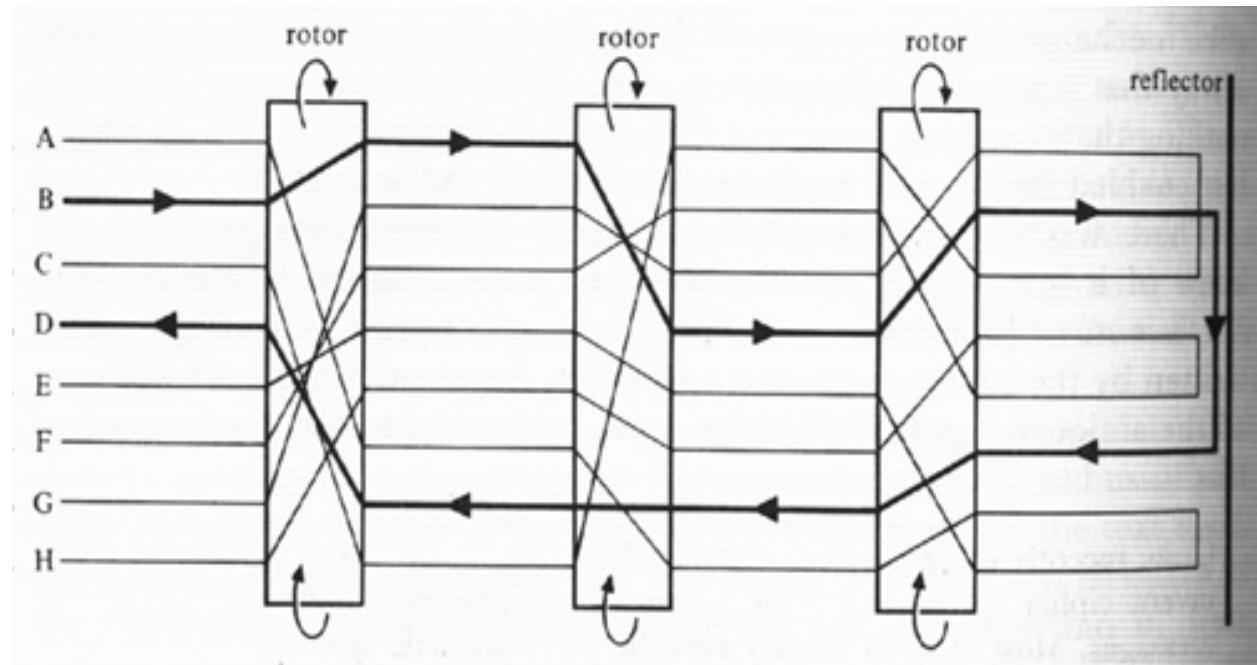
- Crytograms from the Chinese gold bars



<http://www.iacr.org/misc/china/china.html>

# Some Examples

- Enigma, Germany coding machine in *World War II*.



# Some Examples

- Sigaba, used by U.S. during *World War II*.



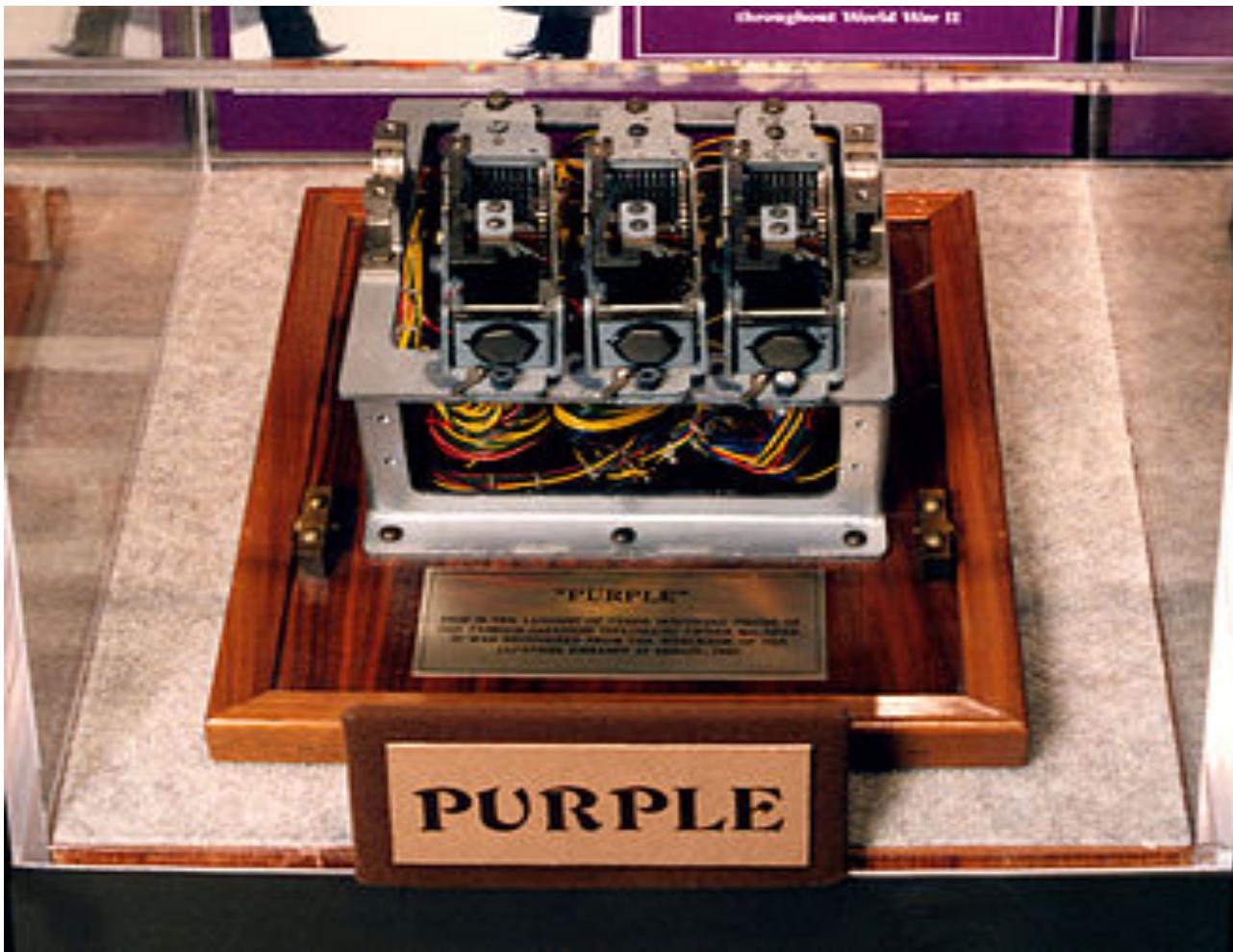
# Some Examples

- Japanese “Enigma” Rotor Cipher Machine



# Some Examples

- Japanese Purple Machine (97-shiki obun inji-ki)



# People Working in Breaking Codes



Alan Turing  
(1912-1954)



Claude E. Shannon  
(1916-2001)

# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography

# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography



# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography



# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography



# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography



# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography



They need agree in advance on the *secret key k*.

# Cryptography History

- History (until 1970's)

“*Symmetric*” cryptography



They need agree in advance on the *secret key k*.

*Q*: How can they do this?

# Cryptography History

## ■ History (until 1970's)

“*Symmetric*” cryptography



They need agree in advance on the *secret key k*.

*Q*: How can they do this?

*Q*: What if Bob could send Alice a “special key” useful only for *encryption* but no help for *decryption*?

只有一种角色的钥匙 既加密又解密

# Caesar Cipher

- Key:  $k = 0, 1, \dots, 25$

Encryption: encode  $i$  as  $(i + k) \bmod 26$

Decryption: decode  $j$  as  $(j - k) \bmod 26$

# Caesar Cipher

- Key:  $k = 0, 1, \dots, 25$

Encryption: encode  $i$  as  $(i + k) \bmod 26$

Decryption: decode  $j$  as  $(j - k) \bmod 26$

plaintext: SEND REINFORCEMENT

Key: 2

ciphertext: UGPF TGKPHQTEGOGPV

# Caesar Cipher

- Key:  $k = 0, 1, \dots, 25$

Encryption: encode  $i$  as  $(i + k) \bmod 26$

Decryption: decode  $j$  as  $(j - k) \bmod 26$

plaintext: SEND REINFORCEMENT

Key: 2

ciphertext: UGPF TGKPHQTEGOGPV

Problem: only 26 possibilities for keys!

# Caesar Cipher

- Key:  $k = 0, 1, \dots, 25$

Encryption: encode  $i$  as  $(i + k) \bmod 26$

Decryption: decode  $j$  as  $(j - k) \bmod 26$

plaintext: SEND REINFORCEMENT

Key: 2

ciphertext: UGPF TGKPHQTEGOGPV

Problem: only 26 possibilities for keys!

Kerchoff's Principle (1883): System should be secure even if algorithms are known, as long as key is secret.

# Substitution Cipher

- Key: table mapping each letter to another letter

A	B	C		Z
V	R	E		D

# Substitution Cipher

- Key: table mapping each letter to another letter

A	B	C		Z
V	R	E		D

Encryption & Decryption: letter by letter according to table

# Substitution Cipher

- Key: table mapping each letter to another letter

A	B	C		Z
V	R	E		D

Encryption & Decryption: letter by letter according to table

# of possible keys:  $26! \approx 4 \times 10^{26}$

# Substitution Cipher

- Key: table mapping each letter to another letter

A	B	C		Z
V	R	E		D

Encryption & Decryption: letter by letter according to table

# of possible keys:  $26! \approx 4 \times 10^{26}$

However, substitution cipher is still **insecure!**

# Substitution Cipher

- **Key:** table mapping each letter to another letter

A	B	C		Z
V	R	E		D

**Encryption & Decryption:** letter by letter according to table

**# of possible keys:**  $26! \approx 4 \times 10^{26}$

However, substitution cipher is still **insecure!**

**Key observation:** can recover plaintext using *statistics* on *letter frequencies*.

# Substitution Cipher

Table 1: Relative frequencies of the letters of the English language

Letter	Relative Frequency (%)	Letter	Relative Frequency (%)
a	8.167	n	6.749
b	1.492	o	7.507
c	2.782	p	1.929
d	4.253	q	0.095
e	12.702	r	5.987
f	2.228	s	6.327
g	2.015	t	9.056
h	6.094	u	2.758
i	6.966	v	0.978
j	0.153	w	2.360
k	0.772	x	0.150
l	4.025	y	1.974
m	2.406	z	0.074

置换密族可用概率试

# Substitution Cipher

Table 2: Number of Diagraphs Expected in 2,000 Letters of English Text

th	-	50	at	-	25	st	-	20
er	-	40	en	-	25	io	-	18
on	-	39	es	-	25	le	-	18
an	-	38	of	-	25	is	-	17
re	-	36	or	-	25	ou	-	17
he	-	33	nt	-	24	ar	-	16
in	-	31	ea	-	22	as	-	16
ed	-	30	ti	-	22	de	-	16
ne	-	30	to	-	22	rt	-	16
ha	-	26	it	-	20	ve	-	16

Table 3: The 15 Most Common Trigraphs in the English Language

1	-	the	6	-	tio	11	-	edt
2	-	and	7	-	for	12	-	tis
3	-	tha	8	-	nde	13	-	oft
4	-	ent	9	-	has	14	-	sth
5	-	ion	10	-	nce	15	-	men

# Substitution Cipher

- LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYLXZIXLIKIIXPPIJVSZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

# Substitution Cipher

- LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYLXZIXLIKIIXPPIJVSZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

**I** – *most common letter*

**LI** – *most common pair*

**XLI** – *most common triple*

# Substitution Cipher

■ LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYlxzixlikiiXPIJVsZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

I – *most common letter*

I = e

LI – *most common pair*

L = h

XLI – *most common triple*

X = t

# Substitution Cipher

■ LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYLXZIXLIKIIXPPIJVSZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

I – *most common letter*

I = e

LI – *most common pair*

L = h

XLI – *most common triple*

X = t

LIVI = he?e

# Substitution Cipher

■ LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYLYZIXLIKIIXPPIJVSZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

I – *most common letter*

I = e

LI – *most common pair*

L = h

XLI – *most common triple*

X = t

LIVI = he?e

V = r

# Substitution Cipher

■ LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYLXZIXLIKIIXPPIJVSZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

I – *most common letter*

I = e

LI – *most common pair*

L = h

XLI – *most common triple*

X = t

LIVI = he?e

V = r

E = a

Y = g

# Substitution Cipher

■ LIVITCSWPIYVEWHEVSRIQMXXLEYVEOIEWHRXEXIPFE  
MVEWHKVSTYLYZIXLIIKIIXPPIJVSZEYPERRGERIMWQL  
MGLMXQERIWGPSRIHMXQEREKI

I – *most common letter*

I = e

LI – *most common pair*

L = h

XLI – *most common triple*

X = t

LIVI = he?e

V = r

E = a

Y = g

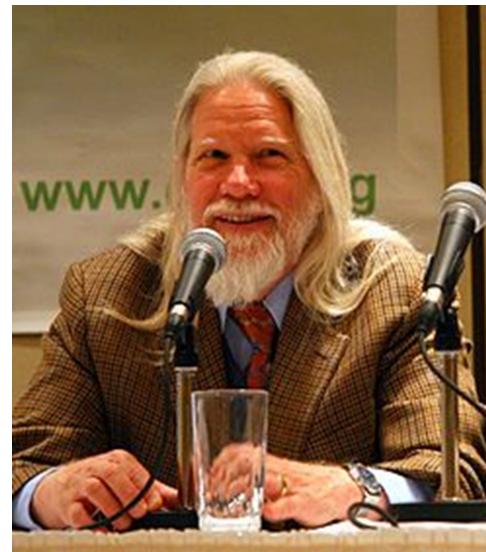
HereUpOnLeGrandAroseWithAGraveAndStatelyAirAndBroug  
MeTheBeetleFromAGlassCaselnWhichItWasEnclosedIt-  
WasABe

# Cryptography History

- History (from 1976)
  - ◊ W. Diffie, M. Hellman, “New direction in cryptography”, *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.

“We stand today on the brink of a revolution in cryptography.”

$\text{DES} \rightarrow 3\text{DES} \rightarrow \text{AES}$



Bailey W. Diffie

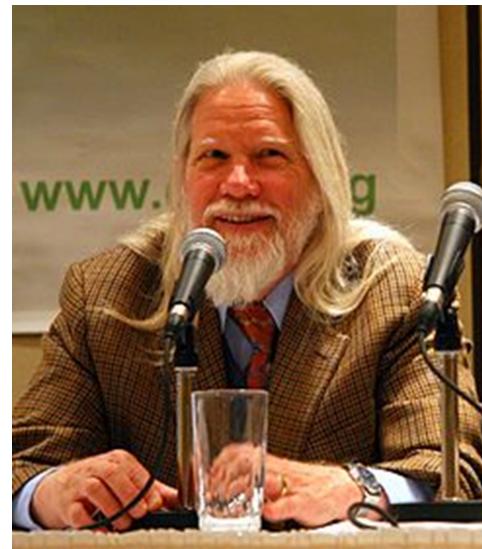
Martin E. Hellman

# Cryptography History

## ■ History (from 1976)

- ◊ W. Diffie, M. Hellman, “New direction in cryptography”, *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.

“We stand today on the brink of a revolution in cryptography.”



## 2015 Turing Award

Bailey W. Diffie

Martin E. Hellman

2015	<a href="#">Martin E. Hellman</a> <a href="#">Whitfield Diffie</a>	For fundamental contributions to <b>modern cryptography</b> . Diffie and Hellman's groundbreaking 1976 paper, "New Directions in Cryptography," <sup>[39]</sup> introduced the ideas of public-key cryptography and digital signatures, which are the foundation for most regularly-used security protocols on the internet today. <sup>[40]</sup>
------	---	--

# Public Key Cryptography

- Alice wants to send a message to Bob



# Public Key Cryptography

- Alice wants to send a message to Bob



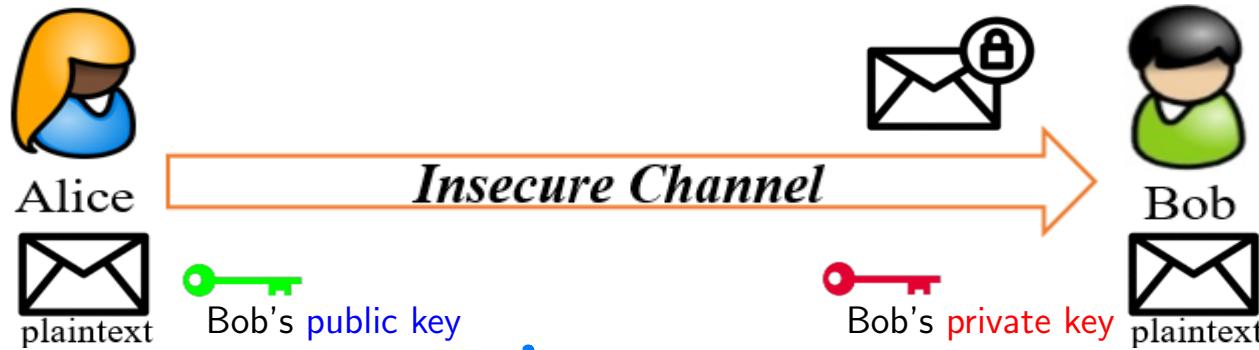
# Public Key Cryptography

- Alice wants to send a message to Bob



# Public Key Cryptography

- Alice wants to send a message to Bob



数学上单向函数  
一个方向好计算，另一个方向很难

# Public Key Cryptography

- Alice wants to send a message to Bob



Ronald L. Rivest



Adi Shamir



Leonard M. Adleman

R. Rivest, A. Shamir, L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”,  
*Communications of the ACM*, vol. 21-2, pages 120-126, 1978.

# RSA Public Key Cryptosystem

## ■ Rivest-Shamir-Adleman

2002 **Turing Award**

2002

[Ronald L. Rivest](#),  
[Adi Shamir](#) and  
[Leonard M. Adleman](#)

For [their ingenious contribution](#) for making [public-key cryptography](#) useful in practice.

# RSA Public Key Cryptosystem

## ■ Rivest-Shamir-Adleman

2002 **Turing Award**

2002

[Ronald L. Rivest](#),  
[Adi Shamir](#) and  
[Leonard M. Adleman](#)

For their ingenious contribution for making [public-key cryptography](#) useful in practice.

Pick two **large** primes,  $p$  and  $q$ . Let  $n = pq$ , then  $\phi(n) = (p - 1)(q - 1)$ . Encryption and decryption keys  $e$  and  $d$  are selected such that

- $\gcd(e, \phi(n)) = 1$
- $ed \equiv 1 \pmod{\phi(n)}$

# RSA Public Key Cryptosystem

## ■ Rivest-Shamir-Adleman

2002 **Turing Award**

2002

[Ronald L. Rivest](#),  
[Adi Shamir](#) and  
[Leonard M. Adleman](#)

For their ingenious contribution for making [public-key cryptography](#) useful in practice.

Pick two **large** primes,  $p$  and  $q$ . Let  $n = pq$ , then  $\phi(n) = (p - 1)(q - 1)$ . Encryption and decryption keys  $e$  and  $d$  are selected such that

- $\gcd(e, \phi(n)) = 1$
- $ed \equiv 1 \pmod{\phi(n)}$

$C = M^e \pmod{n}$  (RSA encryption)

$M = C^d \pmod{n}$  (RSA decryption)

# RSA Public Key Cryptosystem

- $C = M^e \bmod n$  (RSA encryption)
- $M = C^d \bmod n$  (RSA decryption)

**Theorem (Correctness)** : Let  $p$  and  $q$  be two odd primes, and define  $n = pq$ . Let  $e$  be relatively prime to  $\phi(n)$  and let  $d$  be the multiplicative inverse of  $e$  modulo  $\phi(n)$ . For each integer  $x$  such that  $0 \leq x < n$ ,

$$x^{ed} \equiv x \pmod{n}.$$

# RSA Public Key Cryptosystem

- $C = M^e \bmod n$  (RSA encryption)

- $M = C^d \bmod n$  (RSA decryption)

**Theorem (Correctness)** : Let  $p$  and  $q$  be two odd primes, and define  $n = pq$ . Let  $e$  be relatively prime to  $\phi(n)$  and let  $d$  be the multiplicative inverse of  $e$  modulo  $\phi(n)$ . For each integer  $x$  such that  $0 \leq x < n$ ,

$$x^{ed} \equiv x \pmod{n}.$$

**Q** : How to prove this?

# RSA Public Key Cryptosystem: Example

Parameters:	$p$	$q$	$n$	$\phi(n)$	$e$	$d$
	5	11	55	40	7	23

# RSA Public Key Cryptosystem: Example

Parameters:	$p$	$q$	$n$	$\phi(n)$	$e$	$d$
	5	11	55	40	7	23

**Public key:** (7, 55)

**Private key:** 23

# RSA Public Key Cryptosystem: Example

Parameters:	$p$	$q$	$n$	$\phi(n)$	$e$	$d$
	5	11	55	40	7	23

**Public key:** (7, 55)

**Private key:** 23

**Encryption:**  $M = 28, C = M^7 \bmod 55 = 52$

**Decryption:**  $M = C^{23} \bmod 55 = 28$

# RSA Public Key Cryptosystem: Parameters

**Parameters:**  $p$      $q$      $n$      $\phi(n)$      $e$      $d$

**Public key:**  $(e, n)$

**Private key:**  $d$

$p, q, \phi(n)$  must be kept **secret!**

# RSA Public Key Cryptosystem: Parameters

**Parameters:**  $p$      $q$      $n$      $\phi(n)$      $e$      $d$

**Public key:**  $(e, n)$

**Private key:**  $d$

$p, q, \phi(n)$  must be kept **secret!**

**Q :** Why?

# RSA Public Key Cryptosystem: Parameters

**Parameters:**  $p$      $q$      $n$      $\phi(n)$      $e$      $d$

**Public key:**  $(e, n)$

**Private key:**  $d$

$p, q, \phi(n)$  must be kept **secret!**

**Q :** Why?

**Comment:** It is believed that determining  $\phi(n)$  is equivalent to factoring  $n$ . Meanwhile, determining  $d$  given  $e$  and  $n$ , appears to be at least as time-consuming as the integer factoring problem.

# RSA Public Key Cryptosystem: Parameters

**Parameters:**  $p$      $q$      $n$      $\phi(n)$      $e$      $d$

**Public key:**  $(e, n)$

**Private key:**  $d$

$p, q, \phi(n)$  must be kept **secret**!

**Q :** Why?

**Comment:** It is believed that determining  $\phi(n)$  is equivalent to factoring  $n$ . Meanwhile, determining  $d$  given  $e$  and  $n$ , appears to be at least as time-consuming as the integer factoring problem.

CS 208 – Algorithm Design and Analysis

# The Security of the RSA

In practice, RSA keys are typically 1024 to 2048 bits long.

# The Security of the RSA

In practice, RSA keys are typically 1024 to 2048 bits long.

**Remark:** There are some suggestions for choosing  $p$  and  $q$ .

A. Salomaa, *Public-Key Cryptography*, 2nd Edition, Springer, 1996, pp. 134-136.

# The Security of the RSA

In practice, RSA keys are typically 1024 to 2048 bits long.

**Remark:** There are some suggestions for choosing  $p$  and  $q$ .

A. Salomaa, *Public-Key Cryptography*, 2nd Edition, Springer, 1996, pp. 134-136.

**Q :** Consider the RSA system, where  $n = pq$  is the modulus. Let  $(e, d)$  be a key pair for the RSA. Define

$$\lambda(n) = \text{lcm}(p - 1, q - 1)$$

and compute  $d' = e^{-1} \bmod \lambda(n)$ . Will decryption using  $d'$  instead of  $d$  still work?

# Applications of RSA

- SSL/TLS protocol

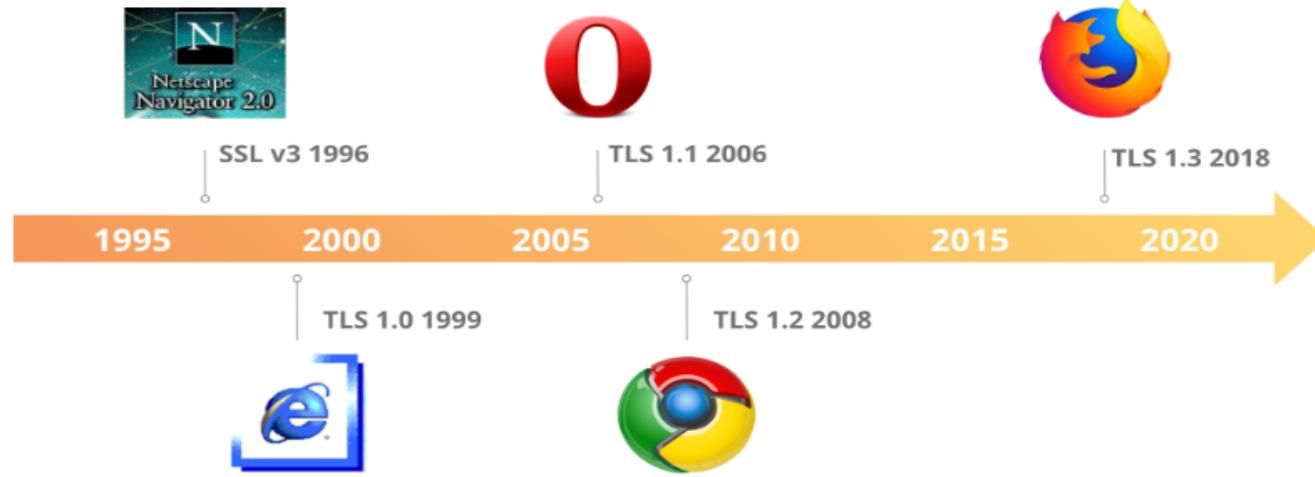
# Applications of RSA

- SSL/TLS protocol



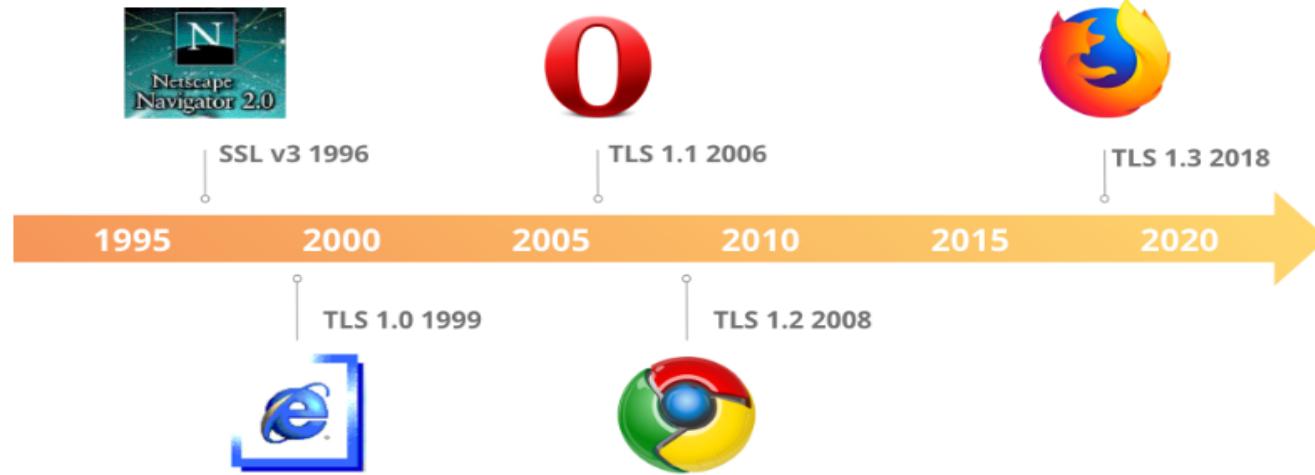
# Applications of RSA

## SSL/TLS protocol



# Applications of RSA

## SSL/TLS protocol

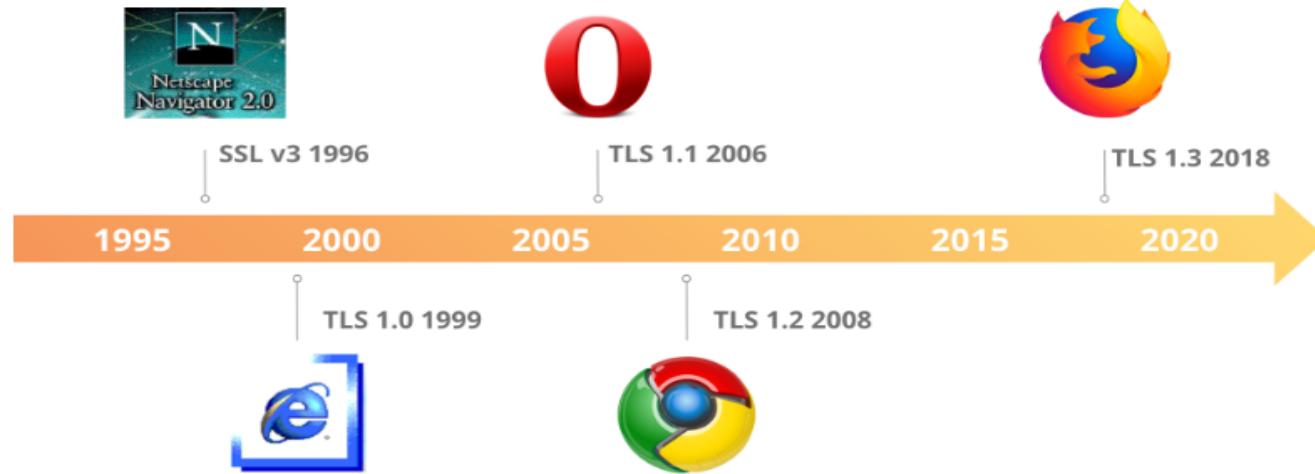


Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
<b>RSA</b>	Yes	Yes	Yes	Yes	Yes	No
<b>DH-RSA</b>	No	Yes	Yes	Yes	Yes	No
<b>DHE-RSA (forward secrecy)</b>	No	Yes	Yes	Yes	Yes	Yes
<b>ECDH-RSA</b>	No	No	Yes	Yes	Yes	No
<b>ECDHE-RSA (forward secrecy)</b>	No	No	Yes	Yes	Yes	Yes

# Applications of RSA

## SSL/TLS protocol

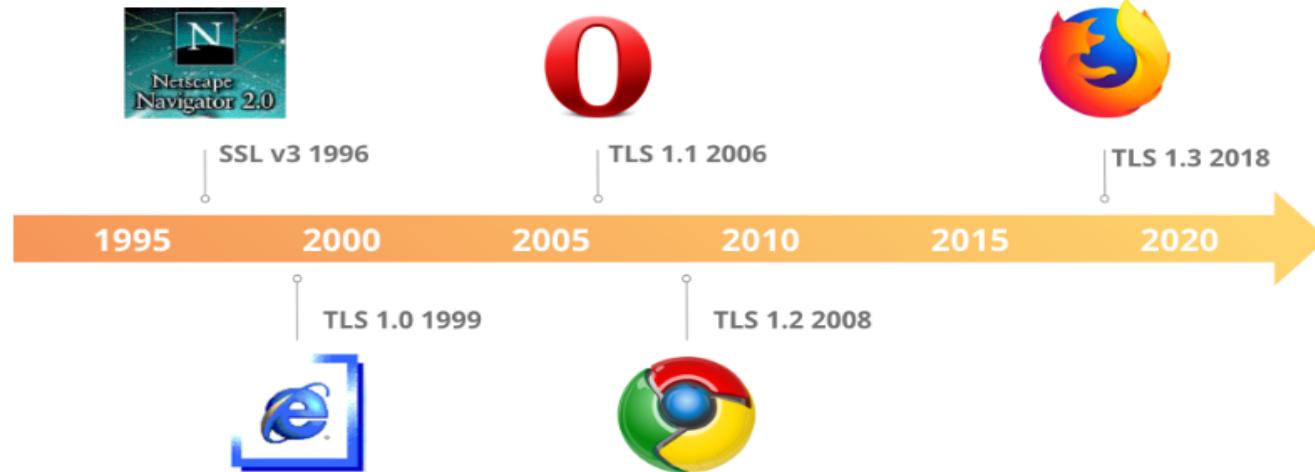


Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
<b>RSA</b>	Yes	Yes	Yes	Yes	Yes	No
<b>DH-RSA</b>	No	Yes	Yes	Yes	Yes	No
<b>DHE-RSA (forward secrecy)</b>	No	Yes	Yes	Yes	Yes	Yes
<b>ECDH-RSA</b>	No	No	Yes	Yes	Yes	No
<b>ECDHE-RSA (forward secrecy)</b>	No	No	Yes	Yes	Yes	Yes

CS 305 – Computer Networks

# Applications of RSA

## SSL/TLS protocol



Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes

CS 305 – Computer Networks

CS 403 – Cryptography and Network Security

# Using RSA for Digital Signature

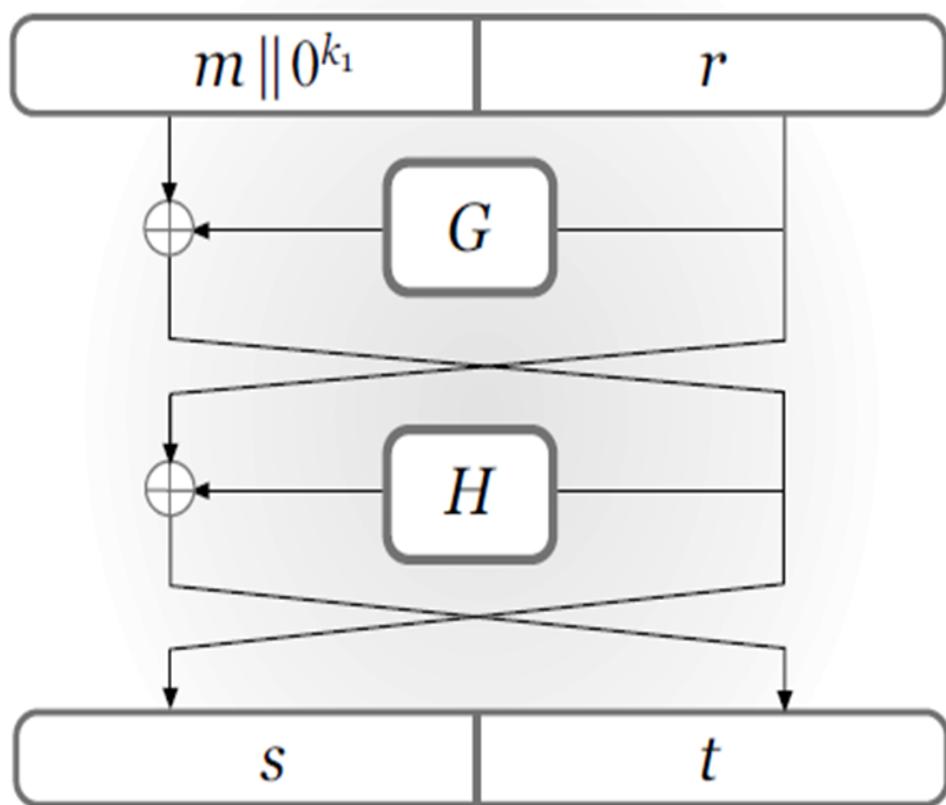
$$S = M^d \bmod n \text{ (RSA signature)}$$

$$M = S^e \bmod n \text{ (RSA verification)}$$

Why?

# RSA-OAEP Standard

- RSA-OAEP (Optimal Asymmetric Encryption Padding) is *IND-CCA2 secure*.
- PKCS#1 V2, RFC2437 Standard



# Next Lecture

- cryptography ...

