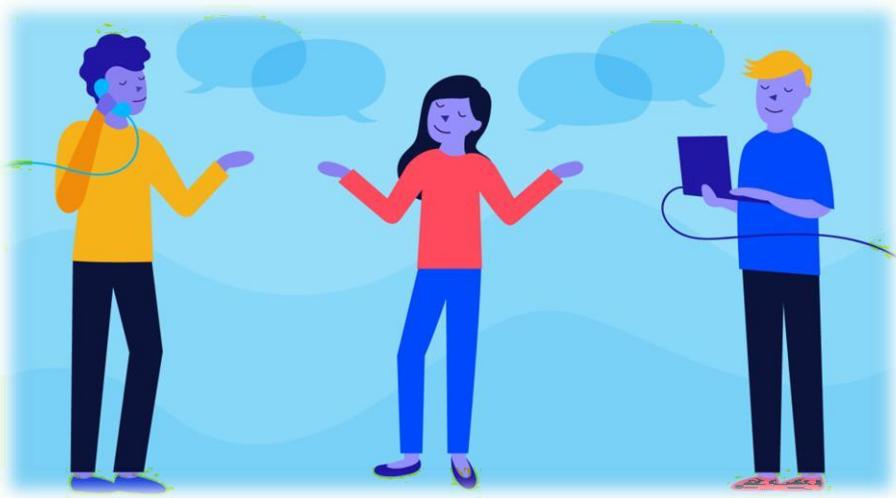


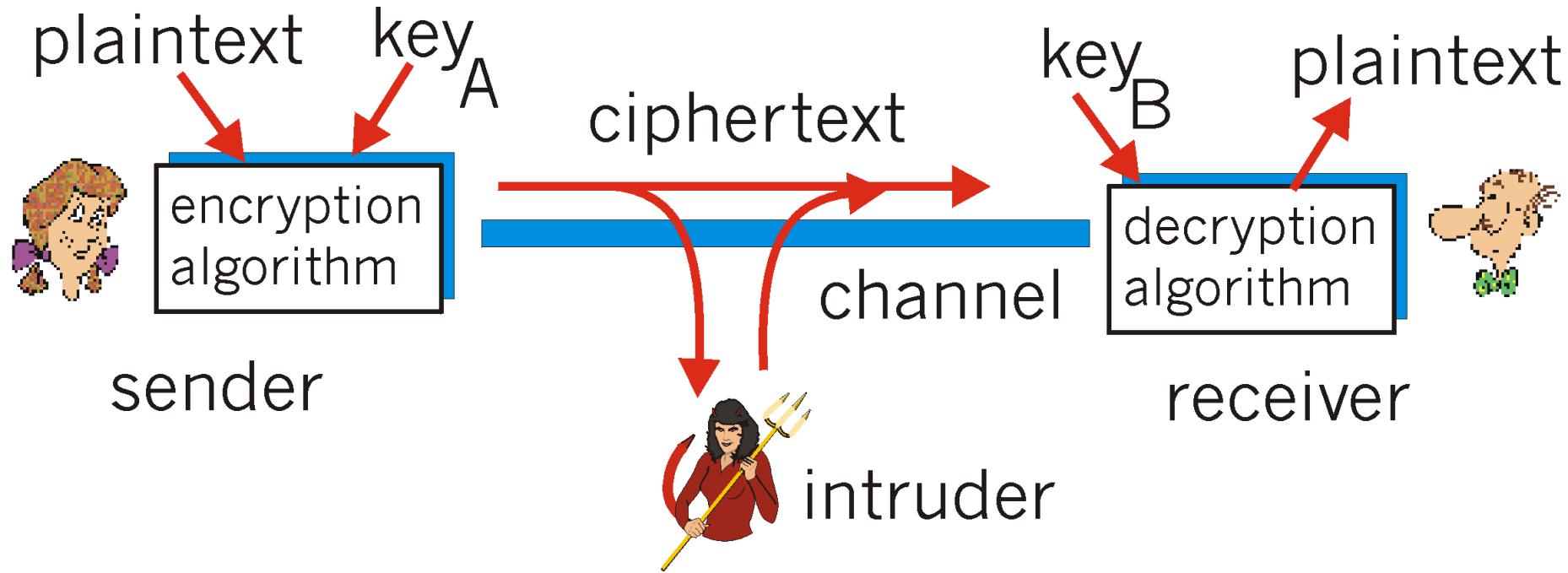
# Tổng quan về lý thuyết mật mã



**Human being** from ages had **two inherent needs**: (a) ***to communicate and share information*** and (b) ***to communicate selectively***.

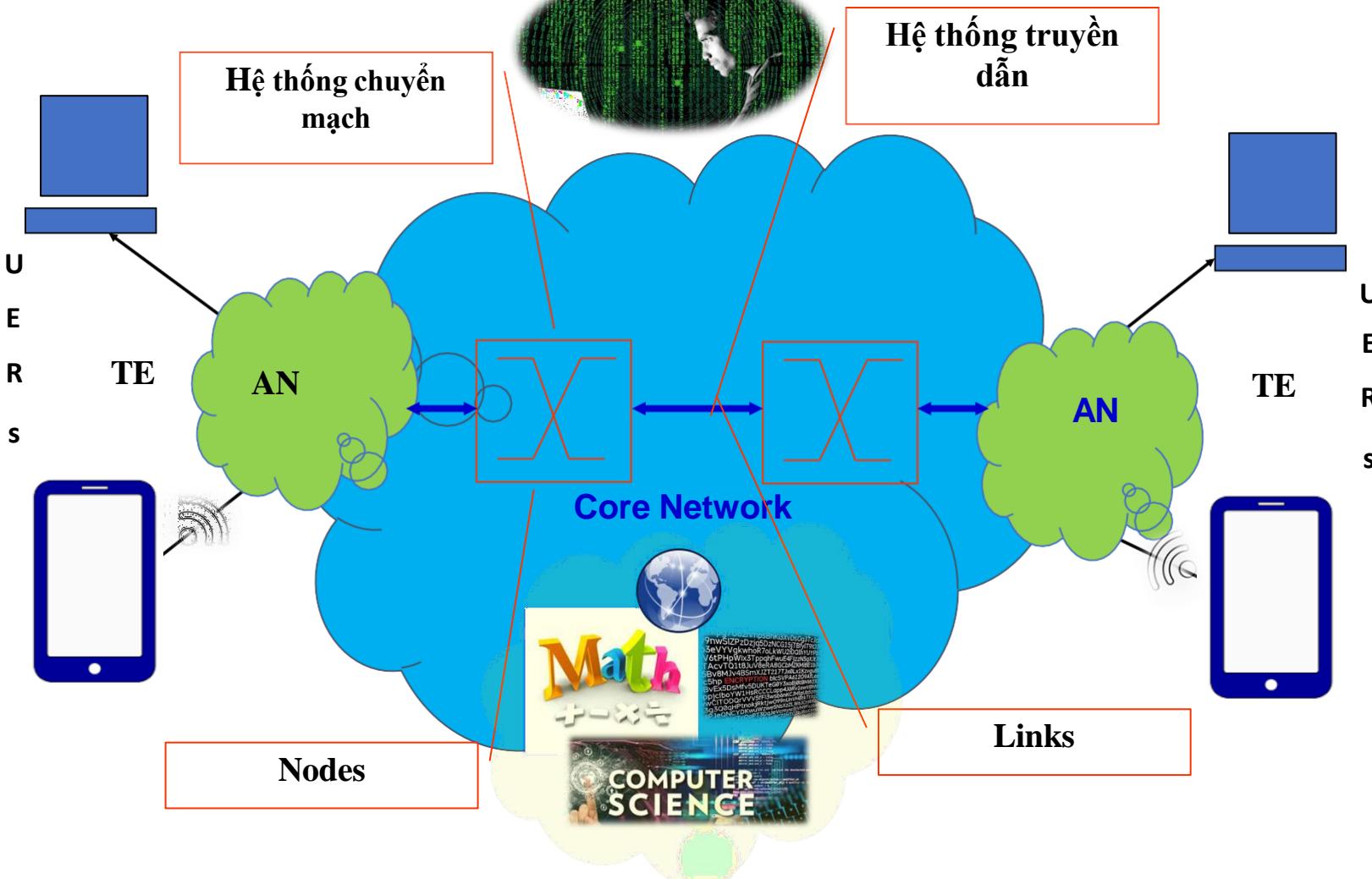
These two needs gave rise to the **art of coding the messages** in such a way that **only the intended people could have access to the information**. Unauthorized people could not extract any information, even if the scrambled messages fell in their hand.

# Tổng quan về lý thuyết mật mã

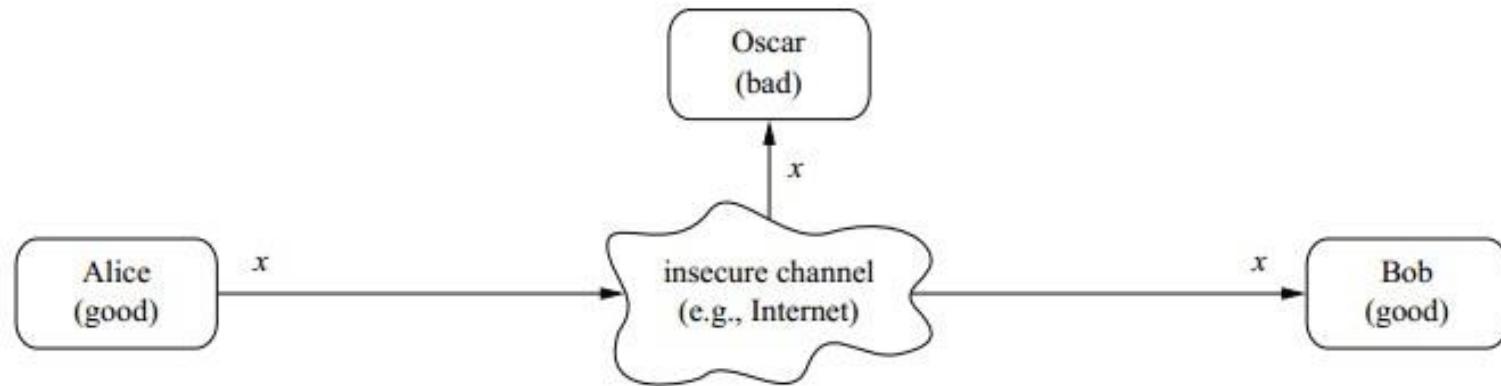


**Cryptography** means **changing the original text** into **unreadable** or **non intelligible format**, so that **third party** **cannot read** it and **message remain protected**.

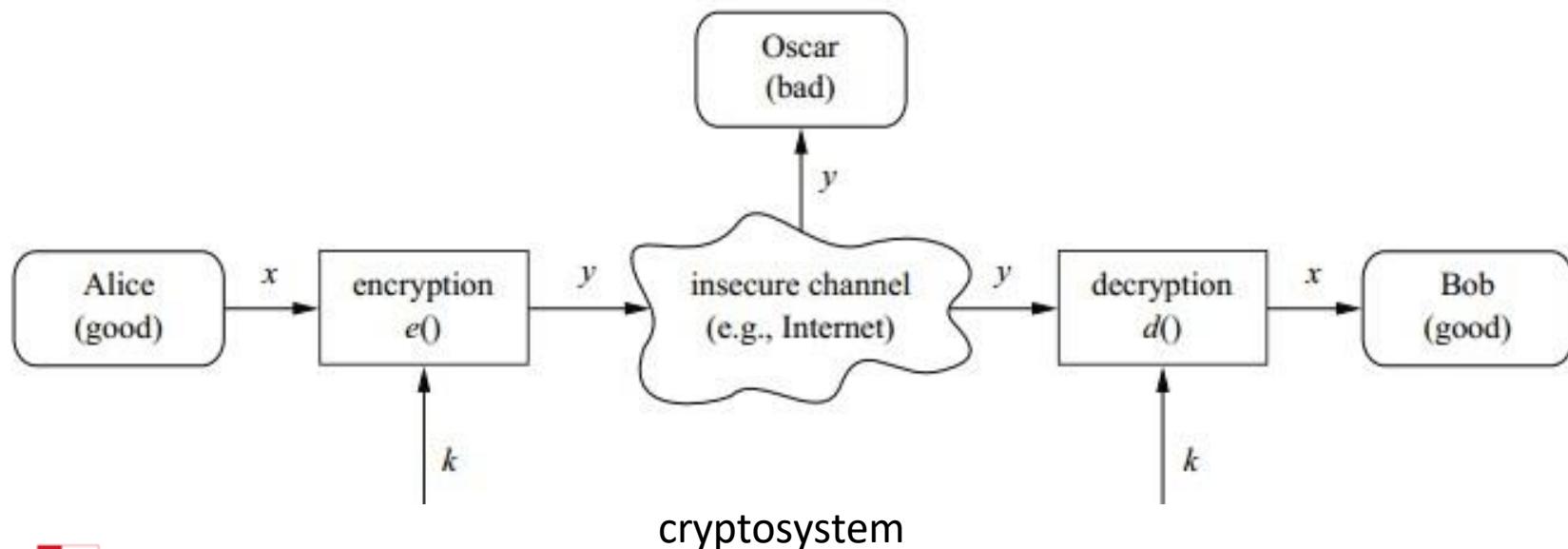
# Tổng quan về lý thuyết mật mã



# Tổng quan về lý thuyết mật mã



Communication over an insecure channel



cryptosystem

Một sơ đồ hệ thống mật mã  
là một bộ năm

$$S = (P, C, K, E, D)$$

Thỏa mãn các điều kiện sau đây:

- Tập nguồn P là tập hữu hạn tất cả các bản tin nguồn cần mã hóa có thể có.
- C là một tập hữu hạn các ký tự bản mã
- K là tập hữu hạn các khóa có thể được sử dụng
- E là một ánh xạ từ  $K \times P$  vào C, được gọi là phép lập mật mã
- D là một ánh xạ từ  $K \times C$  vào P, được gọi là phép giải mã

# Tổng quan về lý thuyết mật mã



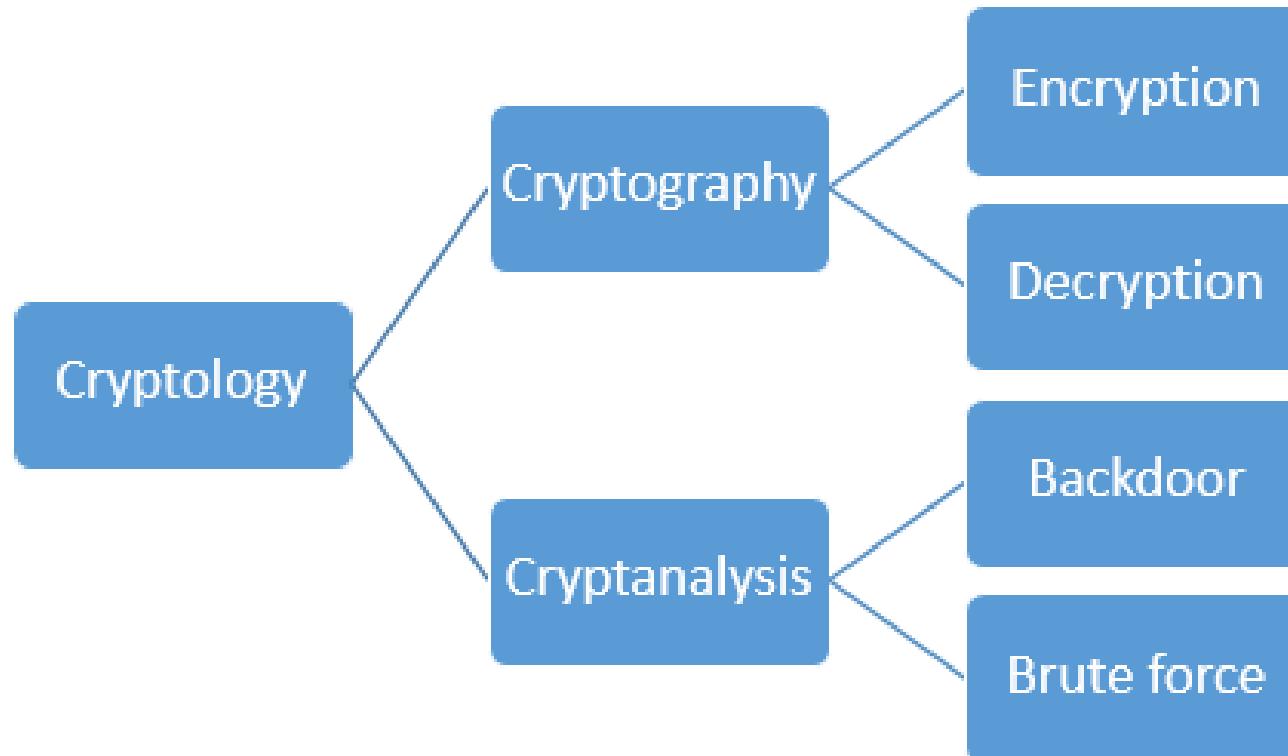
- **Confidentiality** is associated with authentication and authorization, where only *authorized individuals/systems should be able to view sensitive or classified information.*
- *Tính bí mật của thông tin là tính giới hạn về đối tượng được quyền truy xuất đến thông tin.*
- **Integrity** is about *data being trustworthy and not tampered with thus being trustworthy- correct, authentic, and reliable.*
- *Đặc trưng này đảm bảo sự tồn tại nguyên vẹn của thông tin, loại trừ mọi sự thay đổi thông tin có chủ đích hoặc hư hỏng, mất mát thông tin do sự cố thiết bị hoặc phần mềm.*
- **Availability** refers to the ability to *ensure authorized users have timely and reliable access to relevant resources as and when required.*
- *Tính khả dụng của thông tin là tính sẵn sàng của thông tin cho các nhu cầu truy xuất hợp lệ.*

# Tổng quan về lý thuyết mật mã

Mật mã học (Cryptology)

=

Mật mã (Cryptography) + Thám mã  
(Cryptanalysis)



# Tổng quan về lý thuyết mật mã

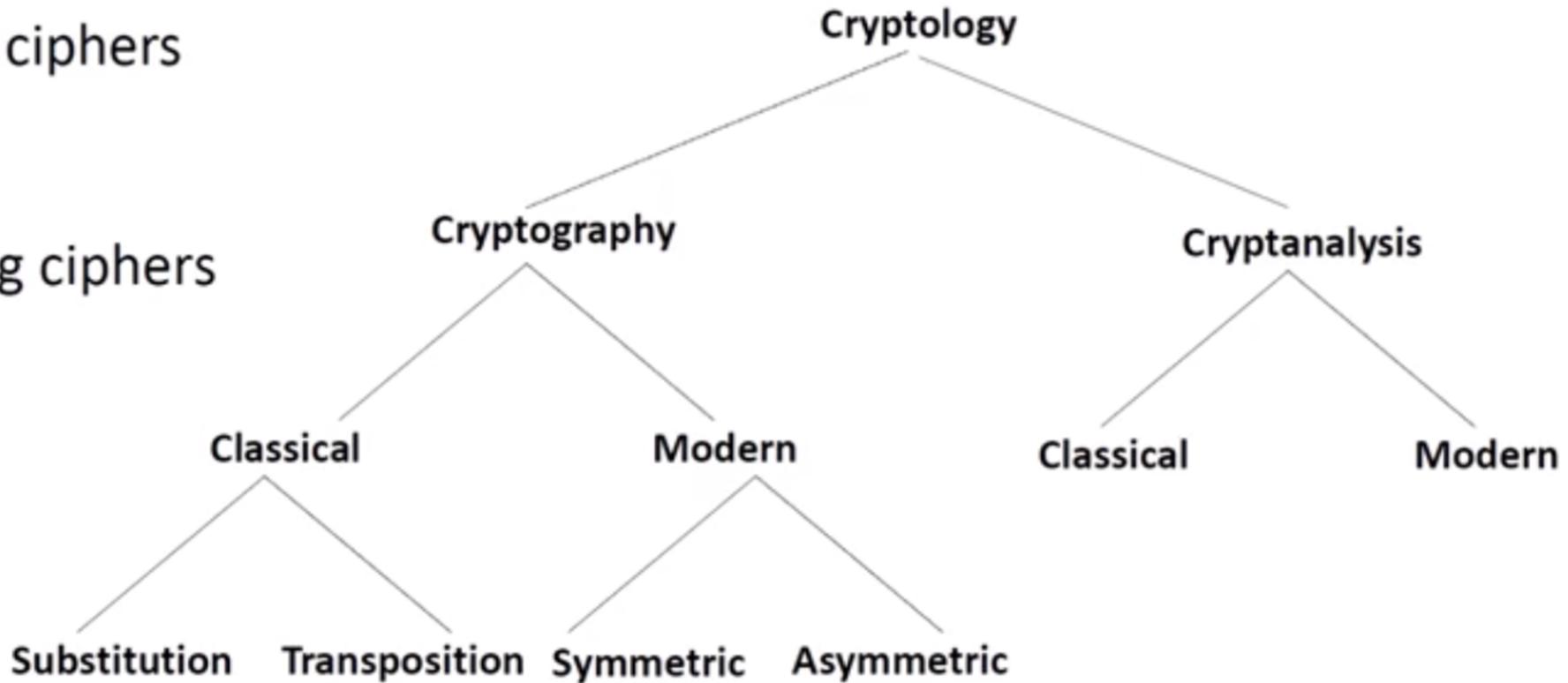
## Mật mã học (Cryptology)

**Cryptography**

Art of making ciphers

**Cryptanalysis**

Art of breaking ciphers



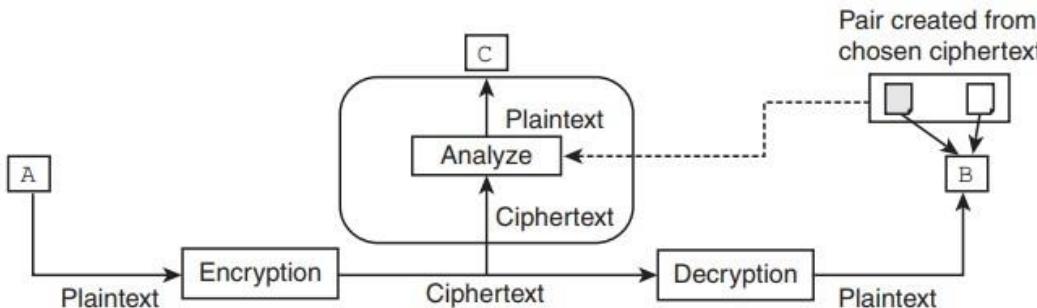
# Hệ mật cổ điển

Classic Cryptography	Modern Cryptography
<p>It manipulates traditional <u>characters</u>, i.e., letters and digits directly.</p>	<p>It operates on <u>binary bit sequences</u>.</p>
<p>It is mainly based on '<u>security through obscurity</u>'. The techniques employed for coding were kept secret and only the parties involved in communication knew about them.</p>	<p>It relies on <u>publicly known mathematical algorithms</u> for coding the information. Secrecy is obtained through a <u>secrete key</u> which is used as the seed for the algorithms. The computational difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information even if he knows the algorithm used for coding.</p>
<p>It requires the <u>entire cryptosystem</u> for communicating confidentially.</p>	<p>Modern cryptography requires parties interested in secure communication to possess the <u>secret key only</u>.</p>

# Hệ mật hiện đại

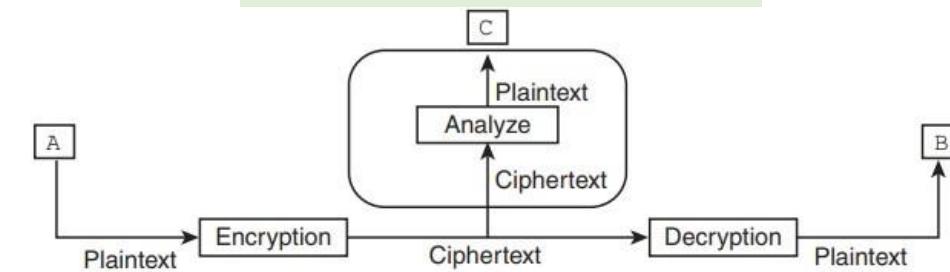
## • Các bài toán thám mã:

1. **Thám mã chỉ biết bản mã**: là bài toán phổ biến nhất, khi người thám mã chỉ biết một bản mật mã Y
2. **Thám mã khi biết cả bản rõ**: người thám mã biết một bản mật mã Y cùng với bản rõ tương ứng X.
3. **Thám mã khi có bản rõ được chọn**: người thám mã có thể chọn một bản rõ X, và biết bản mật mã tương ứng Y. Điều này có thể xảy ra khi người thám mã chiếm được (tạm thời) máy lập mã;
4. **Thám mã khi có bản mã được chọn**: người thám mã có thể chọn một bản mật mã Y, và biết bản rõ tương ứng X. Điều này có thể xảy ra khi người thám mã chiếm được tạm thời máy giải mã.

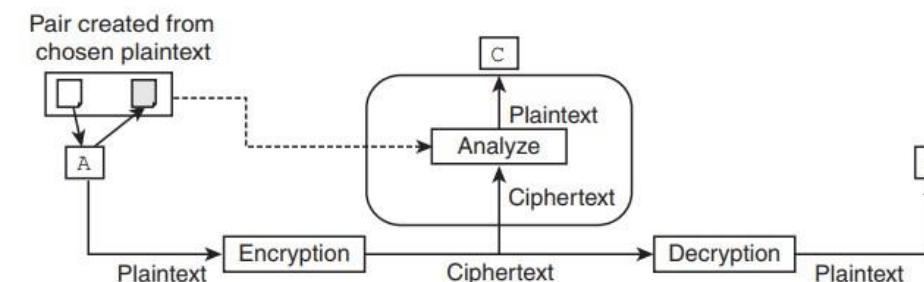
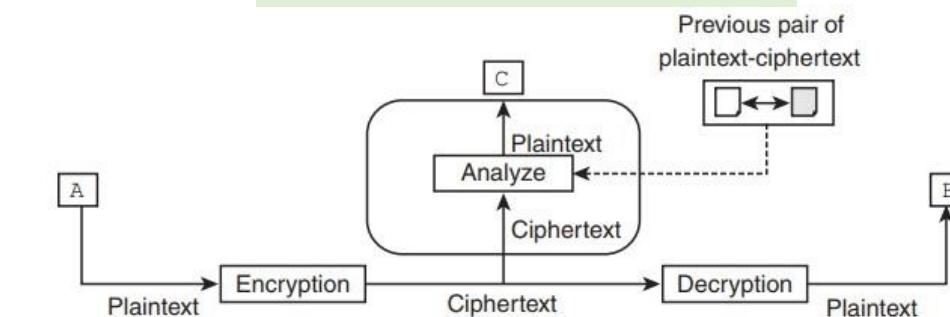


4. Chosen-ciphertext attack

## 1. Ciphertext-only attack



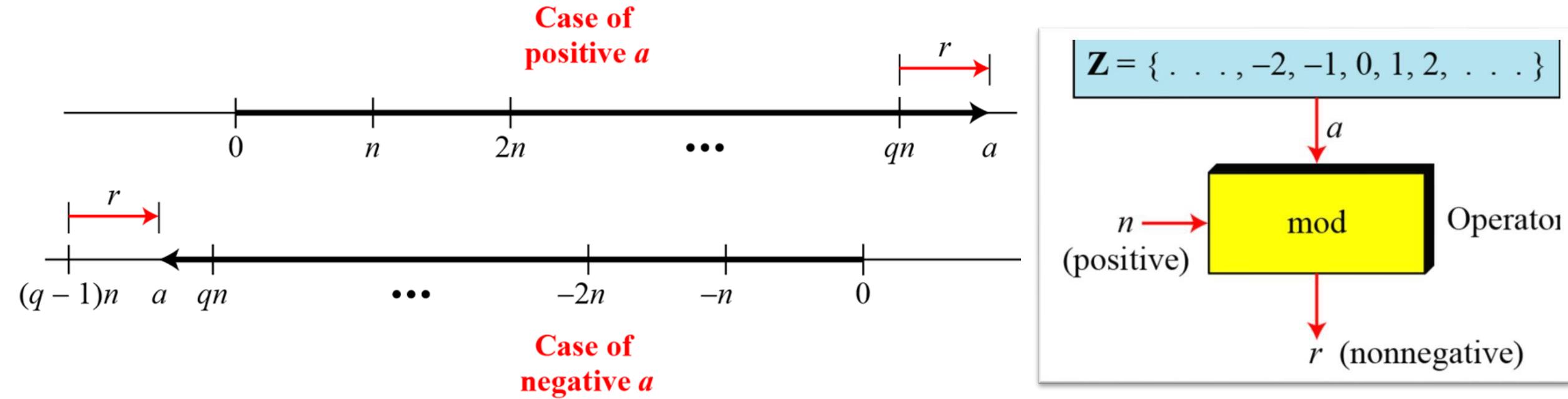
## 2. Known-plaintext attack



3. Chosen-plaintext attack

# Cơ sở toán học của Lý thuyết mật mã

## Modulo Operator



**First Property:**  $(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$

**Second Property:**  $(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$

**Third Property:**  $(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

## Modulo Operator

Example



Find the result of the following operations:

- a.  $27 \bmod 5$
- c.  $-18 \bmod 14$

- b.  $36 \bmod 12$
- d.  $-7 \bmod 10$

# Cơ sở toán học của Lý thuyết mật mã

**Modulo Operator** The modulo operator is shown as `mod`. The second input ( $n$ ) is called the modulus. The output  $r$  is called the residue.

- Let  $a, b, c, n$  be integers with  $n \neq 0$

(1)  $a \equiv 0 \pmod{n}$  iff  $n | a$

(2)  $a \equiv a \pmod{n}$

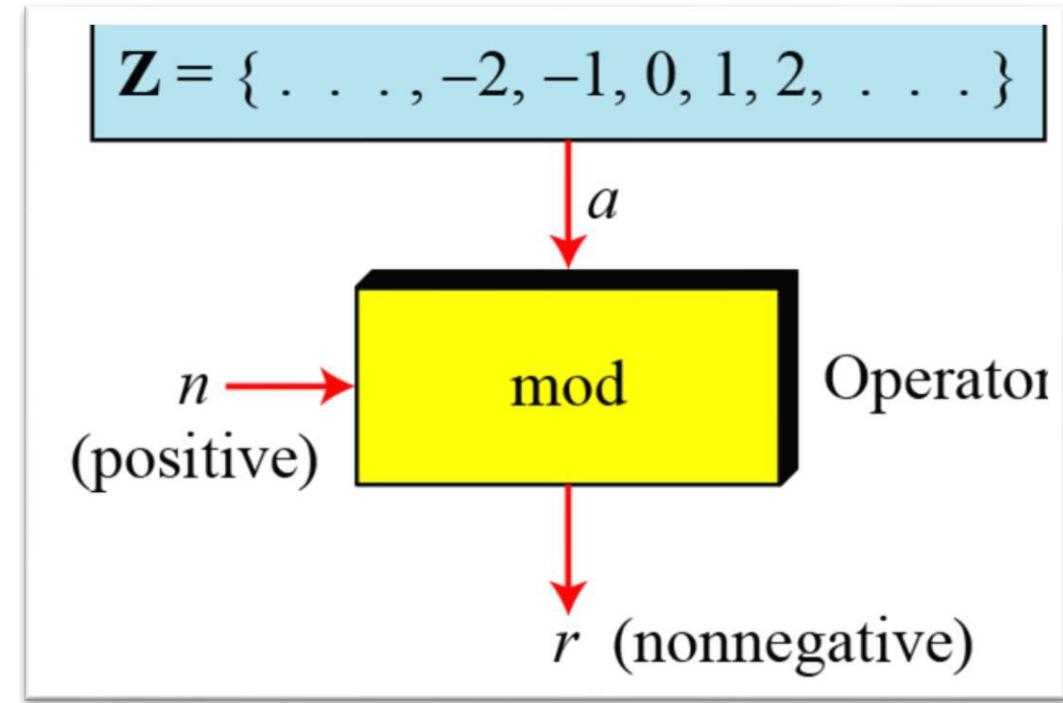
(3)  $a \equiv b \pmod{n}$  iff  $b \equiv a \pmod{n}$

(4)  $a \equiv b$  and  $b \equiv c \pmod{n} \rightarrow a \equiv c \pmod{n}$

(5)  $a \equiv b$  and  $c \equiv d \pmod{n} \rightarrow a+c \equiv b+d,$

$a-c \equiv b-d, ac \equiv bd \pmod{n}$

(6)  $ab \equiv ac \pmod{n}$  with  $n \neq 0$ , and  $\gcd(a, n) = 1$ , then  $b \equiv c \pmod{n}$



# Cơ sở toán học của Lý thuyết mật mã

## Modulo Operator

Example



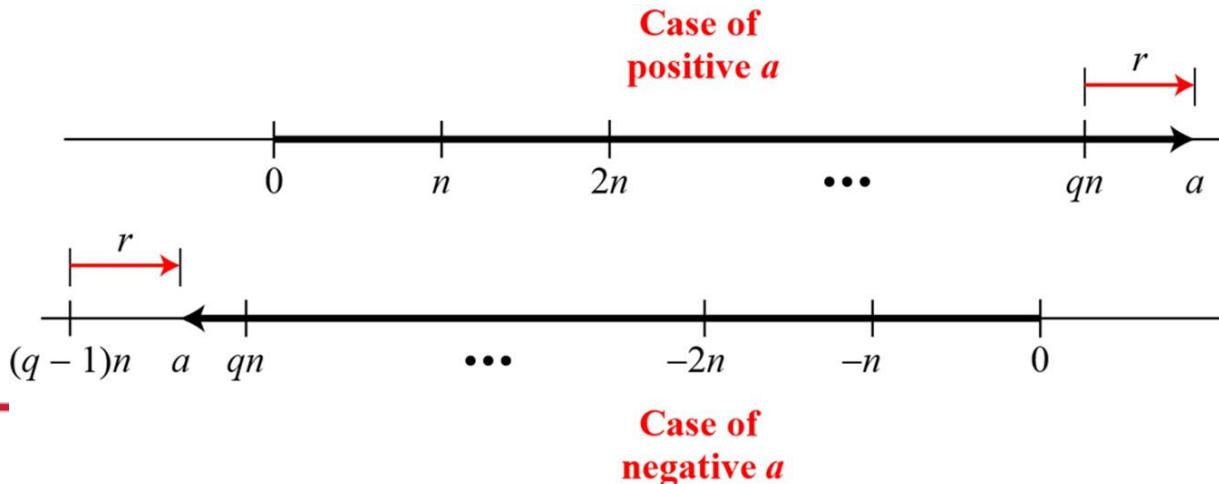
Find the result of the following operations:

- a.  $27 \bmod 5$   
c.  $-18 \bmod 14$

- b.  $36 \bmod 12$   
d.  $-7 \bmod 10$

Solution

- a. Dividing 27 by 5 results in  $r = 2$
- b. Dividing 36 by 12 results in  $r = 0$ .
- c. Dividing  $-18$  by 14 results in  $r = -4$ . After adding the modulus  $r = 10$
- d. Dividing  $-7$  by 10 results in  $r = -7$ . After adding the modulus to  $-7$ ,  $r = 3$ .



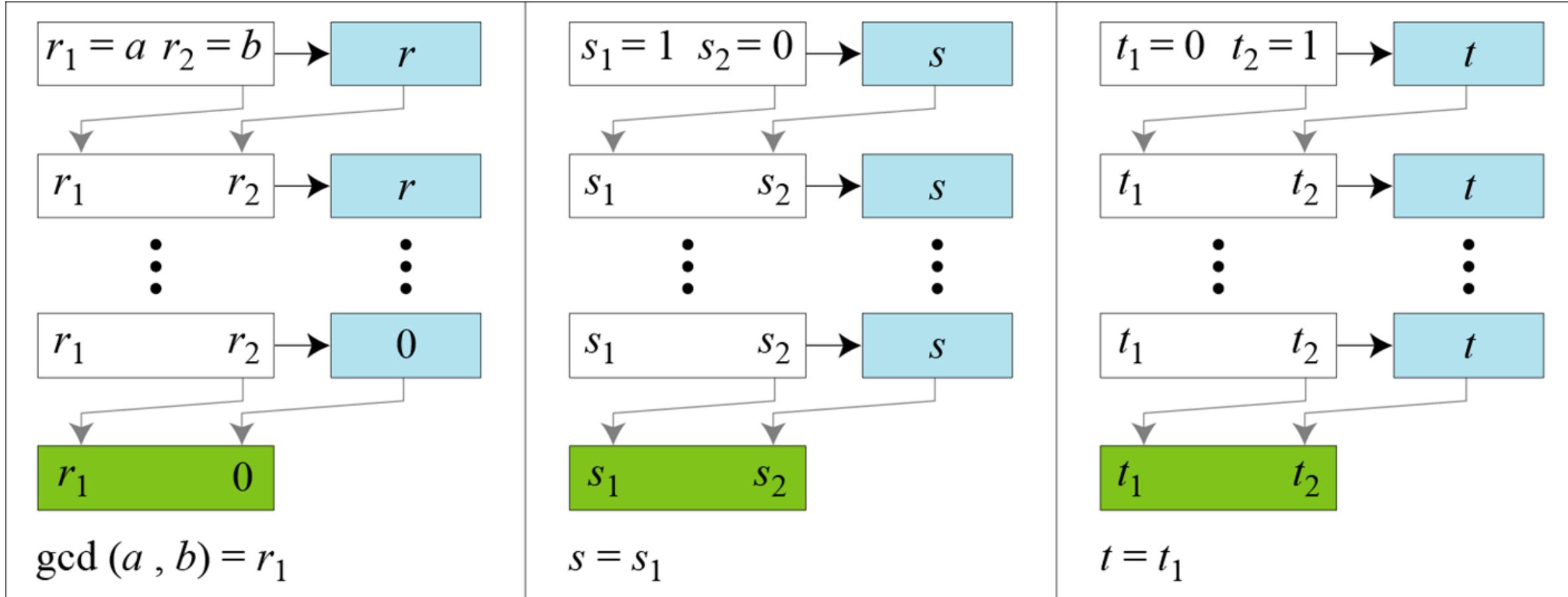
## Extended Euclidean Algorithm

- Given two integers  $a$  and  $b$ , we often need to find other two integers,  $s$  and  $t$ , such that

$$s \times a + t \times b = \gcd(a, b)$$

- The extended Euclidean algorithm can calculate the  $\gcd(a, b)$  and at the same time calculate the value of  $s$  and  $t$ .

## Extended Euclidean Algorithm



## Extended Euclidean Algorithm

Example

Given  $a = 161$  and  $b = 28$ , find  $\gcd(a, b)$  and the values of  $s$  and  $t$ .

Solution

We get  $\gcd(161, 28) = 7$ ,  $s = -1$  and  $t = 6$ .

$s_1$	$s_2$
1	0
$t_1$	$t_2$
0	1

$q$	$r_1$	$r_2$	$r$	$s_1$	$s_2$	$s$	$t_1$	$t_2$	$t$
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

## Set of Residues

The modulo operation creates a set, which in modular arithmetic is referred to as **the set of least residues modulo n**, or  $Z_n$ .

$$Z_n = \{ 0, 1, 2, 3, \dots, (n - 1) \}$$

$$Z_2 = \{ 0, 1 \}$$

$$Z_6 = \{ 0, 1, 2, 3, 4, 5 \}$$

$$Z_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$

Some  $Z_n$  sets

# Cơ sở toán học của Lý thuyết mật mã

**Congruence** To show that two integers are congruent, we use the congruence operator (  $\equiv$  )

Example

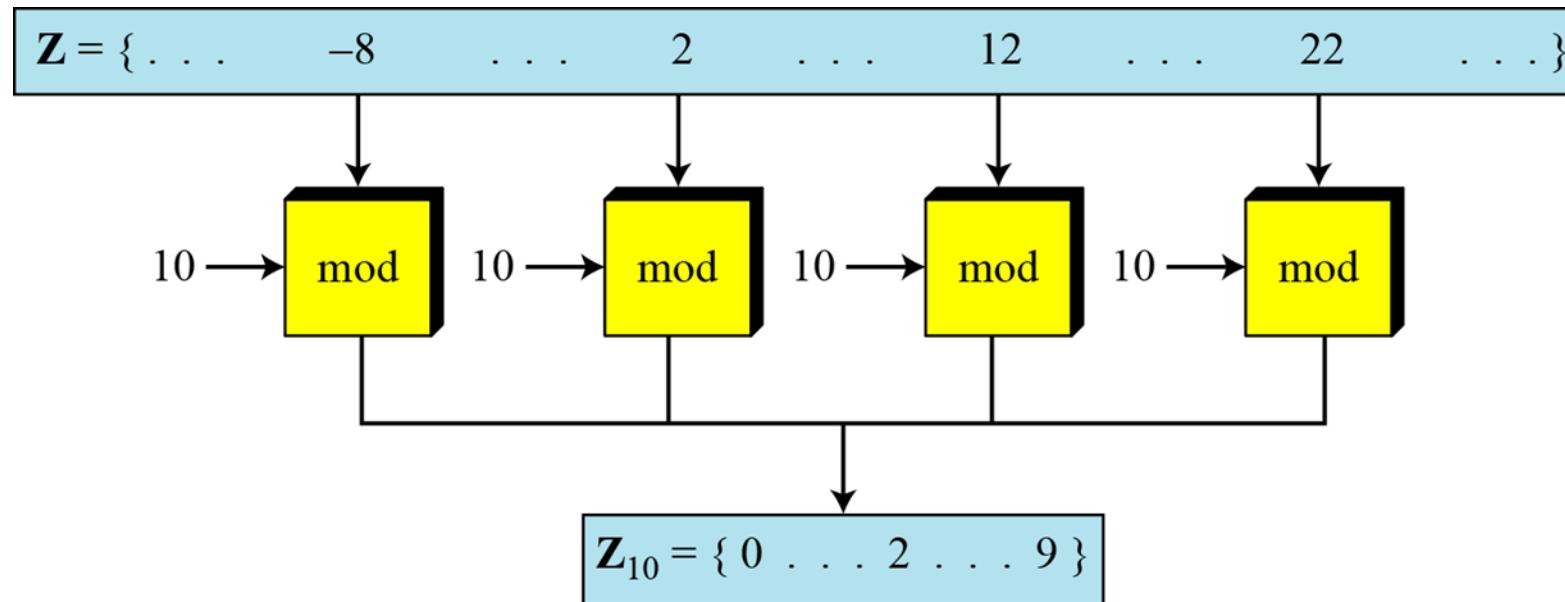


$$2 \equiv 12 \pmod{10}$$

$$3 \equiv 8 \pmod{5}$$

$$13 \equiv 23 \pmod{10}$$

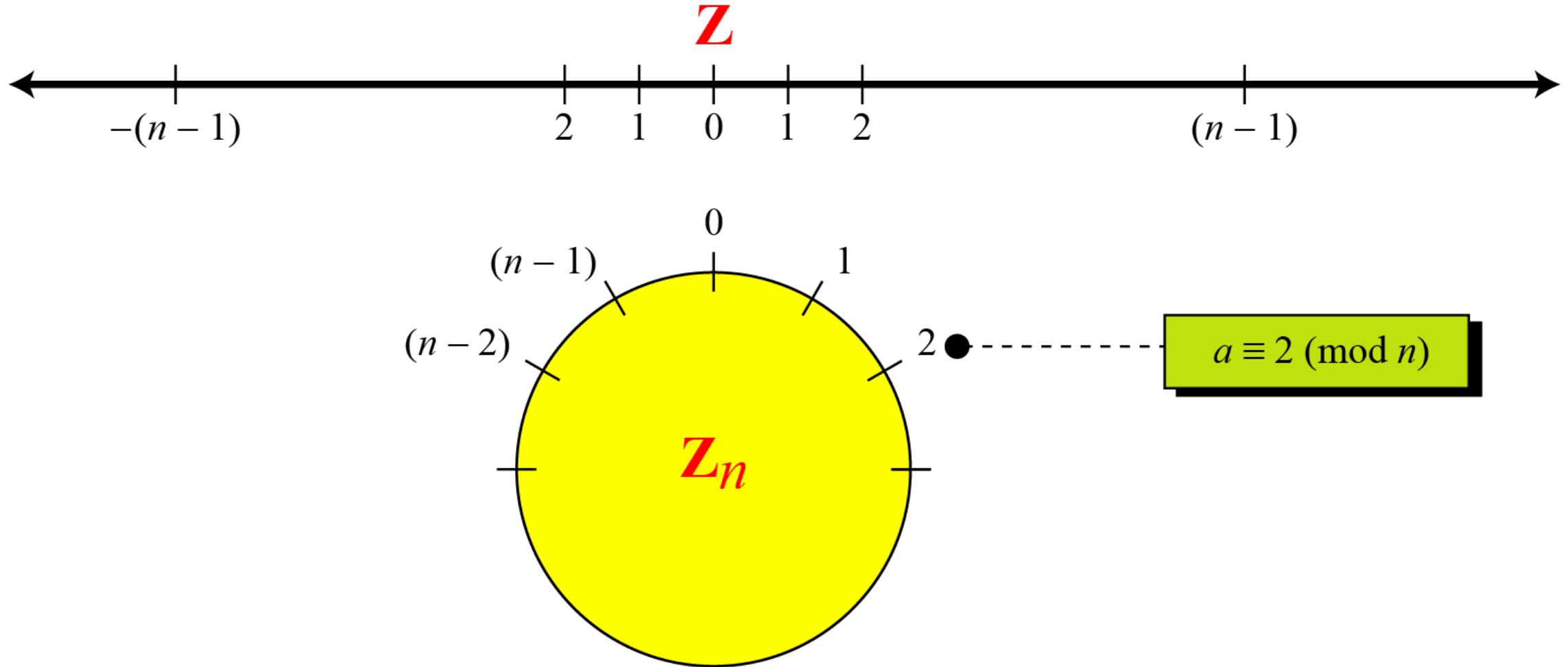
$$8 \equiv 13 \pmod{5}$$



$$-8 \equiv 2 \equiv 12 \equiv 22 \pmod{10}$$

Congruence Relationship

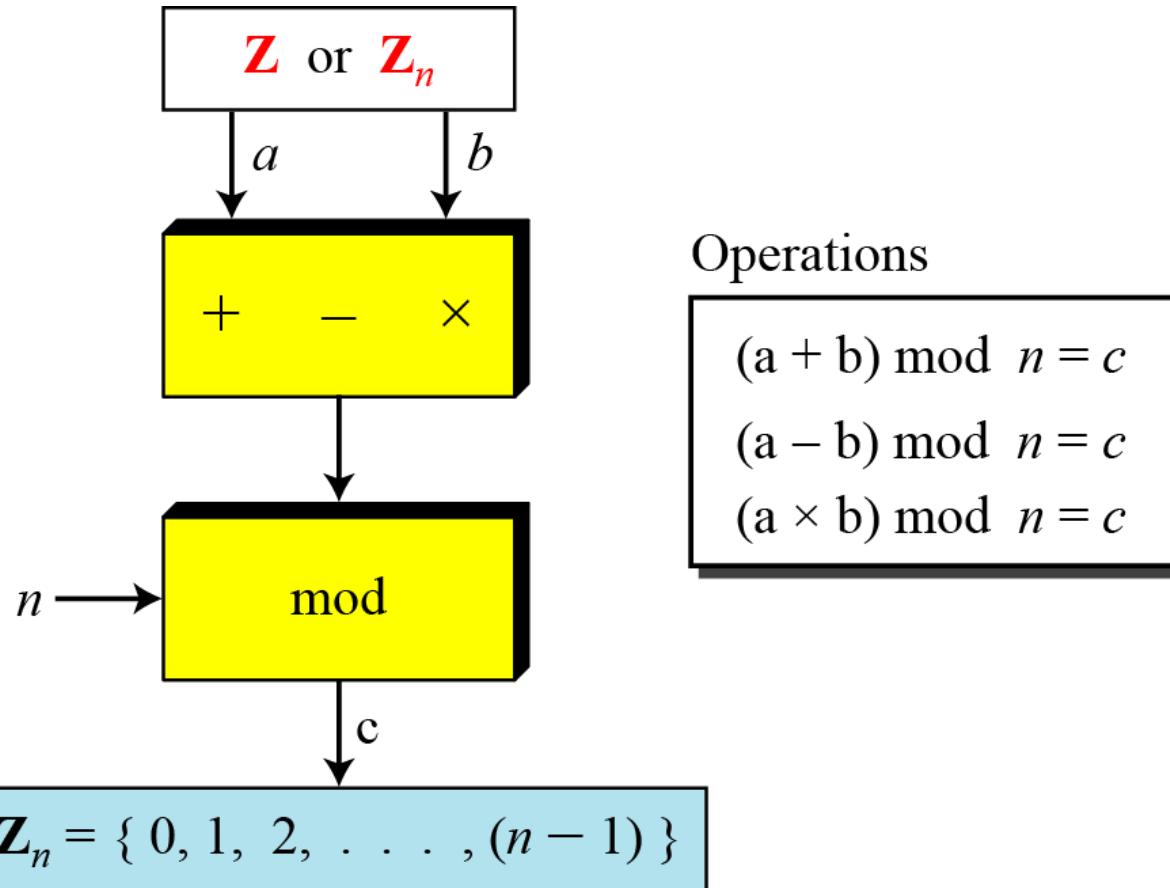
## Comparison of $Z$ and $Z_n$ using graphs



# Cơ sở toán học của Lý thuyết mật mã

## Operation in $Z_n$

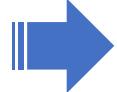
The three binary operations that we discussed for the set  $Z$  can also be defined for the set  $Z_n$ . The result may need to be mapped to  $Z_n$  using the mod operator.



# Cơ sở toán học của Lý thuyết mật mã

## Operation in $Z_n$

Example



Perform the following operations (the inputs come from  $Z_n$ ):

- Add 7 to 14 in  $Z_{15}$ .
- Subtract 11 from 7 in  $Z_{13}$ .
- Multiply 11 by 7 in  $Z_{20}$ .

Solution

$$(14 + 7) \bmod 15$$

$$(7 - 11) \bmod 13$$

$$(7 \times 11) \bmod 20$$

$$(14 + 7) \bmod 15 \rightarrow (21) \bmod 15 = 6$$

$$(7 - 11) \bmod 13 \rightarrow (-4) \bmod 13 = 9$$

$$(7 \times 11) \bmod 20 \rightarrow (77) \bmod 20 = 17$$

# Cơ sở toán học của Lý thuyết mật mã

## Operation in $Z_n$

Example



Perform the following operations

- a. Add 17 to 27 in  $Z_{14}$ .
- b. Subtract 43 from 12 in  $Z_{13}$ .
- c. Multiply 123 by  $-10$  in  $Z_{19}$ .

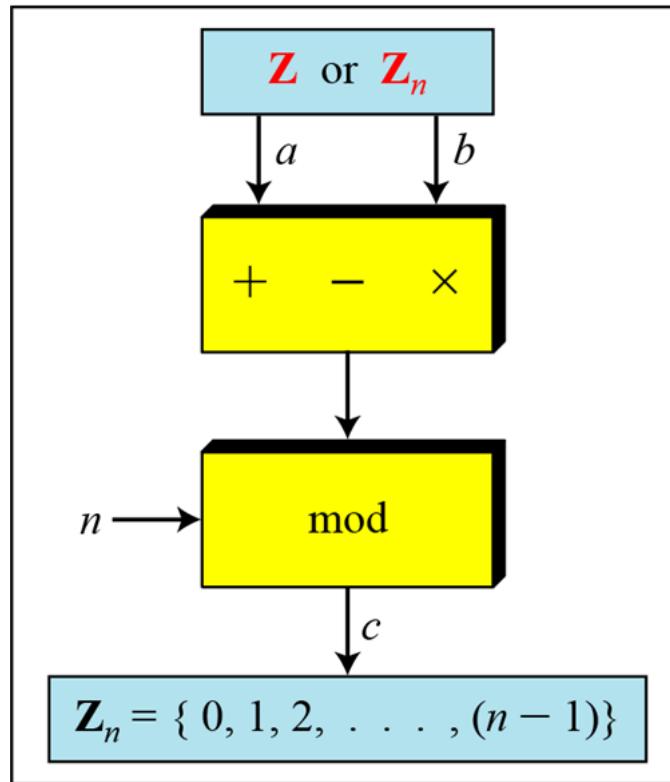
## Operation in $\mathbb{Z}_n$

### Additive Inverse

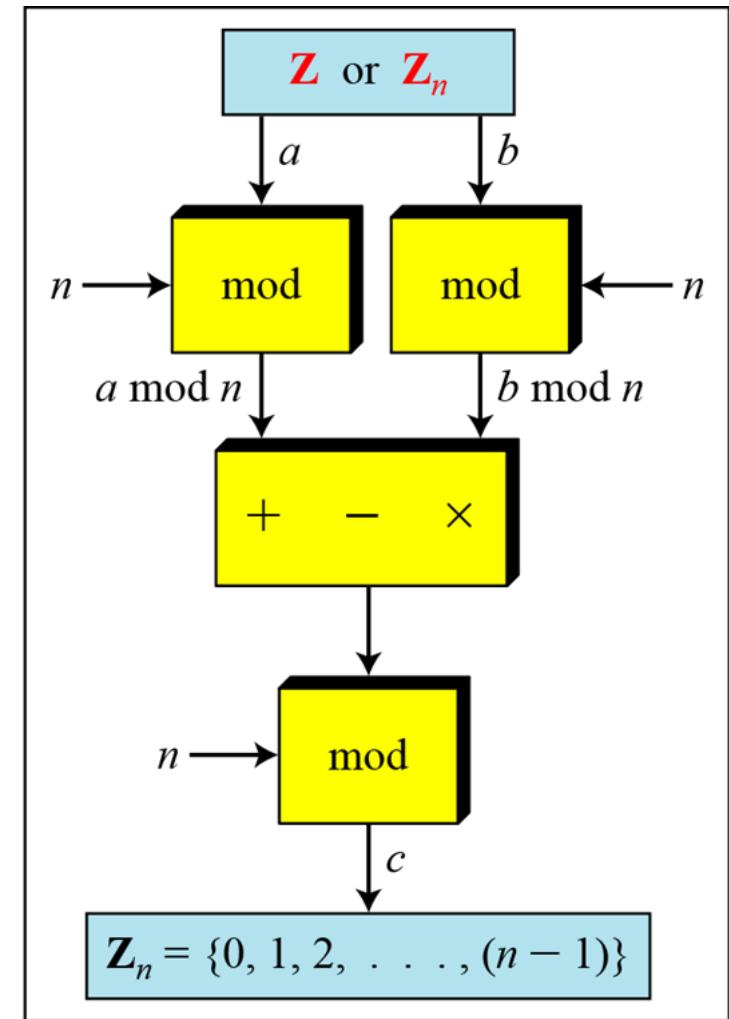
$$a + b \equiv 0 \pmod{n}$$

### Multiplicative Inverse

$$a \times b \equiv 1 \pmod{n}$$



a. Original process



b. Applying properties

## Operation in $Z_n$

### Additive Inverse

$$a + b \equiv 0 \pmod{n}$$

$$10^n \pmod{x} = (10 \pmod{x})^n$$

### Multiplicative Inverse

$$a \times b \equiv 1 \pmod{n}$$

$$(a + b) \pmod{n} = [(a \pmod{n}) + (b \pmod{n})] \pmod{n}$$

$$(a - b) \pmod{n} = [(a \pmod{n}) - (b \pmod{n})] \pmod{n}$$

$$(a \times b) \pmod{n} = [(a \pmod{n}) \times (b \pmod{n})] \pmod{n}$$

## Extended Euclidean Algorithm

Example

Find the multiplicative inverse of 11 in  $\mathbb{Z}_{26}$ .

$t_1$	$t_2$
0	1

Solution

$q$	$r_1$	$r_2$	$r$	$t_1$	$t_2$	$t$
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26
	1	0		-7	26	

The gcd (26, 11) is 1; the inverse of 11 is -7 or 19.

## Z<sub>n</sub> and Z<sub>n</sub>\* sets

Use Z<sub>n</sub> when additive inverses are needed

$$\mathbf{Z}_6 = \{0, 1, 2, 3, 4, 5\}$$

Use Z<sub>n</sub>\* when multiplicative inverses are needed

$$\mathbf{Z}_6^* = \{1, 5\}$$

$$\mathbf{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$$

$$\mathbf{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$$

$$\mathbf{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$\mathbf{Z}_{10}^* = \{1, 3, 7, 9\}$$

# Cơ sở toán học của Lý thuyết mật mã

## Successive Division by 2

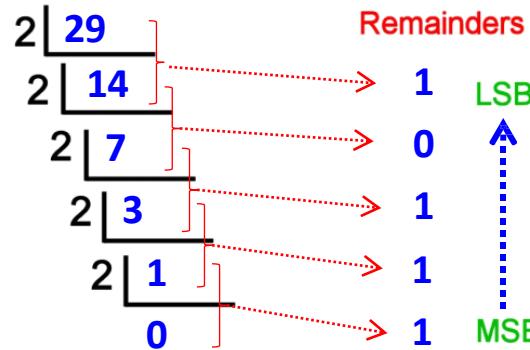
$2 \overline{)29}$	Remainders
$2 \overline{)14}$	1    LSB
$2 \overline{)7}$	0
$2 \overline{)3}$	1
$2 \overline{)1}$	1
0	1    MSB

Read the remainders  
from the bottom up

29 decimal = 11101 binary

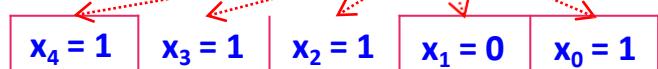
# Cơ sở toán học của Lý thuyết mật mã

Successive Division by 2

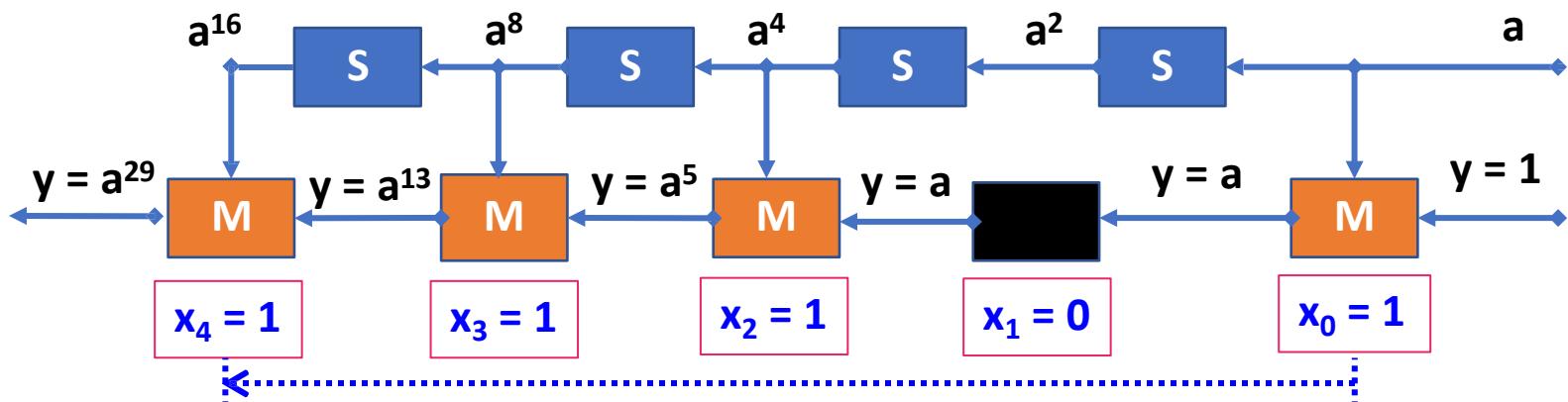


Read the remainders from the bottom up

29 decimal  $\Leftrightarrow$  11101 binary



## Square-and-Multiply method

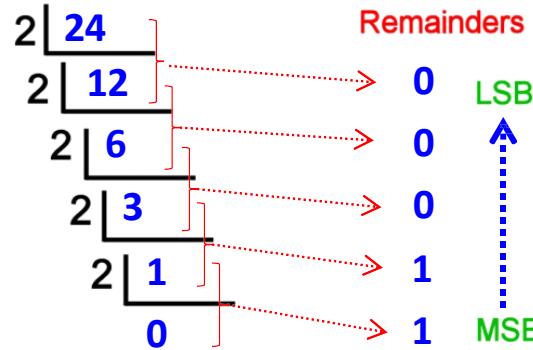


## Ex. Calculation of $17^{29} \bmod 21$

i	$x_i$	Multiplication (Initialization: $y = 1$ )	Squaring (Initialization: $a = 17$ )
0	1	$y = 1 * 17 \bmod 21 = 17$	$\rightarrow$ $a = 17^2 \bmod 21 = 16$
1	0		$\rightarrow$ $a = 16^2 \bmod 21 = 4$
2	1	$y = 17 * 4 \bmod 21 = 5$	$\rightarrow$ $a = 4^2 \bmod 21 = 16$
3	1	$y = 5 * 16 \bmod 21 = 17$	$\rightarrow$ $a = 16^2 \bmod 21 = 4$
4	1	$y = 17 * 4 \bmod 21 = 5$	$\rightarrow$

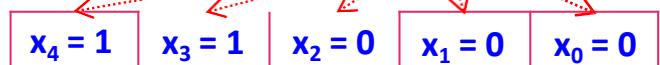
# Cơ sở toán học của Lý thuyết mật mã

Successive Division by 2

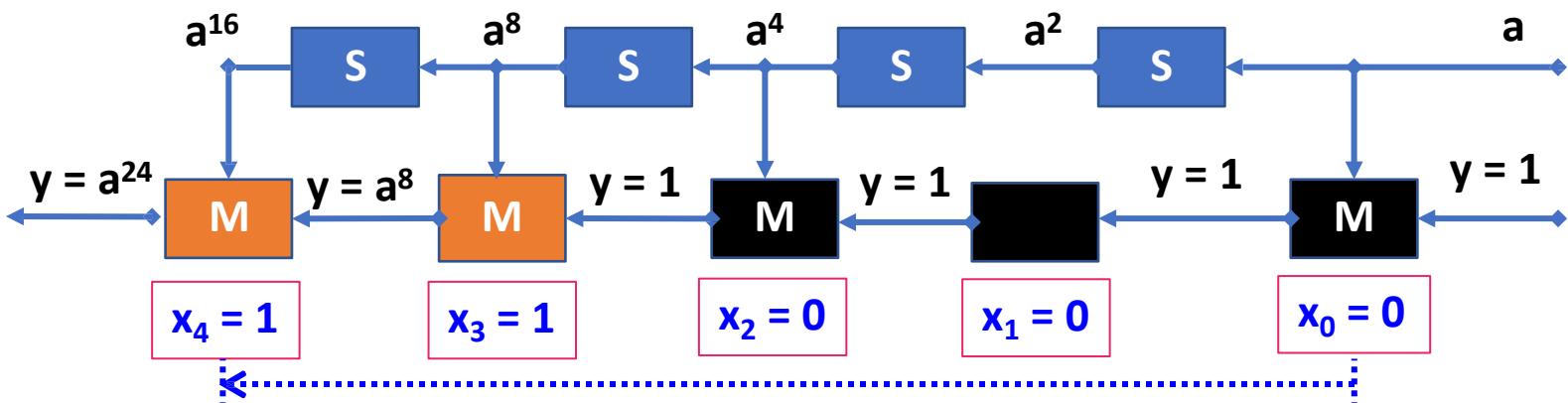


Read the remainders  
from the bottom up

24 decimal  $\Leftrightarrow$  11000 binary



## Square-and-Multiply method



## Ex. Calculation of $21^{24} \bmod 8$

$i$	$x_i$	Multiplication (Initialization: $y = 1$ )	Squaring (Initialization: $a = 21$ )
0	0		$\rightarrow$ $a = 21^2 \bmod 8 = 1$
1	0		$\rightarrow$ $a = 1^2 \bmod 8 = 1$
2	0		$\rightarrow$ $a = 1^2 \bmod 8 = 1$
3	1	$y = 1 * 1 \bmod 8 = 1$	$\rightarrow$ $a = 1^2 \bmod 8 = 1$
4	1	$y = 1 * 1 \bmod 8 = 1$	$\rightarrow$

*Euler's phi-function,  $\phi(n)$ , which is sometimes called the Euler's totient function plays a very important role in cryptography.*

1.  $\phi(1) = 0$ .
2.  $\phi(p) = p - 1$  if  $p$  is a prime.
3.  $\phi(m \times n) = \phi(m) \times \phi(n)$  if  $m$  and  $n$  are relatively prime.
4.  $\phi(p^e) = p^e - p^{e-1}$  if  $p$  is a prime.

A Galois field,  $\text{GF}(p^n)$ , is a finite field with  $p^n$  elements.

A **polynomial** of **degree  $n - 1$**  is an expression of the form

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x^1 + a_0x^0$$

where  $x^i$  is called the **i<sup>th</sup> term** and  $a_i$  is called **coefficient** of the **i<sup>th</sup> term**.

# Cơ sở toán học của Lý thuyết mật mã

Represent the 8-bit word (10011001) using a polynomials

$n$ -bit word

1	0	0	1	1	0	0	1
---	---	---	---	---	---	---	---



Polynomial

$1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0$
---

First simplification

$1x^7 + 1x^4 + 1x^3 + 1x^0$
-----------------------------

Second simplification

$x^7 + x^4 + x^3 + 1$
-----------------------

A Galois field,  $\text{GF}(p^n)$ , is a finite field with  $p^n$  elements.

For the sets of **polynomials** in  $\text{GF}(2^n)$ , a **group of polynomials of degree  $n$**  is defined as the **modulus**. Such polynomials are referred to as **irreducible polynomials**.

Degree	Irreducible Polynomials
1	$(x + 1), (x)$
2	$(x^2 + x + 1)$
3	$(x^3 + x^2 + 1), (x^3 + x + 1)$
4	$(x^4 + x^3 + x^2 + x + 1), (x^4 + x^3 + 1), (x^4 + x + 1)$
5	$(x^5 + x^2 + 1), (x^5 + x^3 + x^2 + x + 1), (x^5 + x^4 + x^3 + x + 1),$ $(x^5 + x^4 + x^3 + x^2 + 1), (x^5 + x^4 + x^2 + x + 1)$

## A Galois field, $\text{GF}(p^n)$ , is a finite field with $p^n$ elements.

Find the result of  $(x^5 + x^2 + x) \otimes (x^7 + x^4 + x^3 + x^2 + x)$  in  $\text{GF}(2^8)$  with irreducible polynomial  $(x^8 + x^4 + x^3 + x + 1)$ . Note that we use the symbol  $\otimes$  to show the multiplication of two polynomials.

### Solution

$$P_1 \otimes P_2 = x^5(x^7 + x^4 + x^3 + x^2 + x) + x^2(x^7 + x^4 + x^3 + x^2 + x) + x(x^7 + x^4 + x^3 + x^2 + x)$$

$$P_1 \otimes P_2 = x^{12} + x^9 + x^8 + x^7 + x^6 + x^9 + x^6 + x^5 + x^4 + x^3 + x^8 + x^5 + x^4 + x^3 + x^2$$

$$P_1 \otimes P_2 = (x^{12} + x^7 + x^2) \bmod (x^8 + x^4 + x^3 + x + 1) = x^5 + x^3 + x^2 + x + 1$$

**A Galois field,  $\text{GF}(p^n)$ , is a finite field with  $p^n$  elements.**

To find the final result, divide the polynomial of degree 12 by the polynomial of degree 8 (the modulus) and keep only the remainder.

$$\begin{array}{r} x^4 + 1 \\ \hline x^8 + x^4 + x^3 + x + 1 \quad \left[ \begin{array}{l} x^{12} + x^7 + x^2 \\ x^{12} + x^8 + x^7 + x^5 + x^4 \\ \hline x^8 + x^5 + x^4 + x^2 \\ x^8 + x^4 + x^3 + x + 1 \\ \hline x^5 + x^3 + x^2 + x + 1 \end{array} \right] \\ \hline \end{array}$$

Remainder   $x^5 + x^3 + x^2 + x + 1$

A Galois field,  $\text{GF}(p^n)$ , is a finite field with  $p^n$  elements.

In  $\text{GF}(2^4)$ , find the inverse of  $(x^2 + 1)$  modulo  $(x^4 + x + 1)$ .

$q$	$r_1$	$r_2$	$r$	$t_1$	$t_2$	$t$
$(x^2 + 1)$	$(x^4 + x + 1)$	$(x^2 + 1)$	$(x)$	$(0)$	$(1)$	$(x^2 + 1)$
$(x)$	$(x^2 + 1)$	$(x)$	$(1)$	$(1)$	$(x^2 + 1)$	$(x^3 + x + 1)$
$(x)$	$(x)$	$(1)$	$(0)$	$(x^2 + 1)$	$(x^3 + x + 1)$	$(0)$
	$(1)$	$(0)$		$(x^3 + x + 1)$	$(0)$	

## Solution

The answer is  $(x^3 + x + 1)$

A Galois field,  $\text{GF}(p^n)$ , is a finite field with  $p^n$  elements.

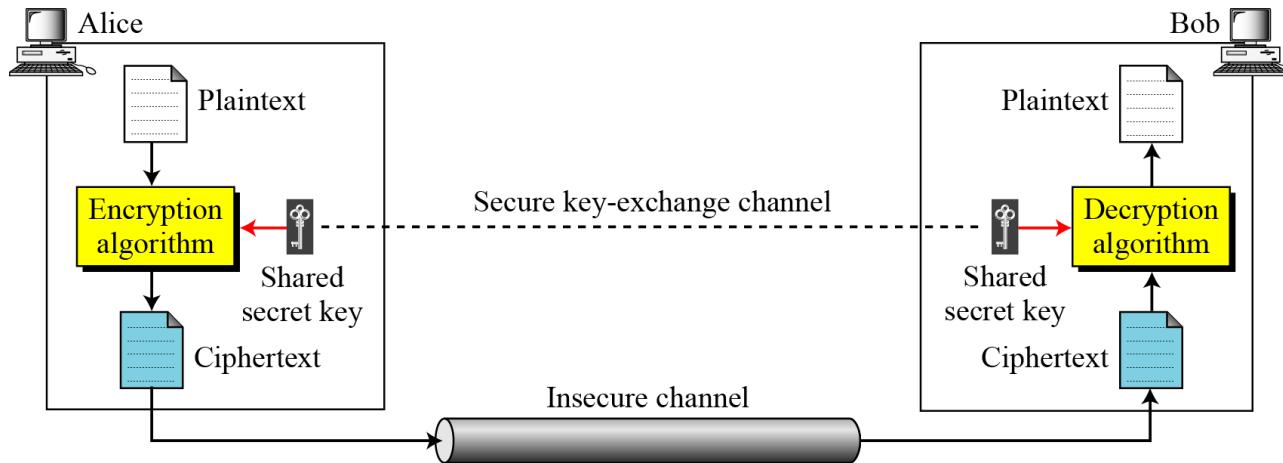
In  $\text{GF}(2^8)$ , find the inverse of  $(x^5)$  modulo  $(x^8 + x^4 + x^3 + x + 1)$  ?

$q$	$r_1$	$r_2$	$r$	$t_1$	$t_2$	$t$
$(x^3)$	$(x^8 + x^4 + x^3 + x + 1) \quad (x^5)$		$(x^4 + x^3 + x + 1)$	$(0)$	$(1)$	$(x^3)$
$(x + 1)$	$(x^5) \quad (x^4 + x^3 + x + 1)$		$(x^3 + x^2 + 1)$	$(1)$	$(x^3)$	$(x^4 + x^3 + 1)$
$(x)$	$(x^4 + x^3 + x + 1) \quad (x^3 + x^2 + 1)$		$(1)$	$(x^3)$	$(x^4 + x^3 + 1)$	$(x^5 + x^4 + x^3 + x)$
$(x^3 + x^2 + 1)$	$(x^3 + x^2 + 1)$	$(1)$	$(0)$	$(x^4 + x^3 + 1)$	$(x^5 + x^4 + x^3 + x)$	$(0)$
	$(1)$	$(0)$		$(x^5 + x^4 + x^3 + x)$	$(0)$	

## Solution

The answer is  $(x^5 + x^4 + x^3 + x)$

# Hệ mật cổ điển



*Plaintext and  
Ciphertext in  $Z_{26}$*

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

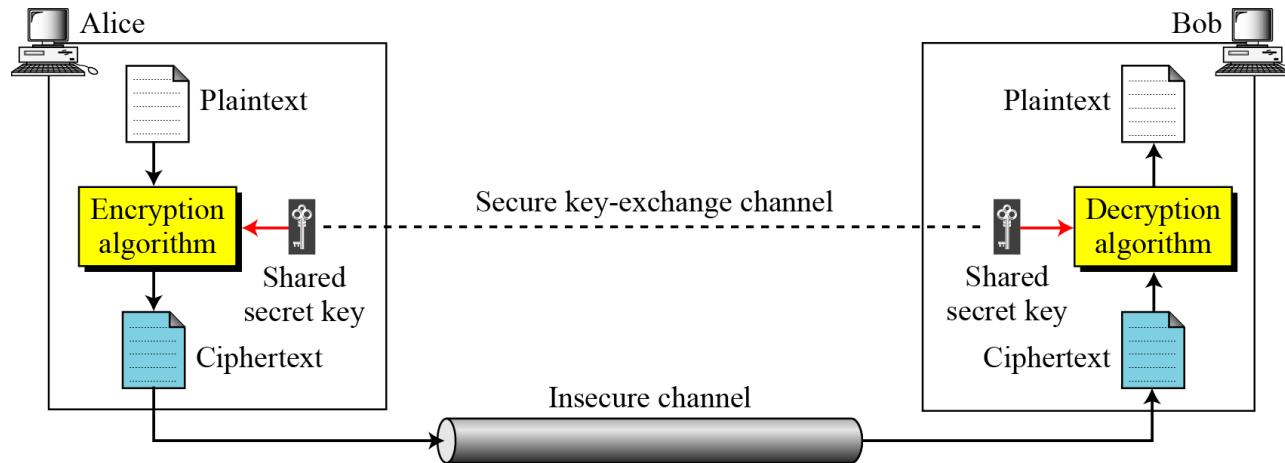
**A Substitution Cipher**  
replaces one symbol with another

*Cesar, Affine, Vigenere Cipher, Hill Cipher...*

**A Transposition Cipher**  
reorders symbols.

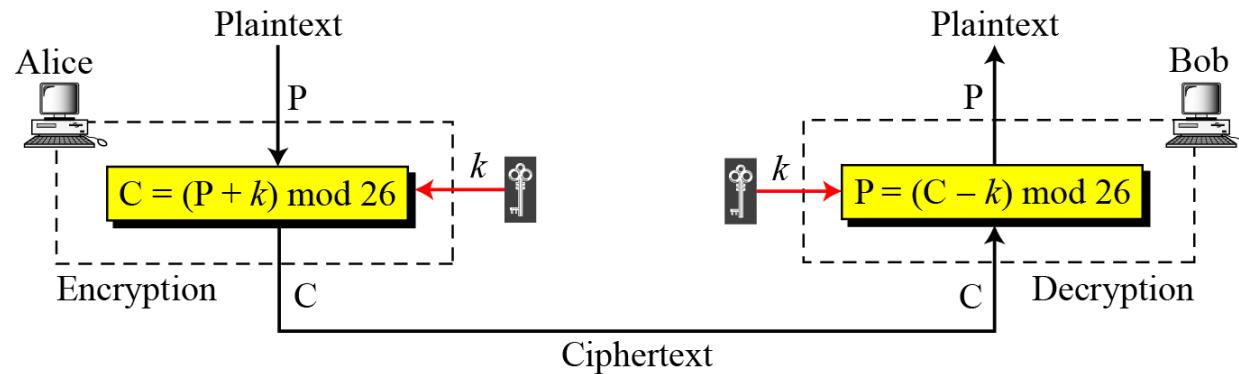
*Rail Fence Cipher, AutoKey ...*

# Hệ mật cổ điển



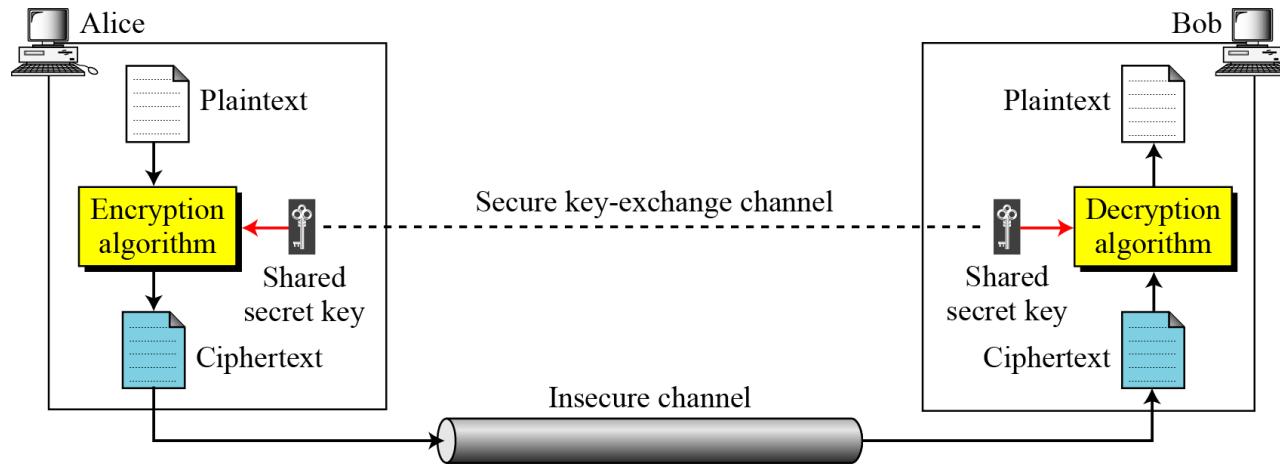
**Plaintext and  
Ciphertext in  $Z_{26}$**

Plaintext →	a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z
Ciphertext →	A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z
Value →	00   01   02   03   04   05   06   07   08   09   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25



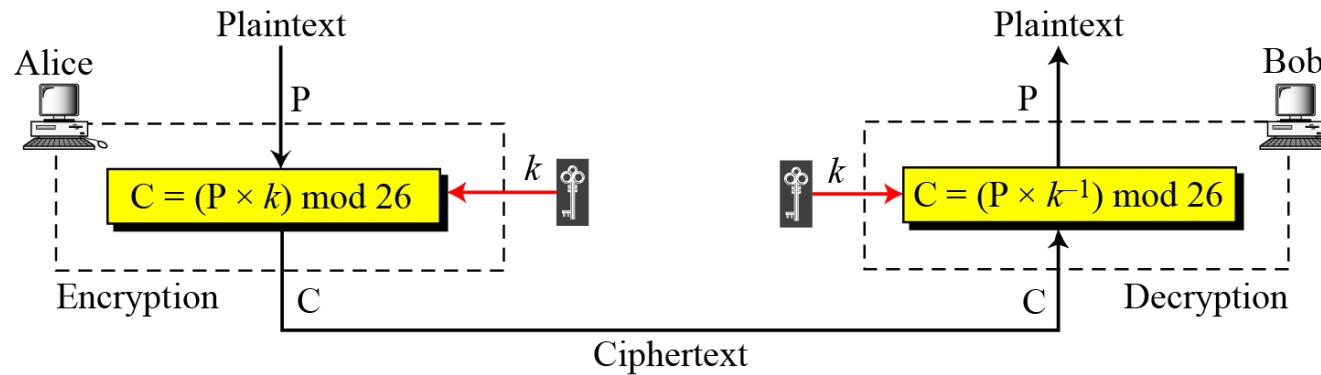
Additive ciphers are sometimes referred to as shift ciphers or Caesar cipher.

# Hệ mật cổ điển



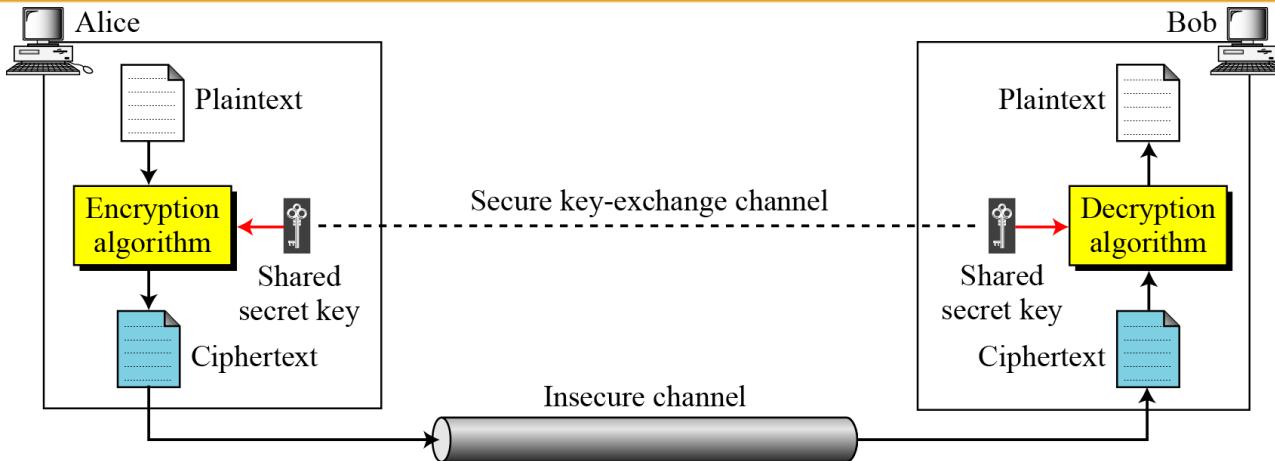
**Plaintext and  
Ciphertext in  $Z_{26}$**

Plaintext →	a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z
Ciphertext →	A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z
Value →	00   01   02   03   04   05   06   07   08   09   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25



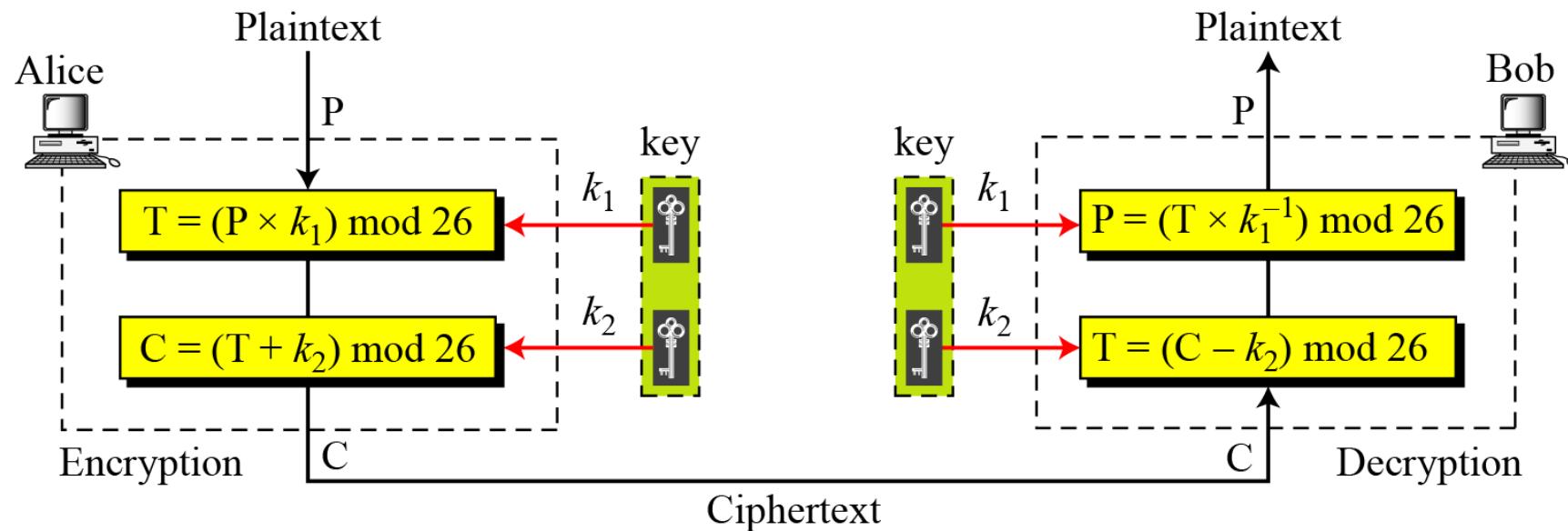
In a multiplicative cipher, the plaintext and ciphertext are integers in  $Z_{26}$ ; the key is an integer in  $Z_{26}^*$ .

# Hệ mật cổ điển



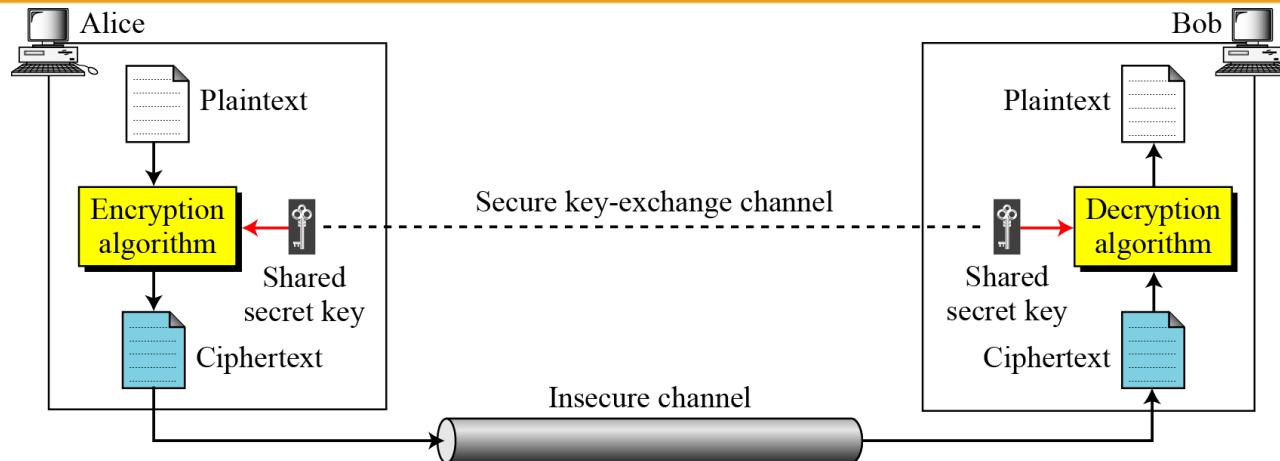
The affine cipher uses a pair of keys in which the first key is from  $Z_{26}^*$  and the second is from  $Z_{26}$ .

The size of the key domain is  $26 \times 12 = 312$ .



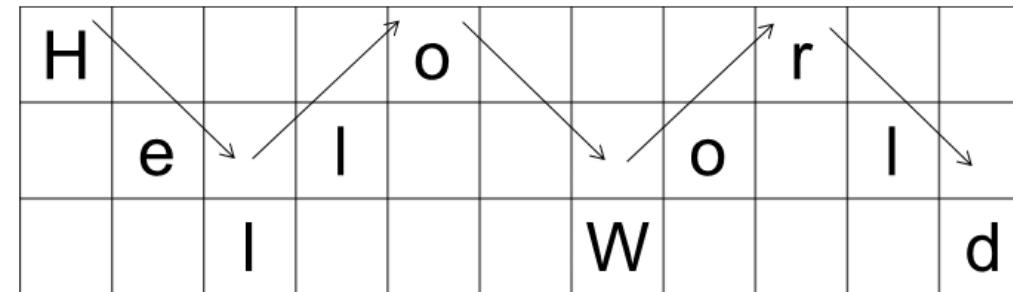
where  $k_1^{-1}$  is the multiplicative inverse of  $k_1$  and  $-k_2$  is the additive inverse of  $k_2$

# Hệ mật cổ điển



## Rail Fence Cipher

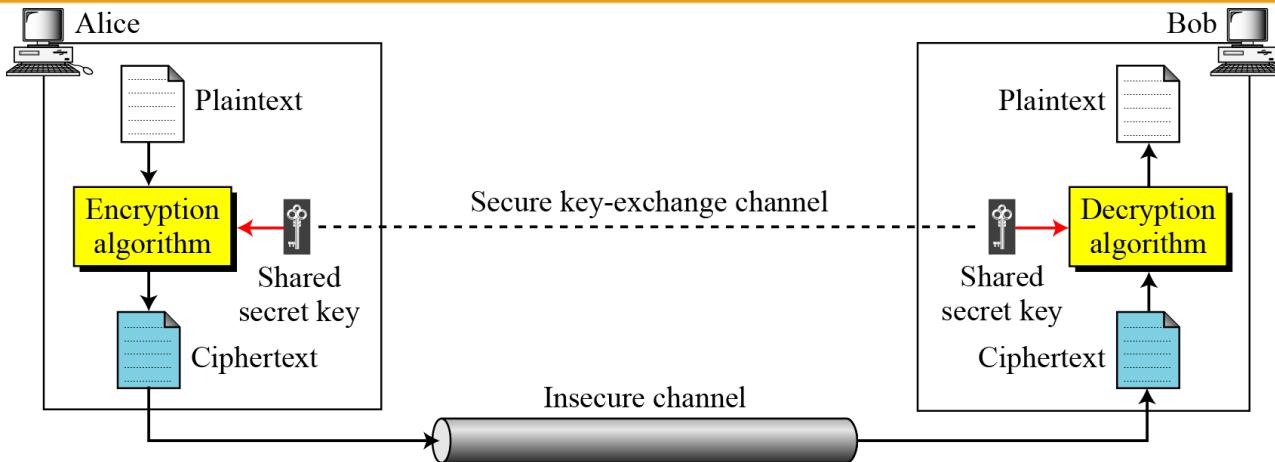
**Original Message:** Hello World



with 3 "rails"

**Encrypted Message:** Horel ollWd

# Hệ mật cổ điển



Plaintext = "HELLO"  
Autokey = N  
Ciphertext = "ULPWZ"

Given plain text is : H E L L O  
Key is : N H E L L

## AutoKey

Plain Text(P)	:	H	E	L	L	O
Corresponding Number:	7	4	11	11	14	
Key(K)	:	N	H	E	L	L
Corresponding Number:	13	7	4	11	11	

Applying the formula: 20 11 15 22 25

Corresponding Letters are : U L P W Z

Hence Ciphertext is: ULPWZ

Let's decrypt:

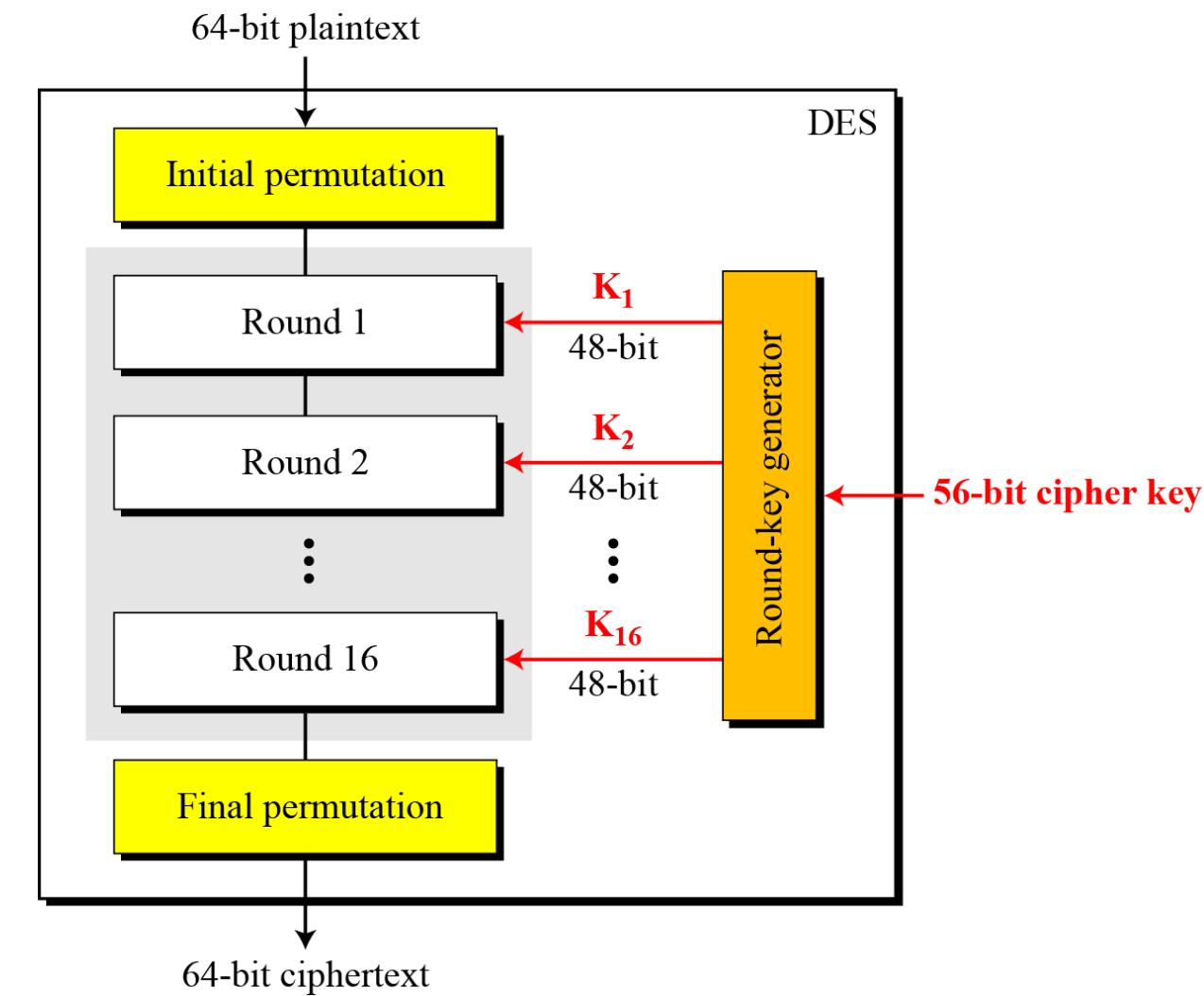
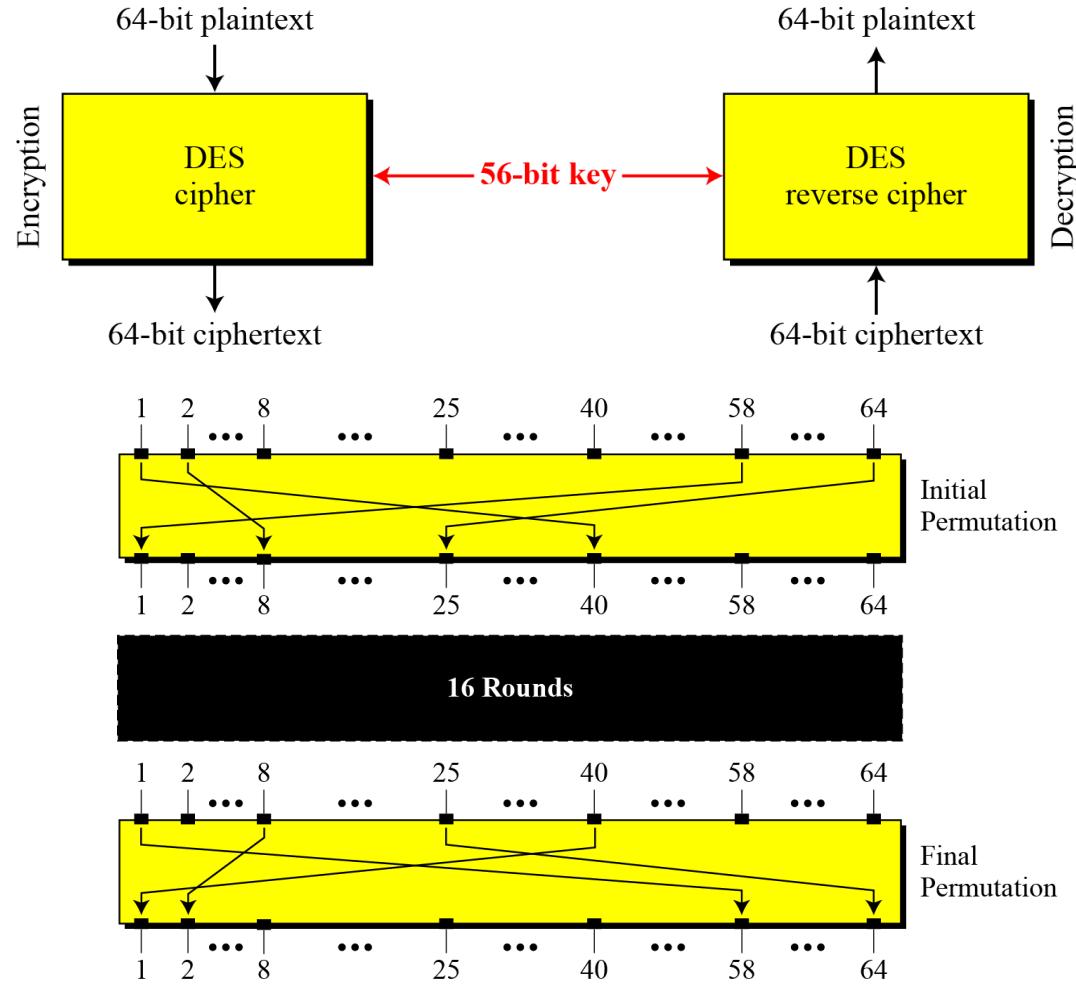
Cipher Text(C)	:	U	L	P	W	Z
Key(K)	:	N	H	E	L	L

Applying the formula: H E L L O

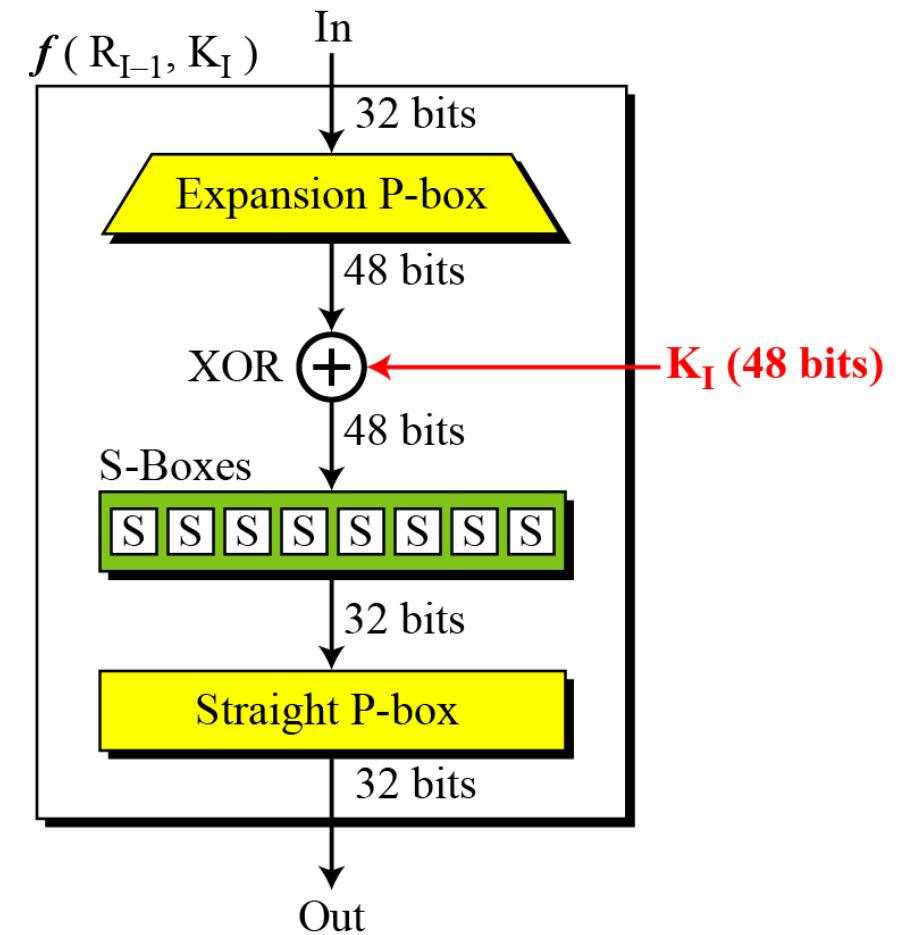
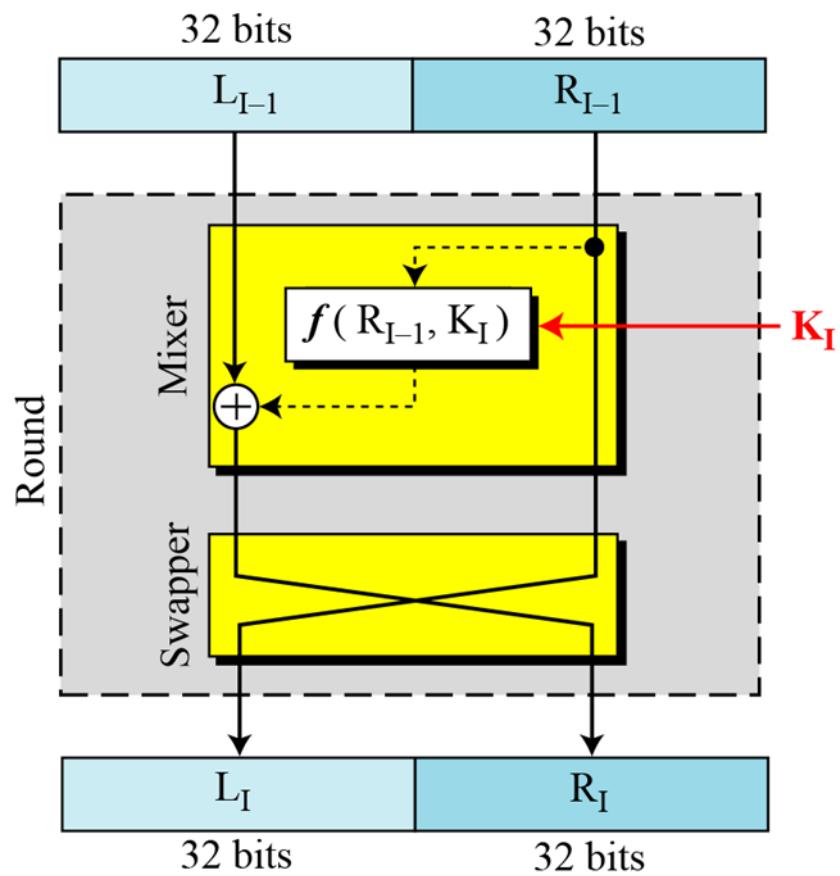
Hence Plaintext is: HELLO

# Hệ mật hiện đại

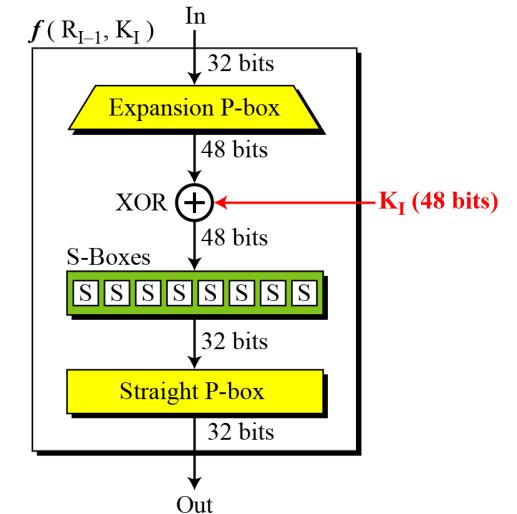
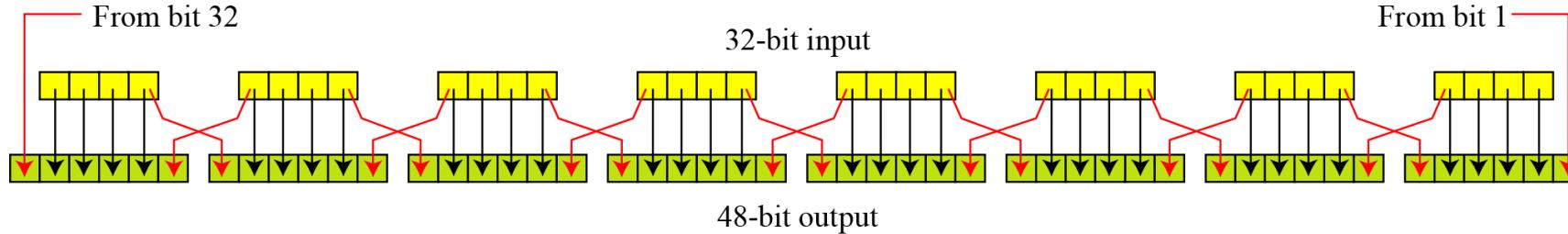
## Data Encryption Standard (DES)



## Data Encryption Standard (DES)



## Data Encryption Standard (DES)



32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

DES uses this table to define this P-box

# Hệ mật hiện đại

## Data Encryption Standard (DES)

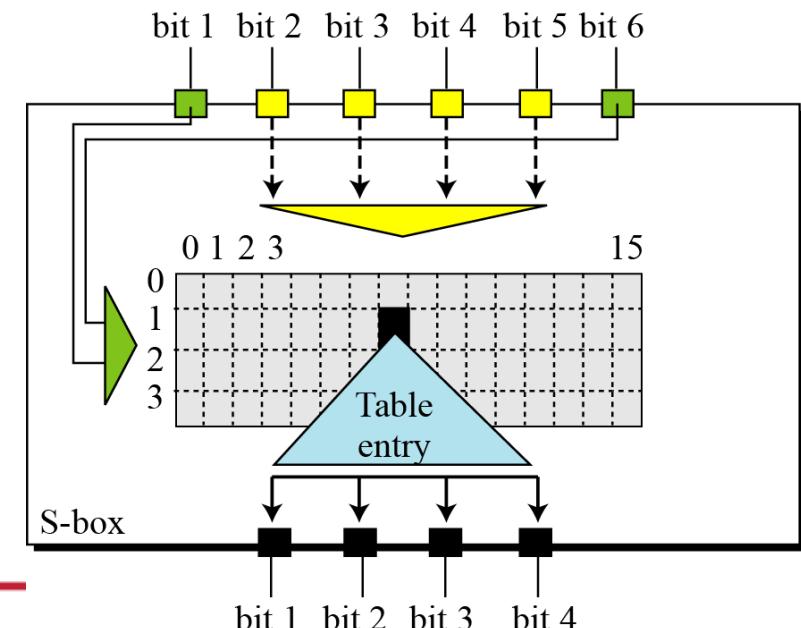
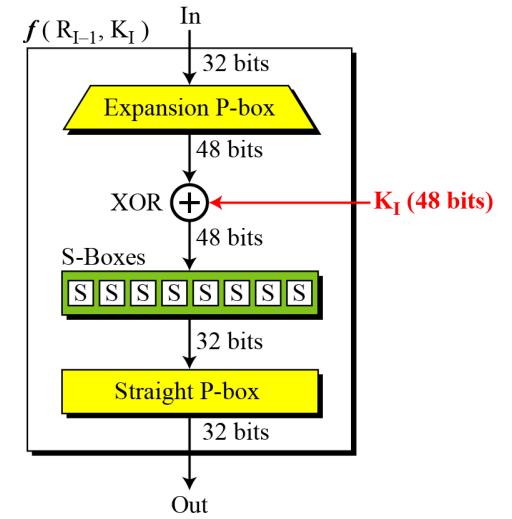
*S-box 1*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

*S-box 2*

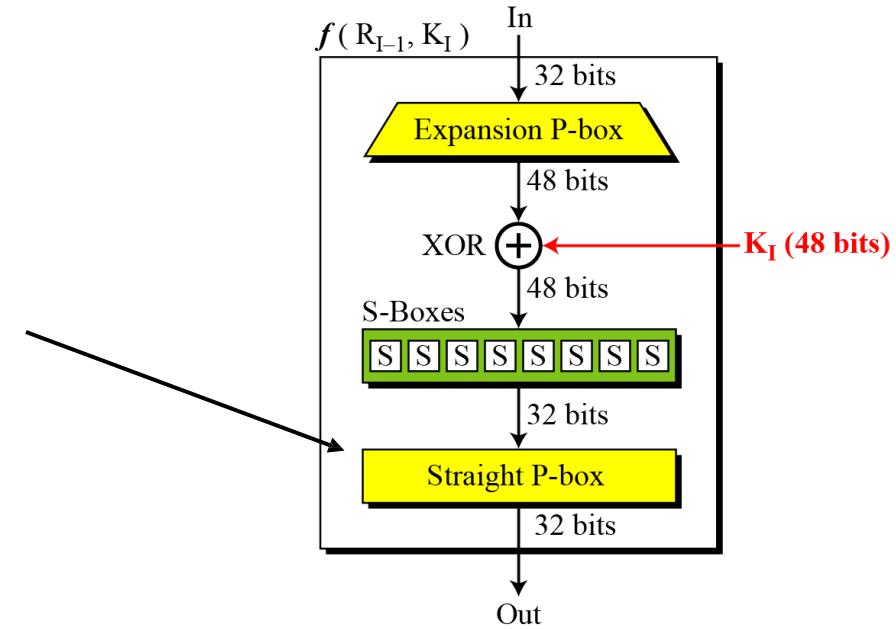
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

*S-box 3...*



## Data Encryption Standard (DES)

### Straight Permutation



16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

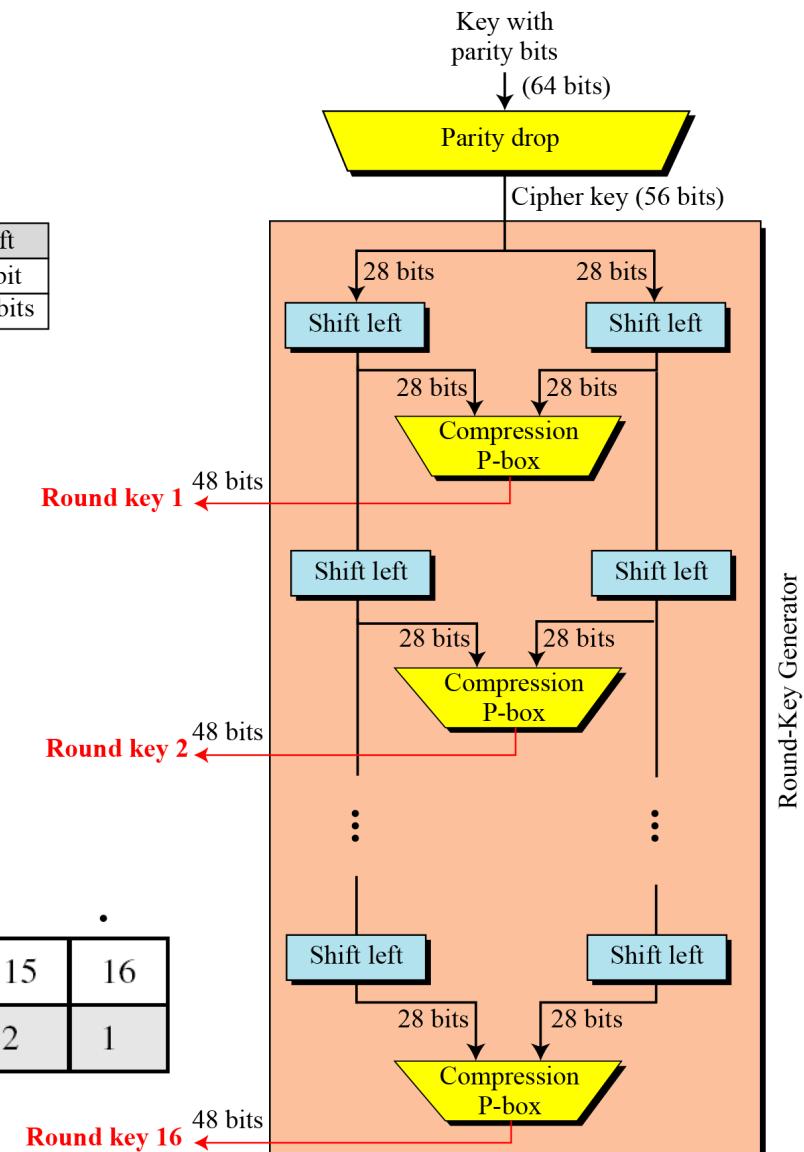
## Data Encryption Standard (DES)

*The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.*

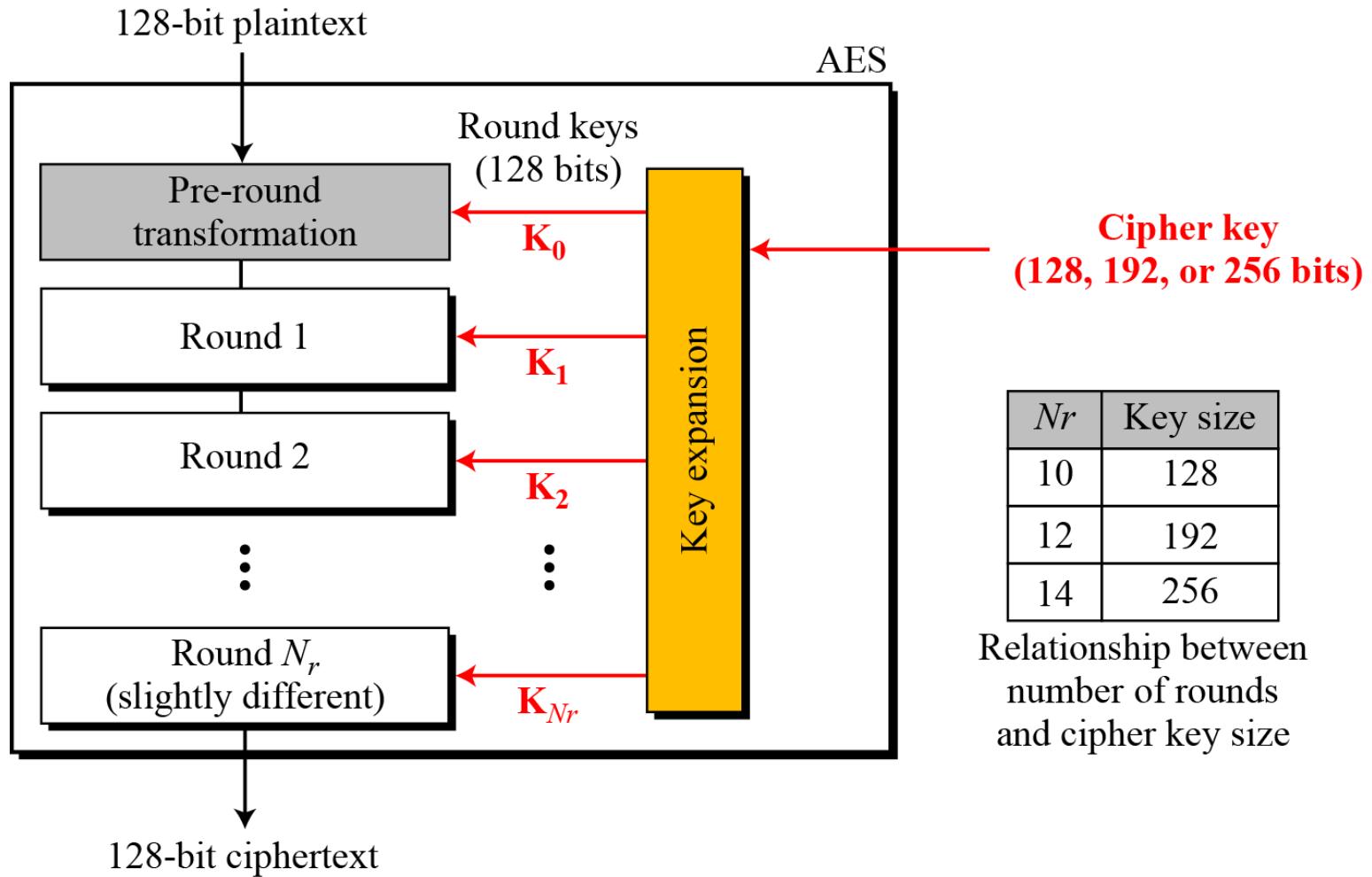
57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

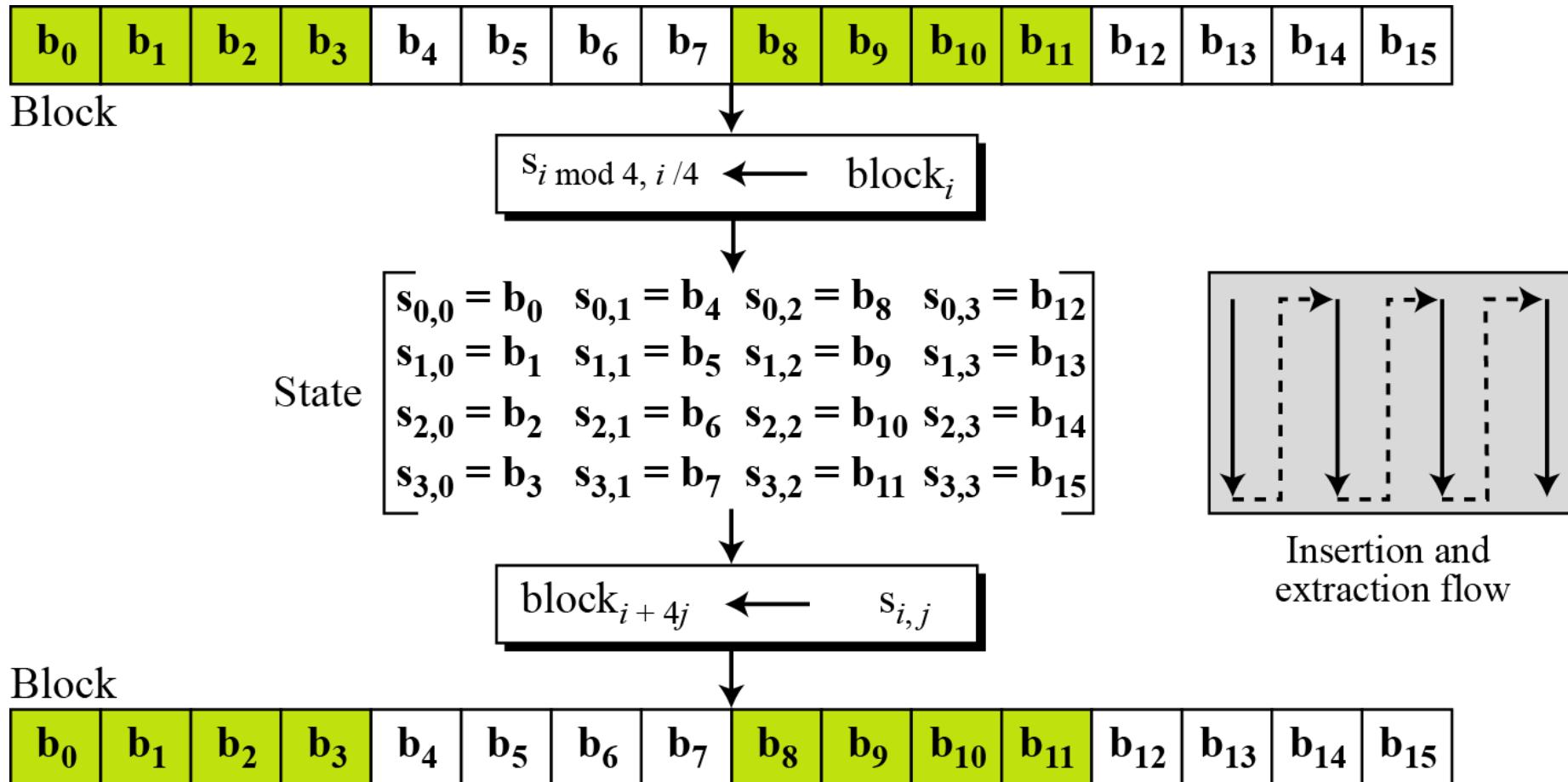
Shifting	
Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



## Advanced Encryption Standard (AES)

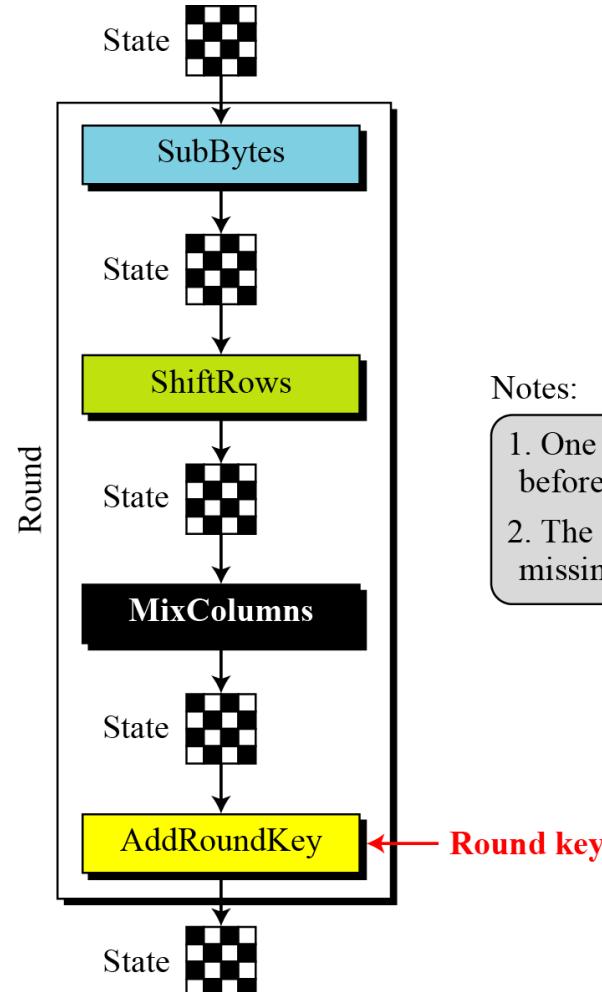


## *Advanced Encryption Standard (AES)*



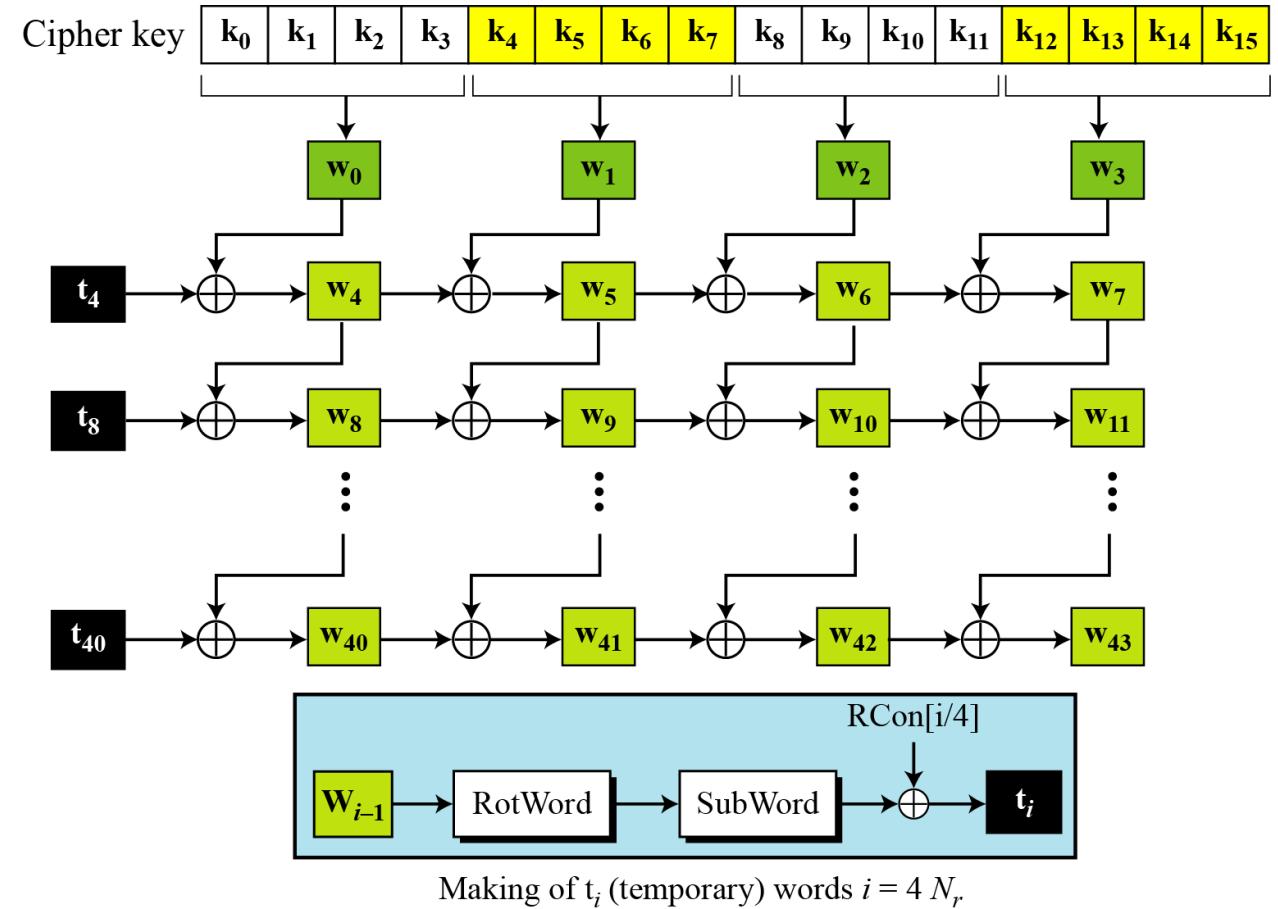
*Block-to-state and state-to-block transformation*

## Advanced Encryption Standard (AES)

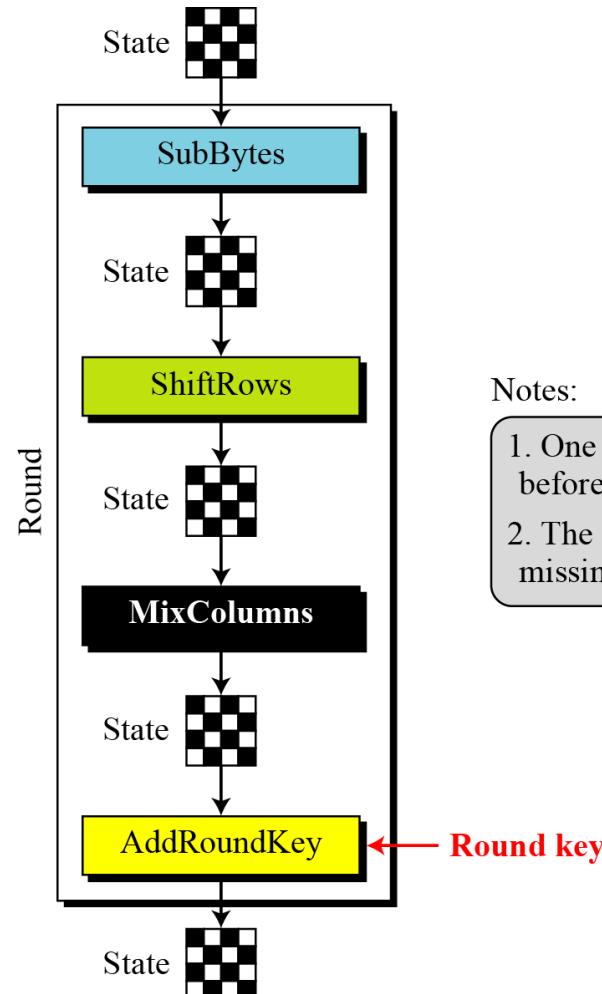


Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

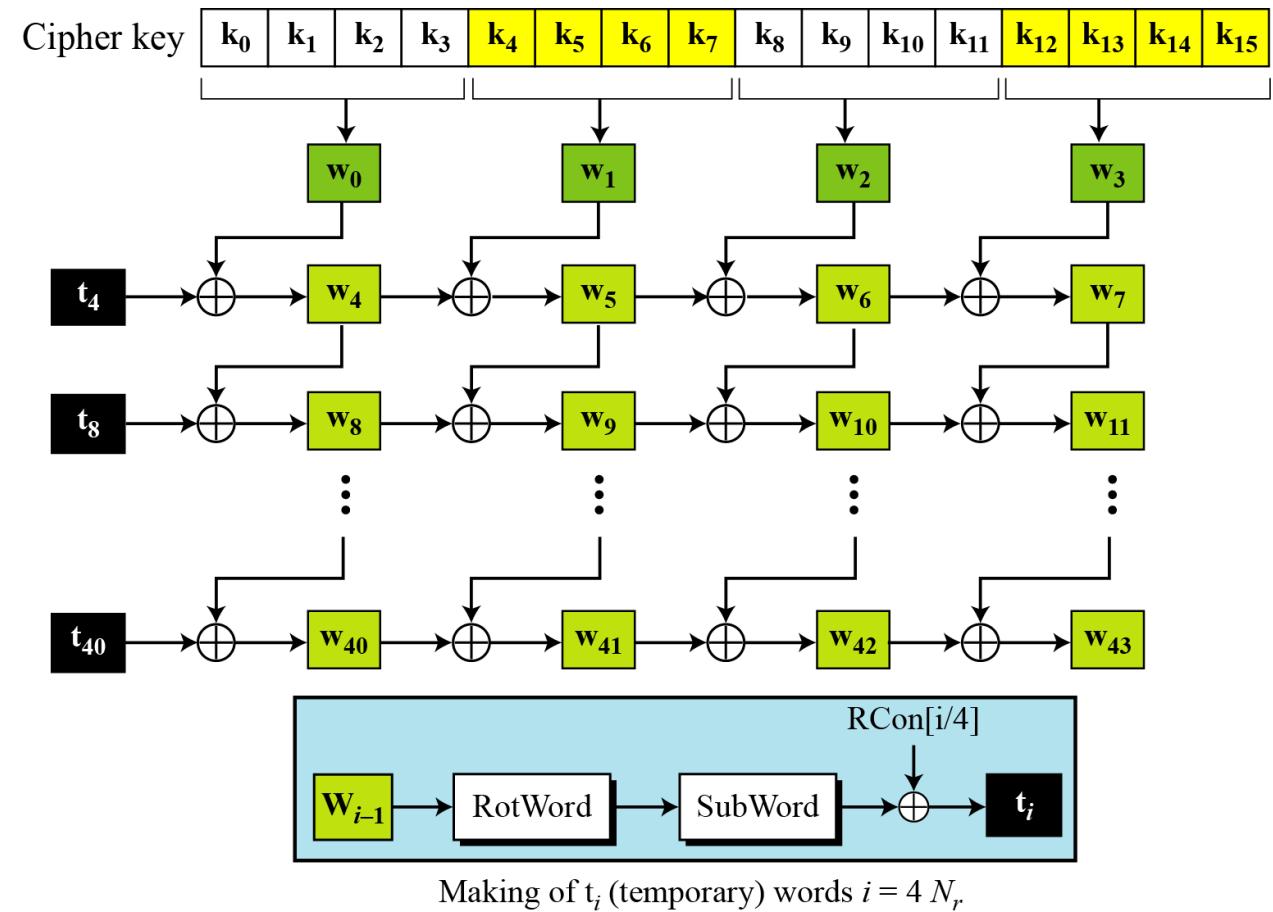


## Advanced Encryption Standard (AES)



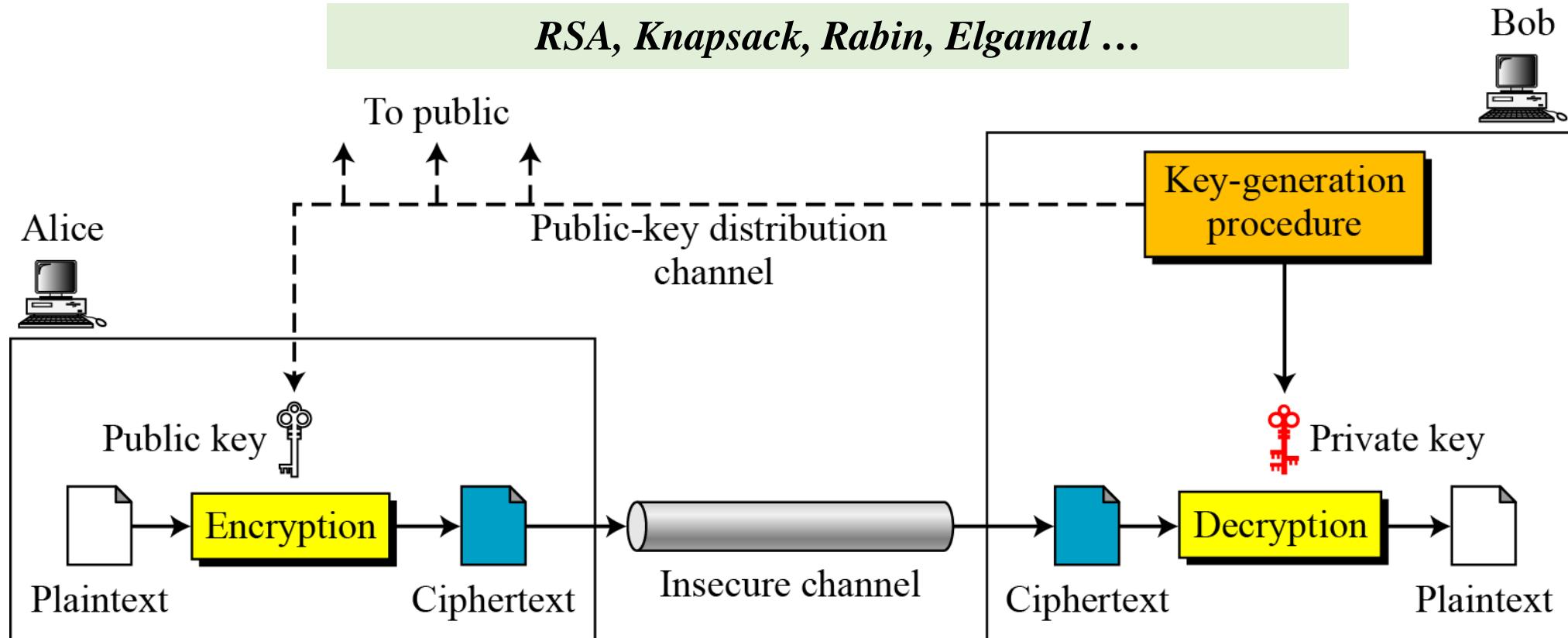
Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.



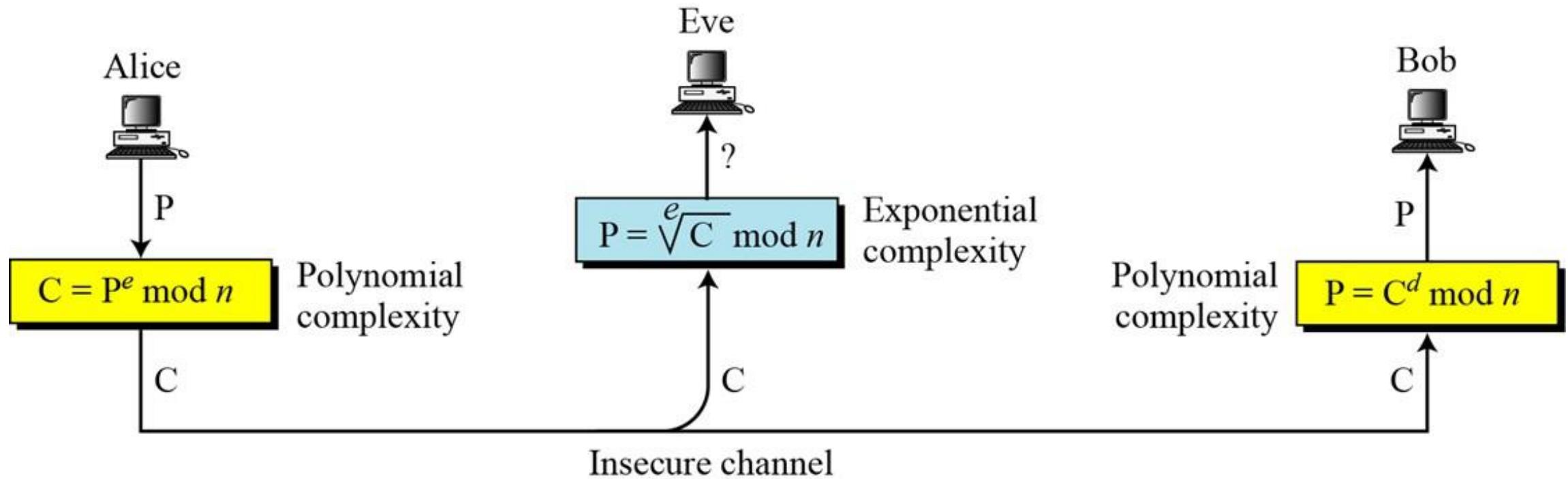
## Asymmetric Cryptography

RSA, Knapsack, Rabin, Elgamal ...



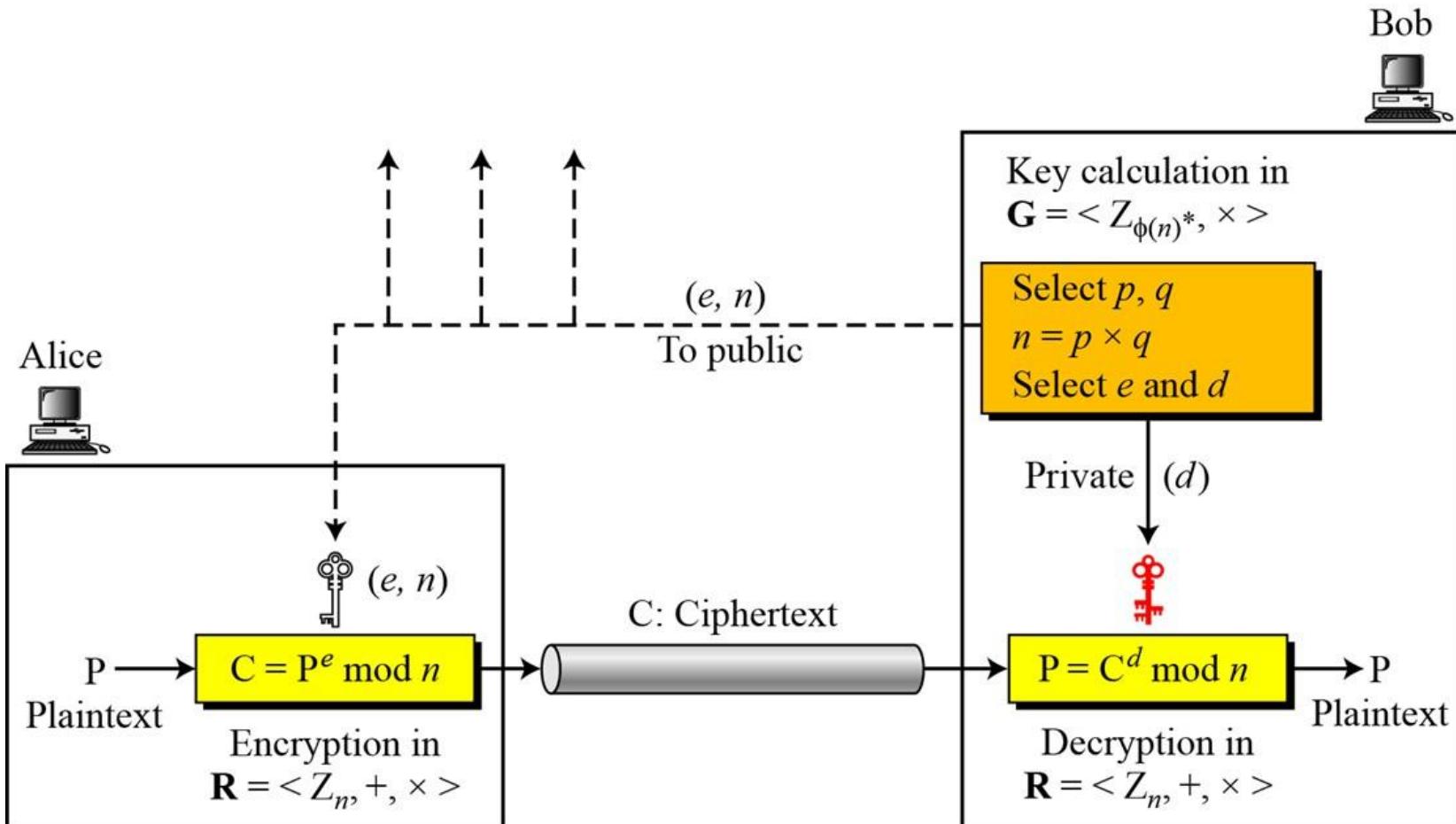
**Asymmetric cryptography uses two separate keys: one private and one public.**

## *Asymmetric Cryptography*



The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).

## Asymmetric Cryptography

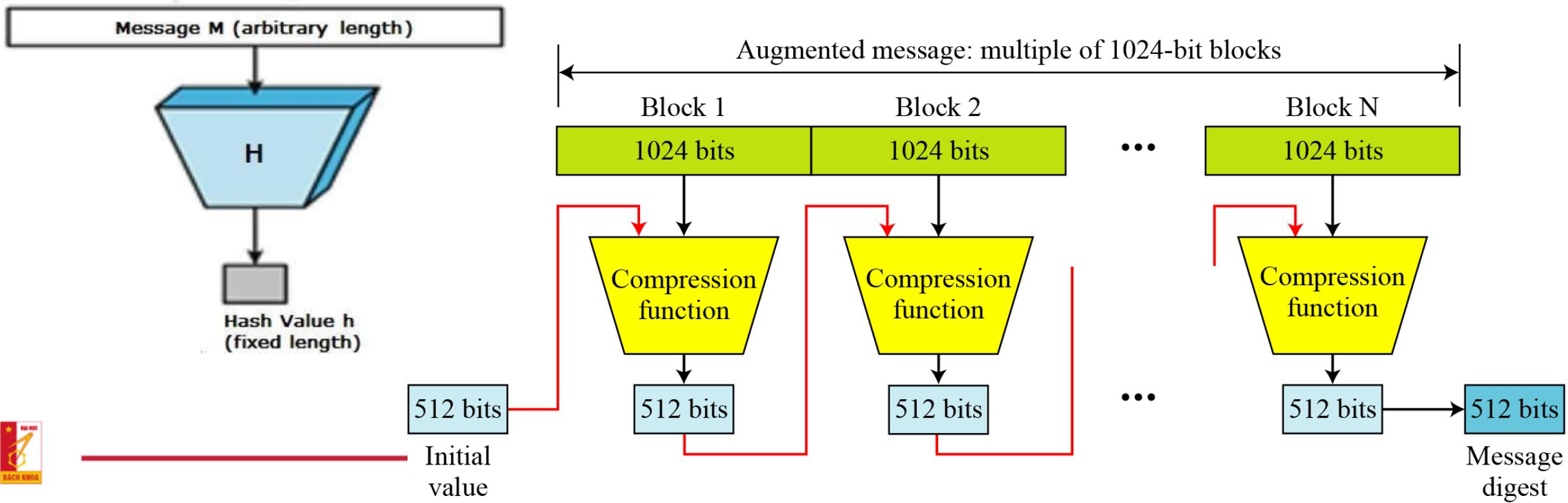
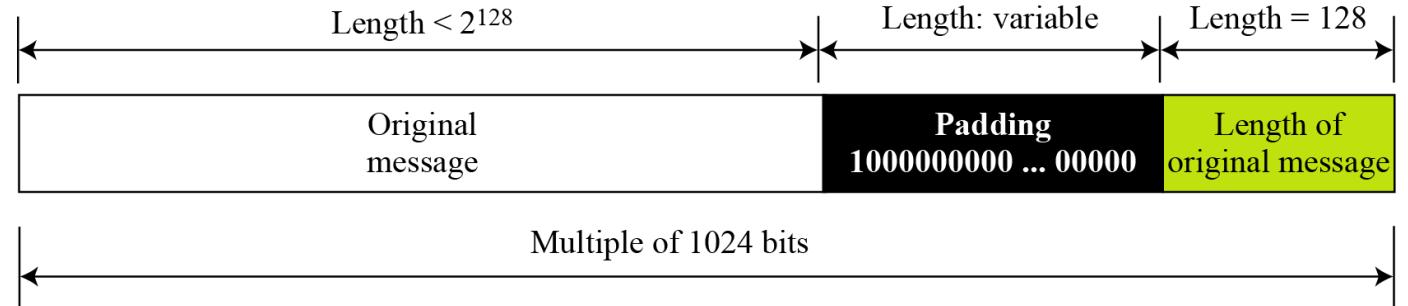


Encryption, decryption, and key generation in RSA

# Hệ mật hiện đại



## Hash Function



# **Chương 3**

## **Traditional Symmetric-Key Ciphers**

**Mật mã khóa đối xứng  
cổ điển**

## Objectives

- Xác định các thuật ngữ và khái niệm về mật mã khóa đối xứng
- Nhấn mạnh hai loại mật mã truyền thống: mật mã thay thế (substitution) và chuyển vị (transposition)
- Mô tả các loại phân tích mật mã (thám mã) được sử dụng để phá vỡ hệ mật mã đối xứng
- Giới thiệu các khái niệm về mật mã dòng và mật mã khối
- Thảo luận về một số mật mã nổi trội được sử dụng trong quá khứ, chẳng hạn như máy Enigma

## Objectives

- To define the terms and the concepts of symmetric key ciphers**
- To emphasize the two categories of traditional ciphers: substitution and transposition ciphers**
- To describe the categories of cryptanalysis used to break the symmetric ciphers**
- To introduce the concepts of the stream ciphers and block ciphers**
- To discuss some very dominant ciphers used in the past, such as the Enigma machine**

## 3-1 INTRODUCTION

*Hình 3.1 sau đây cho thấy ý tưởng chung đằng sau một mật mã khóa đối xứng. Thông điệp gốc từ Alice đến Bob được gọi là bản rõ; thông điệp được gửi qua kênh được gọi là bản mật. Để tạo bản mật từ bản rõ, Alice sử dụng một thuật toán mật mã hóa và một khóa bí mật. Để khôi phục bản rõ từ bản mật, Bob sử dụng một thuật toán giải mã và cùng một khóa bí mật.*

**Topics discussed in this section:**

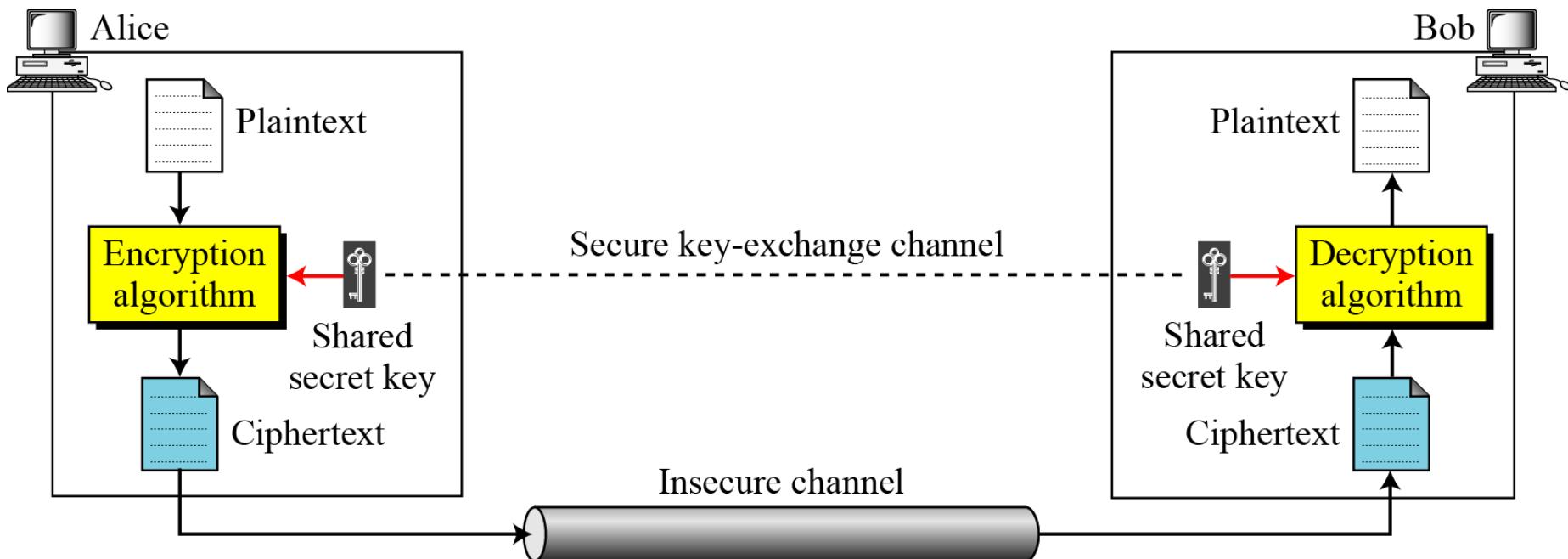
**3.1.1 Kerckhoff's Principle**

**3.1.2 Cryptanalysis**

**3.1.3 Categories of Traditional Ciphers**

### 3.1 *Continued*

**Figure 3.1 General idea of symmetric-key cipher**  
**Hình 3.1 Ý tưởng chung về mật mã khóa đối xứng**



### 3.1 *Continued*

*If  $P$  is the plaintext,  $C$  is the ciphertext, and  $K$  is the key,  
Nếu  $P$  là bản rõ,  $C$  là bản mã và  $K$  là khóa,*

Encryption:  $C = E_k(P)$

Decryption:  $P = D_k(C)$

In which,  $D_k(E_k(x)) = E_k(D_k(x)) = x$

*We assume that Bob creates  $P_1$ ; we prove that  $P_1 = P$ :  
Chúng ta giả định rằng Bob tạo  $P_1$ ; chúng tôi chứng minh  
rằng  $P_1 = P$ :*

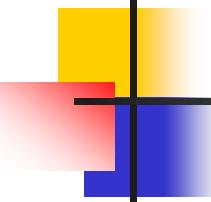
**Alice:**  $C = E_k(P)$

**Bob:**  $P_1 = D_k(C) = D_k(E_k(P)) = P$

### 3.1 *Continued*

**Figure 3.2** Locking and unlocking with the same key  
*Hình 3.2 Khóa và mở khóa bằng cùng một chìa khóa*





### **3.1.1 *Kerckhoff's Principle***

Based on **Kerckhoff's principle**, one should always assume that the adversary, Eve, knows the encryption/decryption algorithm. The resistance of the cipher to attack must be based only on the secrecy of the key.

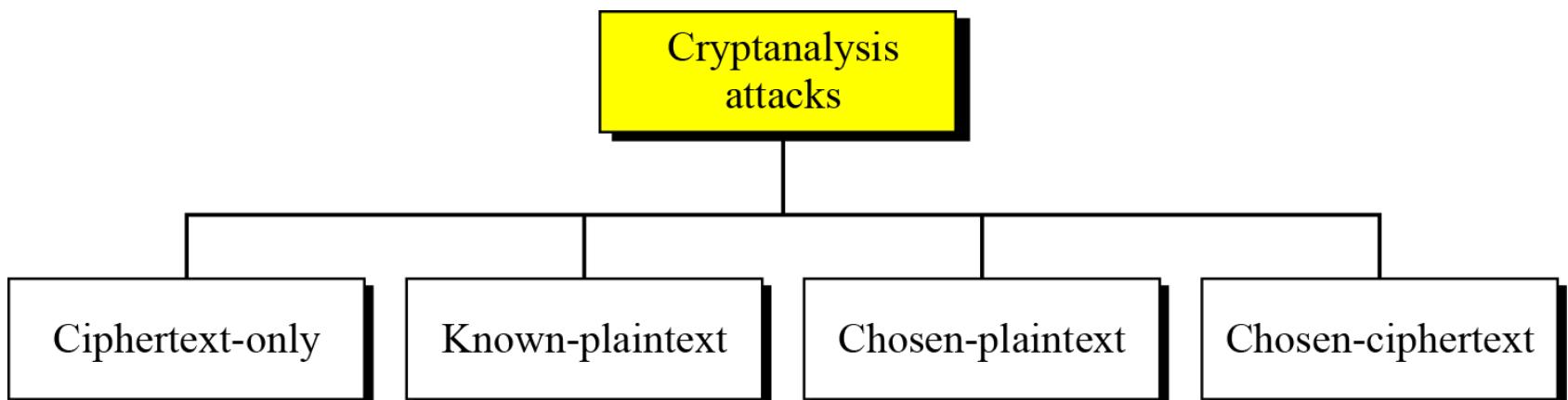
Dựa trên nguyên tắc của Kerckhoff, người ta nên luôn giả định rằng đối thủ, gọi là Eve, biết thuật toán mật mã hóa / giải mật mã. Khả năng chống lại sự tấn công của mật mã chỉ được dựa vào tính bí mật của khóa.

### 3.1.2 Cryptanalysis

As cryptography is the science and art of creating secret codes, **cryptanalysis** is the science and art of breaking those codes.

Vì mật mã là khoa học và nghệ thuật tạo ra các mã bí mật, nên phá mật mã là khoa học và nghệ thuật để phá các mật mã đó.

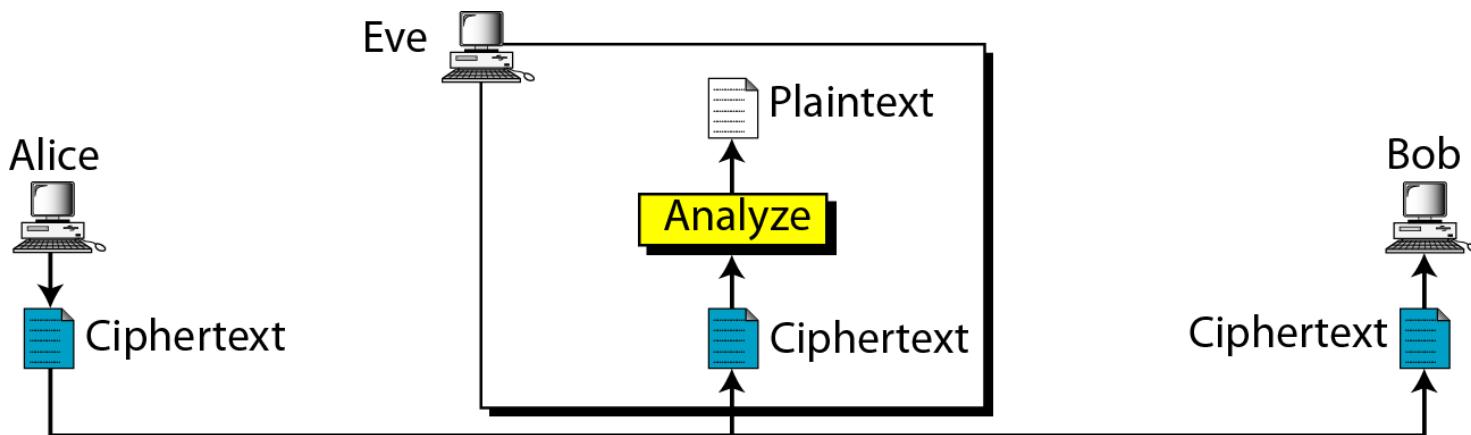
**Figure 3.3** *Cryptanalysis attacks*



## 3.1.2 *Continued*

### Ciphertext-Only Attack

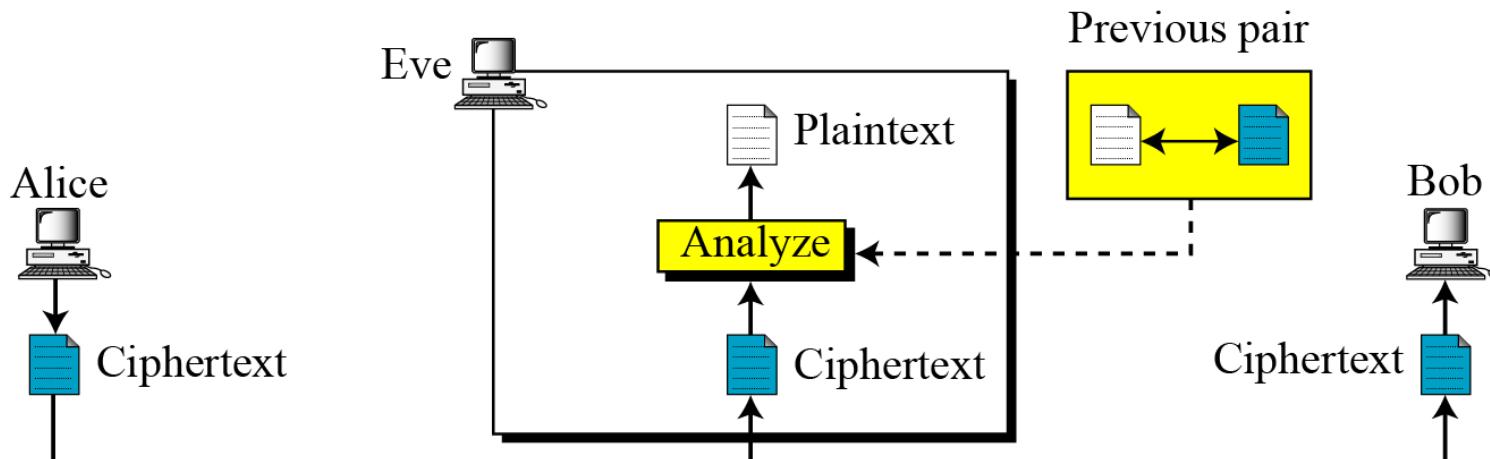
**Figure 3.4 Ciphertext-only attack**  
*Tấn công chỉ bằng văn bản mã*



## 3.1.2 *Continued*

### Known-Plaintext Attack

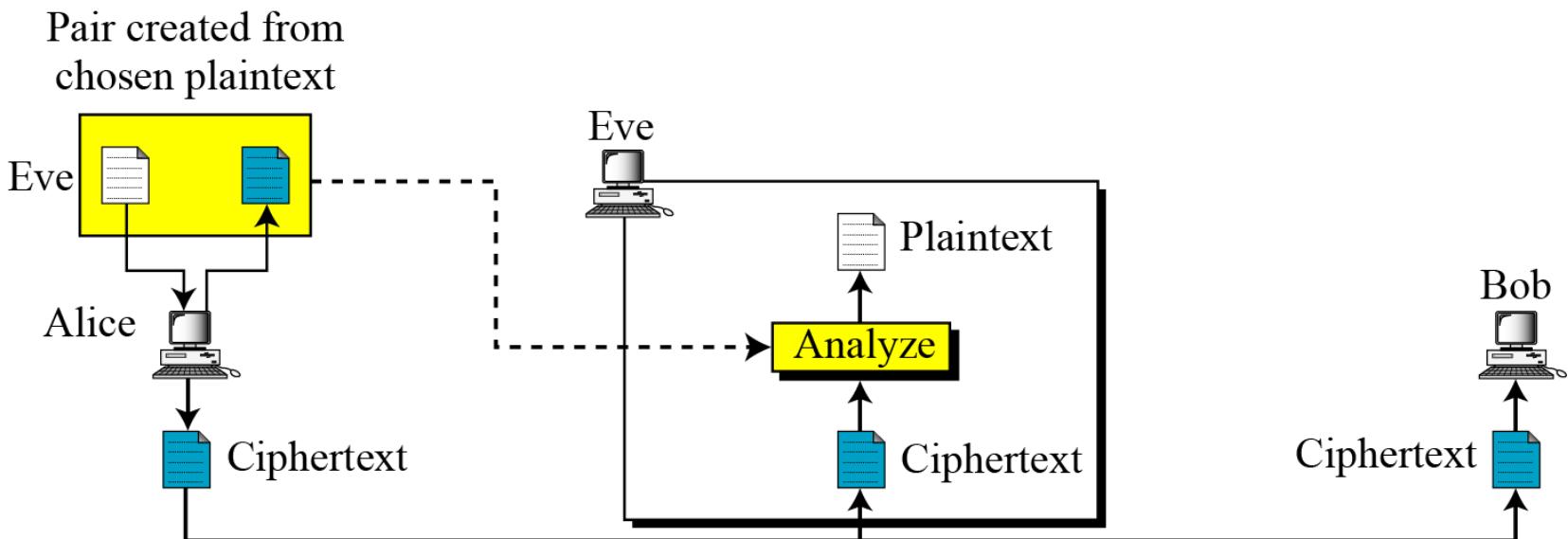
**Figure 3.5 Known-plaintext attack**  
*Tấn công văn bản rõ đã biết*



## 3.1.2 *Continued*

### Chosen-Plaintext Attack

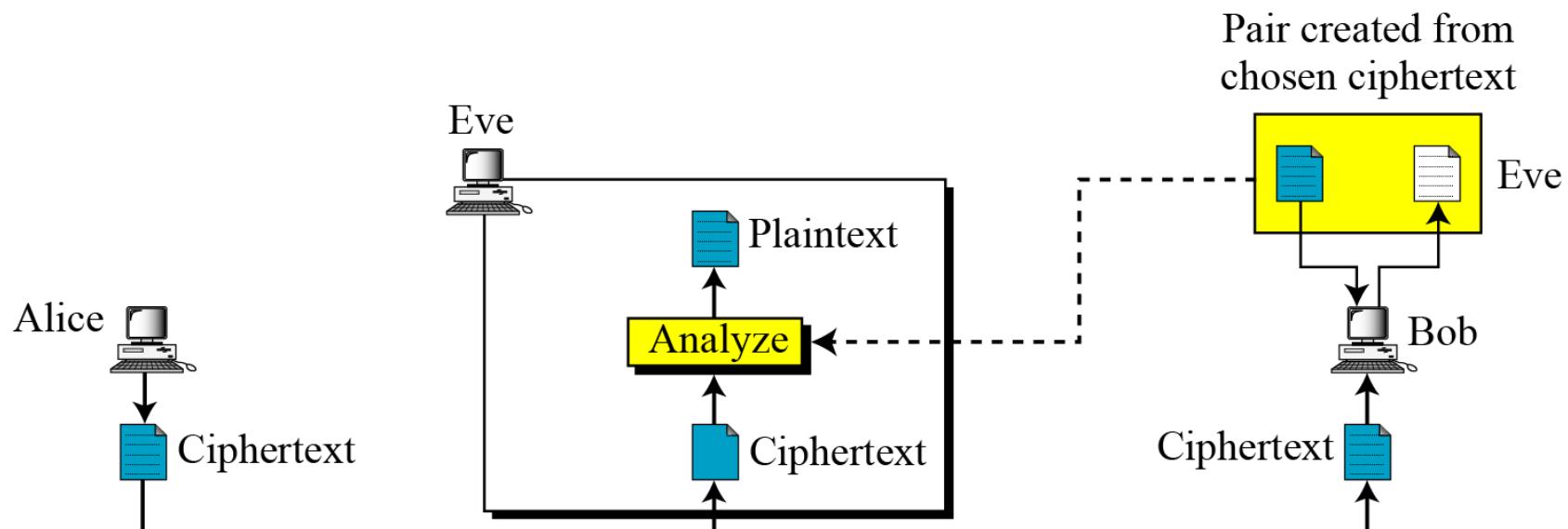
**Figure 3.6 Chosen-plaintext attack**  
*Cuộc tấn công plaintext được chọn*



## 3.1.2 *Continued*

### Chosen-Ciphertext Attack

**Figure 3.7 Chosen-ciphertext attack**  
*Tấn công bản mã được chọn*



## 3-2 SUBSTITUTION CIPHERS: MÃ THAY THẾ

A substitution cipher replaces one symbol with another. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers.

Một mật mã thay thế thay thế một ký hiệu bằng một ký hiệu khác. Mật mã thay thế có thể được phân loại là mật mã đơn ký tự hoặc mật mã đa ký tự.

**Note**

**A substitution cipher replaces one symbol with another.**

**Topics discussed in this section:**

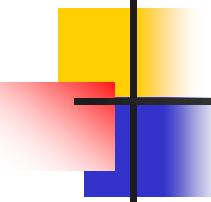
- 3.2.1 Monoalphabetic Ciphers: Hệ mật thay thế đơn ký tự**
- 3.2.2 Polyalphabetic Ciphers: Hệ mật thay thế đa ký tự**

### *3.2.1 Monoalphabetic Ciphers: Mã đơn ký tự*

#### *Note*

**In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always one-to-one.**

*Trong phép thay thế đơn ký tự, mối quan hệ giữa một ký hiệu trong bản rõ với một ký hiệu trong bản mã luôn là một-một.*



### 3.2.1 *Continued*

#### Example 3.1

The following shows a plaintext and its corresponding ciphertext. The cipher is probably monoalphabetic because both *l*'s (els) are encrypted as *O*'s.

**Plaintext:** hello

**Ciphertext:** KHOOR

#### Example 3.2

The following shows a plaintext and its corresponding ciphertext. The cipher is not monoalphabetic because each *l* (el) is encrypted by a different character.

### 3.2.1 *Continued*

#### Additive Cipher

The simplest monoalphabetic cipher is the additive cipher. This cipher is sometimes called a **shift cipher** and sometimes a **Caesar cipher**, but the term **additive cipher** better reveals its mathematical nature.

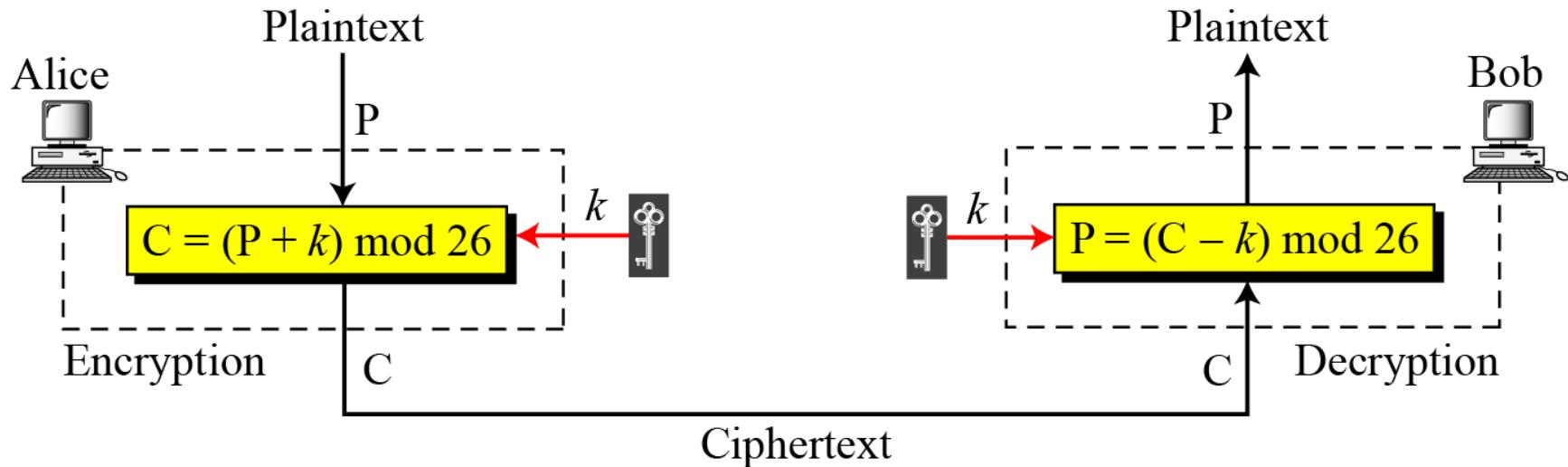
*Mật mã đơn pha đơn giản nhất là mật mã cộng. Mật mã này đôi khi được gọi là mật mã dịch chuyển và đôi khi là mật mã Caesar, nhưng thuật ngữ mật mã cộng thêm tiết lộ rõ hơn bản chất toán học của nó.*

**Figure 3.8 Plaintext and Ciphertext in  $Z_{26}$**

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

### 3.2.1 *Continued*

**Figure 3.9 Additive cipher: Mã cộng**



Hai quá trình mã hóa và giải mã hóa là thuận nghịch với nhau?  
 $P = (C - k) \text{ mod } 26 = (P + k - k) \text{ mod } 26 = P$

**Note**

**When the cipher is additive, the plaintext, ciphertext, and key are integers in  $\mathbb{Z}_{26}$ .**

### 3.2.1 *Continued*

#### Example 3.3

Use the additive cipher with key = 15 to *encrypt* the message “hello”.

#### Solution

Chúng ta áp dụng thuật toán mã hóa cho bản rõ, từng ký tự một:

Plaintext: h → 07

Encryption:  $(07 + 15) \text{ mod } 26$

Ciphertext: 22 → W

Plaintext: e → 04

Encryption:  $(04 + 15) \text{ mod } 26$

Ciphertext: 19 → T

Plaintext: l → 11

Encryption:  $(11 + 15) \text{ mod } 26$

Ciphertext: 00 → A

Plaintext: l → 11

Encryption:  $(11 + 15) \text{ mod } 26$

Ciphertext: 00 → A

Plaintext: o → 14

Encryption:  $(14 + 15) \text{ mod } 26$

Ciphertext: 03 → D

### 3.2.1 *Continued*

#### Example 3.4

Use the additive cipher with key = 15 to *decrypt* the message “WTAAD”.

#### Solution

We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W → 22

Decryption:  $(22 - 15) \bmod 26$

Plaintext: 07 → h

Ciphertext: T → 19

Decryption:  $(19 - 15) \bmod 26$

Plaintext: 04 → e

Ciphertext: A → 00

Decryption:  $(00 - 15) \bmod 26$

Plaintext: 11 → l

Ciphertext: A → 00

Decryption:  $(00 - 15) \bmod 26$

Plaintext: 11 → l

Ciphertext: D → 03

Decryption:  $(03 - 15) \bmod 26$

Plaintext: 14 → o

### 3.2.1 *Continued*

#### Shift Cipher and Caesar Cipher

Historically, additive ciphers are called **shift ciphers**. Julius Caesar used an additive cipher to communicate with his officers. For this reason, additive ciphers are sometimes referred to as the Caesar cipher. Caesar used a key of 3 for his communications.

##### Note

Additive ciphers are sometimes referred to as shift ciphers or Caesar cipher.

Về mặt lịch sử, mật mã cộng được gọi là *mật mã dịch chuyển*. Julius Caesar đã sử dụng một mật mã cộng để liên lạc với các sĩ quan của mình. Vì lý do này, mật mã cộng đôi khi được gọi là *mật mã Caesar*. Caesar đã sử dụng khóa 3 để liên lạc thông tin.

## 3.2.1 *Continued*

### Example 3.5

Eve has intercepted the ciphertext “UVACLYFZLJBYL”. Show how she can use a brute-force attack to break the cipher.

Eve đã chặn bản mã “UVACLYFZLJBYL”. Chỉ ra cách cô ấy có thể sử dụng một cuộc tấn công bạo lực để phá vỡ mật mã.

### Solution

Eve tries keys from 1 to 7. With a key of 7, the plaintext is “not very secure”, which makes sense.

Eve thử các khóa từ 1 đến 7. Với khóa 7, bản rõ là "không an toàn lắm", điều này có lý.

Ciphertext: UVACLYFZLJBYL

<b>K = 1</b>	→	<b>Plaintext:</b> tuzbkxeykiaxk
<b>K = 2</b>	→	<b>Plaintext:</b> styajwdxjhwj
<b>K = 3</b>	→	<b>Plaintext:</b> rsxzivcwigyvi
<b>K = 4</b>	→	<b>Plaintext:</b> qrwyhubvhfxuh
<b>K = 5</b>	→	<b>Plaintext:</b> pqvxgtaugewtg
<b>K = 6</b>	→	<b>Plaintext:</b> opuwfsztfdvsf
<b>K = 7</b>	→	<b>Plaintext:</b> notverysecure

### 3.2.1 *Continued*

**Bảng 3.1 Tần suất của các ký tự trong tiếng Anh**

Letter	Frequency	Letter	Frequency	Letter	Frequency	Letter	Frequency
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

**Table 3.2 tần suất xuất của diagrams and trigrams**

Digram	TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI, OF
Trigram	THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR, DTH

### 3.2.1 *Continued*

#### Example 3.6

Eve đã chặn đoạn mã sau đây. Sử dụng một cuộc tấn công thống kê, tìm ra bản rõ.

XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILY-  
VVCFIJSVIXLIWIPPIVVGIMZIWQSVISJJIVW

#### Solution

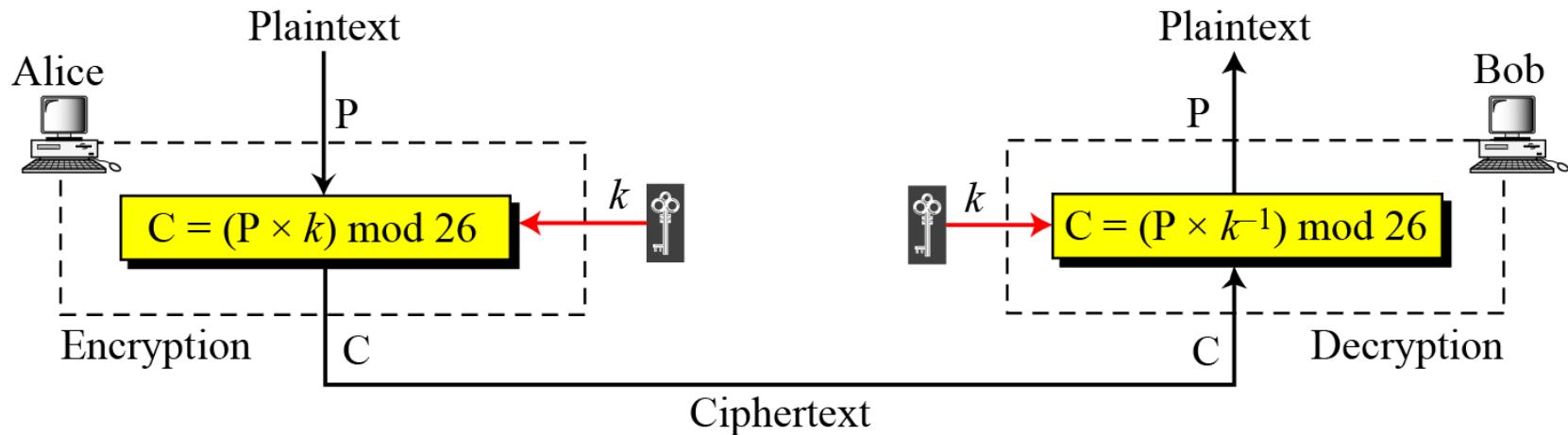
Khi Eve lập bảng tần suất của các chữ cái trong bản mã này, cô ấy nhận được: I = 14, V = 13, S = 12, v.v. Ký tự phổ biến nhất là I với 14 lần xuất hiện. Điều này có nghĩa là key = 4.

the house is now for sale for four million dollars it is worth more hurry before the seller receives more offers

### 3.2.1 *Continued*

#### Multiplicative Ciphers: Mã nhân

Figure 3.10 lược đồ mã nhân (*Multiplicative cipher*)



**Note**

In a multiplicative cipher, the plaintext and ciphertext are integers in  $Z_{26}$ ; the key is an integer in  $Z_{26}^*$ .

### 3.2.1 *Continued*

#### Example 3.7

Miền khóa (key domain) đối với bất kỳ mã nhân (multiplicative cipher) nào là gì, gồm những số nào?

#### Solution

Khóa cần tìm phải thuộc tập  $Z_{26}^*$ . Tập này có 12 số như sau: **1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25**.

#### Example 3.8

Chúng ta sử dụng mã nhân để mã hóa bản tin “hello” với khóa là 7. Khi đó, bản mã là “XCZZU”.

Plaintext: h → 07  
Plaintext: e → 04  
Plaintext: l → 11  
Plaintext: l → 11  
Plaintext: o → 14

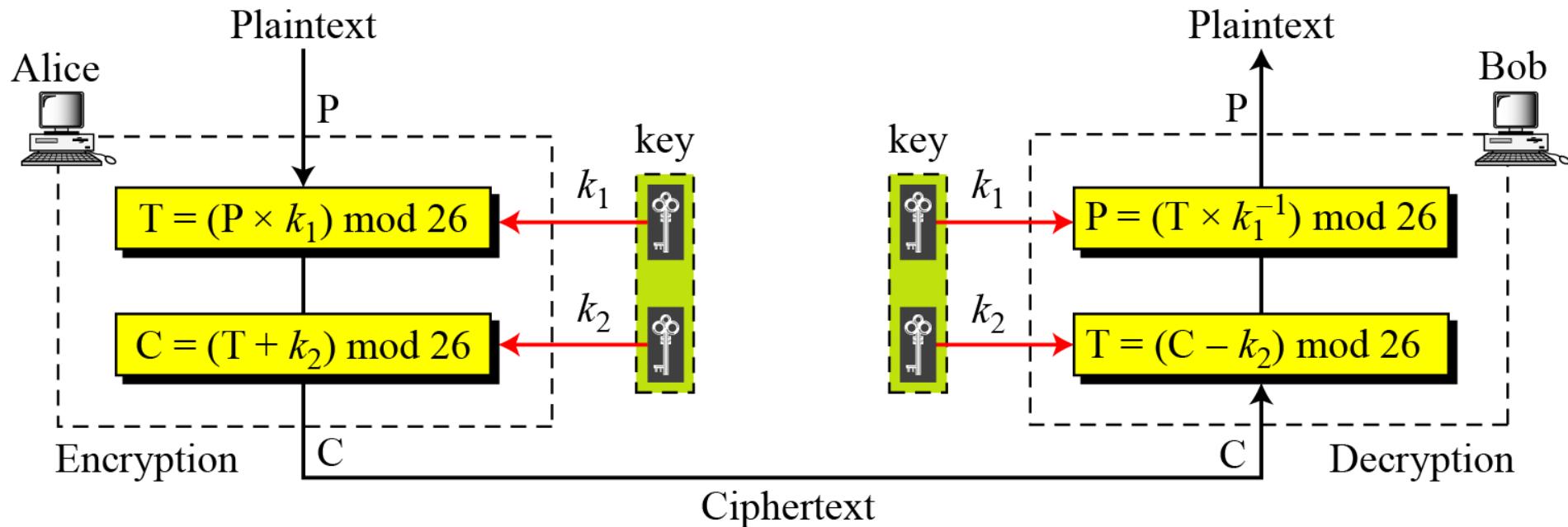
Encryption:  $(07 \times 07) \bmod 26$   
Encryption:  $(04 \times 07) \bmod 26$   
Encryption:  $(11 \times 07) \bmod 26$   
Encryption:  $(11 \times 07) \bmod 26$   
Encryption:  $(14 \times 07) \bmod 26$

ciphertext: 23 → X  
ciphertext: 02 → C  
ciphertext: 25 → Z  
ciphertext: 25 → Z  
ciphertext: 20 → U

### 3.2.1 *Continued*

#### Affine Ciphers: Mā Affine

**Figure 3.11** *Affine cipher*



$$C = (P \times k_1 + k_2) \bmod 26$$

$$P = ((C - k_2) \times k_1^{-1}) \bmod 26$$

where  $k_1^{-1}$  is the multiplicative inverse of  $k_1$  and  $-k_2$  is the additive inverse of  $k_2$

### 3.2.1 *Continued*

#### Example 3.09

The affine cipher uses a pair of keys in which the first key is from  $Z_{26}^*$  and the second is from  $Z_{26}$ . The size of the key domain is  $12 \times 26 = 312$ .

Mật mã affine sử dụng một cặp khóa, trong đó khóa đầu tiên là từ  $Z_{26}^*$  và khóa thứ hai là từ  $Z_{26}$ . Kích thước của miền khóa là  $12 \times 26 = 312$ .

#### Example 3.10

Use an affine cipher to encrypt the message “hello” with the key pair  $(7, 2)$ .

Sử dụng mật mã affine để mã hóa thông báo “hello” bằng cặp khóa  $(7, 2)$ .

$$P: h \rightarrow 07$$

$$\text{Encryption: } (07 \times 7 + 2) \bmod 26$$

$$C: 25 \rightarrow Z$$

$$P: e \rightarrow 04$$

$$\text{Encryption: } (04 \times 7 + 2) \bmod 26$$

$$C: 04 \rightarrow E$$

$$P: l \rightarrow 11$$

$$\text{Encryption: } (11 \times 7 + 2) \bmod 26$$

$$C: 01 \rightarrow B$$

$$P: l \rightarrow 11$$

$$\text{Encryption: } (11 \times 7 + 2) \bmod 26$$

$$C: 01 \rightarrow B$$

$$P: o \rightarrow 14$$

$$\text{Encryption: } (14 \times 7 + 2) \bmod 26$$

$$C: 22 \rightarrow W$$

## 3.2.1 *Continued*

### Example 3.11

Use the affine cipher to decrypt the message “ZEBBW” with the key pair  $(7, 2)$  in modulus 26.

Sử dụng mật mã affine để giải mã thông báo “ZEBBW” bằng cặp khóa  $(7, 2)$  trong mô-đun 26.

### Solution

C: Z  $\rightarrow$  25

Decryption:  $((25 - 2) \times 7^{-1}) \bmod 26$

P:07  $\rightarrow$  h

C: E  $\rightarrow$  04

Decryption:  $((04 - 2) \times 7^{-1}) \bmod 26$

P:04  $\rightarrow$  e

C: B  $\rightarrow$  01

Decryption:  $((01 - 2) \times 7^{-1}) \bmod 26$

P:11  $\rightarrow$  l

C: B  $\rightarrow$  01

Decryption:  $((01 - 2) \times 7^{-1}) \bmod 26$

P:11  $\rightarrow$  l

C: W  $\rightarrow$  22

Decryption:  $((22 - 2) \times 7^{-1}) \bmod 26$

P:14  $\rightarrow$  o

### Example 3.12

The additive cipher is a special case of an affine cipher in which  $k_1 = 1$ . The multiplicative cipher is a special case of affine cipher in which  $k_2 = 0$ .

Mật mã cộng là một trường hợp đặc biệt của mật mã affine, trong đó  $k_1 = 1$ .

Mật mã nhân là một trường hợp đặc biệt của mật mã affine trong đó  $k_2 = 0$ .

### 3.2.1 *Continued*

#### Monoalphabetic Substitution Cipher

Because additive, multiplicative, and affine ciphers have small key domains, they are very vulnerable to brute-force attack.

*Do mật mã cộng, nhân và mã Affine có miền khóa nhỏ, chúng rất dễ bị tấn công cực mạnh (brute-force).*

A better solution is to create a mapping between each plaintext character and the corresponding ciphertext character. Alice and Bob can agree on a table showing the mapping for each character.

*Một giải pháp tốt hơn là tạo một ánh xạ giữa mỗi ký tự bản rõ và ký tự bản mã tương ứng. Alice và Bob có thể đồng ý về một bảng hiển thị ánh xạ cho mỗi ký tự.*

**Figure 3.12** An example key for monoalphabetic substitution cipher

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

### 3.2.1 *Continued*

#### Example 3.13

We can use the key in Figure 3.12 to encrypt the message  
Chúng ta có thể sử dụng khóa trong hình 3.12 để mã hóa thông  
điệp sau đây:

this message is easy to encrypt but hard to find the key

The ciphertext is

ICFVQRVVNEFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS

### 3.2.2 Polyalphabetic Ciphers: Mã đa ký tự

In polyalphabetic substitution, each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many.

Trong thay thế đa chữ cái, mỗi lần xuất hiện của một ký tự có thể có một thay thế khác nhau. Mỗi quan hệ giữa một ký tự trong bản rõ với một ký tự trong bản mã là một-nhiều.

#### Autokey Cipher

$$P = P_1 P_2 P_3 \dots$$

$$C = C_1 C_2 C_3 \dots$$

$$k = (k_1, P_1, P_2, \dots)$$

Encryption:  $C_i = (P_i + k_i) \bmod 26$

Decryption:  $P_i = (C_i - k_i) \bmod 26$

## 3.2.2 *Continued*

### Example 3.14

Assume that Alice and Bob agreed to use an autokey cipher with initial key value  $k_1 = 12$ . Now Alice wants to send Bob the message “Attack is today”. Enciphering is done character by character.

*Giả sử rằng Alice và Bob đã đồng ý sử dụng một mật mã autokey với giá trị khóa ban đầu là  $k_1 = 12$ . Nay giờ, Alice muốn gửi cho Bob thông báo “Attack is today”. Mã hóa được thực hiện theo từng ký tự.*

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	12	00	19	19	00	02	10	08	18	19	14	03	00
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	M	T	M	T	C	M	S	A	L	H	R	D	Y

$$P = P_1 P_2 P_3 \dots$$

$$C = C_1 C_2 C_3 \dots$$

$$k = (k_1, P_1, P_2, \dots)$$

$$\text{Encryption: } C_i = (P_i + k_i) \bmod 26$$

$$\text{Decryption: } P_i = (C_i - k_i) \bmod 26$$

## 3.2.2 *Continued*

### Playfair Cipher: Mã Playfair → hệ đa ký tự

Figure 3.13 Ví dụ về khóa bí mật trong mật mã Playfair

Secret Key =  
 $5 \times 5$

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

1) Nếu cặp 2 chữ nằm cùng hàng → thay bởi các chữ cái bên phải

2) Nếu cặp 2 chữ nằm cùng cột → thay bởi các chữ cái bên dưới

3) Các trường hợp khác còn lại → mỗi chữ cái được thay thế bởi chữ cái khác cùng hàng, nhưng trên cùng cái cột mà chữ cái cùng cặp

$P = \text{“BA CH KH OA HA NO IX”}$   
 $K = \text{“DTVT”}$

$P \quad C$  Ma trận khóa  $5 \times 5$

$BA \rightarrow GD$   
 $CH \rightarrow BI$   
 $KH \rightarrow LI$   
 $OA \rightarrow RT$   
 $HA \rightarrow MD$   
 $NO \rightarrow OP$   
 $IX \rightarrow LU$

$D \ T \ V \ T \ A$   
 $B \ C \ E \ F \ G$   
 $H \ I \ K \ L \ M$   
 $N \ O \ P \ Q \ R$   
 $S \ U \ W \ X \ Y$

$C = \text{“GDBILIRTMDOPLU”}$

#### Example 3.15

Let us encrypt the plaintext “hello” using the key in Figure 3.13.

he → EC

Plaintext: hello

lx → QZ

Ciphertext: ECQZBX

## 3.2.2 *Continued*

### Vigenere Cipher

$$P = P_1 P_2 P_3 \dots$$

$$C = C_1 C_2 C_3 \dots$$

$$K = [(k_1, k_2, \dots, k_m), (k_1, k_2, \dots, k_m), \dots]$$

$$\text{Encryption: } C_i = P_i + k_i$$

$$\text{Decryption: } P_i = C_i - k_i$$

#### Example 3.16

We can encrypt the message “She is listening” using the 6-character keyword “PASCAL”.

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	15	00	18	02	00	11	15	00	18	02	00	11	15	00
C's values:	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Ciphertext:	H	H	W	K	S	W	X	S	L	G	N	T	C	G

## 3.2.2 *Continued*

### Example 3.16

Let us see how we can encrypt the message “She is listening” using the 6-character keyword “PASCAL”. The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream (as many times as needed).

Xem cách có thể mã hóa thông điệp “She is listening” bằng cách sử dụng từ khóa 6 ký tự “PASCAL”. Luồng khóa ban đầu là (15, 0, 18, 2, 0, 11). Luồng khóa là sự lặp lại của luồng khóa ban đầu này (dùng nhiều lần nếu cần).

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	15	00	18	02	00	11	15	00	18	02	00	11	15	00
C's values:	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Ciphertext:	H	H	W	K	S	W	X	S	L	G	N	T	C	G

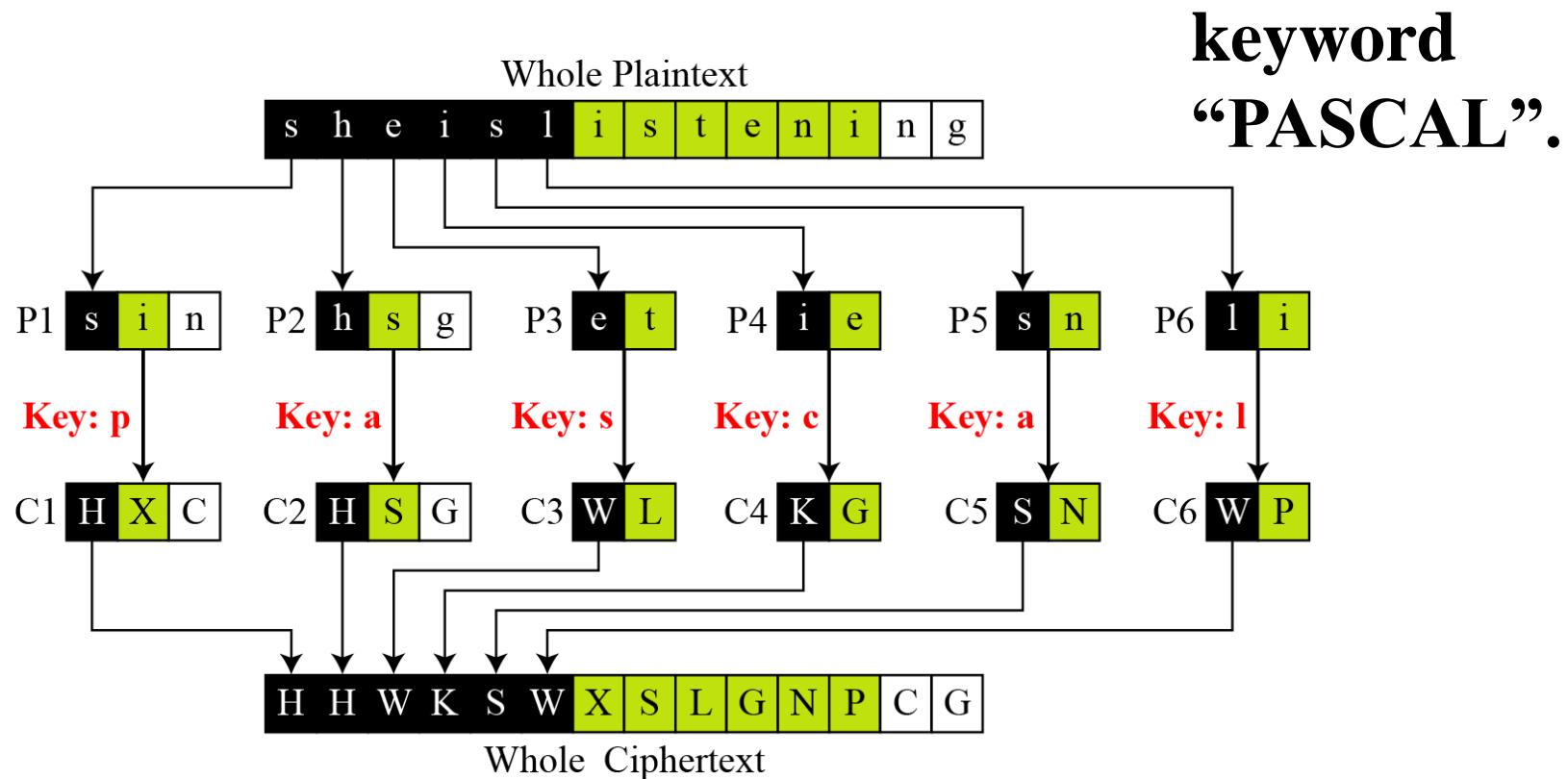
## 3.2.2 *Continued*

### Example 3.17

Mật mã Vigenere có thể được xem là sự kết hợp của m mật mã cộng.

**Figure 3.14** A Vigenere cipher as a combination of m additive ciphers

Mật mã Vigenere là sự kết hợp của m mật mã cộng



## 3.2.2 *Continued*

### Example 3.18

Using Example 3.18, we can say that the additive cipher is a special case of Vigenere cipher in which  $m = 1$ .

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	v	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

**Table 3.3**  
*A Vigenere Table*

## 3.2.2 *Continued*

### Vigenere Cipher (Crypanalysis)

#### Example 3.19

Giả sử chúng tôi đã chặn đoạn mã sau:

LIOMWGFEGGDVWGHHCQUCRHRWAGWIOWQLKGZETKKMEVLWPCZVGTH-  
VTSGXQOVGCSVETQLTJSUMVVVEUVLXEWSLGZMVVWLGYHCUSWXQH-  
KVGSHEEVFLCFDGVSUMPHKIRZDMPHHBVWWJWIXGFWLTSHGJOUEEHH-  
VUCFVGOWICQLTJSUXGLW

Kiểm tra Kasiski về sự lặp lại của các đoạn ba ký tự cho kết quả như trong Bảng 3.4.

<i>String</i>	<i>First Index</i>	<i>Second Index</i>	<i>Difference</i>
JSU	68	168	100
SUM	69	117	48
VWV	72	132	60
MPH	119	127	8

## 3.2.2 *Continued*

### Example 3.19

Let us assume we have intercepted the following ciphertext:

LIOMWGEGGDVWGHHCQUCRHRWAGWIOWQLKGZETKKMEVLWPCZVGTH-  
VTSGXQOVGCSVETQLTJSUMVWVEUVLXEWSLGFMVVWLGYHCUSWXQH-  
KVGSHEEVFLCFDGVSUMPHKIRZDMPHHBVWVWJWIXGFWLTSHGJOUEEHH-  
VUCFVGOWICQLTJSUXGLW

The Kasiski test for repetition of three-character segments yields the results shown in Table 3.4.

<i>String</i>	<i>First Index</i>	<i>Second Index</i>	<i>Difference</i>
JSU	68	168	100
SUM	69	117	48
VWV	72	132	60
MPH	119	127	8

## 3.2.2 *Continued*

### Example 3.19 (Continued)

The greatest common divisor of differences is 4, which means that the key length is multiple of 4. First try  $m = 4$ .

*Ước chung lớn nhất của các khác biệt là 4, có nghĩa là độ dài khóa là bội số của 4. Đầu tiên hãy thử  $m = 4$ .*

C1 : LWGWCRAOKTEPGTQCTJVUEGVGUQGECVPRPVJGTJEUGCJG

P1 : jueuapymircneroarhtsthihytrahcieixsthcarrehe

C2 : IGGGQHGWGKVCTSOSQS WVWFVYSHSVF SHZHWWF SOHCOQSL

P2 : ussscts is who feaeceihcetes oecatn pn ther hctecex

C3 : OFDHURWQZKLZHGVVLUVLSZWHWKHF DUKDHVIWHUHF WL UW

P3 : lcaerotnwhi wed ssirsi irhkete hretl t i ideat rairt

C4 : MEVHCWILEMWVVXGETMEXMLCXVELGMIMBWXLGEVVITX

P4 : i ardy sehaisrrt capia fpwt et hecarha esf terect pt

Trong trường hợp này, bản rõ có ý nghĩa.

---

Julius Caesar used a cryptosystem in his wars, which is now referred to as Caesar cipher.

It is an additive cipher with the key set to three. Each character in the plaintext is shifted three characters to create ciphertext.

---

## 3.2.2 *Continued*

### Hill Cipher: Mă Hill

Figure 3.15 Key in the Hill cipher

$$K = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1m} \\ k_{21} & k_{22} & \dots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{m1} & k_{m2} & \dots & k_{mm} \end{bmatrix}$$

$$\begin{aligned} C_1 &= P_1 k_{11} + P_2 k_{21} + \dots + P_m k_{m1} \\ C_2 &= P_1 k_{12} + P_2 k_{22} + \dots + P_m k_{m2} \\ &\dots \\ C_m &= P_1 k_{1m} + P_2 k_{2m} + \dots + P_m k_{mm} \end{aligned}$$

**Note**

*Ma trận khóa trong mật mã Hill cần có một nghịch đảo nhân.*

The key matrix in the Hill cipher needs to have a multiplicative inverse.

## 3.2.2 *Continued*

### Example 3.20

For example, the plaintext “code is ready” can make a  $3 \times 4$  matrix when adding extra bogus character “z” to the last block and removing the spaces. The ciphertext is “OHKNIHGKLSS”.

*Ví dụ, bản rõ “code is ready” có thể tạo ma trận  $3 \times 4$  khi thêm ký tự không có thật “z” vào khối cuối cùng và loại bỏ khoảng trắng. Bản mã là “OHKNIHGKLSS”.*

**Figure 3.16 Example 3.20**

$$\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}^C = \begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}^P \begin{bmatrix} 09 & 07 & 11 & 13 \\ 04 & 07 & 05 & 06 \\ 02 & 21 & 14 & 09 \\ 03 & 23 & 21 & 08 \end{bmatrix}^K$$

**a. Encryption**

$$\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}^P = \begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}^C \begin{bmatrix} 02 & 15 & 22 & 03 \\ 15 & 00 & 19 & 03 \\ 09 & 09 & 03 & 11 \\ 17 & 00 & 04 & 07 \end{bmatrix}^{K^{-1}}$$

**b. Decryption**

## 3.2.2 *Continued*

### Example 3.21

Assume that Eve knows that  $m = 3$ . She has intercepted three plaintext/ciphertext pair blocks (not necessarily from the same message) as shown in Figure 3.17.

*Giả sử rằng Eve biết rằng  $m = 3$ . Cô ấy đã chặn ba khối cặp bản rõ / bản mã (không nhất thiết phải từ cùng một thông điệp) như trong Hình 3.17.*

**Figure 3.17 Example 3.21**

$$\begin{bmatrix} 05 & 07 & 10 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 03 & 06 & 00 \end{bmatrix}$$

$$\begin{bmatrix} 13 & 17 & 07 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 14 & 16 & 09 \end{bmatrix}$$

$$\begin{bmatrix} 00 & 05 & 04 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 03 & 17 & 11 \end{bmatrix}$$

P

C

### 3.2.2 *Continued*

#### Example 3.21 (Continued)

She makes matrices P and C from these pairs. Because P is invertible, she inverts the P matrix and multiplies it by C to get the K matrix as shown in Figure 3.18.

Cô ấy tạo ma trận P và C từ các cặp này. Vì P khả nghịch nên cô ấy đảo ngược ma trận P và nhân nó với C để được ma trận K như trong hình 3.18.

**Figure 3.18 Example 3.21**

$$\begin{bmatrix} 02 & 03 & 07 \\ 05 & 07 & 09 \\ 01 & 02 & 11 \end{bmatrix} = \begin{bmatrix} 21 & 14 & 01 \\ 00 & 08 & 25 \\ 13 & 03 & 08 \end{bmatrix} \begin{bmatrix} 03 & 06 & 00 \\ 14 & 16 & 09 \\ 03 & 17 & 11 \end{bmatrix}$$

**K**                    **P<sup>-1</sup>**                    **C**

Now she has the key and can break any ciphertext encrypted with that key.

### 3.2.2 *Continued*

#### One-Time Pad: Bảng dùng một lần

One of the goals of cryptography is perfect secrecy. A study by Shannon has shown that perfect secrecy can be achieved if each plaintext symbol is encrypted with a key randomly chosen from a key domain. This idea is used in a cipher called one-time pad, invented by **Vernam**.

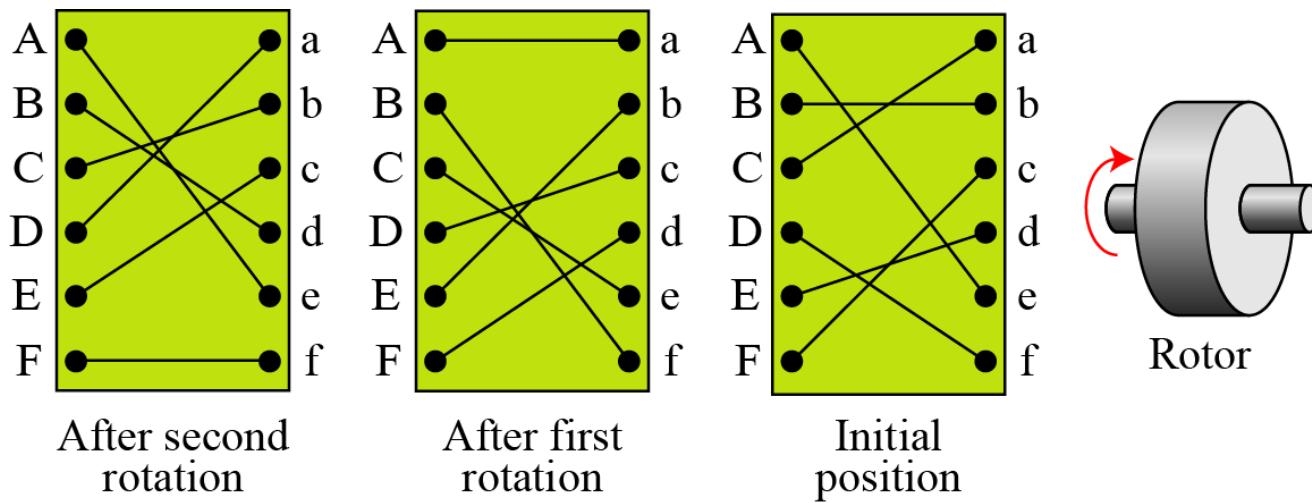
*Một trong những mục tiêu của mật mã là bí mật hoàn hảo. Một nghiên cứu của Shannon đã chỉ ra rằng có thể đạt được sự bí mật hoàn hảo nếu mỗi ký hiệu bản rõ được mã hóa bằng một khóa được chọn ngẫu nhiên từ một miền khóa.*

*Ý tưởng này được sử dụng trong một mật mã được gọi là bảng một lần, do Vernam phát minh.*

## 3.2.2 *Continued*

### Rotor Cipher

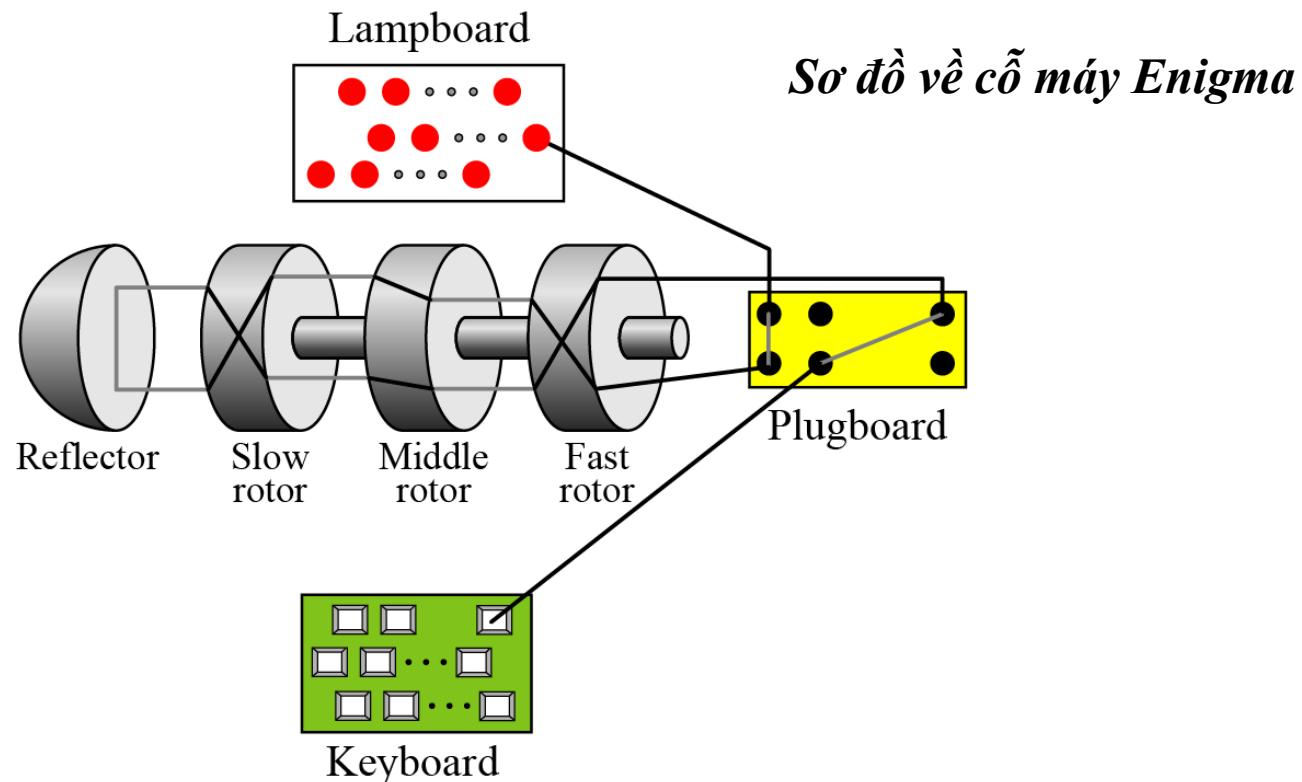
**Figure 3.19 A rotor cipher**



## 3.2.2 *Continued*

### Enigma Machine

**Figure 3.20** *A schematic of the Enigma machine*



## 3-3 TRANSPOSITION CIPHERS: MÃ CHUYỂN VỊ

*A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols.*

*Mật mã chuyển vị không thay thế một ký hiệu này cho một ký hiệu khác, thay vào đó nó thay đổi vị trí của các ký hiệu.*

**Note**

→ *Mật mã chuyển vị sắp xếp lại các ký hiệu.*

**A transposition cipher reorders symbols.**

**Topics discussed in this section:**

**3.3.1 Keyless Transposition Ciphers**

**3.3.2 Keyed Transposition Ciphers**

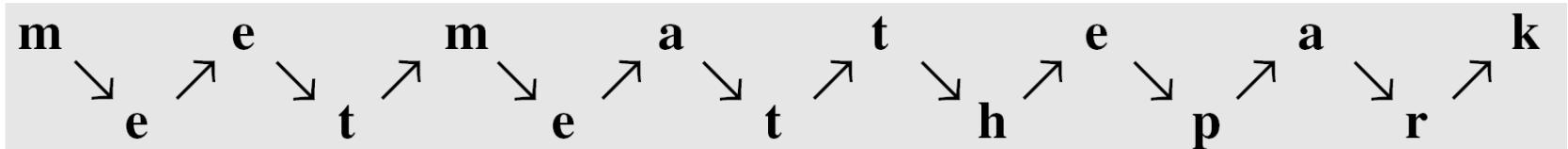
**3.3.3 Combining Two Approaches**

### 3.3.1 Keyless Transposition Ciphers: *Mã chuyển vị không khóa*

Simple transposition ciphers, which were used in the past, are keyless.

#### Example 3.22

Một ví dụ điển hình về mật mã không khóa sử dụng phương pháp đầu tiên là mật mã hàng rào đường sắt (**rail fence cipher**). Bản mã được tạo để đọc từng dòng mẫu. Ví dụ: để gửi tin nhắn “Meet me at the park” cho Bob, Alice viết



She then creates the ciphertext “**MEMATEAKETETHPR**”.

### 3.3.1 *Continued*

#### Example 3.23

Alice and Bob can agree on the number of columns and use the second method. Alice writes the same plaintext, row by row, in a table of four columns.

m	e	e	t
m	e	a	t
t	h	e	p
a	r	k	

She then creates the ciphertext “MMTAEEHREAEKTP”.

### 3.3.1 *Continued*

#### Example 3.24

The cipher in Example 3.23 is actually a transposition cipher. The following shows the **permutation** of each character in the plaintext into the ciphertext based on the positions.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
01	05	09	13	02	06	10	13	03	07	11	15	04	08	12

The second character in the plaintext has moved to the fifth position in the ciphertext; the third character has moved to the ninth position; and so on. Although the characters are permuted, there is a pattern in the permutation: (01, 05, 09, 13), (02, 06, 10, 13), (03, 07, 11, 15), and (08, 12). In each section, the difference between the two adjacent numbers is 4.

### 3.3.2 *Keyed Transposition Ciphers:*

#### *Mã chuyển vị có khóa*

The keyless ciphers permute the characters by using writing plaintext in one way and reading it in another way. The permutation is done on the whole plaintext to create the whole ciphertext. Another method is to *divide the plaintext into groups of predetermined size, called blocks*, and then use a key to permute the characters in each block separately.

*Các mật mã không khóa hoán vị các ký tự bằng cách viết bản rõ theo cách này và đọc theo cách khác. Hoán vị được thực hiện trên toàn bộ bản rõ để tạo ra toàn bộ bản mã.*

*Một phương pháp khác là chia bản rõ thành các nhóm có kích thước xác định trước, được gọi là các khối, sau đó sử dụng một khóa để hoán vị các ký tự trong mỗi khối riêng biệt.*

## 3.3.2 *Continued*

### Example 3.25

Alice needs to send the message “Enemy attacks tonight” to Bob..

e n e m y      a t t a c k s      o n i g h t z

The key used for encryption and decryption is a *permutation key*, which shows how the characters are permuted.

Encryption ↓

3	1	4	5	2
1	2	3	4	5

↑ Decryption

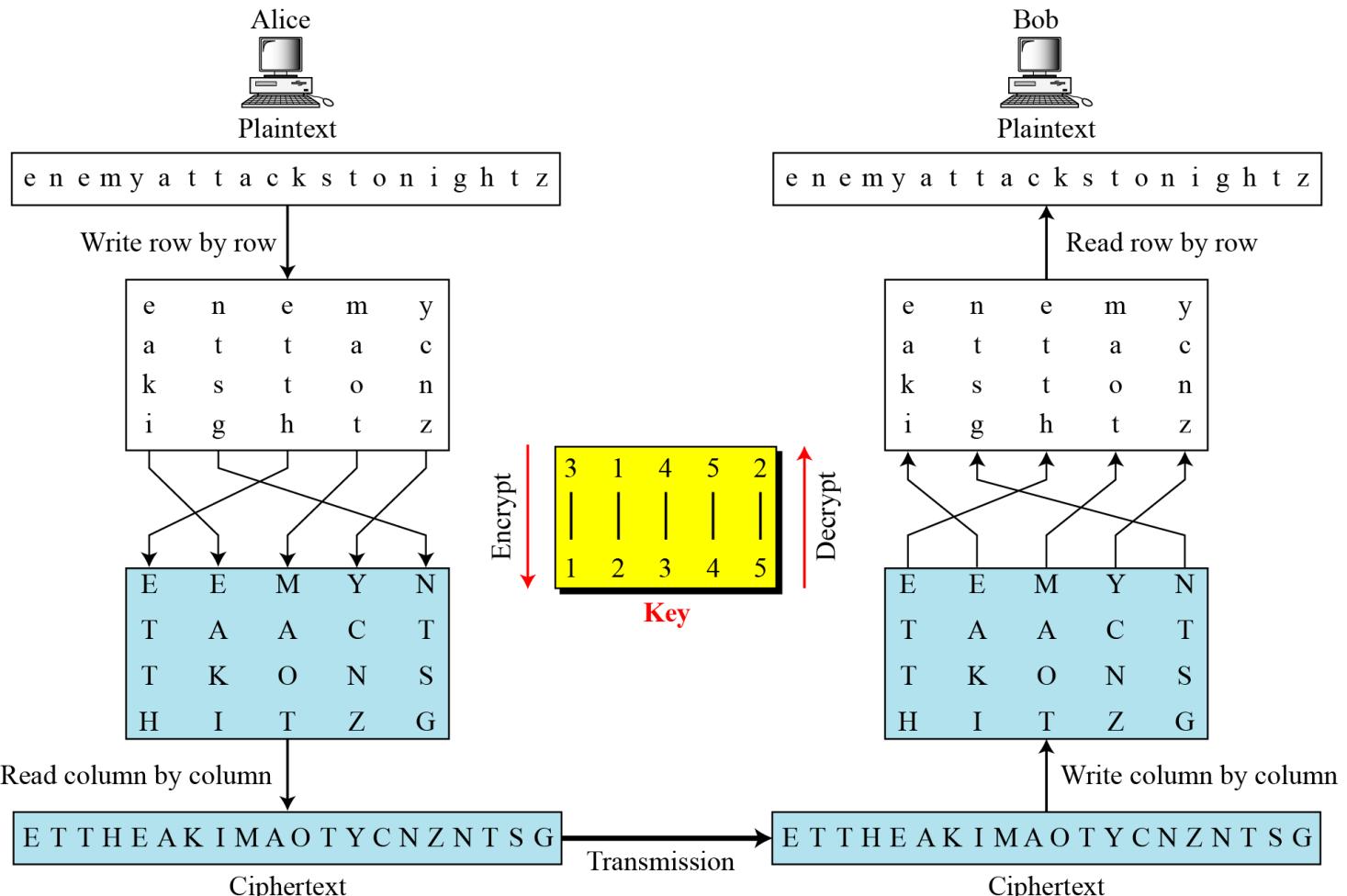
The permutation yields

E E M Y N      T A A C T      T K O N S      H I T Z G

### 3.3.3 Combining Two Approaches: Mã kết hợp chuyển vị có/không có khóa

Example 3.26

Figure 3.21



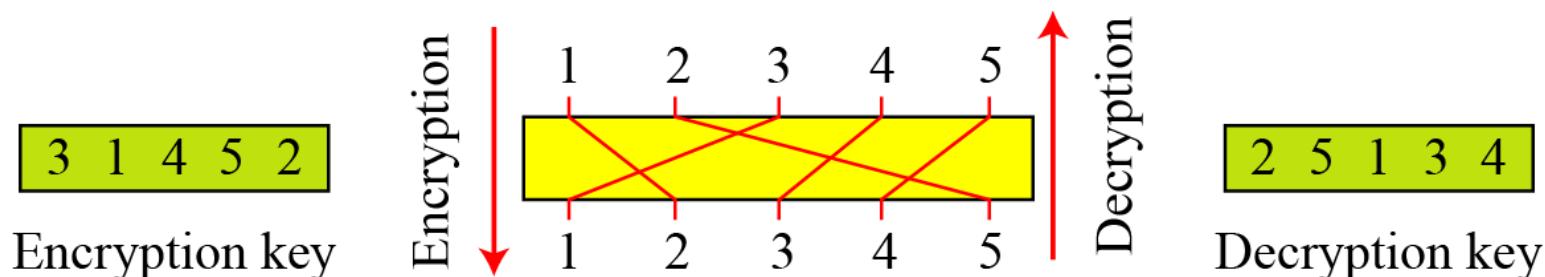
### 3.3.3 *Continued*

#### Keys

In Example 3.21, a single key was used in two directions for the column exchange: downward for encryption, upward for decryption. It is customary to create two keys.

*Một khóa duy nhất đã được sử dụng theo hai hướng để trao đổi cột: hướng xuống để mã hóa, hướng lên để giải mã. Theo thông lệ, bạn nên tạo hai khóa.*

**Figure 3.22** Encryption/decryption keys in transpositional ciphers



### 3.3.3 *Continued*

**Figure 3.23** Đảo khóa trong mật mã chuyển vị

Encryption key

2 6 3 1 4 7 5

2 6 3 1 4 7 5 Add index

1 2 3 4 5 6 7

1 2 3 4 5 6 7 Swap Hoán đổi

2 6 3 1 4 7 5

4 1 3 5 7 2 6 Sort Sắp xếp

Decryption key

a. Manual process

Given: EncKey [index]

index  $\leftarrow 1$

while (index  $\leq$  Column)

{

DecKey[EncKey[index]]  $\leftarrow$  index

index  $\leftarrow$  index + 1

}

Return : DecKey [index]

b. Algorithm

### 3.3.3 *Continued*

#### Using Matrices

We can use matrices to show the encryption/decryption process for a transposition cipher.

Chúng ta có thể sử dụng ma trận để biểu thị quá trình mã hóa / giải mã cho mật mã chuyen vị.

#### Example 3.27

Figure 3.24 Biểu diễn khóa dưới dạng ma trận trong mật mã chuyen vị

$$\begin{matrix} & 3 & 1 & 4 & 5 & 2 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \left[ \begin{array}{ccccc} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{array} \right] & \times & \left[ \begin{array}{ccccc} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array} \right] & = & \left[ \begin{array}{ccccc} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{array} \right] \\ \text{Plaintext} & & \text{Encryption key} & & \text{Ciphertext} \end{matrix}$$

### 3.3.3 *Continued*

#### Example 3.27

Figure 3.24 shows the encryption process. Multiplying the  $4 \times 5$  plaintext matrix by the  $5 \times 5$  encryption key gives the  $4 \times 5$  ciphertext matrix.

Hình 3.24 cho thấy quá trình mã hóa. Nhân ma trận bản rõ  $4 \times 5$  với khóa mã hóa  $5 \times 5$  sẽ cho ma trận mã hóa  $4 \times 5$ .

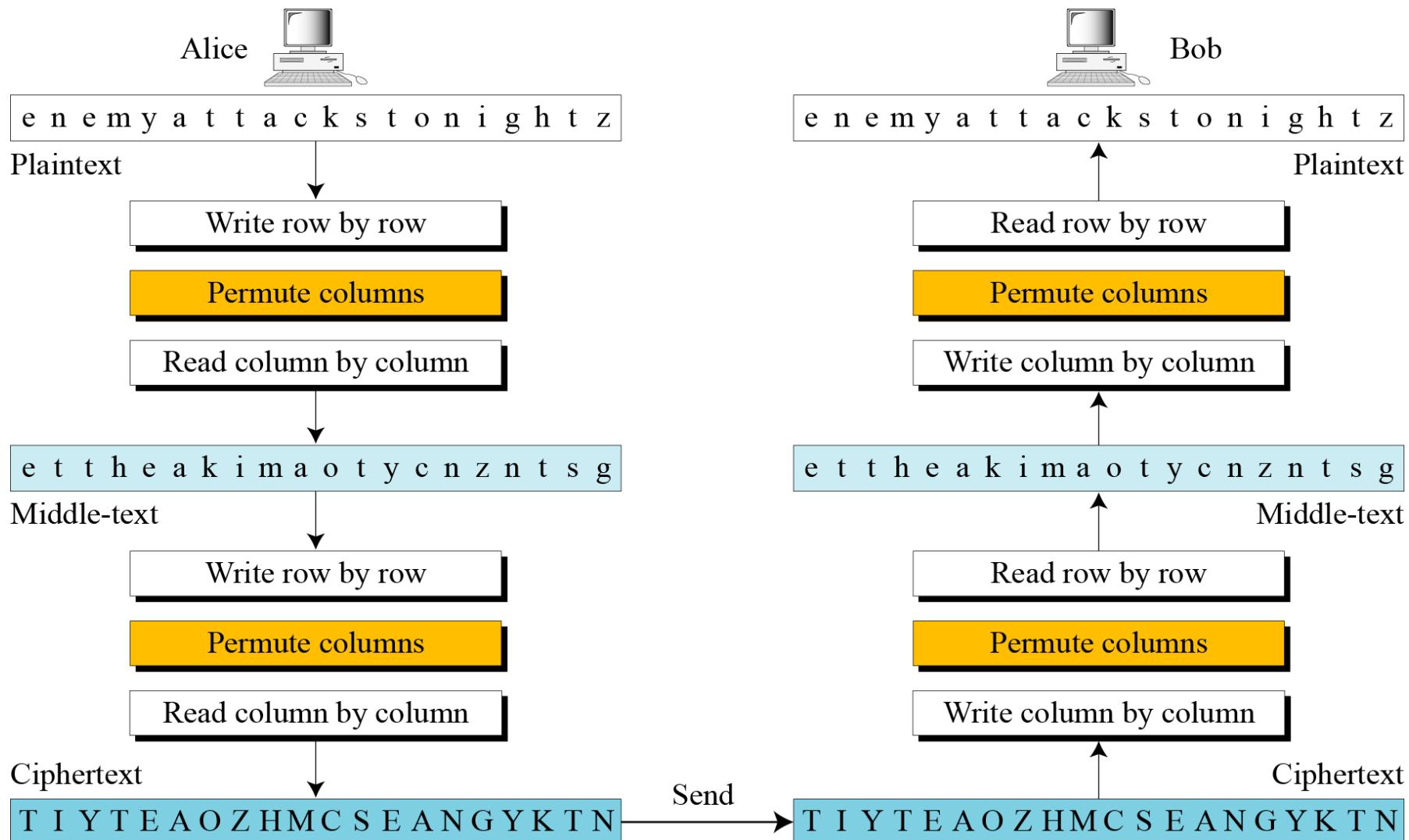
Figure 3.24 Biểu diễn khóa dưới dạng ma trận trong mật mã chuyển vị

$$\begin{bmatrix} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{bmatrix}_{\text{Plaintext}} \times \begin{bmatrix} 3 & 1 & 4 & 5 & 2 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}_{\text{Encryption key}} = \begin{bmatrix} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{bmatrix}_{\text{Ciphertext}}$$

### 3.3.3 *Continued*

## Double Transposition Ciphers

Figure 3.25 Mật mã chuyển vị kép



## 3-4 STREAM AND BLOCK CIPHERS

*Tài liệu chia mật mã đối xứng thành hai loại lớn: mật mã dòng và mật mã khối. Mặc dù các định nghĩa thường được áp dụng cho mật mã hiện đại, cách phân loại này cũng áp dụng cho mật mã truyền thống.*

### *Topics discussed in this section:*

- 3.4.1 Stream Ciphers**
- 3.4.2 Block Ciphers**
- 3.4.3 Combination**

### 3.4.1 Stream Ciphers

Call the plaintext stream  $P$ , the ciphertext stream  $C$ , and the key stream  $K$ .

$$P = P_1 P_2 P_3, \dots$$

$$C = C_1 C_2 C_3, \dots$$

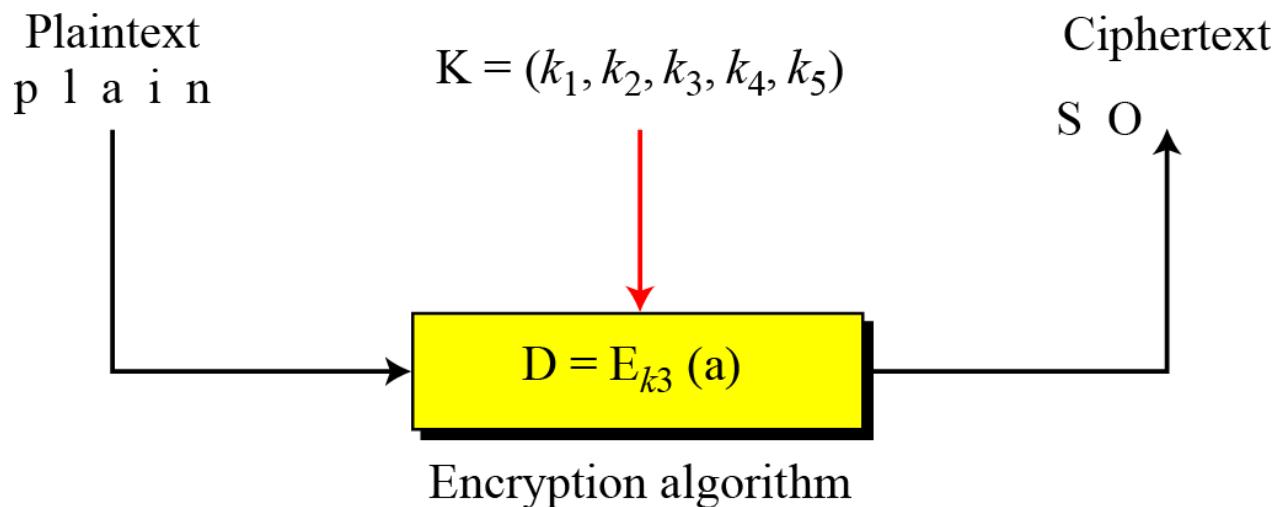
$$K = (k_1, k_2, k_3, \dots)$$

$$C_1 = E_{k1}(P_1)$$

$$C_2 = E_{k2}(P_2)$$

$$C_3 = E_{k3}(P_3) \dots$$

**Figure 3.26 Stream cipher**



### 3.4.1 *Continued*

#### Example 3.30

Mật mã cộng có thể được phân loại là mật mã dòng trong đó dòng khóa là giá trị lặp lại của khóa. Nói cách khác, dòng khóa được coi là một dòng khóa được xác định trước hoặc  $K = (k, k, \dots, k)$ . Tuy nhiên, trong mật mã này, mỗi ký tự trong bản mã chỉ phụ thuộc vào ký tự tương ứng trong bản rõ, vì dòng khóa được tạo ra một cách độc lập.

#### Example 3.31

Các mật mã thay thế đơn ký tự được thảo luận trong chương này cũng là mật mã dòng. Tuy nhiên, mỗi giá trị của dòng khóa trong trường hợp này là ánh xạ của ký tự bản rõ hiện tại với ký tự bản mã tương ứng trong bảng ánh xạ.

## 3.4.1 *Continued*

### Example 3.32

Mật mã Vigenere cũng là mật mã dòng theo định nghĩa. Trong trường hợp này, dòng khóa là sự lặp lại của m giá trị, trong đó m là kích thước của từ khóa. Nói cách khác,

$$K = (k_1, k_2, \dots, k_m, k_1, k_2, \dots, k_m, \dots)$$

### Example 3.33

Chúng ta có thể thiết lập một tiêu chí để phân chia mật mã dòng dựa trên các dòng chính của chúng. Chúng ta có thể nói rằng mật mã dòng là mật mã đơn ký tự nếu giá trị của ki không phụ thuộc vào vị trí của ký tự bản rõ trong dòng bản rõ; nếu không, mật mã là đa ký tự.

### 3.4.1 *Continued*

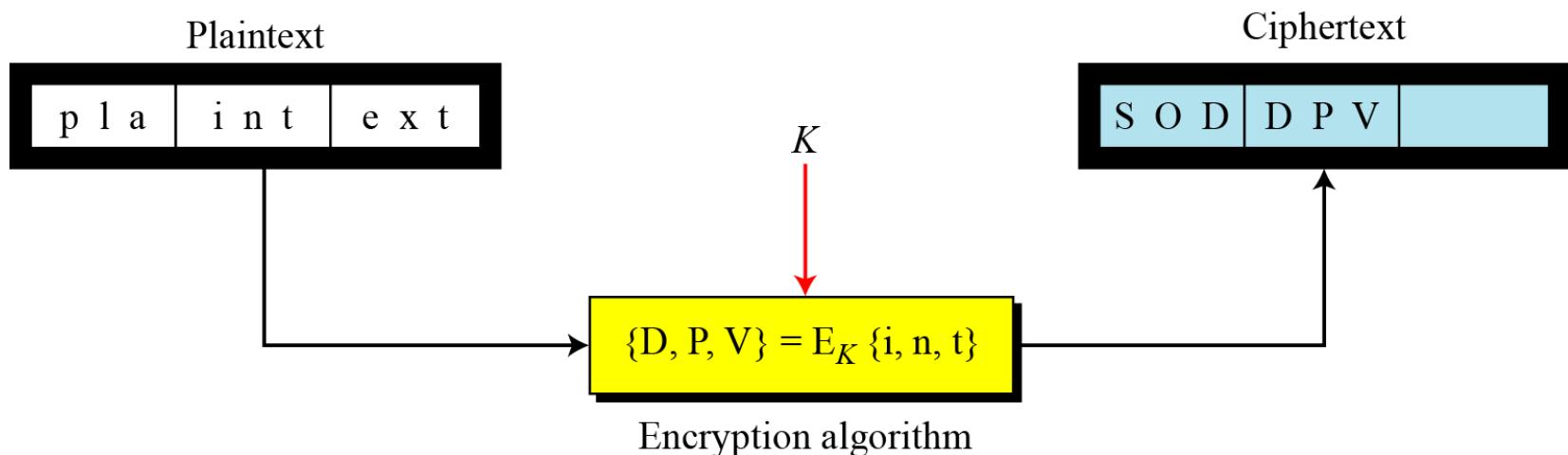
#### Example 3.33 (Continued)

- Additive ciphers are definitely monoalphabetic because  $k_i$  in the key stream is fixed; it does not depend on the position of the character in the plaintext.
- Monoalphabetic substitution ciphers are monoalphabetic because  $k_i$  does not depend on the position of the corresponding character in the plaintext stream; it depends only on the value of the plaintext character.
- Vigenere ciphers are polyalphabetic ciphers because  $k_i$  definitely depends on the position of the plaintext character. However, the dependency is cyclic. The key is the same for two characters  $m$  positions apart.

### 3.4.2 Block Ciphers

Trong mật mã khối, một nhóm các ký hiệu bản rõ có kích thước m ( $m > 1$ ) được mã hóa cùng nhau tạo ra một nhóm bản mã có cùng kích thước. Một khóa duy nhất được sử dụng để mã hóa toàn bộ khối ngay cả khi khóa được tạo từ nhiều giá trị. Hình 3.27 cho thấy khái niệm về mật mã khối.

Figure 3.27 Block cipher



## 3.4.2 *Continued*

### Example 3.34

Mật mã Playfair là mật mã khối. Kích thước của khối là  $m = 2$ . Hai ký tự được mã hóa cùng nhau.

### Example 3.35

Mật mã Hill là mật mã khối. Một khối văn bản rõ, có kích thước từ 2 trở lên được mã hóa cùng nhau bằng một khóa duy nhất (ma trận). Trong các mật mã này, giá trị của mỗi ký tự trong bản mã phụ thuộc vào tất cả các giá trị của các ký tự trong bản rõ. Mặc dù khóa được tạo ra từ các giá trị  $m \times m$ , nó được coi là một khóa duy nhất.

### Example 3.36

Từ định nghĩa về mật mã khối, rõ ràng là mọi mật mã khối đều là mật mã đa ký tự vì mỗi ký tự trong khối bản mã ( $C$ ) phụ thuộc vào tất cả các ký tự trong khối bản rõ ( $P$ ).

### **3.4.3 Combination**

In practice, blocks of plaintext are encrypted individually, but they use a stream of keys to encrypt the whole message block by block. In other words, the cipher is a block cipher when looking at the individual blocks, but it is a stream cipher when looking at the whole message considering each block as a single unit.

*Trong thực tế, các khối văn bản rõ được mã hóa riêng lẻ, nhưng chúng sử dụng một luồng các khóa để mã hóa toàn bộ thông điệp theo từng khối. Nói cách khác, mật mã là một mật mã khối khi nhìn vào các khối riêng lẻ, nhưng nó là một mật mã dòng khi xem xét toàn bộ thông điệp - coi mỗi khối là một đơn vị duy nhất.*

# **Thuật toán mật mã khoá đối xứng hiện đại**

## Nội dung

- ❑ Phân biệt giữa mật mã khoá đối xứng cổ điển và hiện đại.
- ❑ Giới thiệu mật mã khối hiện đại và đặc điểm của chúng.
- ❑ Giải thích lý do tại sao các thuật toán mật mã khối hiện đại cần được thiết kế như mật mã thay thế (substitution ciphers).
- ❑ Giới thiệu các thành phần của mật mã khối (block ciphers) như hộp P và hộp S.

## Nội dung

- Thảo luận về mật mã tích (product ciphers) và phân biệt giữa hai loại: mã hoá Feistel và không-Feistel.
- Thảo luận về hai loại tấn công được thiết kế đặc biệt cho mật mã khối hiện đại: thám mã tuyến tính và vi sai (differential and linear cryptanalysis).
- Giới thiệu mật mã dòng (stream ciphers) và phân biệt giữa mật mã dòng đồng bộ và không đồng bộ.
- Thảo luận về thanh ghi dịch hồi tiếp tuyến tính và không tuyến để thực hiện các thuật toán mã dòng.

# 5-1 MẬT MÃ KHỐI HIỆN ĐẠI

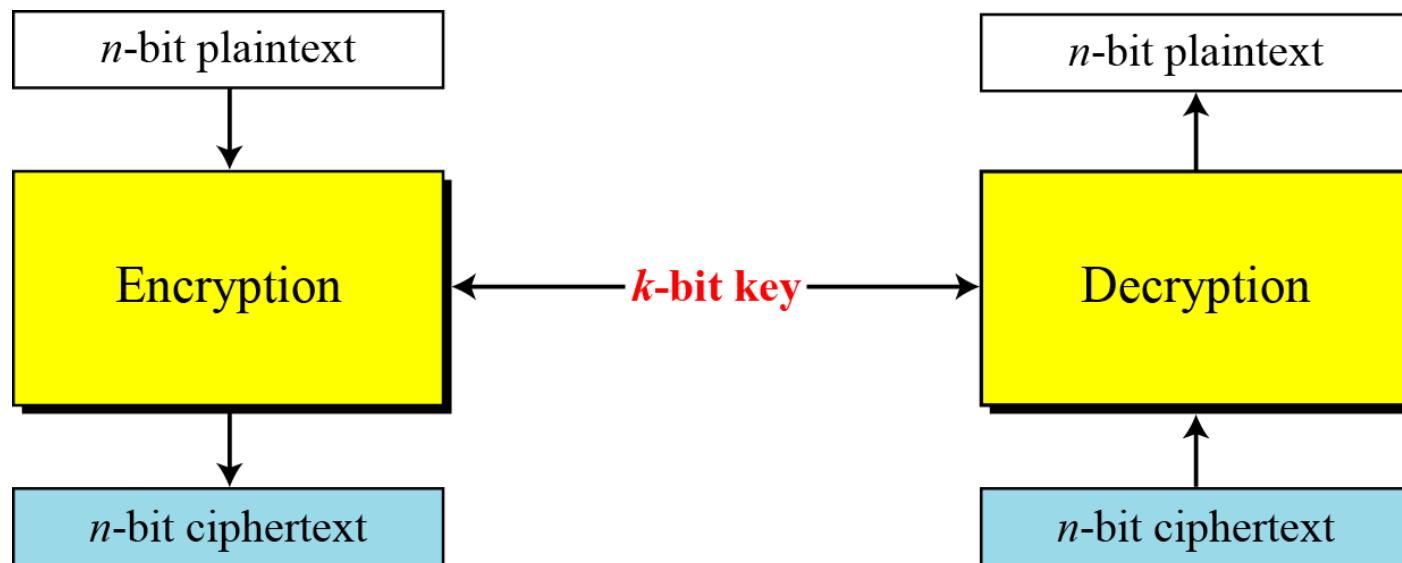
*Mật mã khối hiện đại khóa đối xứng thực hiện mật mã hóa khối **n-bit** **bản rõ** hoặc giải mật mã khối n-bit của bản mật. Các thuật toán mật mã hóa / giải mật mã đều dùng một khóa k-bit.*

## Nội dung:

- 5.1.1** Thay thế, chuyển vị (Substitution or Transposition)
- 5.1.2** Mật mã khối là các nhóm hoán vị (Block Ciphers as Permutation Groups)
- 5.1.3** Thành phần của mật mã khối hiện đại
- 5.1.4** Hệ mật tích (Product Ciphers)
- 5.1.5** Phân loại hệ mật tích (Two Classes of Product Ciphers)
- 5.1.6** Tấn công trong mật mã khối (Attacks on Block Ciphers)

## 5.1 Tiếp...

**Hình 5.1 Mô hình mã khôi hiện đại**



## 5.1.1 Thay thế, chuyển vị

Một mật mã khôi hiện đại có thể được thiết kế để hoạt động như là một mật mã **thay thế** hoặc một **mật mã chuyển vị**.

### Note

Để chống lại **tấn công tìm kiếm đầy đủ**, một mật mã khôi hiện đại cần phải được thiết kế như một **mật mã thay thế**.

## 5.1.1 Tiếp...

### Ví dụ 5.2

Giả sử rằng chúng ta có một mật mã khối  $n = 64$ . Nếu có 10 bit 1 trong bản mã, có bao nhiêu thử nghiệm lỗi nào mà Eve cần làm để khôi phục lại bản rõ từ bản mật bị chặn trong mỗi trường hợp sau?

- a. Mật mã được thiết kế như là một mật mã thay thế?
- b. Mật mã được thiết kế như là một mật mã chuyển vị?

### Giải

- a. Trong trường hợp đầu tiên, Eve không biết có bao nhiêu số 1 trong bản rõ. Eve cần thử tất cả  $2^{64}$  khối 64-bit có thể có để tìm một cái có nghĩa.
- b. Trong trường hợp thứ hai, Eve biết rằng có chính xác 10 số 1 trong bản rõ. Eve có thể khởi chạy một tấn công tìm kiếm toàn diện bằng cách sử dụng những khối 64-bit có chính xác 10 số 1.

## 5.1.2 Mã khối là các nhóm hoán vị

*Mã khối chuyển vị khóa kích thước đầy đủ*

*Chúng ta có thể có  $n!$  khóa, khi đó mỗi khóa sẽ có độ dài  $\lceil \log_2 n! \rceil$  bits.*

### Ví dụ 5.3

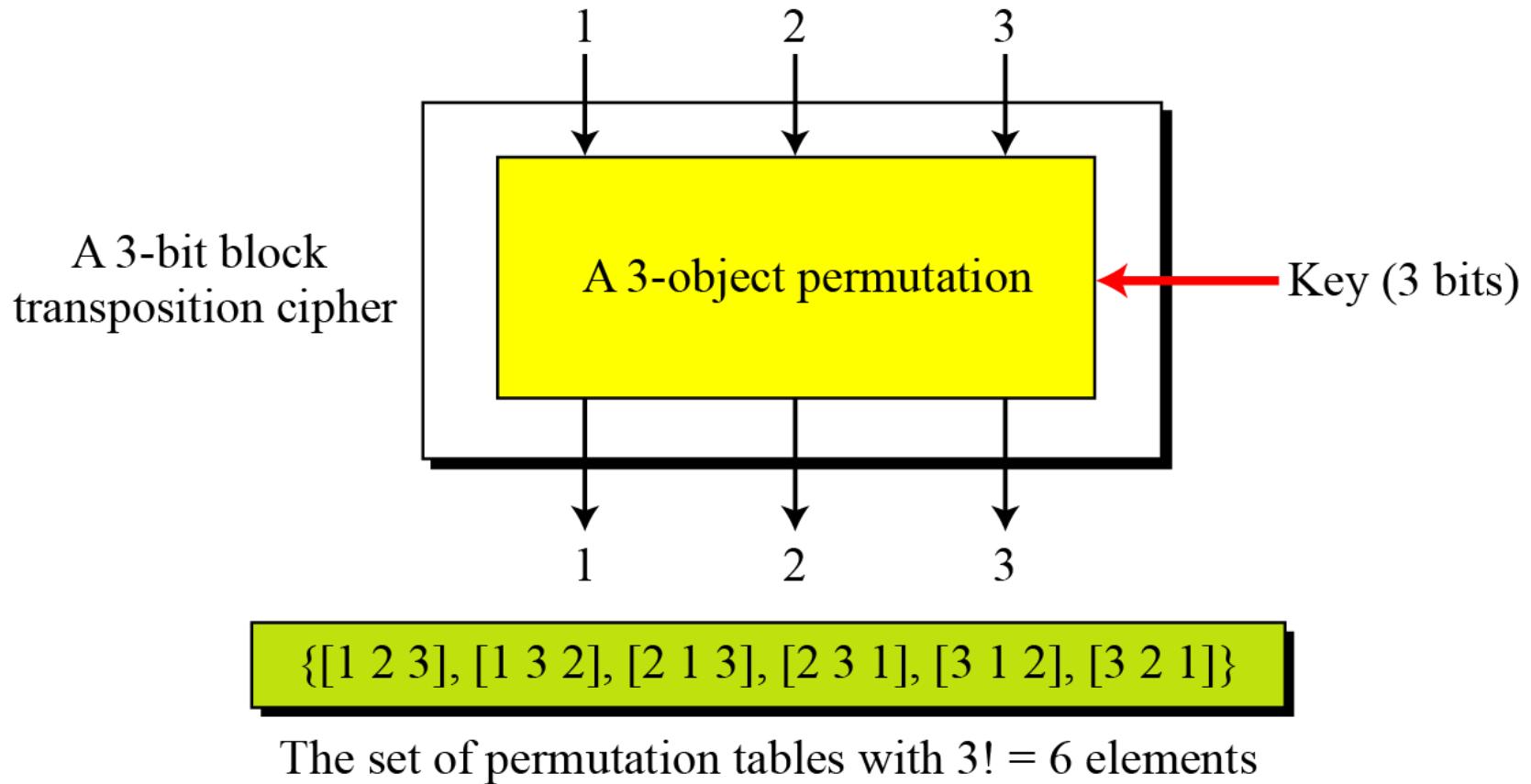
Chỉ ra mô hình và tập các bảng hoán vị cho mã chuyển vị khối 3-bit (mỗi khối có kích thước 3 bits)

### Giải

Tập các bảng là  $3! = 6$  elements, như sau:

## 5.1.2 Tiếp...

**Hình 5.2** *Mã khối chuyen vị được mô hình như là phép hoán vị*



## 5.1.2 Tiếp...

*Mã khối chuyển vị khóa kích thước đầy đủ*

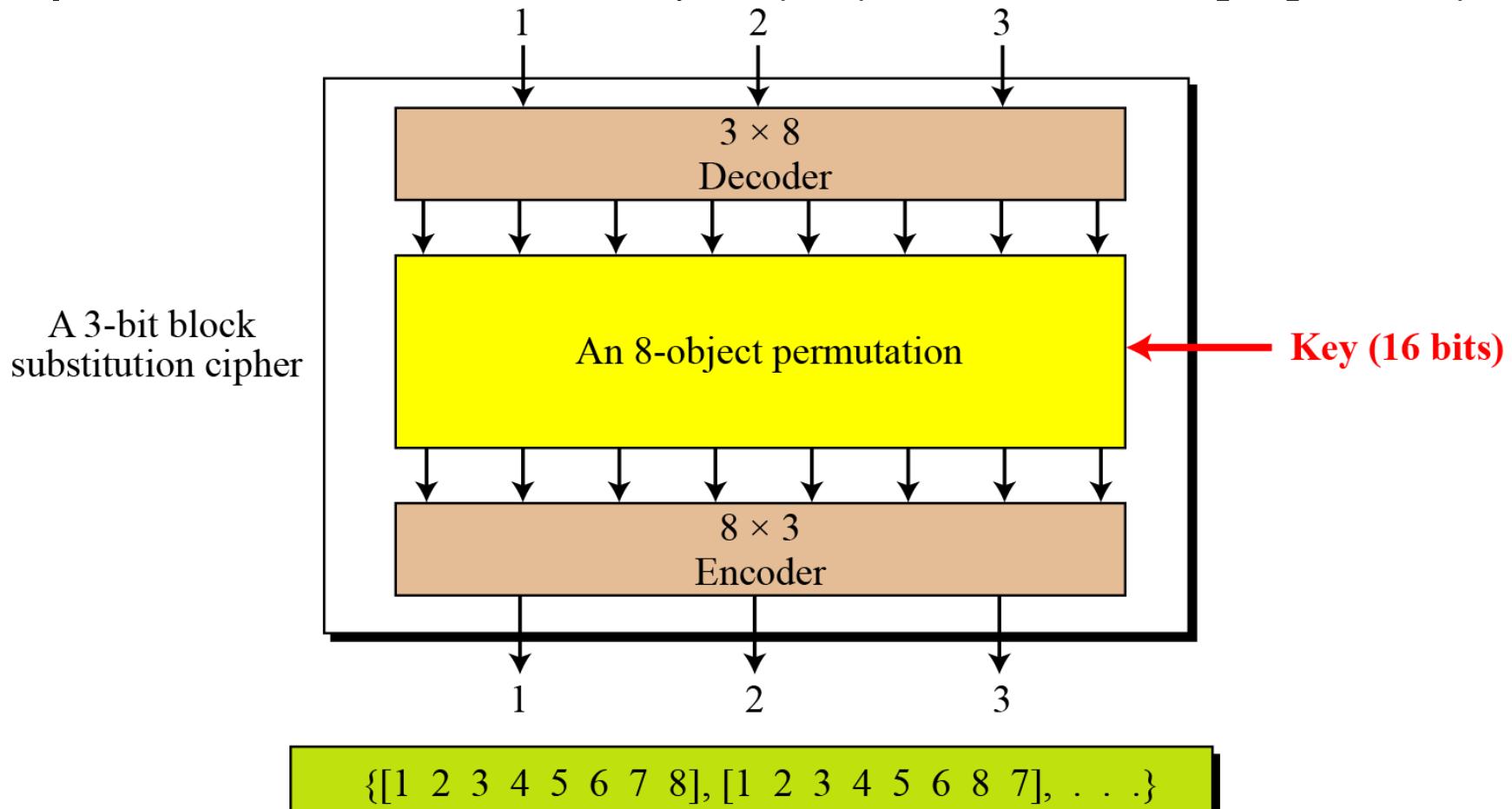
*Mã thay thế khoá kích thước đầy đủ không chuyển đổi các bit; nó thay thế các bit. Chúng ta có thể mô hình hóa mật mã thay thế như một phép hoán vị nếu có thể giải mã đầu vào và mã hoá đầu ra.*

### Ví dụ 5.4

**Cho biết mô hình và tập các bảng hoán vị cho mã hóa thay thế khối 3-bit**

## 5.1.2 Tiếp...

Hình 5.3 Mã khối chuyen vị được mô hình như là phép hoán vị



The set of permutation tables with  $8! = 40,320$  elements

Hình 5.3 sau chỉ ra tập các bảng hoán vị. Khóa có độ dài  $\lceil \log_2 40,320 \rceil = 16$  bits.

## 5.1.2 Tiếp...

**Note**

A full-size key  $n$ -bit transposition cipher or a substitution block cipher can be modeled as a permutation, but their key sizes are different:

- Transposition: the key is  $\lceil \log_2 n! \rceil$  bits long.
- Substitution: the key is  $\lceil \log_2(2^n)! \rceil$  bits long.

**Note**

Một mật mã khóa một phần là một nhóm trong phép toán kết hợp nếu nó là một phân nhóm của mật mã khoá kích thước đầy đủ tương ứng.

### 5.1.3 Thành phần của mã khối hiện đại

Mã khoá khối hiện đại thường là các m<sup>át</sup> m<sup>ã</sup> thay thế khoá, trong đó khóa có thể chỉ cho phép ánh xạ một phần từ đầu vào tới đầu ra.

*Modern block ciphers normally are keyed substitution ciphers in which the key allows only partial mappings from the possible inputs to the possible outputs.*

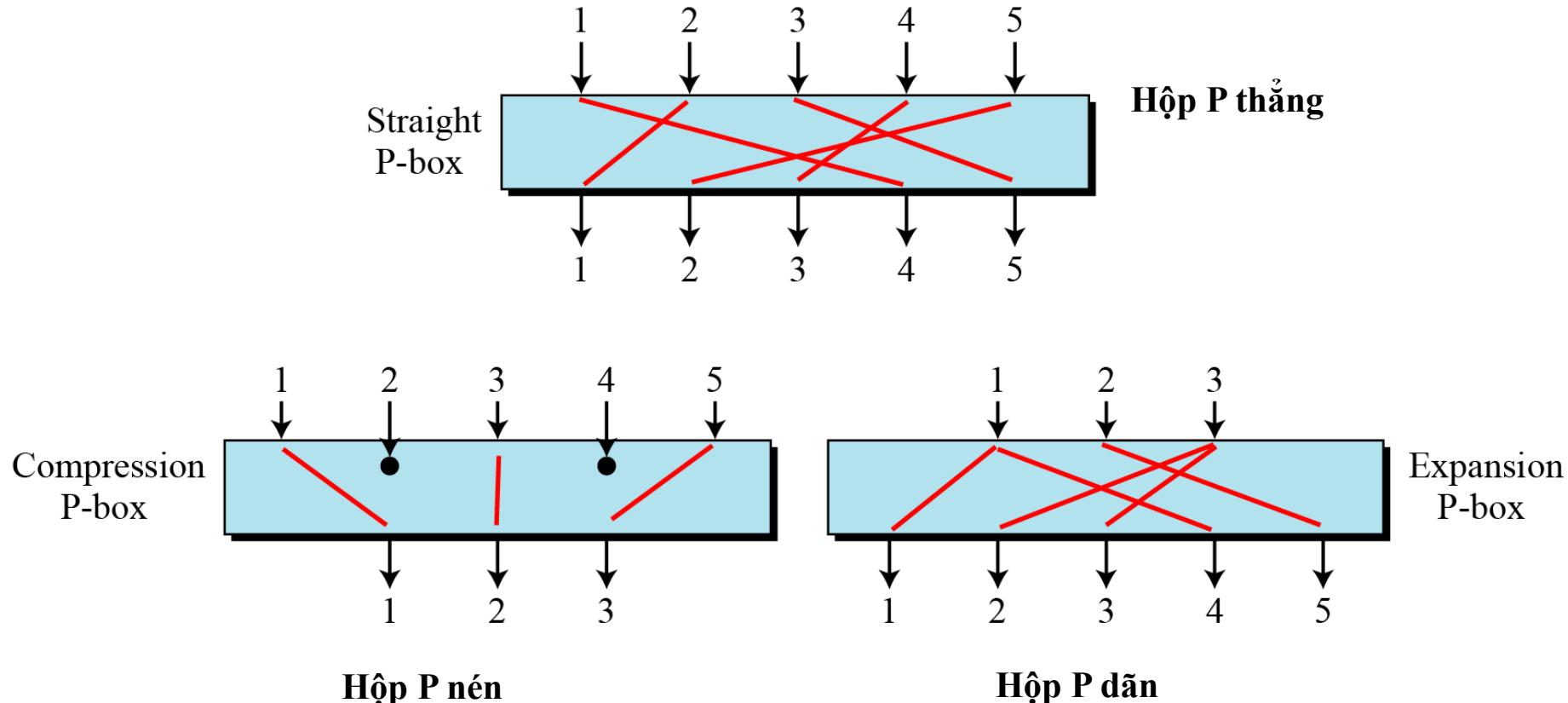
#### P-Boxes

**P-box** (hộp hoán vị) tương đương với m<sup>át</sup> m<sup>ã</sup> chuyen đổi truyền thống cho các ký tự. Nó chuyen vị các bit.

A P-box (permutation box) parallels the traditional transposition cipher for characters. It transposes bits.

## 5.1.3 Tiếp...

Hình 5.4 3 loại P-boxes

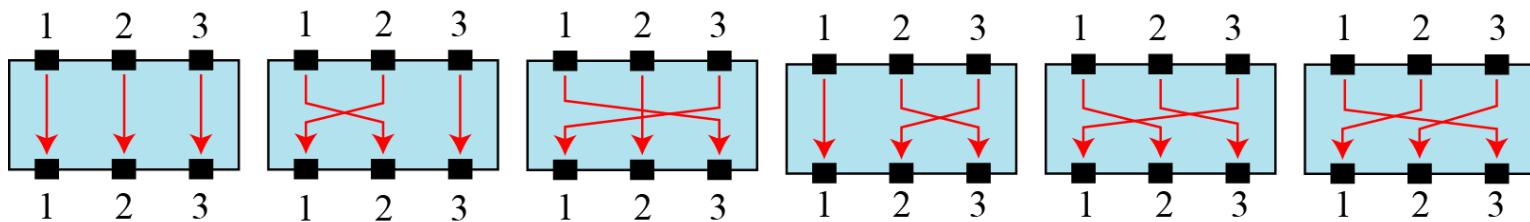


## 5.1.3 Tiếp...

### Ví dụ 5.5

Hình 5.5 chỉ ra 6 trường hợp ánh xạ của hộp  $3 \times 3$  P-box.

**Hình 5.5** *The possible mappings of a  $3 \times 3$  P-box*



## 5.1.3 Tiếp...

### Straight P-Boxes

Bảng 5.1 *Example of a permutation table for a straight P-box*

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

## 5.1.2 Tiếp...

### Ví dụ 5.6

Design an  $8 \times 8$  permutation table for a straight P-box that moves the two middle bits (bits 4 and 5) in the input word to the two ends (bits 1 and 8) in the output words. Relative positions of other bits should not be changed.

### Giải

We need a straight P-box with the table [4 1 2 3 6 7 8 5]. The relative positions of input bits 1, 2, 3, 6, 7, and 8 have not been changed, but the first output takes the fourth input and the eighth output takes the fifth input.

## 5.1.3 Tiếp...

### Compression P-Boxes

*Hộp nén P là hộp P với n đầu vào và đầu ra m, trong đó  $m < n$ .*

*A compression P-box is a P-box with n inputs and m outputs where  $m < n$ .*

**Bảng 5.2 Example of a  $32 \times 24$  permutation table**

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

## 5.1.3 *Tiếp...*

### Compression P-Box

**Bảng 5.2** *Example of a  $32 \times 24$  permutation table*

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

## 5.1.3 Tiếp...

### Expansion P-Boxes

*Hộp mở rộng P là hộp P với n đầu vào và đầu ra m, trong đó  $m > n$ .*

*An expansion P-box is a P-box with n inputs and m outputs where  $m > n$ .*

**Bảng 5.3** *Example of a  $12 \times 16$  permutation table*

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

## **5.1.3 Tiết...**

### **P-Boxes: Tính khả nghịch**

**Note**

**Hộp P-box có tính khả nghịch, tuy nhiên hộp P-boxes nén và mở rộng thì không.**

## 5.1.3 Tiếp...

### Ví dụ 5.7

Hình 5.6 cho thấy làm thế nào để đảo ngược một bảng hoán vị đại diện như một bảng một chiều.

**Hình 5.6** *Inverting a permutation table*

1. Original table  
Bảng gốc

6	3	4	5	2	1
---	---	---	---	---	---

2. Add indices  
Thêm chỉ mục

6	3	4	5	2	1
1	2	3	4	5	6

3. Swap contents  
and indices  
Tráo đổi nội dung  
và chỉ mục

1	2	3	4	5	6
6	3	4	5	2	1

4. Sort based  
on indices  
Sắp xếp dựa vào  
chỉ mục

6	5	2	3	4	1
1	2	3	4	5	6

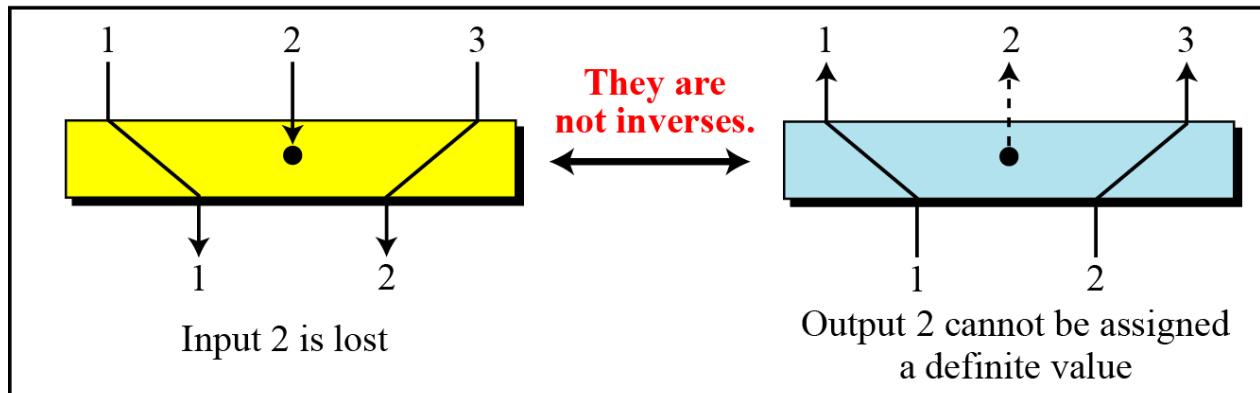
6	5	2	3	4	1
---	---	---	---	---	---

5. Inverted table  
Bảng đảo ngược

## 5.1.3 Tiếp...

**Hình 5.7 Các hộp nén và mở rộng không có tính đảo ngược**  
**Compression and expansion P-boxes are non-invertible**

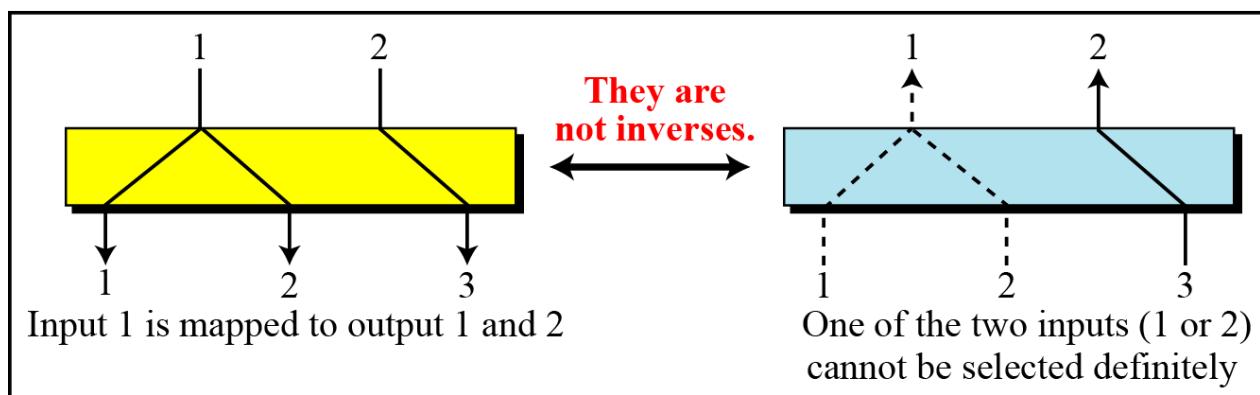
Compression P-box



They are  
not inverses.

Output 2 cannot be assigned  
a definite value

Expansion P-box



They are  
not inverses.

One of the two inputs (1 or 2)  
cannot be selected definitely

## 5.1.3 Tiếp...

### S-Box

*Hộp S-box (hộp thay thế) có thể được coi như một mảng thay thế thu nhỏ.*

*An S-box (substitution box) can be thought of as a miniature substitution cipher.*

#### Note

**An S-box is an  $m \times n$  substitution unit, where  $m$  and  $n$  are not necessarily the same.**

### 5.1.3 Tiếp...

#### Ví dụ 5.8

Trong một S-box với ba đầu vào và hai đầu ra, chúng ta có

$$y_1 = x_1 \oplus x_2 \oplus x_3 \quad y_2 = x_1$$

Hộp S-box là tuyến tính vì  $a_{1,1} = a_{1,2} = a_{1,3} = a_{2,1} = 1$  và  $a_{2,2} = a_{2,3} = 0$ .

Mỗi quan hệ có thể được **biểu diễn** bằng ma trận:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

## 5.1.3 Tiếp...

### Ví dụ 5.9

Trong một hộp S với ba đầu vào và hai đầu ra, chúng ta có

$$y_1 = (x_1)^3 + x_2 \quad y_2 = (x_1)^2 + x_1 x_2 + x_3$$

trong đó phép nhân và phép cộng nằm trong GF (2).

Hộp S-box là phi tuyến vì không có mối quan hệ tuyến tính giữa đầu vào và đầu ra.

### 5.1.3 Tiếp...

#### Ví dụ 5.10

**Bảng dưới đây xác định mối quan hệ đầu vào / đầu ra cho hộp S-box có kích thước  $3 \times 2$ .**

**Phần bit ngoài cùng bên trái của đầu vào xác định hàng; hai bit ngoài cùng bên phải của đầu vào xác định cột. Hai bit đầu ra là các giá trị trên mặt cắt ngang của hàng và cột được chọn.**

Leftmost  
bit

Rightmost  
bits

	00	01	10	11
0	00	10	01	11
1	10	00	11	01

Output bits

**Dựa trên bảng này, một đầu vào của 010 cho ra kết quả đầu ra 01. Một đầu vào của 101 cho kết quả đầu ra là 00.**

## 5.1.3 Tiếp...

### S-Boxes: Invertibility

*Hộp S-box có thể có hoặc không thể đảo ngược được. Trong hộp S-box đảo ngược, số bit đầu vào bằng với số bit đầu ra.*

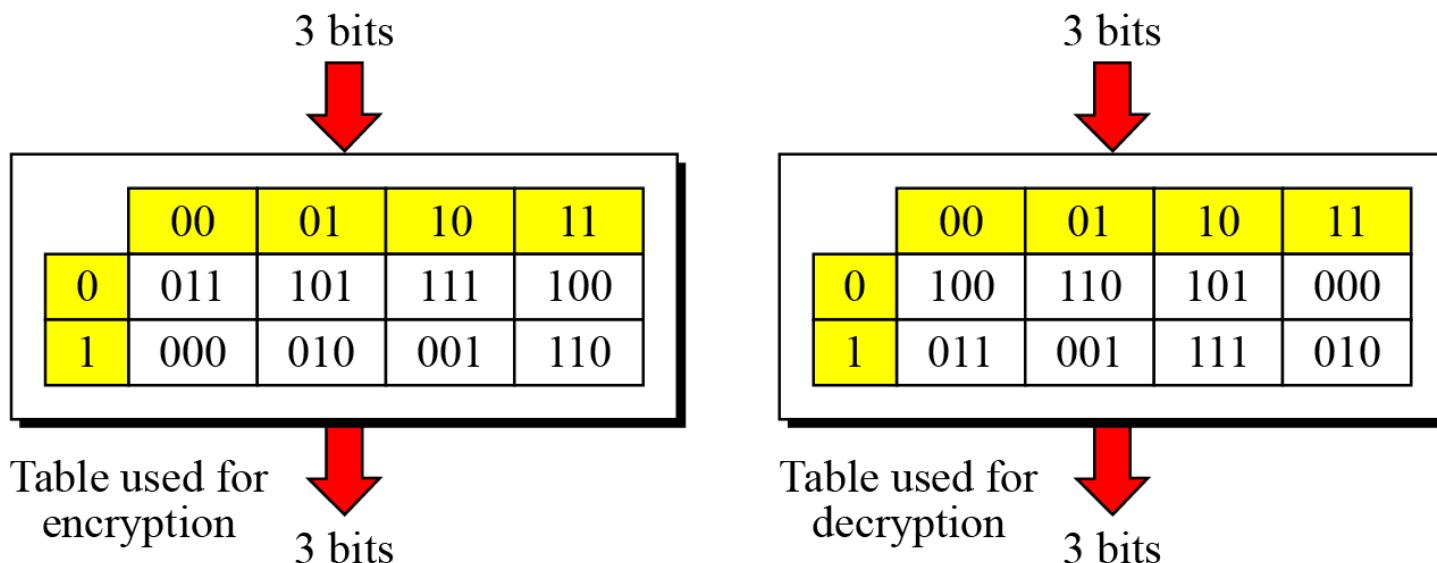
*An S-box may or may not be invertible. In an invertible S-box, the number of input bits should be the same as the number of output bits.*

## 5.1.3 Tiếp...

### Ví dụ 5.11

Hình 5.8 cho thấy một ví dụ về một hộp S-nghịch đảo. Ví dụ, nếu đầu vào cho ô bên trái là 001, đầu ra là 101. Đầu vào 101 trong bảng bên phải tạo ra 001, trong đó cho thấy hai bảng là đảo ngược của nhau.

**Hình 5.8 bảng S-box trong Ví dụ 5.11**

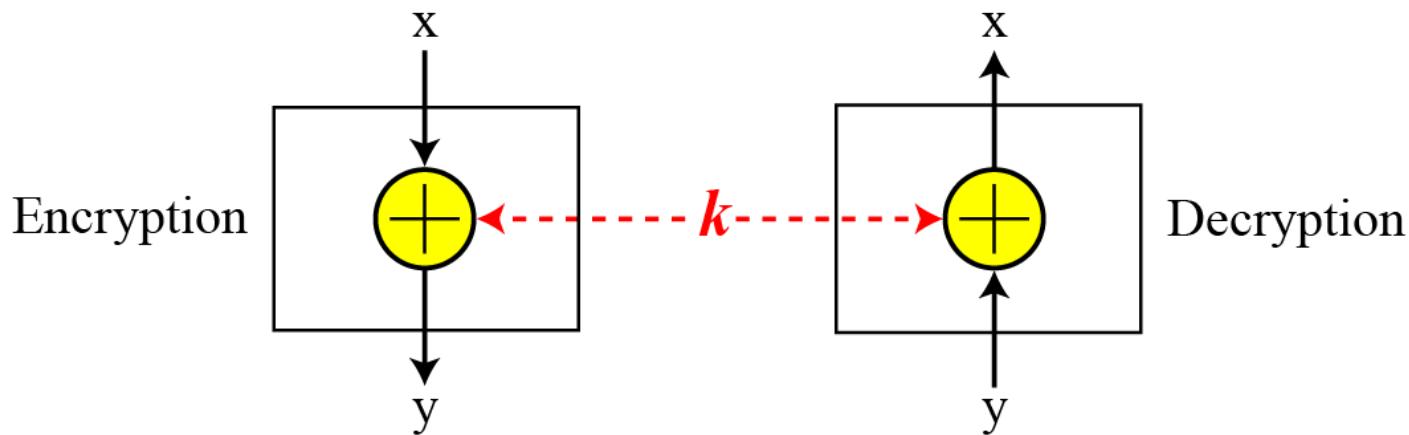


## 5.1.3 Tiếp...

### Exclusive-Or

Một thành phần quan trọng trong hầu hết các mật mã khối là phép toán XOR.

Hình 5.9 Tính thuận nghịch của toán tử XOR



- ✓ XOR là 1 toán tử nhị phân (tức là toán tử hai ngôi có 2 tham số - giống như phép cộng).
- ✓ Theo tên của nó, exclusive - OR, nó dễ dàng để suy ra (đúng, hoặc sai). Bảng chân lý cho toán tử XOR.

A	B	A XOR B
T	T	F
T	F	T
F	T	T
F	F	F

### 5.1.3 Tiếp...

#### Exclusive-Or (tiếp...)

Như chúng ta đã thảo luận trong chương 4, phép cộng và phép trừ trong trường  $GF(2^n)$  được thực hiện bằng một phép toán đơn gọi là XOR.

Năm tính chất của phép XOR trong trường  $GF(2^n)$  làm cho phép toán này trở thành một thành phần khá quan trọng trong việc thực hiện mật mã khối: đóng, kết hợp, giao hoán, tồn tại tính duy nhất và tồn tại nghịch đảo.

### 5.1.3 Tiếp...

#### Exclusive-Or (tiếp...)

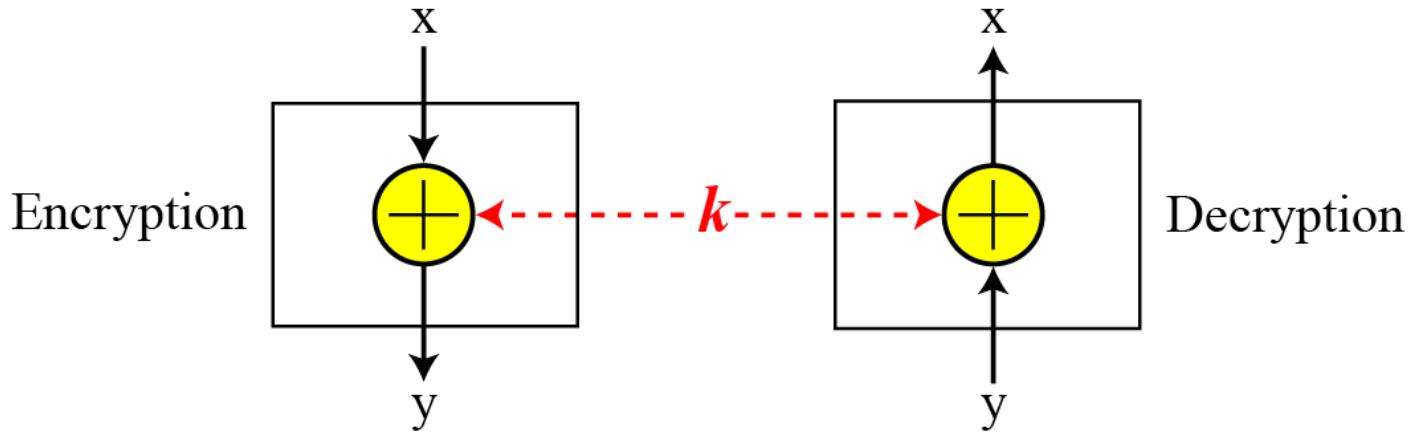
Nghịch đảo một thành phần trong mật mã có ý nghĩa nếu thành phần đại diện cho một phép toán đơn (một đầu vào và một đầu ra). Ví dụ, một hộp không khóa P-box hoặc hộp S-box có thể được thực hiện nghịch đảo bởi chúng có một đầu vào và một đầu ra. Một phép XOR là một toán tử nhị phân. **Phép nghịch đảo của XOR chỉ có ý nghĩa nếu một trong các đầu vào là cố định (giống nhau trong mã hóa và giải mã).**

Ví dụ, nếu một trong các đầu vào là khoá, thông thường là giống nhau trong **mã hóa và giải mã**, khi đó phép XOR là **tự nghịch đảo**, như thể hiện trong hình 5.9.

## 5.1.1 Tiếp...

Hình 5.9 *Invertibility of the exclusive-or operation*

Nghịch đảo phép toán XOR



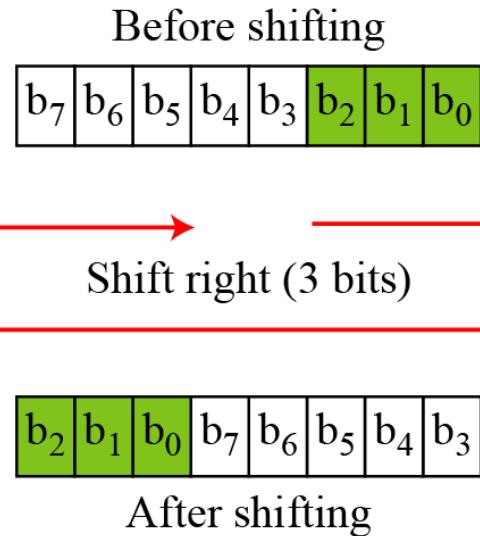
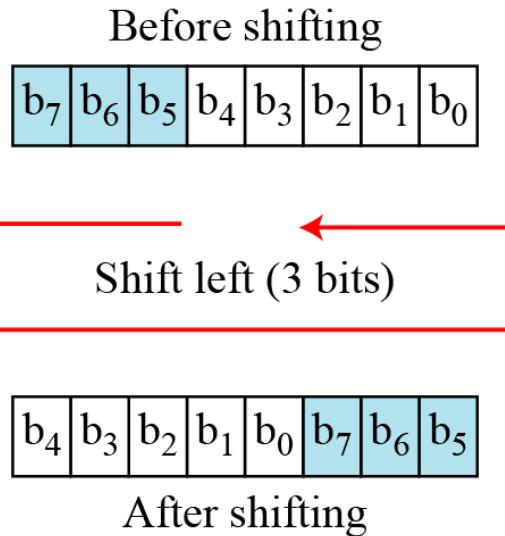
## 5.1.3 Tiếp...

### Dịch vòng tròn - Circular Shift

Một thành phần khác trong một số thuật toán mật mã khôi hiện đại là phép toán dịch chuyển tròn.

Hình 5.10 Dịch vòng từ mã 8-bit sang trái hoặc phải

$n (=8)$ :  
số lượng  
bit trong  
một từ  
(word)  
 $k (=3)$  số  
lần dịch,  
là số có  
định  
được cho  
trước.



Toán tử dịch  
vòng trái hoặc  
phải là có tính  
nghịch đảo:  
phép dịch trái  
dùng cho mã  
thì dịch phải  
dùng cho giải  
mã hóa

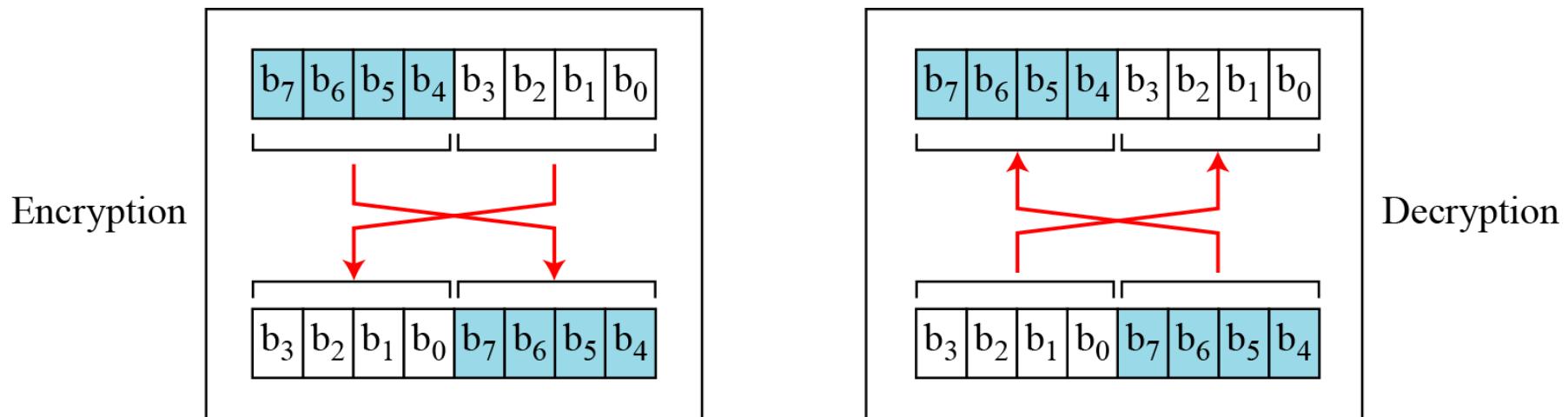
Toán tử dịch  
vòng thực  
hiện trong  
modulo  $n$ . Vì  
nếu  $k=0$  hoặc  
 $k=n$  tức là  
không có phép  
dịch. Nếu  $k > n$   
thì phép dịch  
là  $k \bmod n$  bit

## 5.1.3 Tiếp...

### Hoán đổi - Swap

*Thuật toán hoán đổi là một trường hợp đặc biệt của phép dịch chuyển vòng tròn khi  $k = n / 2$  (n chẵn).*

**Hình 5.11** Phép toán hoán đổi cho từ mã 8-bit

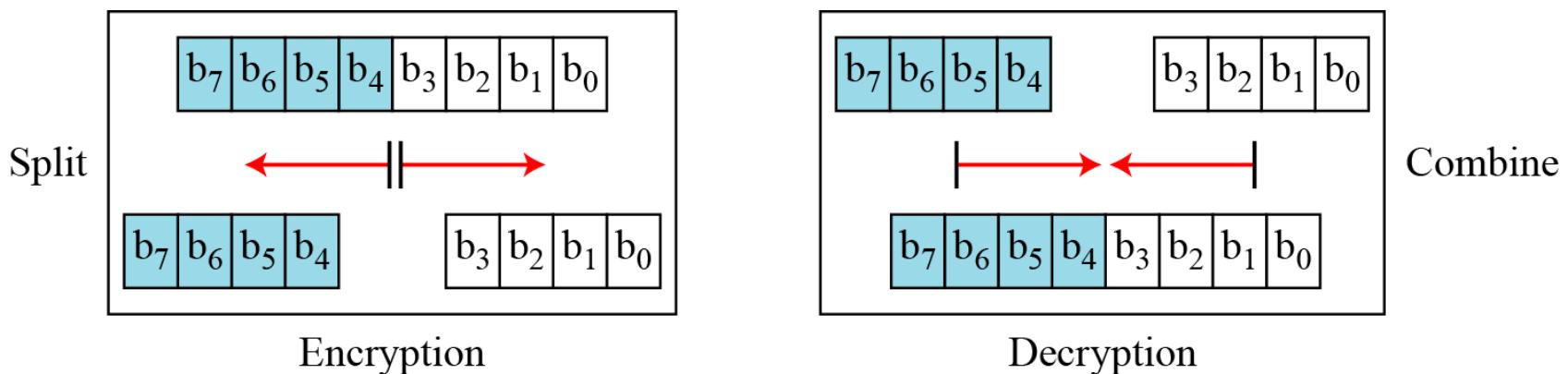


## 5.1.3 Tiếp...

### Chia và Kết hợp - Split and Combine

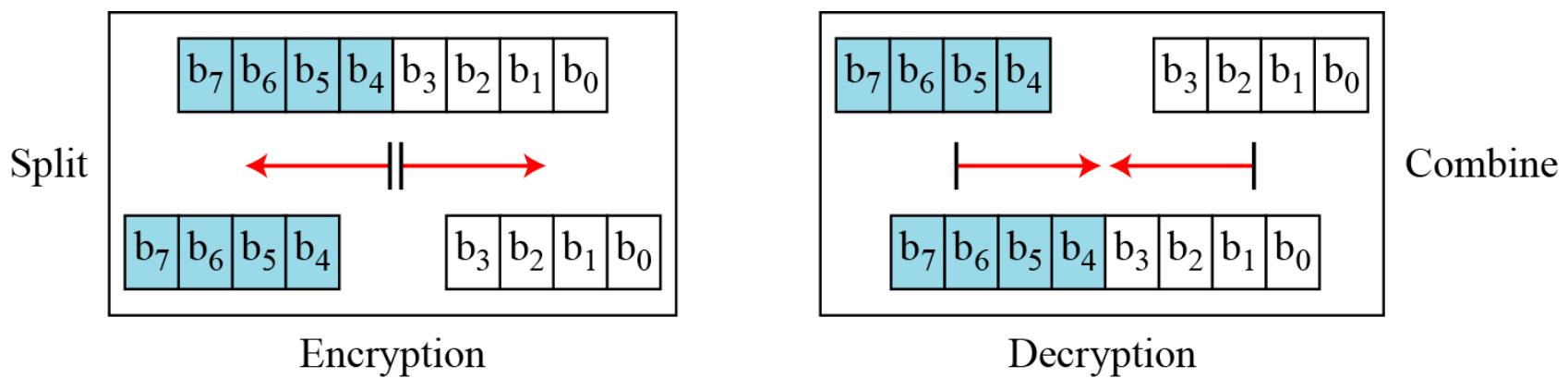
*Hai phép toán khác trong một số mật mã khối là chia và kết hợp. Nếu phép chia dùng cho mã hóa thì phép kết hợp dùng cho giải mã hóa và ngược lại.*

**Hình 5.12** *Phép toán chia và kết hợp trong từ mã 8-bit*



## 5.1.3 Tiếp...

Hình 5.12 Phéo toán chia và kết hợp trong từ mã 8-bit



## 5.1.4 Mật mã tích - Product Ciphers

Shannon giới thiệu khái niệm về mật mã tích. Mã hóa tích là một mật mã phức tạp kết hợp sự **thay thế, hoán vị** và các thành phần khác như đã được thảo luận trong các phần trước.

Hai đặc điểm quan trọng đối với mã khôi này là **sự diffusion và confusion**

## 5.1.4 Tiếp...

### Khuếch tán (diffusion)

Ý tưởng của khuếch tán là để ẩn giấu mối quan hệ giữa bản mã và bản rõ.

Tức là nếu một bit trong bản tin plaintext thay đổi thì nhiều hoặc tất cả bit trong bản tin ciphertext bị thay đổi.

*The idea of diffusion is to hide the relationship between the ciphertext and the plaintext.*

#### Note

Diffusion hides the relationship between the ciphertext and the plaintext.

## 5.1.4 Tiếp...

### Nhầm lẫn (Confusion)

Ý tưởng của sự nhầm lẫn là để giấu mối quan hệ giữa bản mã và khóa.

Tức là, nếu một bit đơn trong khóa bị thay đổi thì tất cả các bit trong bản tin ciphertext cũng sẽ bị thay đổi.

### Confusion

*The idea of confusion is to hide the relationship between the ciphertext and the key.*

#### Note

**Confusion hides the relationship between the ciphertext and the key.**

## 5.1.4 Tiếp...

### Vòng

*Khuếch tán và nhầm lẫn có thể đạt được bằng cách sử dụng mật mã tích được lắp lại, mỗi lần lắp lại là sự kết hợp của các hộp S-box, P-box và các thành phần khác.*

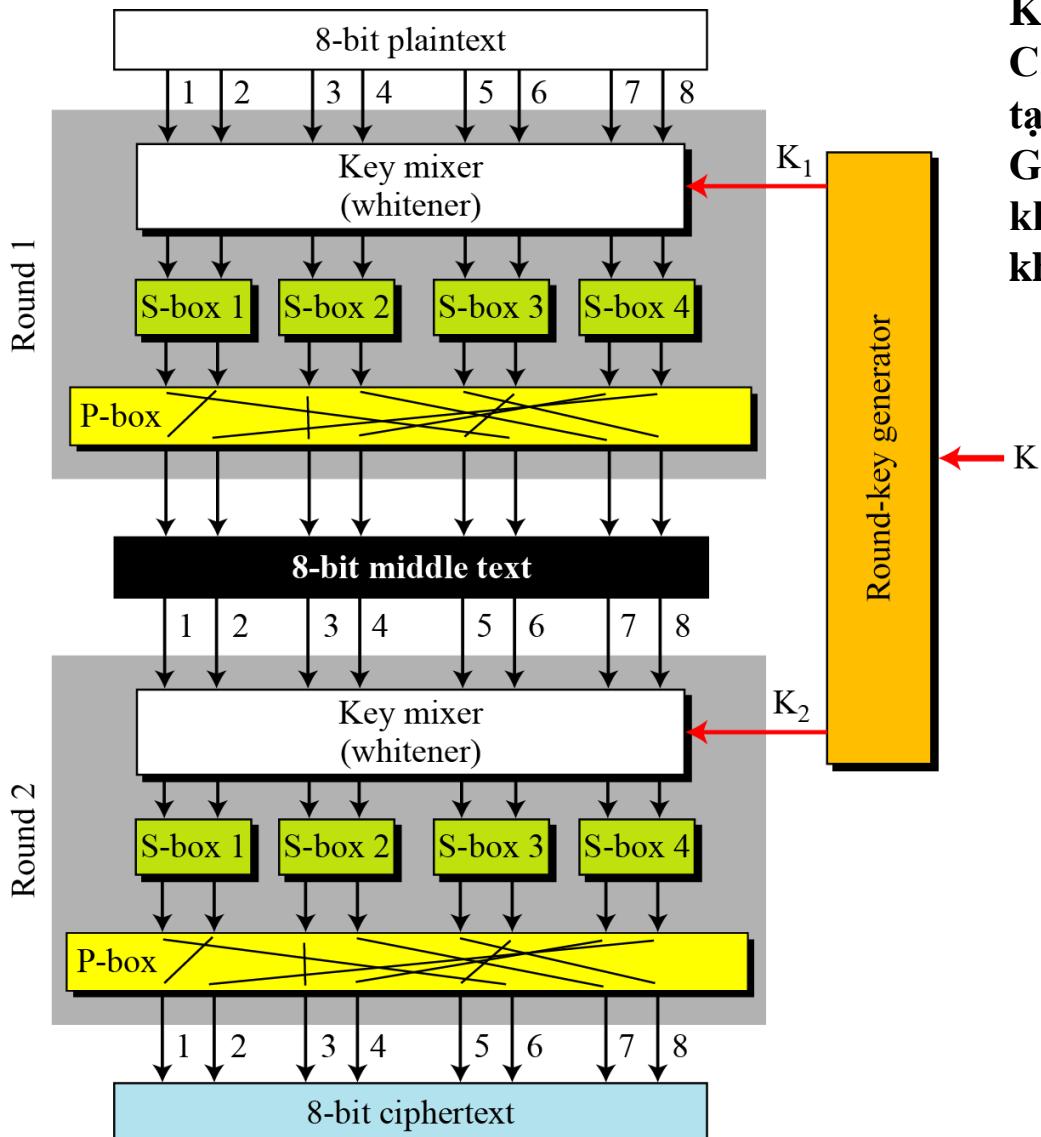
*(Một vòng được thực hiện bằng cách sử dụng kết hợp các phép biến đổi, các thành phần hộp S và hộp P)*

### Rounds

*Diffusion and confusion can be achieved using iterated product ciphers where each iteration is a combination of S-boxes, P-boxes, and other components.*

## 5.1.4 Continued

Hình 5.13 Một mật mã tích tạo ra bởi 2 vòng



### Khóa:

Các khóa trong mỗi vòng được sinh / tạo ra bởi bộ tạo khóa gọi là Key Generator hoặc là Key Schedule. Bộ tạo khóa sẽ sinh cho mỗi vòng một khóa khác nhau.

Trong ví dụ bên: có 3 quá trình chuyển đổi được thực hiện trong mỗi vòng:

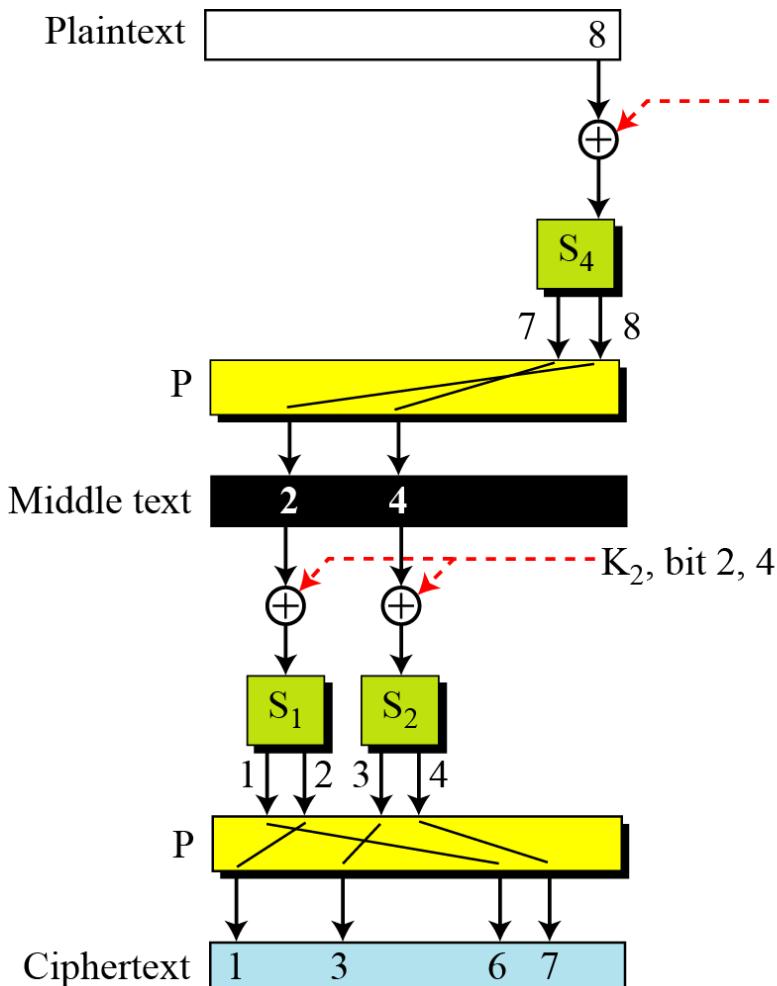
- + 8 bit dữ liệu vào được trộn với khóa  $k_1$  thông qua bộ trộn khóa (Key mixer) để thực hiện làm trắng (whiten) bản tin gốc (ẩn giấu các bit vào bởi khóa  $k_1$ ). Thông thường chúng ta sẽ thực hiện toán tử XOR với từ mã 8bit vào và 8 bit khóa  $k_1$ .

- + 8 bit ra của bộ trên được nhóm thành 2 bit với nhau để đưa vào bốn hộp S. Qua đó, giá trị các bit bị thay đổi do quá trình chuyển đổi của hộp S.

- + 8 bit đầu ra của 4 hộp S chuyển đến một hộp P để hoán vị trí các bit vào / ra tạo nên 8 bit ra của vòng 1. 8 bit này sẽ là đầu vào cho vòng 2 tiếp theo.

## 5.1.4 Continued

Hình 5.14 Khuếch tán và nhầm lẫn trong mật mã khối trong hệ mật mã khối nhân



**Quá trình Khuếch tán:** Khi sử dụng các hộp S và P, quá trình khuếch tán được đảm bảo. Đầu tiên hai bit số 7 và số 8 sau khi XOR với khóa K1 qua bộ S4 .... → ẩn giấu giữa C và P

**Quá trình nhầm lẫn:** bốn bit ra 1,3,6,7 bị tác động bởi các bit 8, 2 ,4 của khóa K1 và K2 → ẩn giấu giữa C và K.

## 5.1.5 Two Classes of Product Ciphers

*Mã khoá khối hiện đại là tất cả các mật mã tích, chúng được chia thành hai lớp.*

*Modern block ciphers are all **product ciphers**, but they are divided into two classes.*

- 1. Mã Feistel (Feistel ciphers), ví dụ hệ mật DES (sử dụng các thành phần có khả tính khả nghịch)**
- 2. Mã không Feistel (Non-Feistel ciphers), ví dụ hệ mật mã AES (**

## 5.1.5 Continued

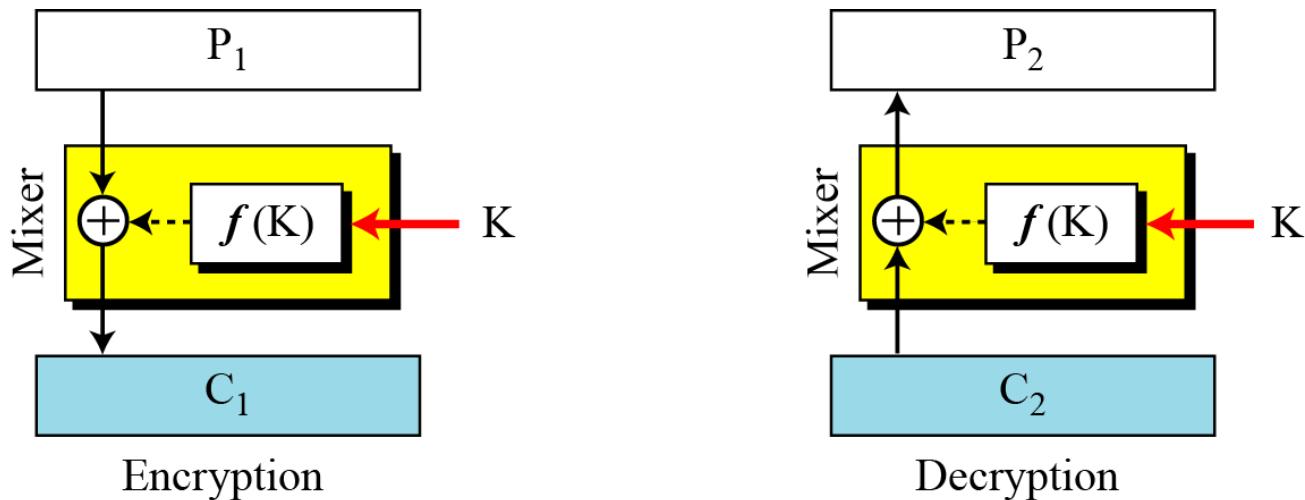
### *Mã Feistel (Feistel Ciphers)*

*Feistel thiết kế một mật mã rất thông minh và thú vị, đã được sử dụng trong nhiều thập kỷ. Một mật mã Feistel có thể có ba loại thành phần: **tự nghịch đảo, nghịch đảo, và không nghịch đảo.***

*Feistel designed a very intelligent and interesting cipher that has been used for decades. A Feistel cipher can have three types of components: **self-invertible, invertible, and noninvertible.***

## 5.1.5 Continued

Hình 5.15 Ý tưởng đầu tiên trong thiết kế mật mã Feistel



$f(K)$  không phải là hàm thuận nghịch, đóng vai trò quan trọng trong mật mã Feistel.  
Khóa  $K$  dùng chung cho mã hóa và giải mã hóa.

Bộ trộn Mixer là quá trình thực hiện phép XOR và hàm  $f(K)$ .

Quá trình mã hóa và giải mã hóa là thuận nghịch: Nếu  $C_1=C_2$  thì  $P_1=P_2$ .

$C_1=P_1 \text{ XOR } f(K)$

$P_2=C_2 \text{ XOR } f(K)=P_1 \text{ XOR } f(K) \text{ XOR } f(K) = P_1 \text{ XOR } (00\dots 0) = P_1$ .

**Note**

**Khuyếch tán ẩn giấu mối quan hệ giữa bản mã và bản rõ.**

## 5.1.3 *Continued*

### Ví dụ 5.12

Bản rõ và bản mã có chiều dài 4 bit và khóa là 3 bit. Giả sử rằng hàm  $f(K)$  có các bit đầu tiên và thứ ba của khóa, biến đổi hai bit này thành số thập phân, sau đó bình phương số đó, và biến đổi kết quả thành một mẫu nhị phân 4 bit.

Hãy cho biết kết quả của việc mã hóa và giải mã nếu bản chính gốc là 0111 và khóa là K=101?

**Giải**

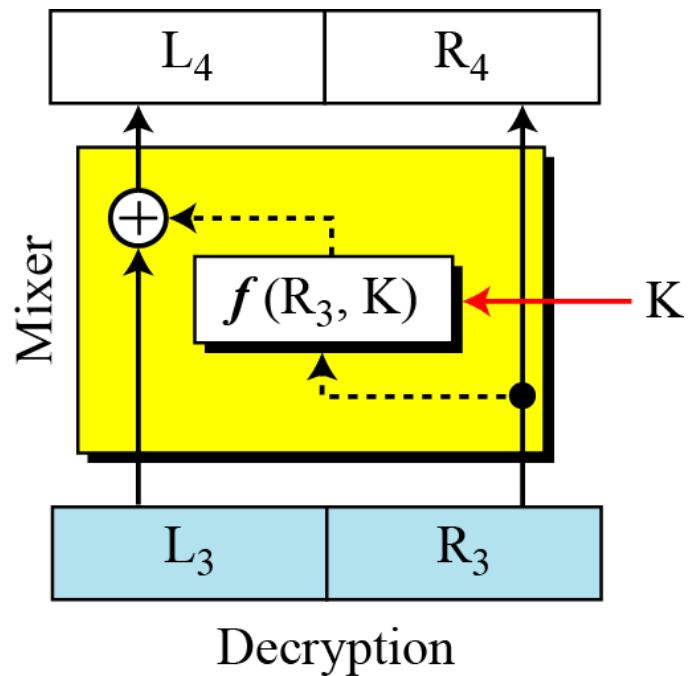
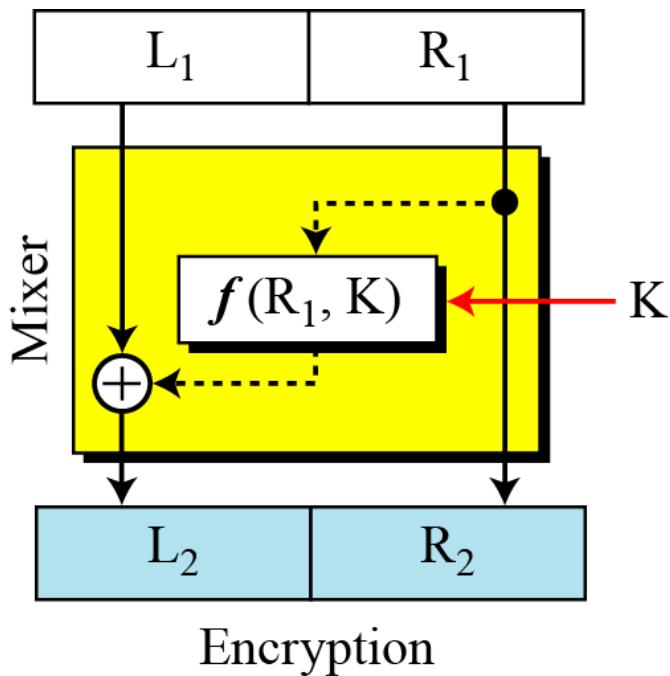
The function extracts the first and second bits to get 11 in binary or 3 in decimal. The result of squaring is 9, which is 1001 in binary.

**Encryption:**  $C = P \oplus f(K) = 0111 \oplus 1001 = 1110$

**Decryption:**  $P = C \oplus f(K) = 1110 \oplus 1001 = 0111$

## 5.1.5 Continued

Hình 5.16 Phát triển thiết kế mã



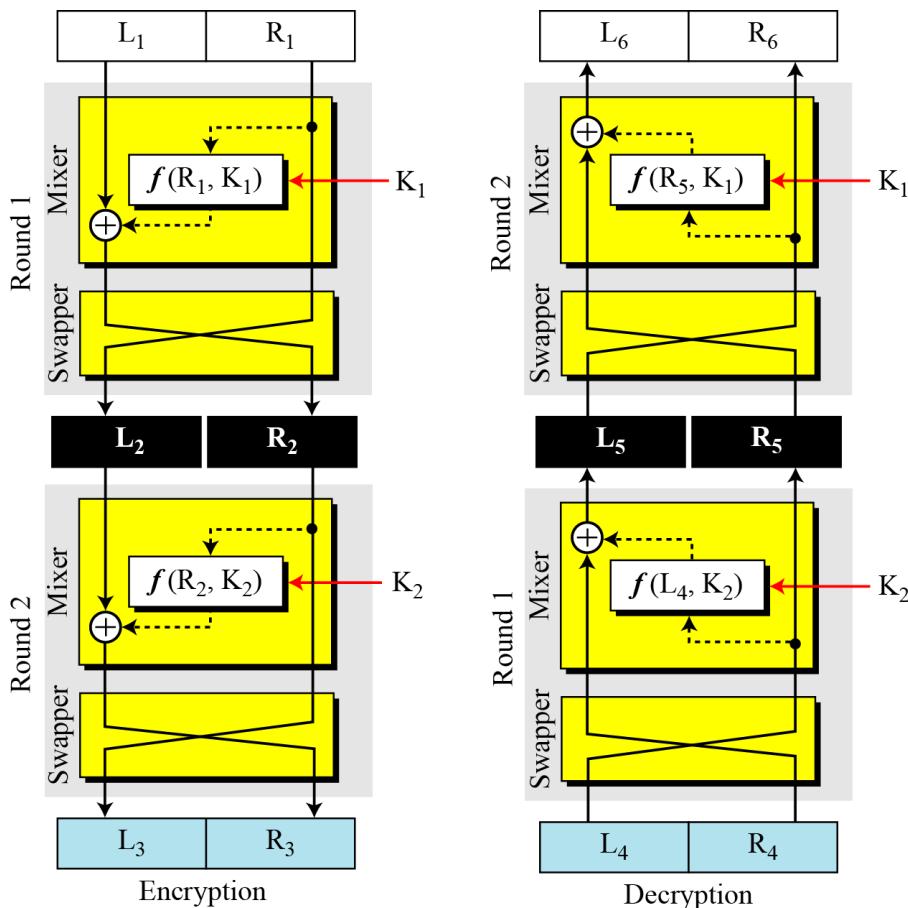
Nếu  $L_3=L_2$  và  $R_3=R_2$ .

$R_4=R_3=R_2=R_1$  (1/2 từ mã bên phải của bản tin  $P$  luôn không bị thay đổi -  
 → Eve dễ dàng tìm được nửa bên phải của bản tin  $P$ ) → cần phải thiết kế  
 thêm bộ hoán đổi (Swapper) cho mỗi vòng.

$$L_4 = L_3 \text{ XOR } f(R_3, K) = L_1 \text{ XOR } f(R_1, K) \text{ XOR } f(R_1, K) = L_1$$

## 5.1.5 Continued

Hình 5.17 Thiết kế nâng cao mã Feistel cuối cùng với 2 vòng



$$L_5 = R_4 \text{ XOR } f(L_4, K_2) = L_2 \text{ XOR } f(R_2, K_2) \text{ XOR } f(L_4, K_2) = L_2$$

$$R_5 = L_4 = L_3 = R_2$$

$$L_6 = R_5 \text{ XOR } f(R_5, K_1) = L_1 \text{ XOR } f(R_1, K_1) \text{ XOR } f(R_5, K_1) = L_1$$

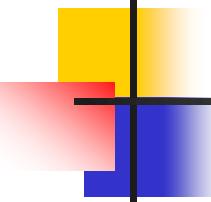
$$R_6 = L_5 = L_2 = R_1$$

## 5.1.5 Continued

### *Mã không Feistrel (Non-Feistel Ciphers)*

*Một mật mã phi Feistel chỉ sử dụng các thành phần có thể ngược đảo. Một thành phần trong mật mã hoá có thành phần tương ứng trong giải mật mã.*

*A non-Feistel cipher uses only invertible components. A component in the encryption cipher has the corresponding component in the decryption cipher.*



## ***5.1.6 Attacks on Block Ciphers***

*Attacks on traditional ciphers can also be used on modern block ciphers, but today's block ciphers resist most of the attacks discussed in Chapter 3.*

## **5.1.5 Continued**

### **Differential Cryptanalysis**

*Eli Biham and Adi Shamir introduced the idea of differential cryptanalysis. This is a chosen-plaintext attack.*

## 5.1.6 *Continued*

### Example 5.13

Assume that the cipher is made only of one exclusive-or operation, as shown in Figure 5.18. Without knowing the value of the key, Eve can easily find the relationship between plaintext differences and ciphertext differences if by plaintext difference we mean  $P_1 \oplus P_2$  and by ciphertext difference, we mean  $C_1 \oplus C_2$ . The following proves that  $C_1 \oplus C_2 = P_1 \oplus P_2$ :

$$C_1 = P_1 \oplus K \quad C_2 = P_2 \oplus K \quad \rightarrow \quad C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

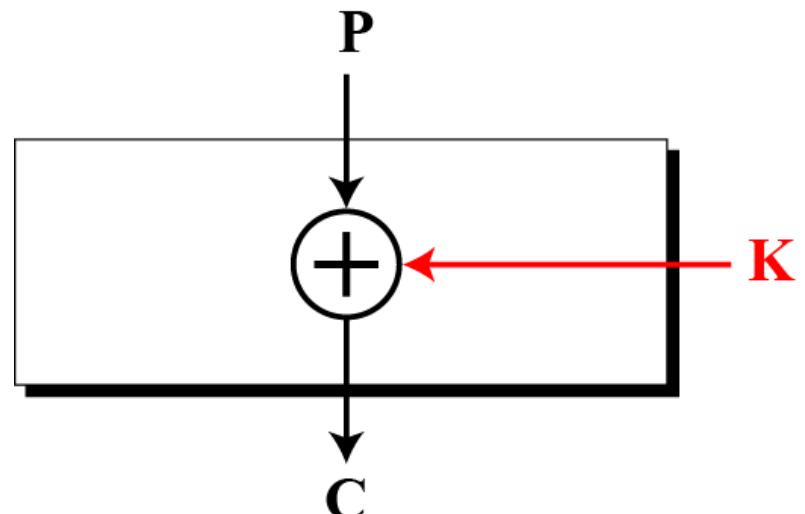


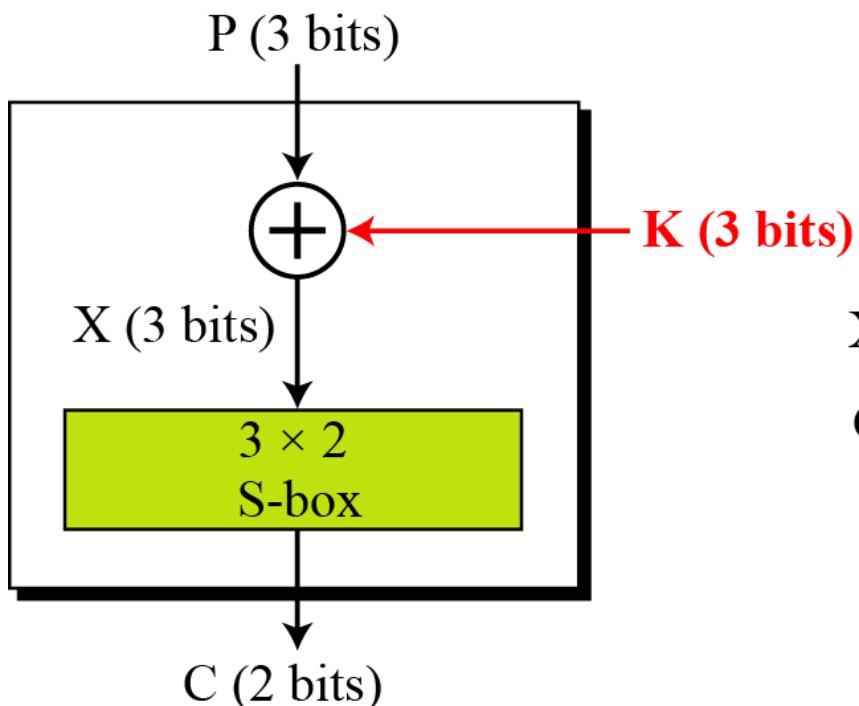
Figure 5.18 *Diagram for Example 5.13*

## 5.1.6 *Continued*

### Example 5.14

We add one S-box to Example 5.13, as shown in Figure 5.19.

**Figure 5.19** *Diagram for Example 5.14*



X	000	001	010	011	100	101	110	111
C	11	00	10	10	01	00	11	00

S-box table

## 5.1.6 Continued

### Example 5.14 Continued

Eve now can create a probabilistic relationship as shown in Table 5.4.

**Table 5.4** Differential input/output

		$C_1 \oplus C_2$			
		00	01	10	11
$P_1 \oplus P_2$	000	8			
	001	2	2		4
	010	2	2	4	
	011		4	2	2
	100	2	2	4	
	101		4	2	2
	110	4		2	2
	111			2	6

## 5.1.6 *Continued*

### Example 5.15

The heuristic result of Example 5.14 can create probabilistic information for Eve as shown in Table 5.5.

**Table 5.5** *Differential distribution table*

		$C_1 \oplus C_2$			
		00	01	10	11
P <sub>1</sub> ⊕ P <sub>2</sub>		000	1	0	0
001		0.25	0.25	0	0.50
010		0.25	0.25	0.50	0
011		0	0.50	0.25	0.25
100		0.25	0.25	0.50	0
101		0	0.50	0.25	0.25
110		0.50	0	0.25	0.25
111		0	0	0.25	0.75

## 5.1.6 *Continued*

### Example 5.16

Looking at Table 5.5, Eve knows that if  $P_1 \oplus P_2 = 001$ , then  $C_1 \oplus C_2 = 11$  with the probability of 0.50 (50 percent). She tries  $C_1 = 00$  and gets  $P_1 = 010$  (chosen-ciphertext attack). She also tries  $C_2 = 11$  and gets  $P_2 = 011$  (another chosen-ciphertext attack). Now she tries to work backward, based on the first pair,  $P_1$  and  $C_1$ ,

$$C_1 = 00 \rightarrow X_1 = 001 \text{ or } X_1 = 111$$

$$\text{If } X_1 = 001 \rightarrow K = X_1 \oplus P_1 = 011$$

$$\text{If } X_1 = 111 \rightarrow K = X_1 \oplus P_1 = 101$$

$$C_2 = 11 \rightarrow X_2 = 000 \text{ or } X_2 = 110$$

$$\text{If } X_2 = 000 \rightarrow K = X_2 \oplus P_2 = 011$$

$$\text{If } X_2 = 110 \rightarrow K = X_2 \oplus P_2 = 101$$

The two tests confirm that  $K = 011$  or  $K = 101$ .

## **5.1.6 *Continued***

### **Note**

**Differential cryptanalysis is based on a nonuniform differential distribution table of the S-boxes in a block cipher.**

### **Note**

**A more detailed differential cryptanalysis is given in Appendix N.**

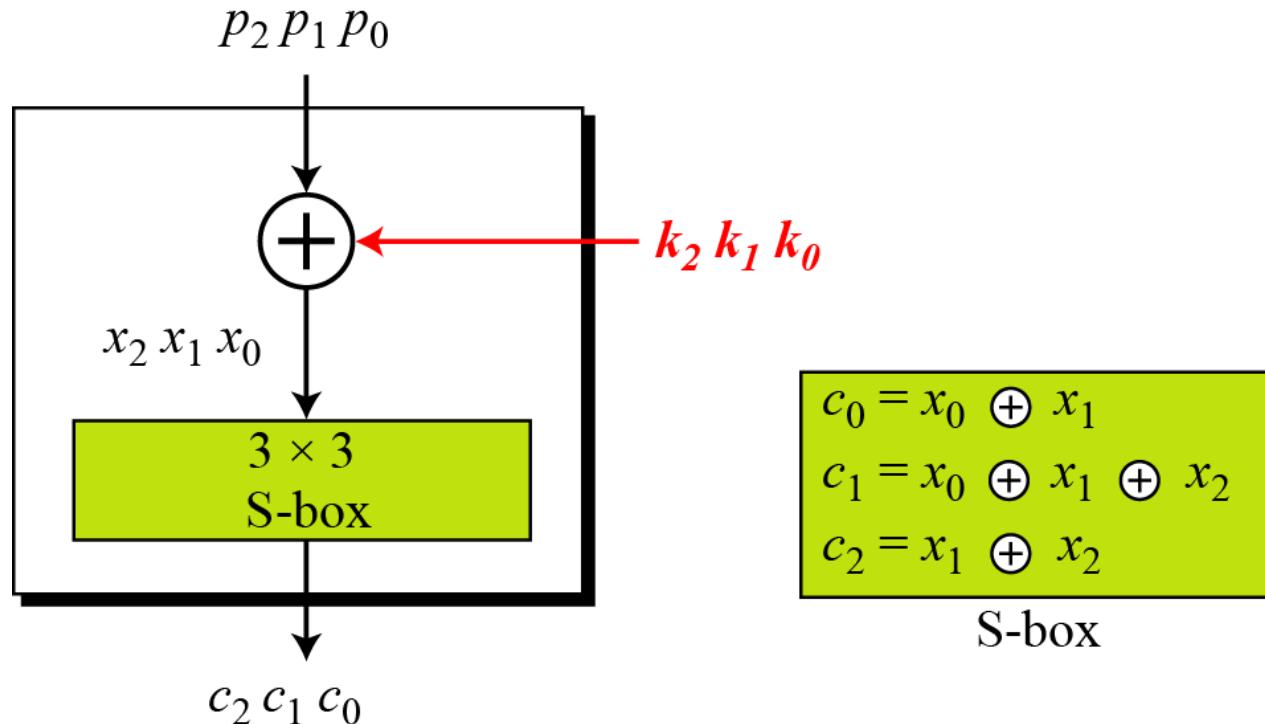
## **5.1.6 Continued**

### *Linear Cryptanalysis*

*Linear cryptanalysis was presented by Mitsuru Matsui in 1993. The analysis uses known plaintext attacks.*

## 5.1.6 Continued

**Figure 5.20** A simple cipher with a linear S-box



## 5.1.6 Continued

$$c_0 = p_0 \oplus k_0 \oplus p_1 \oplus k_1$$

$$c_1 = p_0 \oplus k_0 \oplus p_1 \oplus k_1 \oplus p_2 \oplus k_2$$

$$c_2 = p_1 \oplus k_1 \oplus p_2 \oplus k_2$$

*Solving for three unknowns, we get.*

$$k_1 = (p_1) \oplus (c_0 \oplus c_1 \oplus c_2)$$

$$k_2 = (p_2) \oplus (c_0 \oplus c_1)$$

$$k_0 = (p_0) \oplus (c_1 \oplus c_2)$$

*This means that three known-plaintext attacks can find the values of  $k_0$ ,  $k_1$ , and  $k_2$ .*

## 5.1.6 Continued

*In some modern block ciphers, it may happen that some S-boxes are not totally nonlinear; they can be approximated, probabilistically, by some linear functions.*

$$(k_0 \oplus k_1 \oplus \cdots \oplus k_x) = (p_0 \oplus p_1 \oplus \cdots \oplus p_y) \oplus (c_0 \oplus c_1 \oplus \cdots \oplus c_z)$$

*where  $1 \leq x \leq m$ ,  $1 \leq y \leq n$ , and  $1 \leq z \leq n$ .*

### Note

A more detailed linear cryptanalysis is given in Appendix N.

## 5-2 MODERN STREAM CIPHERS

*In a modern stream cipher, encryption and decryption are done  $r$  bits at a time. We have a plaintext bit stream  $P = p_n \dots p_2 \ p_1$ , a ciphertext bit stream  $C = c_n \dots c_2 \ c_1$ , and a key bit stream  $K = k_n \dots k_2 \ k_1$ , in which  $p_i$ ,  $c_i$ , and  $k_i$  are  $r$ -bit words.*

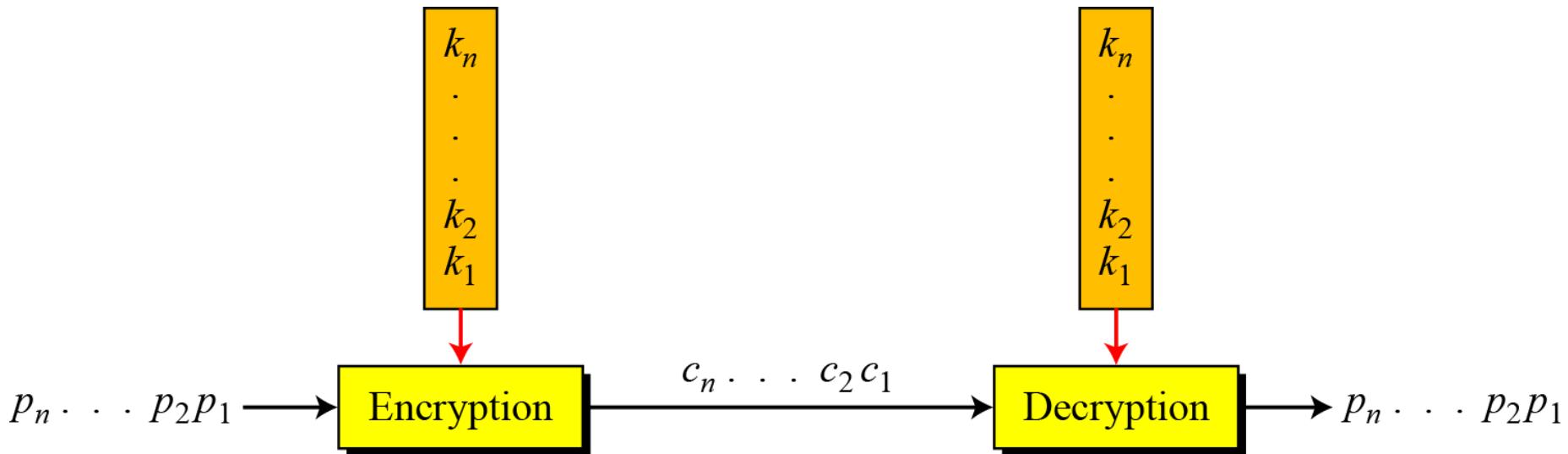
**Topics discussed in this section:**

**5.2.1 Synchronous Stream Ciphers**

**5.2.2 Nonsynchronous Stream Ciphers**

## 5.2 Continued

Figure 5.20 Stream cipher



**Note**

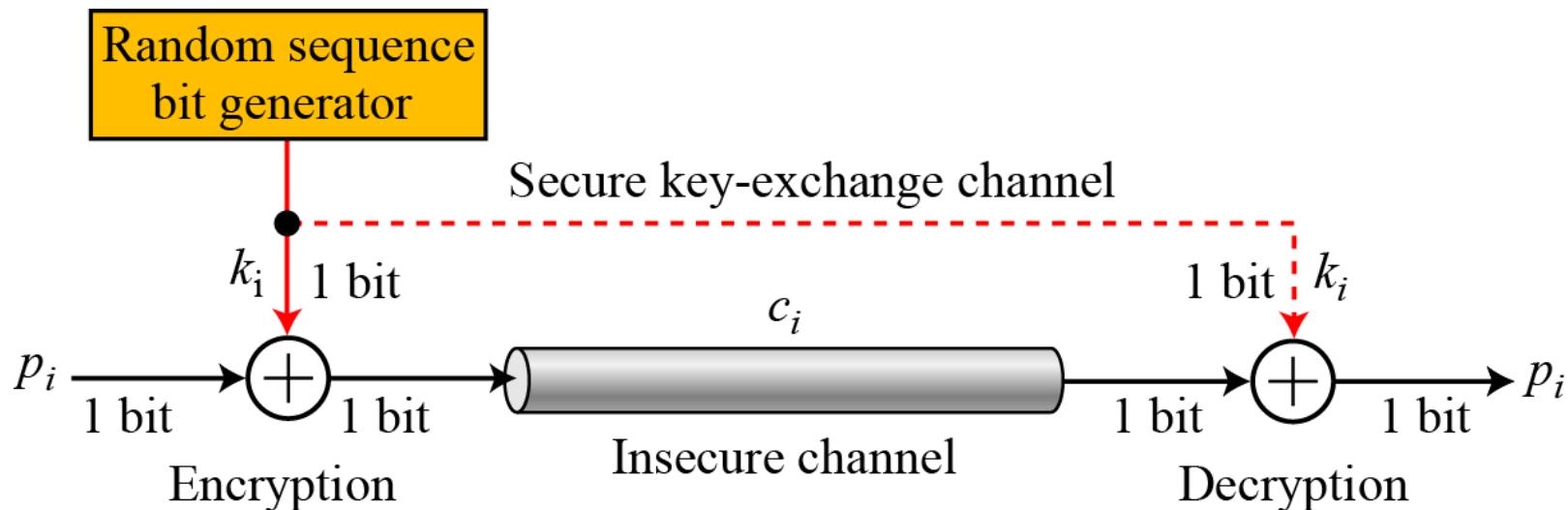
In a modern stream cipher, each  $r$ -bit word in the plaintext stream is enciphered using an  $r$ -bit word in the key stream to create the corresponding  $r$ -bit word in the ciphertext stream.

## 5.2.1 Synchronous Stream Ciphers

**Note**

In a synchronous stream cipher the key is independent of the plaintext or ciphertext.

**Figure 5.22 One-time pad**



## 5.2.1 *Continued*

### Example 5.17

What is the pattern in the ciphertext of a one-time pad cipher in each of the following cases?

- a. The plaintext is made of  $n$  0's.
- b. The plaintext is made of  $n$  1's.
- c. The plaintext is made of alternating 0's and 1's.
- d. The plaintext is a random string of bits.

### Solution

- a. Because  $0 \oplus k_i = k_i$ , the ciphertext stream is the same as the key stream. If the key stream is random, the ciphertext is also random. The patterns in the plaintext are not preserved in the ciphertext.

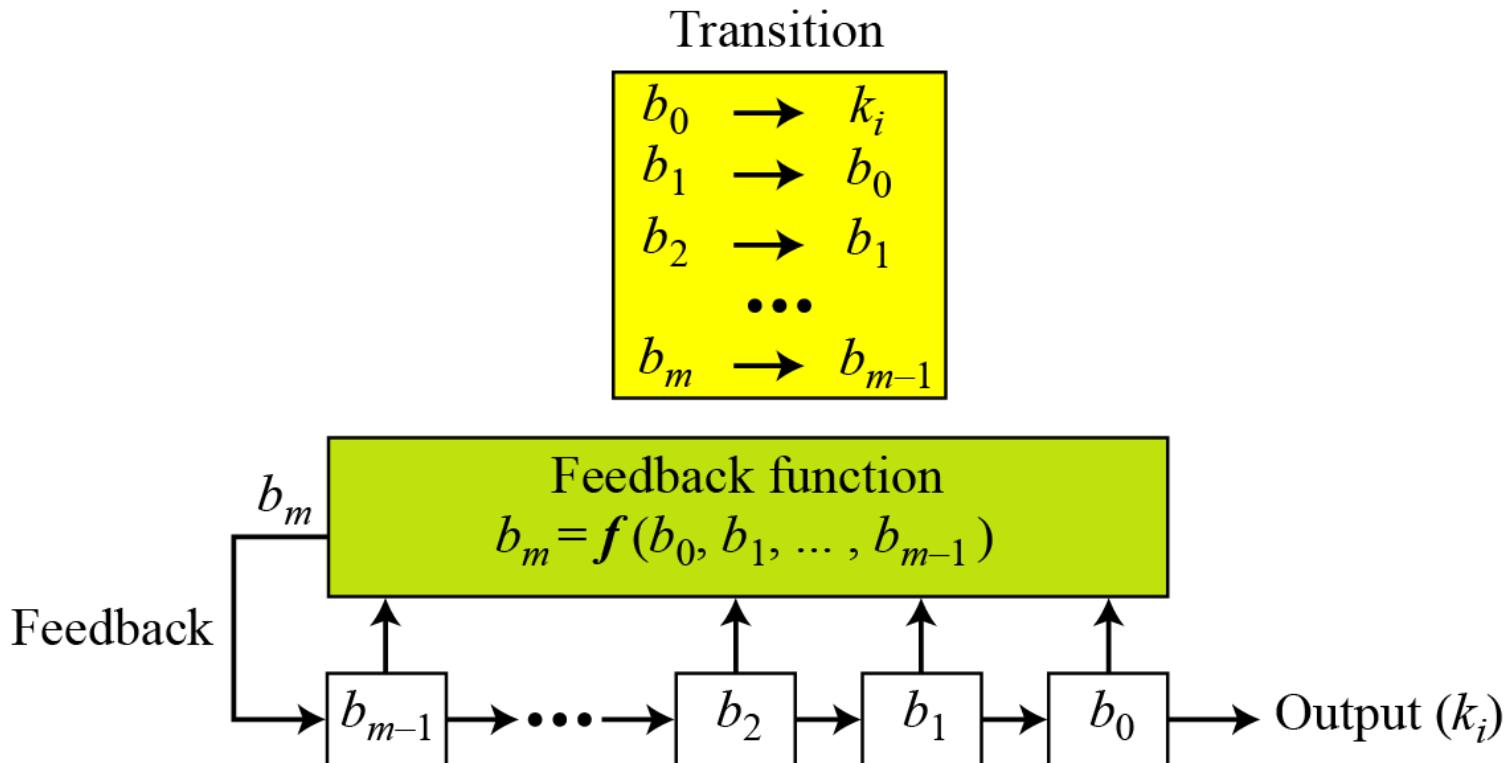
## 5.2.1 *Continued*

### Example 5.7 (Continued)

- b. Because  $1 \oplus k_i = \bar{k}_i$  where  $\bar{k}_i$  is the complement of  $k_i$ , the ciphertext stream is the complement of the key stream. If the key stream is random, the ciphertext is also random. Again the patterns in the plaintext are not preserved in the ciphertext.
- c. In this case, each bit in the ciphertext stream is either the same as the corresponding bit in the key stream or the complement of it. Therefore, the result is also a random string if the key stream is random.
- d. In this case, the ciphertext is definitely random because the exclusive-or of two random bits results in a random bit.

## 5.2.1 Continued

Figure 5.23 Feedback shift register (FSR)



## 5.2.1 *Continued*

### Example 5.18

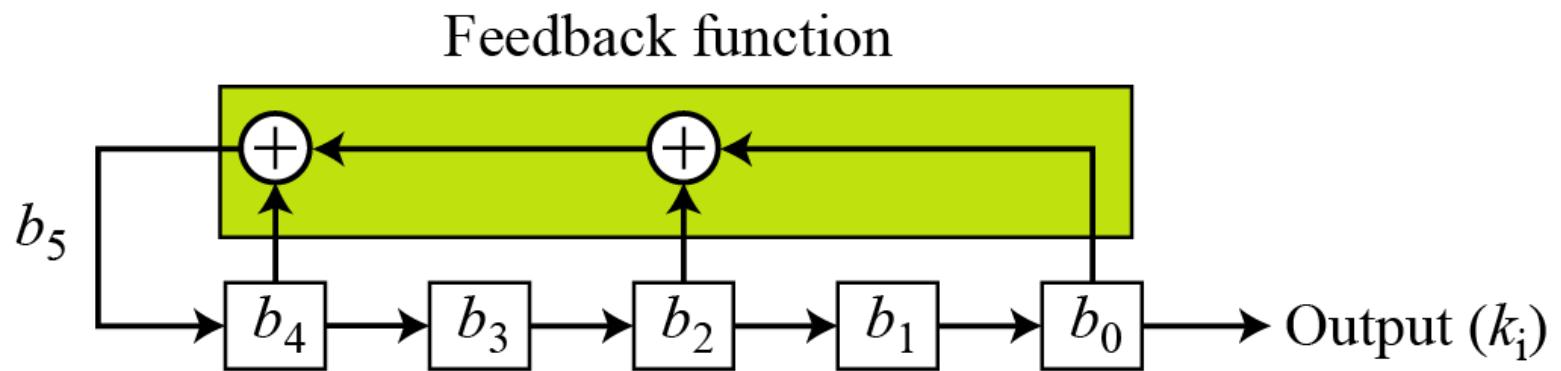
Create a linear feedback shift register with 5 cells in which  $b_5 = b_4 \oplus b_2 \oplus b_0$ .

### Solution

If  $c_i = 0$ ,  $b_i$  has no role in calculation of  $b_m$ . This means that  $b_i$  is not connected to the feedback function. If  $c_i = 1$ ,  $b_i$  is involved in calculation of  $b_m$ . In this example,  $c1$  and  $c3$  are 0's, which means that we have only three connections. Figure 5.24 shows the design.

## 5.2.1 Confidentiality

Figure 5.24 LSF for Example 5.18



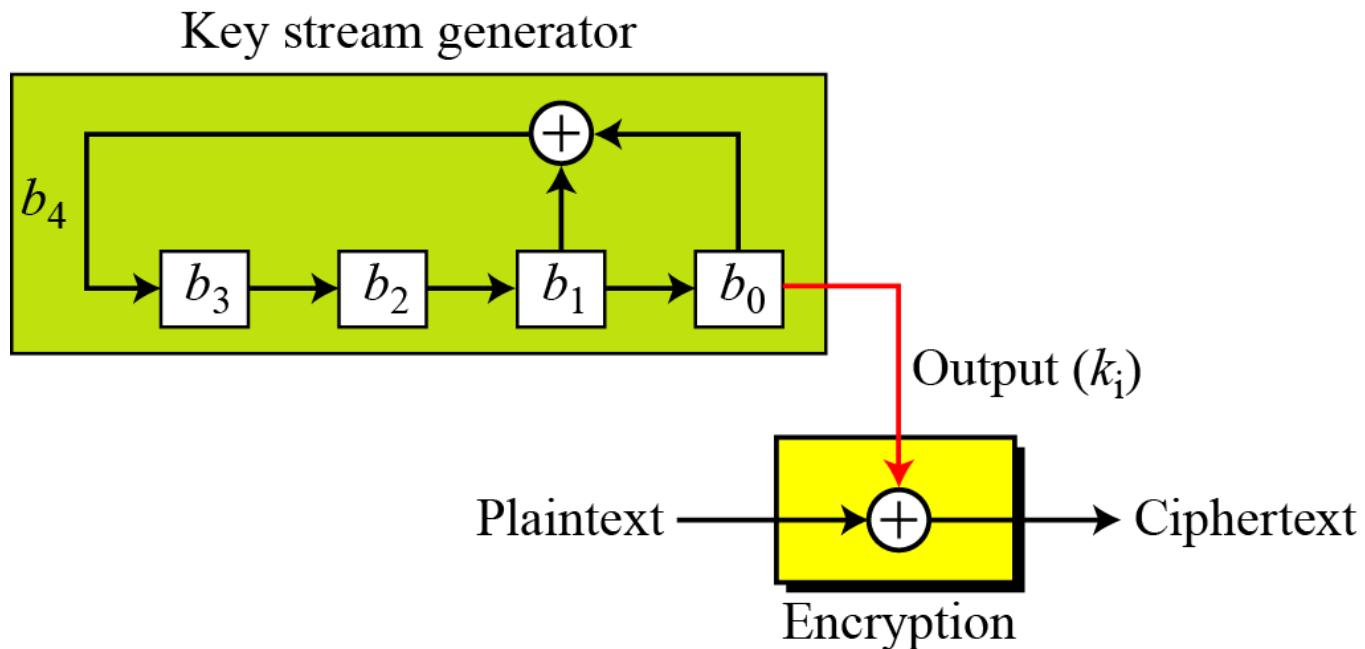
## 5.2.1 *Continued*

### Example 5.19

Create a linear feedback shift register with 4 cells in which  $b_4 = b_1 \oplus b_0$ . Show the value of output for 20 transitions (shifts) if the seed is  $(0001)_2$ .

### Solution

Figure 5.25 LFSR for Example 5.19



## 5.2.1 Continued

### Example 5.19 (Continued)

**Table 4.6** *Cell values and key sequence for Example 5.19*

States	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$	$k_i$
Initial	1	0	0	0	1	
1	0	1	0	0	0	1
2	0	0	1	0	0	0
3	1	0	0	1	0	0
4	1	1	0	0	1	0
5	0	1	1	0	0	1
6	1	0	1	1	0	0
7	0	1	0	1	1	0
8	1	0	1	0	1	1
9	1	1	0	1	0	1
10	1	1	1	0	1	0

## 5.2.1 Continued

### Example 5.19 (Continued)

**Table 4.6** Continued

11	1	1	1	1	0	1
12	0	1	1	1	1	0
13	0	0	1	1	1	1
14	0	0	0	1	1	1
15	1	0	0	0	1	1
16	0	1	0	0	0	1
17	0	0	1	0	0	0
18	1	0	0	1	0	0
19	1	1	0	0	1	0
20	1	1	1	0	0	1

## 5.2.1 *Continued*

### Example 5.19 (Continued)

Note that the key stream is **100010011010111 10001...** This looks like a random sequence at first glance, but if we go through more transitions, we see that the sequence is periodic. It is a repetition of 15 bits as shown below:

100010011010111 100010011010111 100010011010111 100010011010111 ...

The key stream generated from a LFSR is a pseudorandom sequence in which the sequence is repeated after  $N$  bits.

**Note**

The maximum period of an LFSR is to  $2^m - 1$ .

## **5.2.1   Continued**

### **Example 5.20**

**The characteristic polynomial for the LFSR in Example 5.19 is  $(x^4 + x + 1)$ , which is a primitive polynomial. Table 4.4 (Chapter 4) shows that it is an irreducible polynomial. This polynomial also divides  $(x^7 + 1) = (x^4 + x + 1)(x^3 + 1)$ , which means  $e = 2^3 - 1 = 7$ .**

## 5.2.2 *Nonsynchronous Stream Ciphers*

*In a nonsynchronous stream cipher, each key in the key stream depends on previous plaintext or ciphertext.*

**Note**

**In a nonsynchronous stream cipher, the key depends on either the plaintext or ciphertext.**

# **Asymmetric-Key Cryptography**

# **Mật mã Khóa-bất đối xứng**

# Nội dung

- To distinguish between two cryptosystems: symmetric-key and asymmetric-key
- To introduce trapdoor one-way functions and their use in asymmetric-key cryptosystems
- To introduce the knapsack cryptosystem as one of the first ideas in asymmetric-key cryptography
- To discuss the RSA cryptosystem
- To discuss the Rabin cryptosystem
- To discuss the ElGamal cryptosystem

# Nội dung – Mục đích

- Để phân biệt giữa hai hệ thống mật mã: **khóa đối xứng và khóa không đối xứng**
- Giới thiệu các **hàm một chiều của cửa sập** và việc sử dụng chúng trong hệ thống mật mã khóa không đối xứng
- Giới thiệu hệ thống mật mã knapsack như một trong những ý tưởng đầu tiên trong mật mã **khóa bất đối xứng**
- Để thảo luận về hệ thống mật mã **RSA**
- Để thảo luận về hệ thống mật mã **Rabin**
- Để thảo luận về hệ thống mật mã **ElGamal**

# 1 INTRODUCTION

*Mật mã khóa **đối xứng** và **không đối xứng** sẽ tồn tại song song và tiếp tục phục vụ cộng đồng. Chúng ta thực sự tin rằng chúng là sự bổ sung của nhau; ưu điểm của cái này có thể bù đắp cho những bất lợi của cái kia.*

*Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community. We actually believe that they are complements of each other; the advantages of one can compensate for the disadvantages of the other.*

# 1 INTRODUCTION

**Chủ đề thảo luận:**

- 1.1 Keys: Các khóa**
- 1.2 General Idea: Ý tưởng chung**
- 1.3 Need for Both**
- 1.4 Trapdoor One-Way Function: Chức năng một chiều của cửa sập**
- 1.5 Knapsack Cryptosystem: Hệ thống mật mã Knapsack**

# 1 INTRODUCTION

*Note*

Symmetric-key cryptography is based on **sharing secrecy**; asymmetric-key cryptography is based on **personal secrecy**.

Mật mã khóa đối xứng dựa trên việc **chia sẻ bí mật**;  
mật mã khóa không đối xứng dựa trên **bí mật cá nhân**.

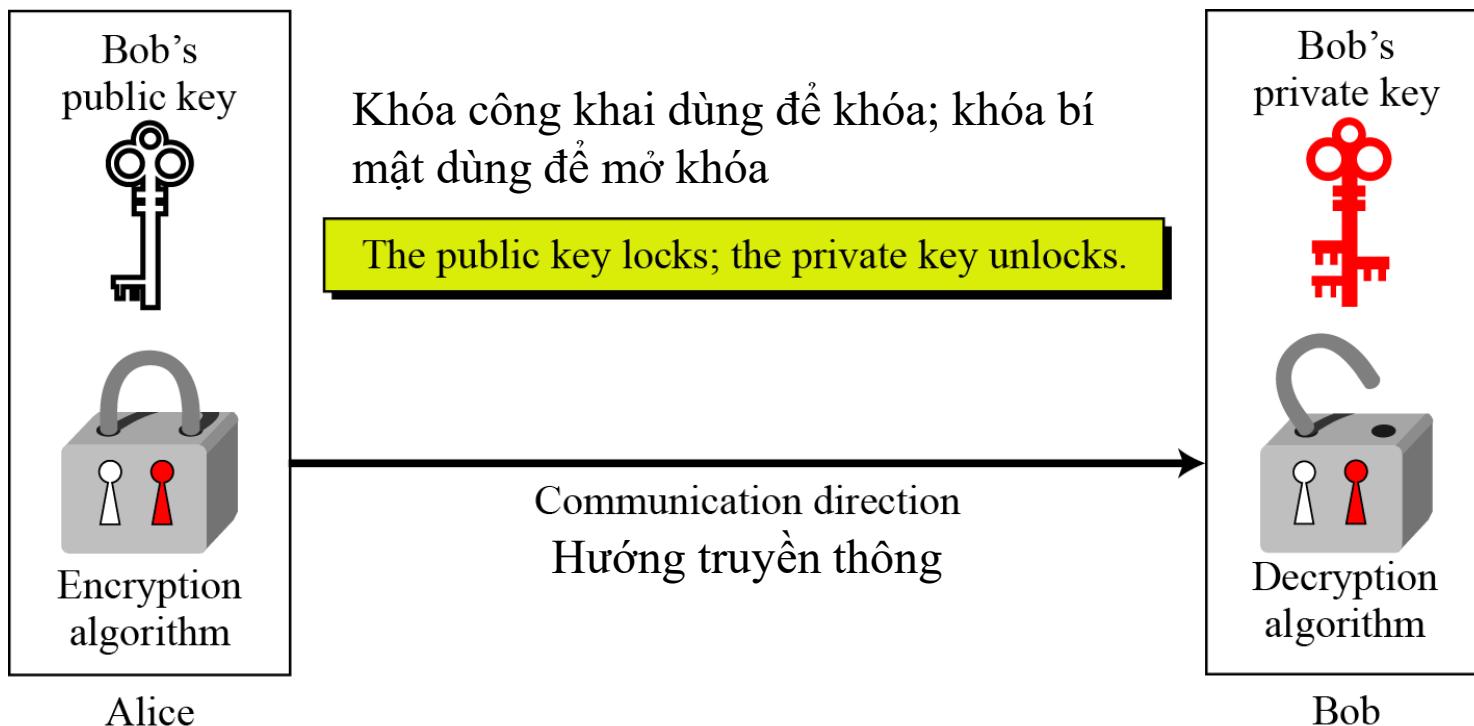
# 1.1 Keys: Các khóa bí mật, công khai

*Asymmetric key cryptography uses two separate keys: one **private** and one **public**.*

*Mật mã khóa bất đối xứng sử dụng hai khóa riêng biệt: một khóa riêng tư (bí mật) và một khóa công khai.*

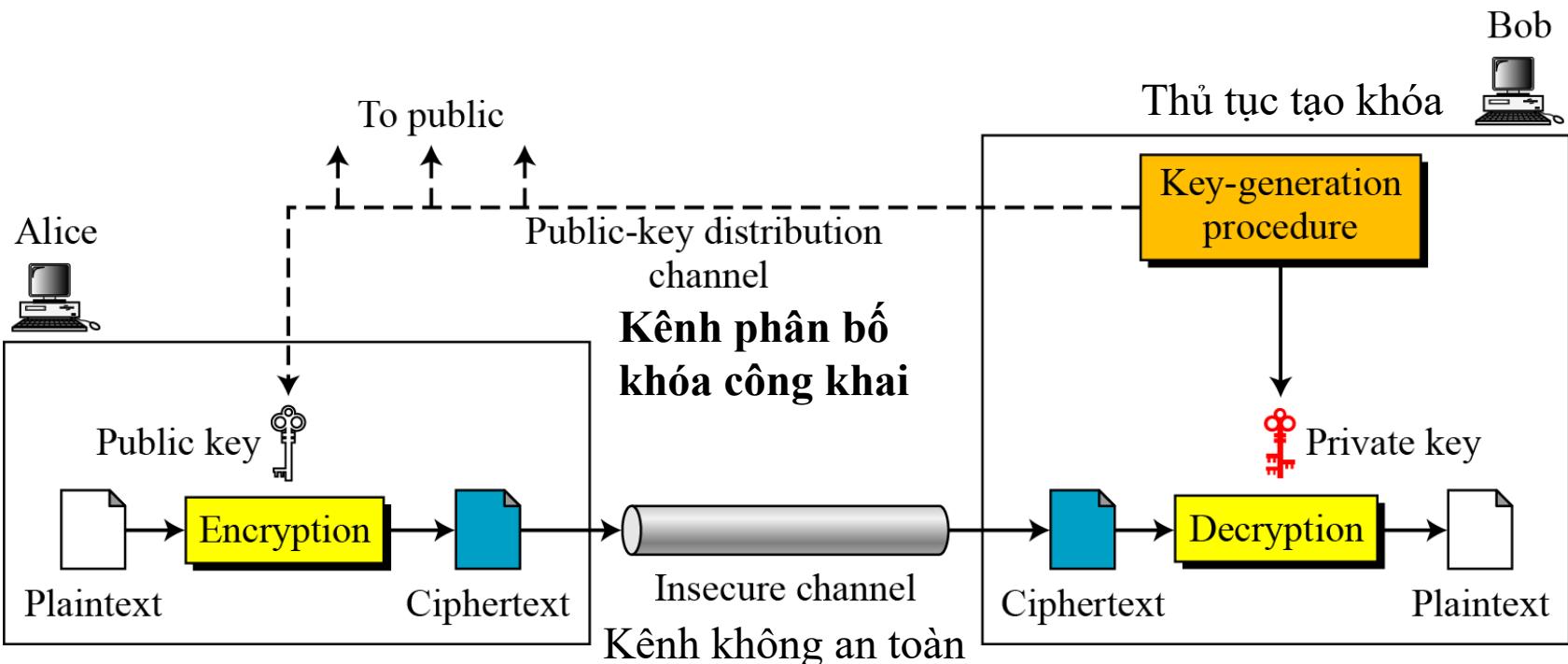
**Figure 1** Locking and unlocking in asymmetric-key cryptosystem

*Khóa và mở khóa trong hệ thống mật mã không đối xứng*



## 1.2 General Idea: Ý tưởng chung

**Figure 2 General idea of asymmetric-key cryptosystem**  
**Ý tưởng chung về hệ thống mật mã khóa không đối xứng**



## 1.2 Continued

### Plaintext/Ciphertext

Unlike in symmetric-key cryptography, plaintext and ciphertext are treated as **integers** in asymmetric-key cryptography.

*Không giống như trong mật mã khóa đối xứng, bản rõ và bản mã được coi là số nguyên trong mật mã khóa không đối xứng.*

### Encryption/Decryption

$$C = f(K_{public}, P); \quad P = g(K_{private}, C)$$

*P, C được coi là số nguyên*

## 1.3 Need for Both: Cần cho cả hai

*There is a very important fact that is sometimes misunderstood: The advent of asymmetric-key cryptography does not eliminate the need for symmetric-key cryptography.*

*Có một thực tế rất quan trọng mà đôi khi bị hiểu nhầm: Sự ra đời của mật mã khóa bất đối xứng không phải là loại bỏ sự cần thiết của mật mã khóa đối xứng.*

## 1.4 Trapdoor One-Way Function

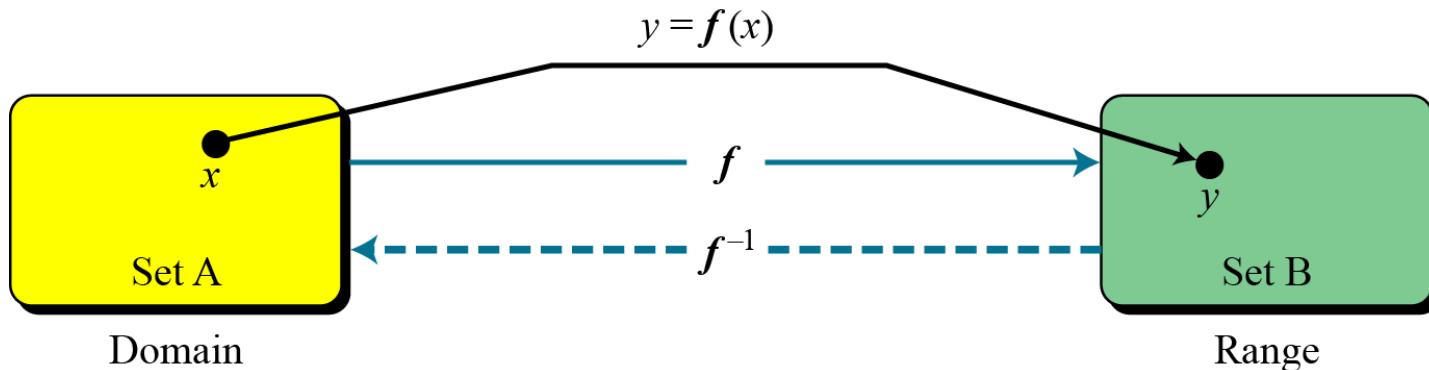
*The main idea behind asymmetric-key cryptography is the concept of the trapdoor one-way function.*

*Ý tưởng chính đằng sau mật mã khóa bất đối xứng là khái niệm về chức năng một chiều của cửa sập.*

### Functions

**Figure 3 A function as rule mapping a domain to a range**

*Một chức năng dưới dạng quy tắc ánh xạ miền tới một phạm vi*



## 1.4 Continued

**One-Way Function (OWF): Hàm một chiều**

1.  $f$  is easy to compute. Cho  $x$ ,  $y=f(x)$  là dễ thực hiện
2.  $f^{-1}$  is difficult to compute. Cho  $y$ , rất khó để thực hiện tìm  $x=f^{-1}(y)$

**Trapdoor One-Way Function (TOWF)**

*Chức năng một chiều của cửa sập*

3. Given  $y$  and a trapdoor,  $x$  can be computed easily. Cho  $y$  và một cửa sập (trapdoor) - là bí mật, có thể tính được  $x$ .

## 1.4 Continued

### Example 1

When  $n$  is large,  $n = p \times q$  is a one-way function. Given  $p$  and  $q$ , it is always easy to calculate  $n$ ; given  $n$ , it is very difficult to compute  $p$  and  $q$ . This is the factorization problem.

Khi  $n$  lớn,  $n = p \times q$  là hàm một chiều. Cho  $p$  và  $q$ , luôn dễ dàng tính được  $n$ ; cho trước  $n$ , rất khó để tính  $p$  và  $q$ . Đây là vấn đề thừa số hóa.

## 1.4 Continued

### Example 2

When  $n$  is large, the function  $y = x^k \text{ mod } n$  is a trapdoor one-way function. Given  $x$ ,  $k$ , and  $n$ , it is easy to calculate  $y$ . Given  $y$ ,  $k$ , and  $n$ , it is very difficult to calculate  $x$ . This is the discrete logarithm problem. However, if we know the trapdoor,  $k'$  such that  $k \times k' = 1 \text{ mod } \phi(n)$ , we can use  $x = y^{k'} \text{ mod } n$  to find  $x$ .

Khi  $n$  lớn, hàm  $y = x^k \text{ mod } n$  là hàm một chiều cửa sập. Với  $x$ ,  $k$  và  $n$ , ta dễ dàng tính được  $y$ . Với  $y$ ,  $k$  và  $n$ , rất khó để tính  $x$ . Đây là bài toán logarit rời rạc. Tuy nhiên, nếu chúng ta biết cửa sập,  $k'$  sao cho  $k \times k' = 1 \text{ mod } \phi(n)$ , chúng ta có thể sử dụng  $x = y^{k'} \text{ mod } n$  để tìm  $x$ .

## 1.5 Knapsack Cryptosystem

### Giới thiệu

- + Ý tưởng đầu tiên được đưa ra bởi Merkle và Hellman
- + Dựa trên hệ thống mật mã khóa công khai (public key)

## 1.5 Knapsack Cryptosystem

**Definition: Định nghĩa**

$a = [a_1, a_2, \dots, a_k]$  and  $x = [x_1, x_2, \dots, x_k]$ .

$a$  là dãy số cho trước,  $x_i$  là các số “0” hoặc “1”

Tổng của các thành phần  $x_i \times a_i$  được đưa vào Knapsack.

$$s = knapsackSum(a, x) = x_1a_1 + x_2a_2 + \dots + x_ka_k$$

Given  $a$  and  $x$ , it is easy to calculate  $s$ . Cho  $a$  và  $x \rightarrow$  dễ dàng tìm được giá trị  $s$ .

However, given  $s$  and  $a$  it is difficult to find  $x$ . Cho  $s$  và  $a \rightarrow$  khó để tìm được giá trị  $x = inv\_knapsackSum(s, a)$ .

**Superincreasing Tuple: Chuỗi tăng cường Tuple**

$$a_i \geq a_1 + a_2 + \dots + a_{i-1}$$

## 1.5 Knapsack Cryptosystem

*Điều kiện Superincreasing Tuple:*

$$a_i \geq a_1 + a_2 + \dots + a_{i-1}$$

*Mỗi thành phần (ngoại trừ a1) lớn hơn hoặc bằng tổng các thành phần trước nó. Khi đó, chúng ta tính toán hàm knapsackSum và Inv\_knapsackSum như slide sau.*

*Chú ý: trong quá trình tính ngược tìm x, cần bắt đầu từ giá trị lớn nhất đến giá trị nhỏ nhất của a.*

# 1.5 Continued

**Algorithm 10.1** *knapsacksum and inv\_knapsackSum for a superincreasing k-tuple*

```
knapsackSum ( $x [1 \dots k]$ ,  $a [1 \dots k]$ )
{
     $s \leftarrow 0$ 
    for ( $i = 1$  to  $k$ )
    {
         $s \leftarrow s + a_i \times x_i$ 
    }
    return  $s$ 
}
```

```
inv_knapsackSum ( $s, a [1 \dots k]$ )
{
    for ( $i = k$  down to 1)
    {
        if  $s \geq a_i$ 
        {
             $x_i \leftarrow 1$ 
             $s \leftarrow s - a_i$ 
        }
        else  $x_i \leftarrow 0$ 
    }
    return  $x [1 \dots k]$ 
}
```

$x=inv\_knapsackSum(s, a).$   
*Bắt đầu từ ai có giá trị max*

# 1.5 Continued

## Example 3

As a very trivial example, assume that  $a = [17, 25, 46, 94, 201, 400]$  and  $s = 272$  are given. Table 10.1 shows how the tuple  $x$  is found using `inv_knapsackSum` routine in Algorithm 10.1. In this case  $x = [0, 1, 1, 0, 1, 0]$ , which means that 25, 46, and 201 are in the knapsack.

**Table 10.1** Values of  $i$ ,  $a_i$ ,  $s$ , and  $x_i$  in Example 10.3

$i$	$a_i$	$s$	$s \geq a_i$	$x_i$	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

# 1.5 Continued

## Example 3

Như một ví dụ rất đơn giản, giả sử rằng  $a = [17, 25, 46, 94, 201, 400]$  và  $s = 272$  được đưa ra. Bảng 10.1 cho thấy cách tìm tuple  $x$  bằng cách sử dụng quy trình `inv_knapsackSum` trong Thuật toán 10.1. Trong trường hợp này  $x = [0, 1, 1, 0, 1, 0]$ , có nghĩa là 25, 46 và 201 nằm trong cái túi (knapsack).

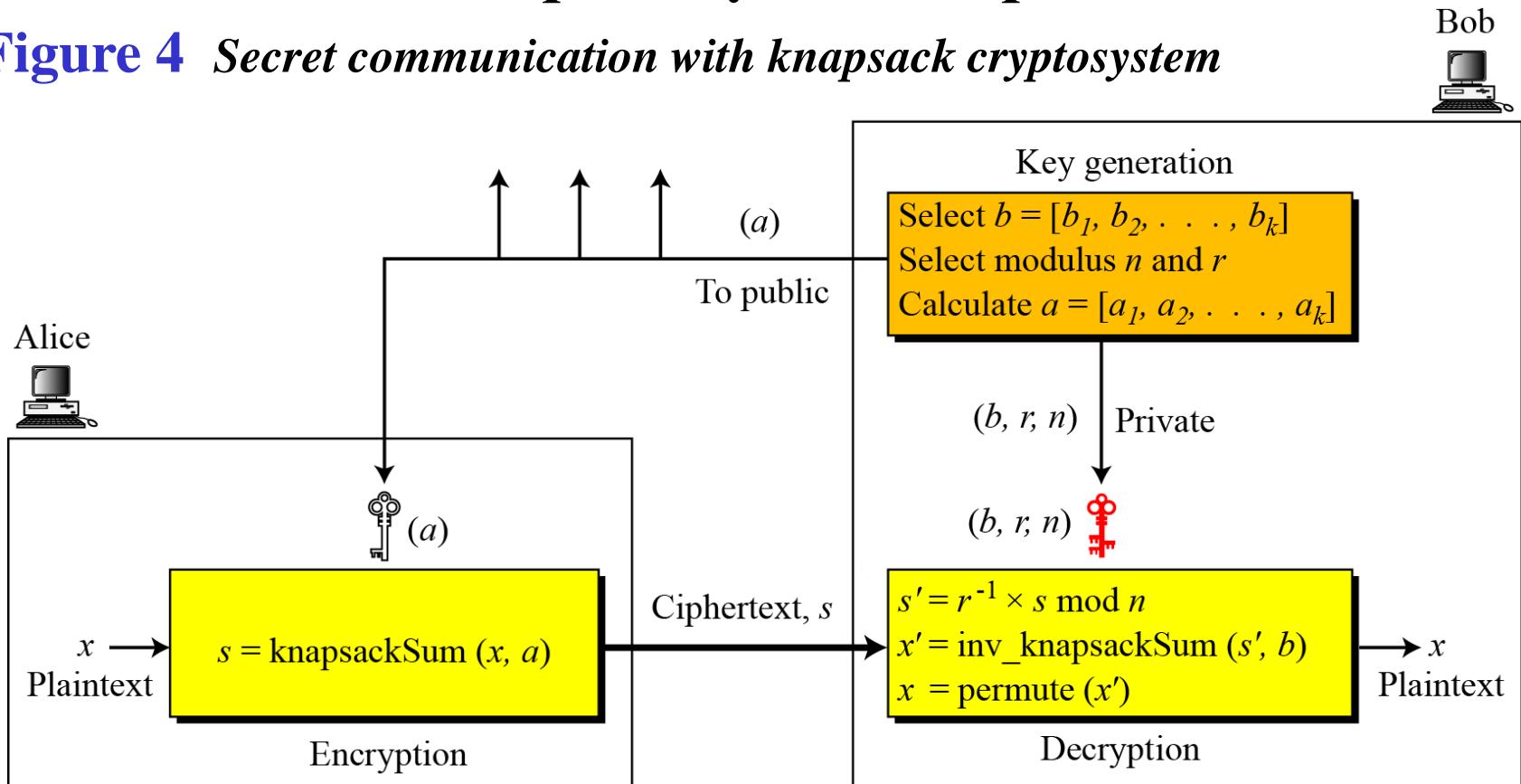
**Table 10.1** Values of  $i$ ,  $a_i$ ,  $s$ , and  $x_i$  in Example 10.3

$i$	$a_i$	$s$	$s \geq a_i$	$x_i$	$s \leftarrow s - a_i \times x_i$
6	400	272	false	$x_6 = 0$	272
5	201	272	true	$x_5 = 1$	71
4	94	71	false	$x_4 = 0$	71
3	46	71	true	$x_3 = 1$	25
2	25	25	true	$x_2 = 1$	0
1	17	0	false	$x_1 = 0$	0

# 1.5 Continued

## *Secret Communication with Knapsacks.* *Giao tiếp bí mật với Knapsacks.*

**Figure 4** Secret communication with knapsack cryptosystem



## 1.5 *Continued*

**Quá trình tạo khóa (Key generation) công khai, bí mật:**

- + Tạo chuỗi tăng dần superincreasing  $k$ -tuple  $b=[b_1, b_2, \dots, b_k]$
- + Chọn một giá trị mô-đun  $n$  thỏa mãn:  $n > b_1 + b_2 + \dots + b_k$ ;
- + Chọn giá trị số nguyên  $r$  ngẫu nhiên, nguyên tố với  $n$  và thỏa mãn điều kiện  $1 \leq r \leq n-1$ ;
- + Tạo một dãy  $t=[t_1, t_2, \dots, t_k]$  với  $t_i = r \times b_i \bmod n$ ;
- + Chọn phép hoán vị  $k$  phần tử và tìm chuỗi mới :  $a = \text{permute}(t)$ ;
- + Khi đó, khóa công khai là dãy  $a$ ; khóa bí mật là  $n, r$ , và dãy  $b$ .

# 1.5 *Continued*

## Quá trình mã hóa:

**Giả sử Alice cần gửi một bản tin nào đó tới Bob**

- + Alice sẽ chuyển đổi bản tin về dạng dãy  $x = [x_1, x_2, \dots, x_k]$ , với  $x_i = "0"$  hoặc  $"1"$ .
- + Alice sử dụng quá trình  $knapsackSum(a, x)$  để tính giá trị  $s$ ;  $s$  chính là giá trị của bản tin mã hóa (ciphertext)

## Quá trình giải mã:

**Giả sử Bob nhận được trọng lượng bản tin mã hóa  $s$ ;**

- + Bob sẽ tính giá trị  $s' = r^{-1} \times s \text{ mod } n$ ;
- + Bob sử dụng quá trình  $inv\_knapsackSum(s', b)$  để tính ra  $x'$ ;
- + Bob thực hiện hoán vị  $x'$  để tìm ra  $x$ . Dãy  $x$  là bản tin gốc.

# 1.5 Continued

## Example 4

Đây là một ví dụ (rất không an toàn) chỉ để mô tả quy trình.

1. Key generation:
  - a. Bob creates the superincreasing tuple  $b = [7, 11, 19, 39, 79, 157, 313]$ .
  - b. Bob chooses the modulus  $n = 900$  and  $r = 37$ , and  $[4 \ 2 \ 5 \ 3 \ 1 \ 7 \ 6]$  as permutation table.
  - c. Bob now calculates the tuple  $t = [259, 407, 703, 543, 223, 409, 781]$ .
  - d. Bob calculates the tuple  $a = \text{permute}(t) = [543, 407, 223, 703, 259, 781, 409]$ .
  - e. Bob publicly announces  $a$ ; he keeps  $n, r$ , and  $b$  secret.
2. Suppose Alice wants to send a single character “g” to Bob.
  - a. She uses the 7-bit ASCII representation of “g”,  $(1100111)_2$ , and creates the tuple  $x = [1, 1, 0, 0, 1, 1, 1]$ . This is the plaintext.
  - b. Alice calculates  $s = \text{knapsackSum}(a, x) = 2165$ . This is the ciphertext sent to Bob.
3. Bob can decrypt the ciphertext,  $s = 2399$ 
  - a. Bob calculates  $s' = s \times r^{-1} \pmod{n} = 2165 \times 37^{-1} \pmod{900} = 527$ .
  - b. Bob calculates  $x' = \text{Inv\_knapsackSum}(s', b) = [1, 1, 0, 1, 0, 1, 1]$ .
  - c. Bob calculates  $x = \text{permute}(x') = [1, 1, 0, 0, 1, 1, 1]$ . He interprets the string  $(1100111)_2$  as the character “g”.

## 1.5 Continued

### Example 4

Đây là một ví dụ (rất không an toàn) chỉ để mô tả quy trình.

#### *Trapdoor*

Calculating the sum of items in Alice's knapsack is actually the multiplication of the row matrix  $x$  by the column matrix  $a$ . The result is a  $1 \times 1$  matrix  $s$ . Matrix multiplication,  $s = x \times a$ , in which  $x$  is a row matrix and  $a$  is a column matrix, is a one-way function. Given  $s$  and  $x$ , Eve cannot find  $a$  easily. Bob, however, has a trapdoor. Bob uses his  $s' = r^{-1} \times s$  and the secret superincreasing column matrix  $b$  to find a row matrix  $x'$  using the *inv\_knapsackSum* routine. The permutation allows Bob to find  $x$  from  $x'$ .

## 2 RSA CRYPTOSYSTEM: HỆ MẬT RSA

*The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).*

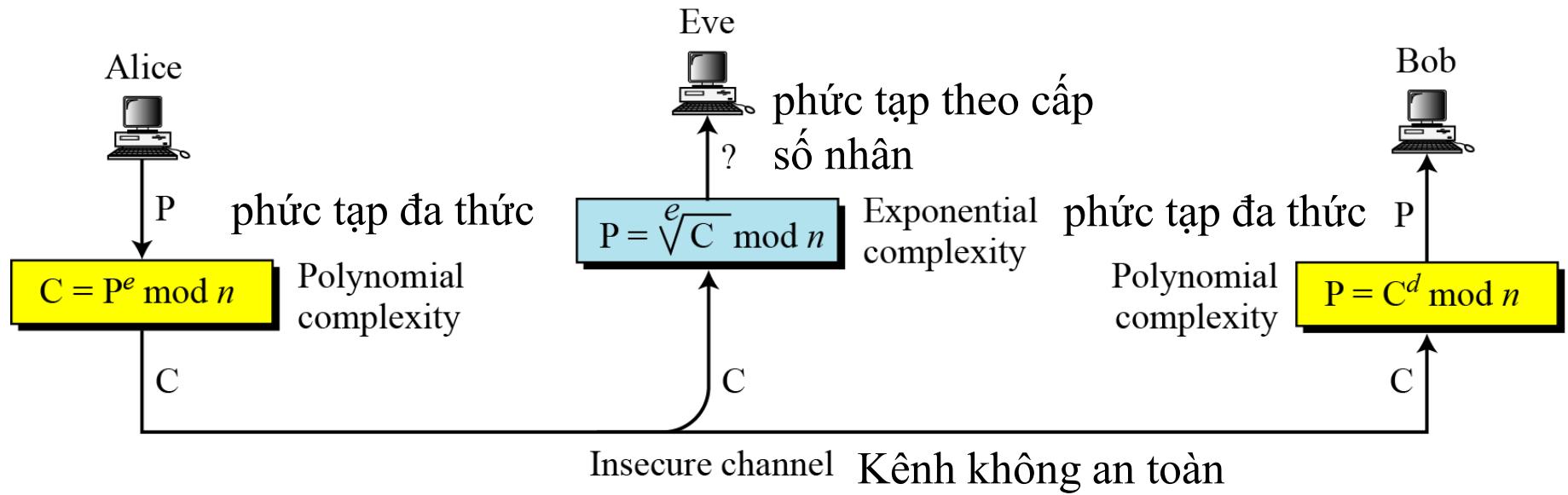
*Thuật toán khóa công khai phổ biến nhất là hệ thống mật mã RSA, được đặt tên cho những người phát minh ra nó (Rivest, Shamir và Adleman).*

### **Topics discussed in this section:**

- 2.1      Introduction**
- 2.2      Procedure**
- 2.3      Some Trivial Examples**
- 2.4      Attacks on RSA**
- 2.5      Recommendations**
- 2.6      Optimal Asymmetric Encryption Padding (OAEP)**
- 2.7      Applications**

## 2.1 Introduction: Giới thiệu

**Figure 5 Complexity of operations in RSA**  
**Độ phức tạp của các hoạt động trong RSA**

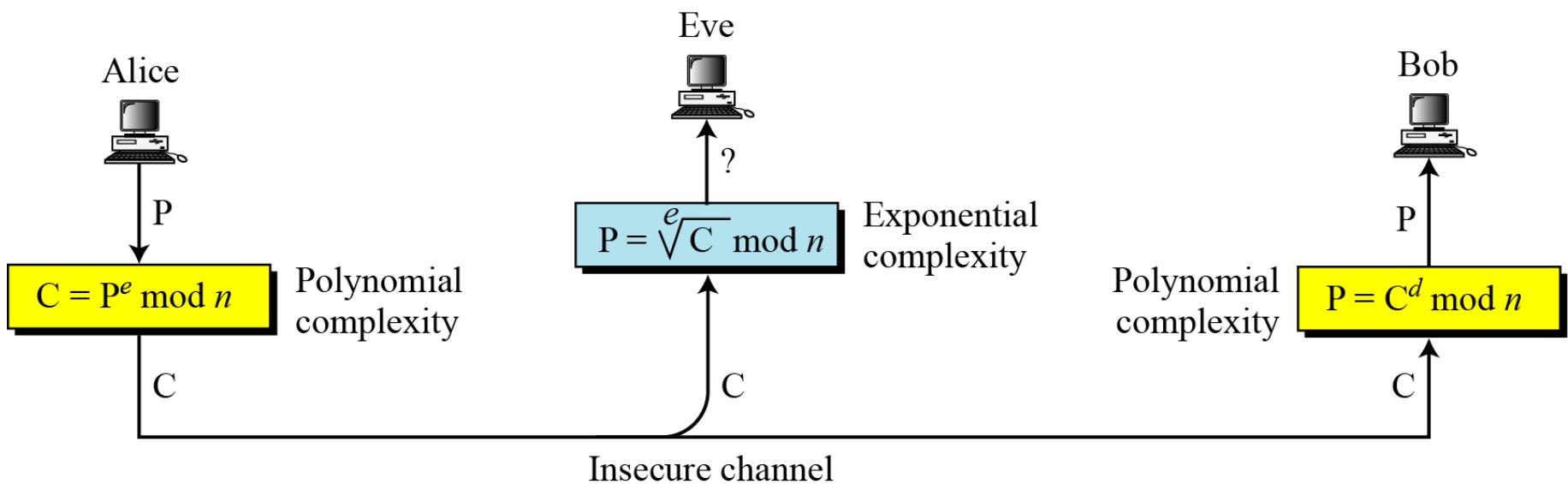


**RSA uses modular exponentiation for encryption/decryption;  
To attack it, Eve needs to calculate  $\sqrt[e]{C} \text{ mod } n$ .**

**RSA sử dụng lũy thừa mô-đun để mã hóa/ giải mã hóa;  
Để tấn công được, Eve cần tính  $\sqrt[e]{C} \text{ mod } n$ .**

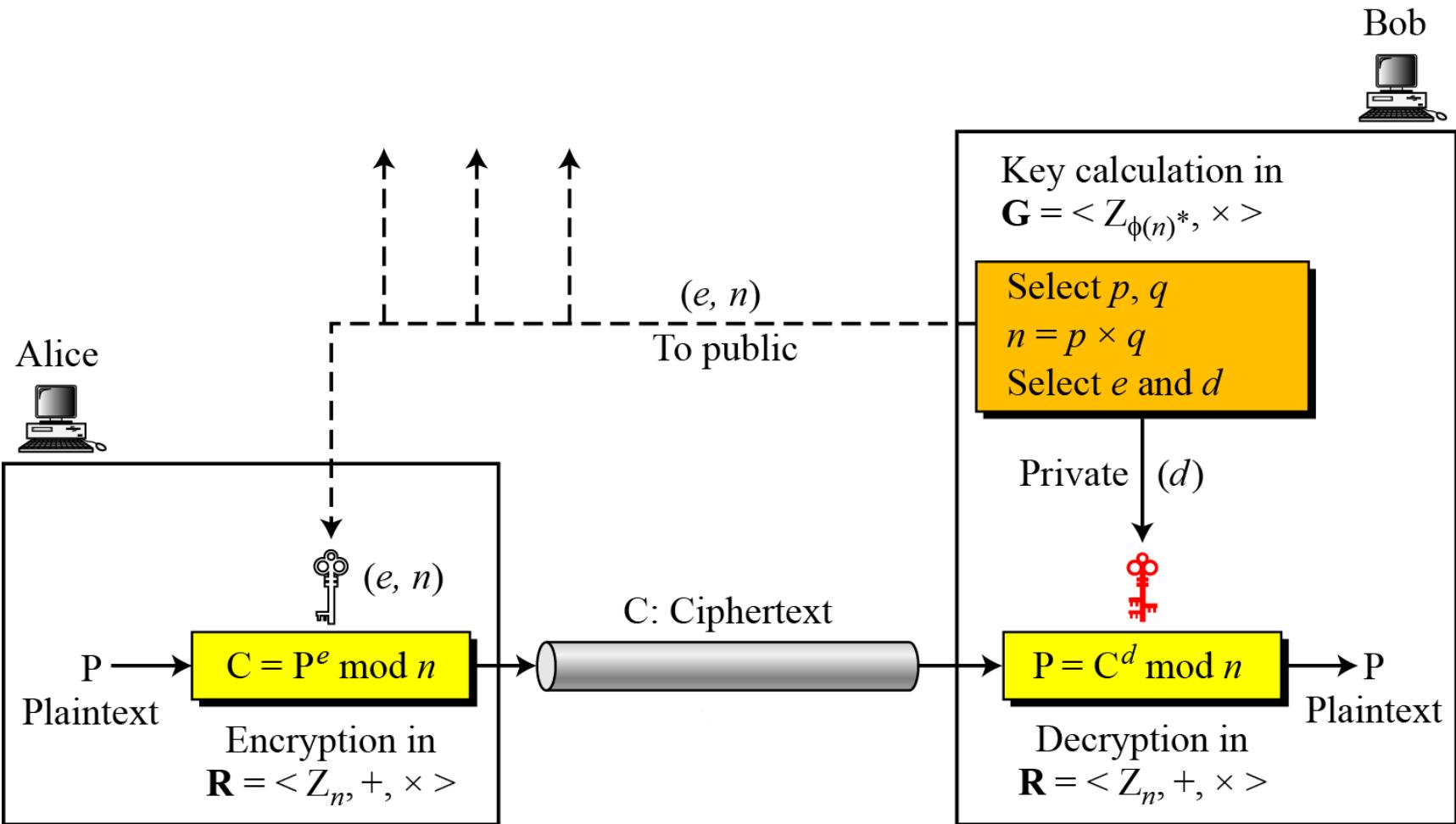
## 2.1 Introduction: Giới thiệu

- RSA sử dụng hai số mũ  $e$  và  $d$ :  $e$  là công khai,  $d$  là bí mật.
- P là bản tin gốc, C là bản tin mật:
- Quá trình mã hóa:  $C = P^e \text{ mod } n$ ;
- Quá trình giải mã hóa:  $P = C^d \text{ mod } n$ ;
- Mô đun  $n$  là số lớn, được tạo ra bởi quá trình tạo khóa;
- Mã hóa , giải mã hóa sử dụng lũy thừa mô đun.



## 2.2 Procedure: Thủ tục

Figure 10.6 Encryption, decryption, and key generation in RSA  
Mã hóa, giải mã và tạo khóa trong RSA



## 2.2 Continued

*Two Algebraic Structures: RSA sử dụng hai cấu trúc đại số*

*Encryption/Decryption Ring:*

$$R = \langle \mathbb{Z}_n, +, \times \rangle$$

*1. Vòng mã hóa / giải mã*

*Key-Generation Group:*

$$G = \langle \mathbb{Z}_{\phi(n)}^*, \times \rangle$$

*2. Nhóm tạo Khóa*

RSA uses two algebraic structures:  
a public ring  $R = \langle \mathbb{Z}_n, +, \times \rangle$  and a private group  $G = \langle \mathbb{Z}_{\phi(n)}^*, \times \rangle$ .

In RSA, the tuple  $(e, n)$  is the public key; the integer  $d$  is the private key.

## 2.2 *Continued*

### Algorithm 10.2 RSA Key Generation

#### RSA\_Key\_Generation

{

Select two large primes  $p$  and  $q$  such that  $p \neq q$ .

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p - 1) \times (q - 1)$

Select  $e$  such that  $1 < e < \phi(n)$  and  $e$  is coprime to  $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$  //  $d$  is inverse of  $e$  modulo  $\phi(n)$

Public\_key  $\leftarrow (e, n)$  // To be announced publicly

Private\_key  $\leftarrow d$  // To be kept secret

return Public\_key and Private\_key

}

## 2.2 Continued

### Encryption

#### Algorithm 10.3 RSA encryption

```
RSA_Encryption (P, e, n)          // P is the plaintext in  $Z_n$  and  $P < n$ 
{
    C  $\leftarrow$  Fast_Exponentiation (P, e, n)    // Calculation of  $(P^e \bmod n)$ 
    return C
}
```

In RSA,  $p$  and  $q$  must be at least 512 bits;  $n$  must be at least 1024 bits.

## 2.2 *Continued*

### *Decryption*

#### **Algorithm 10.4 RSA decryption**

```
RSA_Decryption (C, d, n)          //C is the ciphertext in Zn
{
    P ← Fast_Exponentiation (C, d, n)    // Calculation of (Cd mod n)
    return P
}
```

## 2.2 Continued

**Proof of RSA: Bằng chứng RSA về phép mã hóa và giải mã là nghịch đảo của nhau.**

If  $n = p \times q$ ,  $a < n$ , and  $k$  is an integer, then  $a^{k \times \phi(n) + 1} \equiv a \pmod{n}$ .

$$P_1 = C^d \pmod{n} = (P^e \pmod{n})^d \pmod{n} = P^{ed} \pmod{n}$$

$$ed = k\phi(n) + 1 \quad // d \text{ and } e \text{ are inverses modulo } \phi(n)$$

$$P_1 = P^{ed} \pmod{n} \rightarrow P_1 = P^{k\phi(n)+1} \pmod{n}$$

$$P_1 = P^{k\phi(n)+1} \pmod{n} = P \pmod{n}$$

// Euler's theorem (second version)

## 2.3 Some Trivial Examples: Một số ví dụ

### Example 5

Bob chooses 7 and 11 as  $p$  and  $q$  and calculates  $n = 77$ . The value of  $\phi(n) = (7 - 1)(11 - 1) = 60$ . Now, he chooses two exponents,  $e$  and  $d$ , from  $Z_{60}^*$ . If he chooses  $e = 13$ , then  $d = 37$ . Note that  $e \times d \text{ mod } 60 = 1$  (they are inverses of each). Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5

$$C = 5^{13} = 26 \text{ mod } 77$$

Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26

$$P = 26^{37} = 5 \text{ mod } 77$$

Plaintext: 5

## 2.3 Some Trivial Examples

### Example 5

Bob chọn 7 và 11 là p và q và tính n = 77. Giá trị của  $\phi(n) = (7 - 1)(11 - 1) = 60$ . Bây giờ anh ta chọn hai số mũ, e và d, từ  $Z_{60}^*$ . Nếu anh ta chọn e là 13, thì d là 37. Lưu ý rằng  $e \times d \bmod 60 = 1$  (chúng là nghịch đảo của nhau). Bây giờ hãy tưởng tượng rằng Alice muốn gửi bản rõ 5 cho Bob. Cô ấy sử dụng số mũ công khai 13 để mã hóa 5.

Plaintext: 5

$$C = 5^{13} \bmod 77$$

Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26

$$P = 26^{37} \bmod 77$$

Plaintext: 5

## 2.3 Some Trivial Examples

### Example 6

Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

Plaintext: 63

$$C = 63^{13} \equiv 28 \pmod{77}$$

Ciphertext: 28

Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

Ciphertext: 28

$$P = 28^{37} \equiv 63 \pmod{77}$$

Plaintext: 63

## 2.3 Some Trivial Examples

### Example 6

Bây giờ giả sử rằng một người khác, John, muốn gửi một tin nhắn cho Bob. John có thể sử dụng cùng một khóa công khai được Bob công khai (có thể là trên trang web của anh ấy),  $e=13$ ; John's plaintext là **63**. John tính toán như sau:

Plaintext: 63

$$C = 63^{13} = 28 \text{ mod } 77$$

Ciphertext: 28

Bob nhận bản mã 28 và sử dụng khóa riêng  $d=37$  của mình để giải mã bản mã:

Ciphertext: 28

$$P = 28^{37} = 63 \text{ mod } 77$$

Plaintext: 63

## 2.3 Some Trivial Examples

### Example 7

Jennifer creates a pair of keys for herself. She chooses  $p = 397$  and  $q = 401$ . She calculates  $n = 159197$ . She then calculates  $\phi(n) = 158400$ . She then chooses  $e = 343$  and  $d = 12007$ . Show how Ted can send a message to Jennifer if he knows  $e$  and  $n$ .

Suppose Ted wants to send the message “NO” to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Figure 10.7 shows the process.

## 2.3 Some Trivial Examples

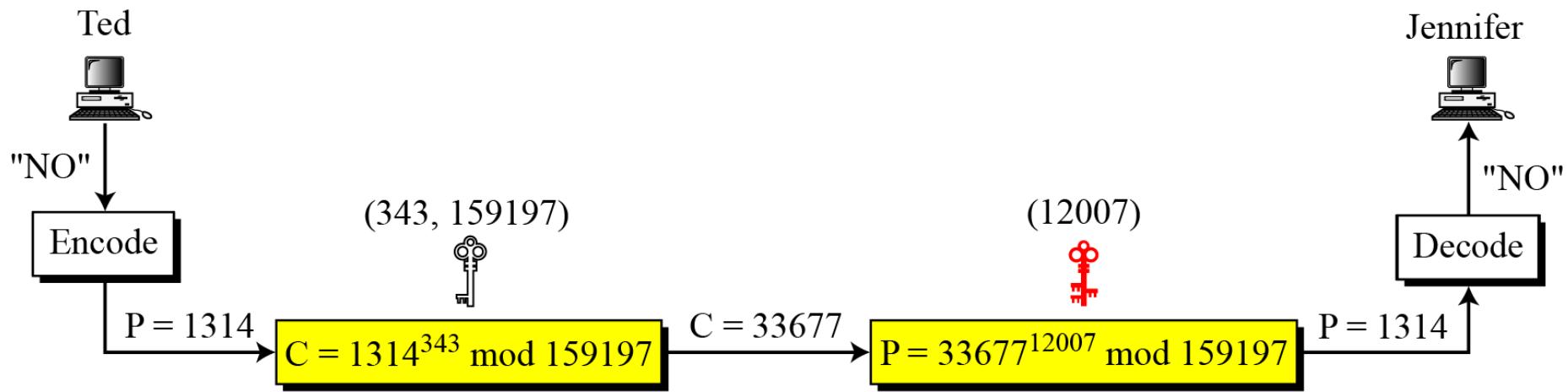
### Example 7

Jennifer tạo ra một cặp chìa khóa cho chính mình. Cô ấy chọn  $p = 397$  và  $q = 401$ . Cô ấy tính  $n = 159197$ . Sau đó cô ấy tính  $\phi(n) = (p-1)(q-1) = 158400$ . Sau đó cô ấy chọn  $e = 343$  và  $d = 12007$ . Hãy chỉ ra cách Ted có thể gửi tin nhắn cho Jennifer nếu anh ấy biết  $e$  và  $n$ .

Gia sư Ted muốn gửi tin nhắn "NU" cho Jennifer. Anh ta thay đổi mỗi ký tự thành một số (từ 00 đến 25), với mỗi ký tự được mã hóa là hai chữ số. Sau đó, anh ta nối hai ký tự được mã hóa và nhận được một số có bốn chữ số. Bản rõ là 1314. Hình 7 cho thấy quá trình này.

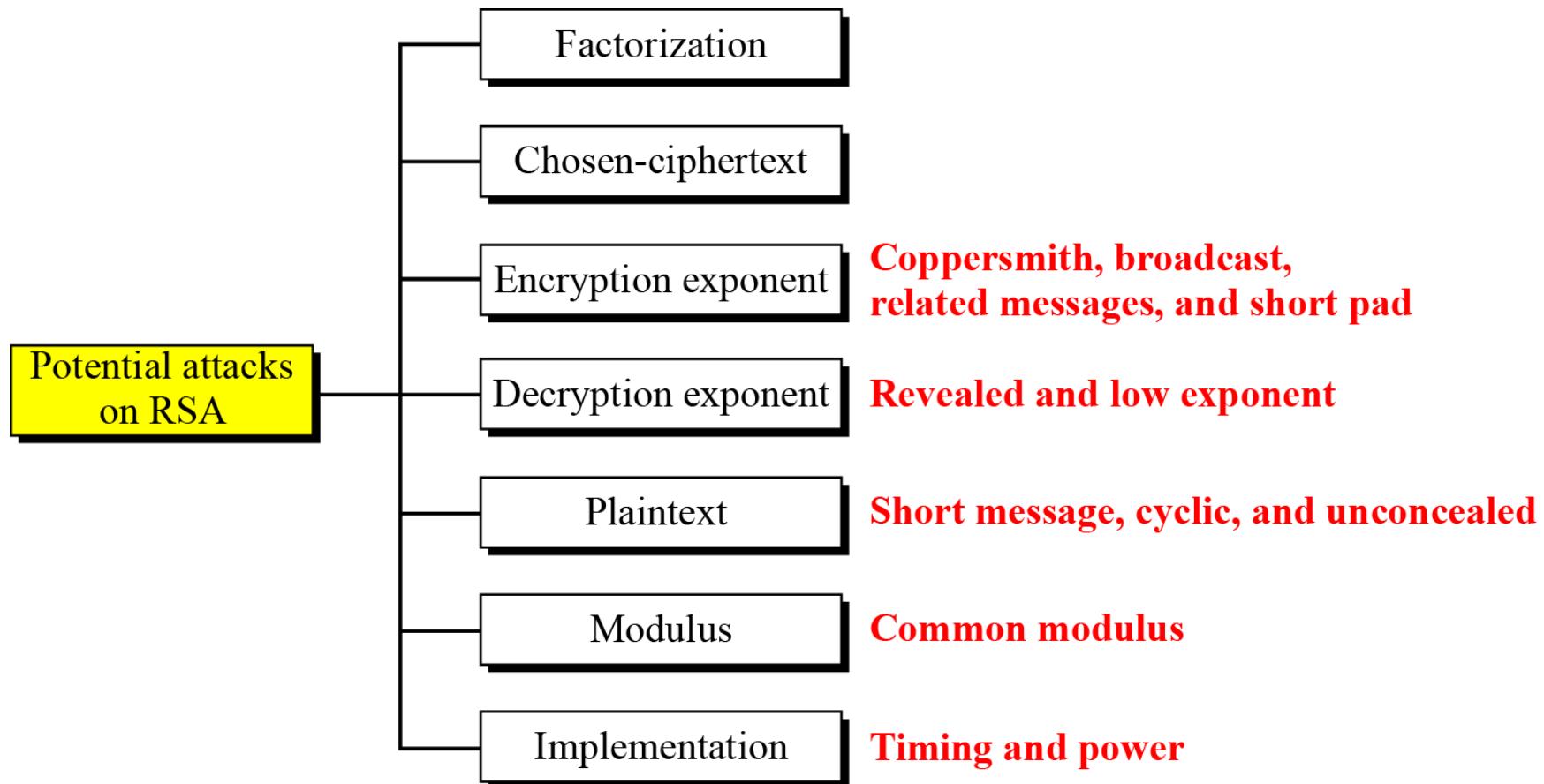
## 2.3 Continued

**Figure 7** Encryption and decryption in Example 7



## 2.4 Attacks on RSA

**Figure 8 Taxonomy of potential attacks on RSA**  
*Phân loại các cuộc tấn công vào RSA*



## 2.6 Continued

### Example 8

Here is a more realistic example. We choose a 512-bit  $p$  and  $q$ , calculate  $n$  and  $\phi(n)$ , then choose  $e$  and test for relative primeness with  $\phi(n)$ . We then calculate  $d$ . Finally, we show the results of encryption and decryption. The integer  $p$  is a 159-digit number.

$p =$

961303453135835045741915812806154279093098455949962158225831508796  
479404550564706384912571601803475031209866660649242019180878066742  
1096063354219926661209

$q =$

120601919572314469182767942044508960015559250546370339360617983217  
314821484837646592153894532091752252732268301071206956046025138871  
45524969000359660045617

## 2.6 *Continued*

### Example 8

Đây là một ví dụ thực tế hơn. Chúng ta chọn  $p$  và  $q$  512 bit, tính  $n$  và  $\phi(n)$ , sau đó chọn  $e$  và kiểm tra tính nguyên tố tương đối với  $\phi(n)$ . Sau đó ta tính  $d$ . Cuối cùng, chúng tôi hiển thị kết quả mã hóa và giải mã. Số nguyên  $p$  là một số có 159 chữ số.

$p =$

961303453135835045741915812806154279093098455949962158225831508796  
479404550564706384912571601803475031209866660649242019180878066742  
1096063354219926661209

$q =$

120601919572314469182767942044508960015559250546370339360617983217  
314821484837646592153894532091752252732268301071206956046025138871  
45524969000359660045617

## 2.6 *Continued*

### Example 8      *Continued*

**The modulus  $n = p \times q$ , có 309 chữ số.**

$$n = \begin{array}{l} 115935041739676149688925098646158875237714573754541447754855261376 \\ 147885408326350817276878815968325168468849300625485764111250162414 \\ 552339182927162507656772727460097082714127730434960500556347274566 \\ 628060099924037102991424472292215772798531727033839381334692684137 \\ 327622000966676671831831088373420823444370953 \end{array}$$

**$\phi(n) = (p - 1)(q - 1)$  có 309 chữ số.**

$$\phi(n) = \begin{array}{l} 115935041739676149688925098646158875237714573754541447754855261376 \\ 147885408326350817276878815968325168468849300625485764111250162414 \\ 552339182927162507656751054233608492916752034482627988117554787657 \\ 013923444405716989581728196098226361075467211864612171359107358640 \\ 614008885170265377277264467341066243857664128 \end{array}$$

## 2.6 *Continued*

### Example 8      *Continued*

**Bob chooses  $e = 35535$  (the ideal is 65537) and tests it to make sure it is relatively prime with  $\phi(n)$ . He then finds the inverse of  $e$  modulo  $\phi(n)$  and calls it  $d$ .**

**Bob chọn  $e = 35535$  (lý tưởng là 65537) và kiểm tra nó để đảm bảo rằng nó tương đối nguyên tố với  $\phi(n)$ . Sau đó anh ta tìm ra nghịch đảo của  $e$  modulo  $\phi(n)$  và gọi nó là  $d$ .**

$e =$	35535
$d =$	580083028600377639360936612896779175946690620896509621804228661113 805938528223587317062869100300217108590443384021707298690876006115 306202524959884448047568240966247081485817130463240644077704833134 010850947385295645071936774061197326557424237217617674620776371642 0760033708533328853214470885955136670294831

## 2.6 *Continued*

### Example 10.8 *Continued*

Alice muốn gửi tin nhắn “THIS IS A TEST”, có thể được thay đổi thành giá trị số bằng cách sử dụng lược đồ mã hóa 00–26 (26 là ký tự khoảng trắng).

$$P = \boxed{1907081826081826002619041819}$$

Bản mã được Alice tính toán là  $C = P^e$ , đó là

$$C = \boxed{475309123646226827206365550610545180942371796070491716523239243054 \\ 452960613199328566617843418359114151197411252005682979794571736036 \\ 101278218847892741566090480023507190715277185914975188465888632101 \\ 148354103361657898467968386763733765777465625079280521148141844048 \\ 14184430812773059004692874248559166462108656}$$

## 2.6 *Continued*

### Example 8      *Continued*

**Bob có thể khôi phục bản rõ từ bản mã bằng cách sử dụng  $P = C^d$ , đó là**

$$P = \begin{vmatrix} 1907081826081826002619041819 \end{vmatrix}$$

**Bản rõ được khôi phục là “THIS IS A TEST” sau khi giải mã.**

### 3 RABIN CRYPTOSYSTEM

*The Rabin cryptosystem can be thought of as an RSA cryptosystem in which the value of  $e$  and  $d$  are fixed. The encryption is  $C \equiv P^2 \pmod{n}$  and the decryption is  $P \equiv C^{1/2} \pmod{n}$ .*

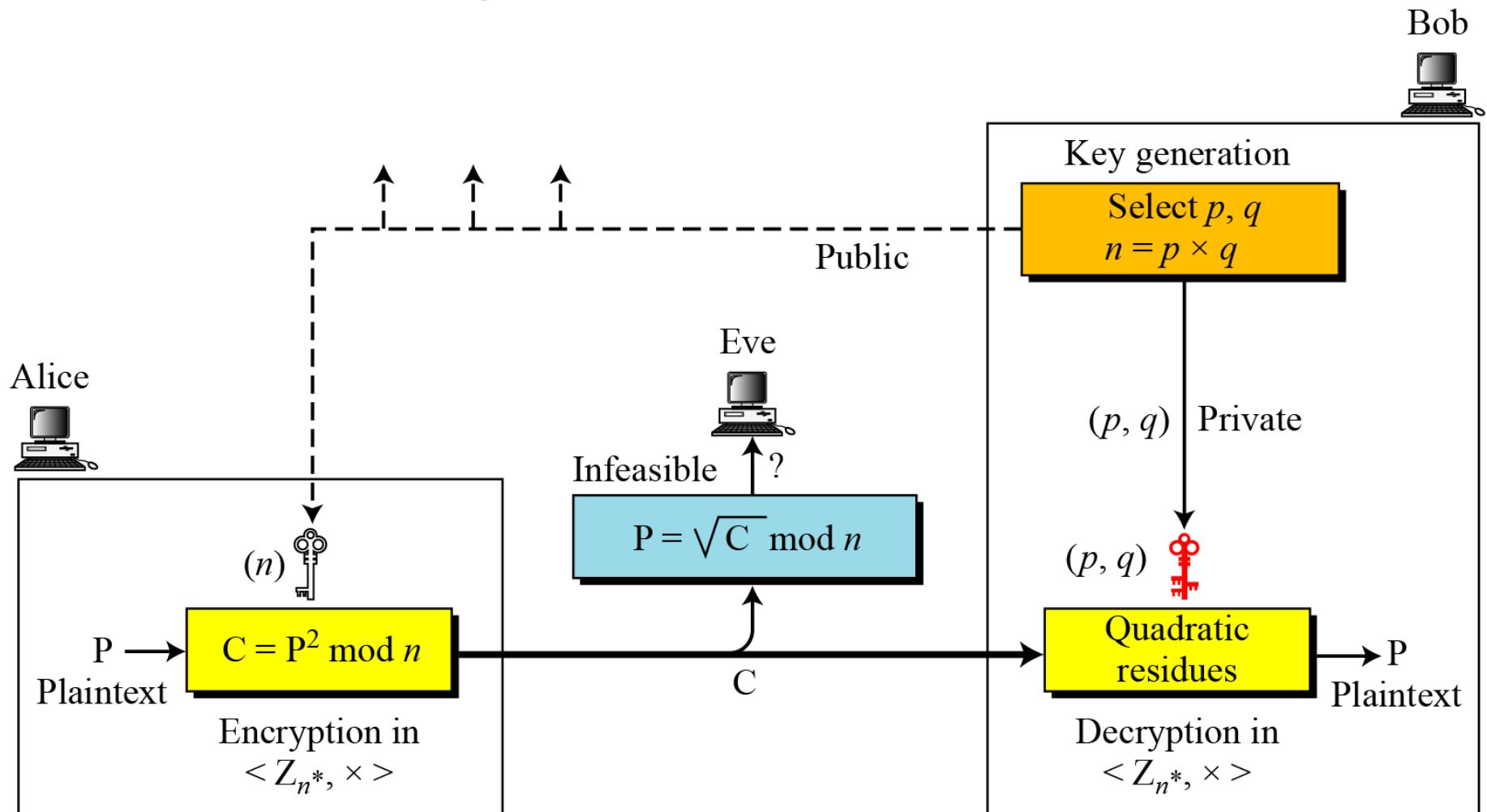
*Hệ thống mật mã Rabin có thể được coi là một hệ thống mật mã RSA trong đó giá trị của  $e$  và  $d$  là cố định. Mã hóa là  $C \equiv P^2 \pmod{n}$  và giải mã là  $P \equiv C^{1/2} \pmod{n}$ .*

**Topics discussed in this section:**

- 3.1    Procedure: Thủ tục
- 3.2    Security of the Rabin System

### 3 Continued

**Figure 10 Rabin cryptosystem**



## 3.1 Procedure

### Key Generation

**Algorithm 10.6** Key generation for Rabin cryptosystem

#### Rabin\_Key\_Generation

{

Choose two large primes  $p$  and  $q$  in the form  $4k + 3$  and  $p \neq q$ .

$n \leftarrow p \times q$

Public\_key  $\leftarrow n$  // To be announced publicly

Private\_key  $\leftarrow (q, n)$  // To be kept secret

return Public\_key and Private\_key

}

## 3.1 Continued

### Encryption

**Algorithm 10.7** *Encryption in Rabin cryptosystem*

```
Rabin_Encryption ( $n$ ,  $P$ )           //  $n$  is the public key;  $P$  is the ciphertext from  $\mathbf{Z}_n^*$ 
{
     $C \leftarrow P^2 \bmod n$            //  $C$  is the ciphertext
    return  $C$ 
}
```

# 3.1 Continued

## Decryption

**Algorithm 10.8 Decryption in Rabin cryptosystem**

```
Rabin_Decryption (p, q, C)           // C is the ciphertext; p and q are private keys
{
     $a_1 \leftarrow +(\text{C}^{(p+1)/4}) \text{ mod } p$ 
     $a_2 \leftarrow -(\text{C}^{(p+1)/4}) \text{ mod } p$ 
     $b_1 \leftarrow +(\text{C}^{(q+1)/4}) \text{ mod } q$ 
     $b_2 \leftarrow -(\text{C}^{(q+1)/4}) \text{ mod } q$ 
    // The algorithm for the Chinese remainder algorithm is called four times.
    P1  $\leftarrow$  Chinese_Remainder ( $a_1, b_1, p, q$ )
    P2  $\leftarrow$  Chinese_Remainder ( $a_1, b_2, p, q$ )
    P3  $\leftarrow$  Chinese_Remainder ( $a_2, b_1, p, q$ )
    P4  $\leftarrow$  Chinese_Remainder ( $a_2, b_2, p, q$ )
    return P1, P2, P3, and P4
}
```

**Note**

Hệ thống mật mã Rabin không có tính xác định: Việc giải mã tạo ra bốn bản rõ.

The Rabin cryptosystem is not deterministic:  
Decryption creates four plaintexts.

## 3.1 Continued

### Example 9

Here is a very trivial example to show the idea.

1. Bob selects  $p = 23$  and  $q = 7$ . Note that both are congruent to 3 mod 4.
2. Bob calculates  $n = p \times q = 161$ .
3. Bob announces  $n$  publicly; he keeps  $p$  and  $q$  private.
4. Alice wants to send the plaintext  $P = 24$ . Note that 161 and 24 are relatively prime; 24 is in  $\mathbb{Z}_{161}^*$ . She calculates  $C = 24^2 = 93$  mod 161, and sends the ciphertext 93 to Bob.

## 3.1 Continued

### Example 9

5. Bob receives 93 and calculates four values:

$$a_1 = +(93^{(23+1)/4}) \bmod 23 = 1 \bmod 23$$

$$a_2 = -(93^{(23+1)/4}) \bmod 23 = 22 \bmod 23$$

$$b_1 = +(93^{(7+1)/4}) \bmod 7 = 4 \bmod 7$$

$$b_2 = -(93^{(7+1)/4}) \bmod 7 = 3 \bmod 7$$

6. Bob takes four possible answers,  $(a_1, b_1)$ ,  $(a_1, b_2)$ ,  $(a_2, b_1)$ , and  $(a_2, b_2)$ , and uses the Chinese remainder theorem to find four possible plaintexts: 116, 24, 137, and 45. Note that only the second answer is Alice's plaintext.

## 4 ELGAMAL CRYPTOSYSTEM

*Besides RSA and Rabin, another public-key cryptosystem is ElGamal. ElGamal is based on the discrete logarithm problem.*

*Ngoài RSA và Rabin, một hệ thống mật mã khóa công khai khác là ElGamal. ElGamal dựa trên bài toán logarit rời rạc*

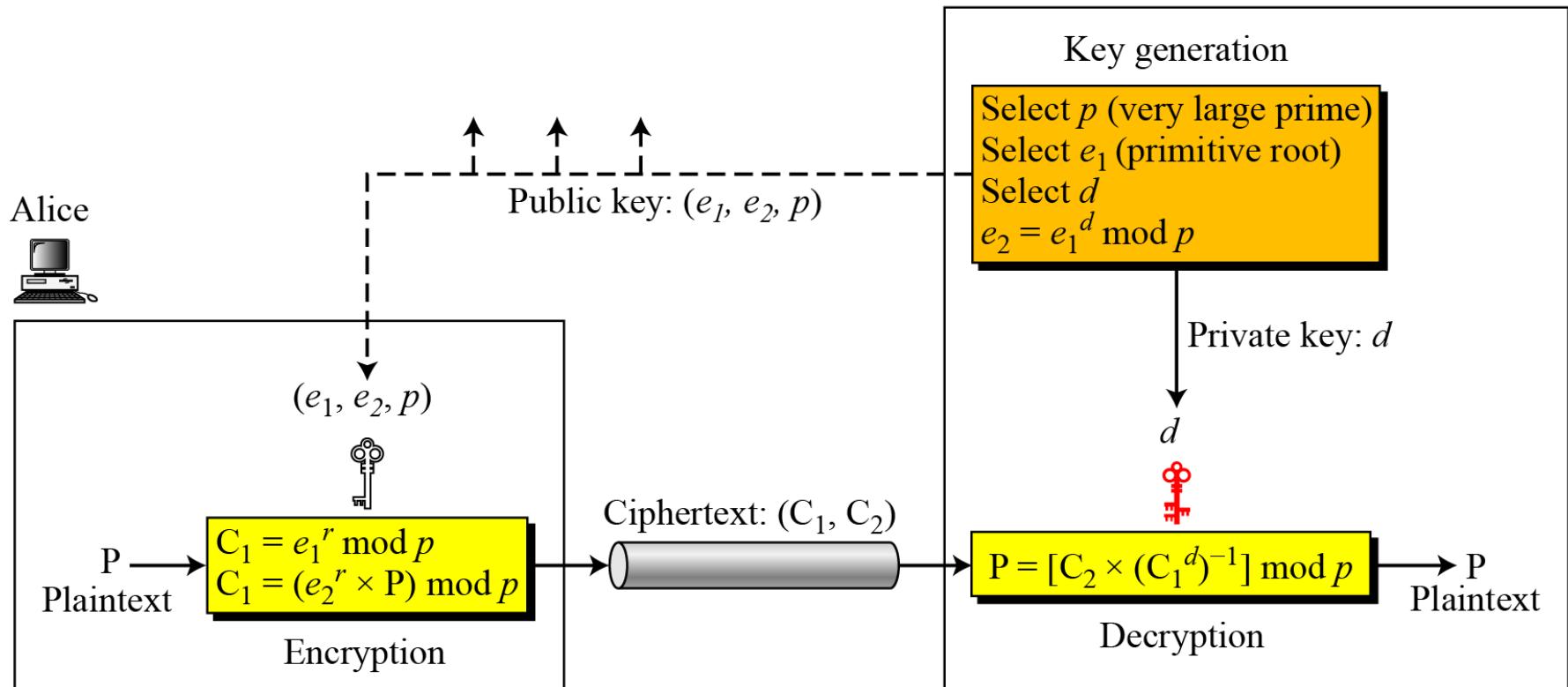
### Topics discussed in this section:

- 4.1 ElGamal Cryptosystem
- 4.2 Procedure
- 4.3 Proof
- 4.4 Analysis
- 4.5 Security of ElGamal
- 4.6 Application

## 4.2 Procedure

**Figure 11 Key generation, encryption, and decryption in ElGamal**

Bob  

$r = \log_{\{e_1\}} e_2 \text{ mod } p$ : bài toán logarit rời rạc

## 4.2 Continued

### Key Generation

**Algorithm 10.9** *ElGamal key generation*

**ElGamal\_Key\_Generation**

{

Select a large prime  $p$

Select  $d$  to be a member of the group  $\mathbf{G} = \langle \mathbf{Z}_p^*, \times \rangle$  such that  $1 \leq d \leq p - 2$

Select  $e_1$  to be a primitive root in the group  $\mathbf{G} = \langle \mathbf{Z}_p^*, \times \rangle$

$e_2 \leftarrow e_1^d \bmod p$

Public\_key  $\leftarrow (e_1, e_2, p)$  // To be announced publicly

Private\_key  $\leftarrow d$  // To be kept secret

return Public\_key and Private\_key

}

**Table 8.3 Powers of Integers, Modulo 19**

## 4.2 Continued

### Algorithm 10.10 ElGamal encryption

```
ElGamal_Encryption ( $e_1, e_2, p, P$ ) // P is the plaintext
{
    Select a random integer  $r$  in the group  $\mathbf{G} = \langle \mathbf{Z}_p^*, \times \rangle$ 
     $C_1 \leftarrow e_1^r \bmod p$ 
     $C_2 \leftarrow (P \times e_2^r) \bmod p$  //  $C_1$  and  $C_2$  are the ciphertexts
    return  $C_1$  and  $C_2$ 
}
```

## 4.2 *Continued*

### Algorithm 10.11 *ElGamal decryption*

```
ElGamal_Decryption (d, p, C1, C2) // C1 and C2 are the ciphertexts
{
    P ← [C2 (C1d)-1] mod p // P is the plaintext
    return P
}
```

**Độ phức tạp hoạt động bit của mã hóa hoặc giải mã trong hệ thống mật mã ElGamal là đa thức.**

**Note**

**The bit-operation complexity of encryption or decryption in ElGamal cryptosystem is polynomial.**

## 4.3 Continued

### Example 10

*Here is a trivial example. Bob chooses  $p = 11$  and  $e_1 = 2$ . and  $d = 3$   $e_2 = e_1^d = 8$ . So the public keys are  $(2, 8, 11)$  and the private key is 3. Alice chooses  $r = 4$  and calculates  $C1$  and  $C2$  for the plaintext 7.*

**Plaintext:** 7

$$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$$

$$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$$

**Ciphertext:** (5, 6)

*Bob receives the ciphertexts (5 and 6) and calculates the plaintext.*

$$[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$$

**Plaintext:** 7

## 4.3 Continued

### Example 10

Đây là một ví dụ nhỏ. Bob chọn  $p = 11$  và  $e_1 = 2$ . và  $d = 3$   $e_2 = e_1^d = 8$ . Vậy khóa công khai là  $(2, 8, 11)$  và khóa riêng là  $3$ . Alice chọn  $r = 4$  và tính  $C1$  và  $C2$  cho bản rõ  $7$ .

**Plaintext:** 7

$$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$$

$$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$$

**Ciphertext:** (5, 6)

Bob nhận bản mã (5 và 6) và tính toán bản rõ.

$$[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$$

**Plaintext:** 7

## 4.3 Continued

### Example 11

Instead of using  $P = [C_2 \times (C_1^d)^{-1}] \bmod p$  for decryption, we can avoid the calculation of multiplicative inverse and use  $P = [C_2 \times C_1^{p-1-d}] \bmod p$  (see Fermat's little theorem in Chapter 9). In Example 10.10, we can calculate  $P = [6 \times 5^{11-1-3}] \bmod 11 = 7 \bmod 11$ .

**Note**

For the ElGamal cryptosystem,  $p$  must be at least 300 digits and  $r$  must be new for each encipherment.

## 4.3 Continued

### Example 12

*Bob uses a random integer of 512 bits. The integer  $p$  is a 155-digit number (the ideal is 300 digits). Bob then chooses  $e_1$ ,  $d$ , and calculates  $e_2$ , as shown below:*

$p =$	115348992725616762449253137170143317404900945326098349598143469219 056898698622645932129754737871895144368891765264730936159299937280 61165964347353440008577
$e_1 =$	2
$d =$	1007
$e_2 =$	978864130430091895087668569380977390438800628873376876100220622332 554507074156189212318317704610141673360150884132940857248537703158 2066010072558707455

## 4.3 Continued

### Example 10

Alice has the plaintext  $P = 3200$  to send to Bob. She chooses  $r = 545131$ , calculates  $C_1$  and  $C_2$ , and sends them to Bob.

$P =$	3200
$r =$	545131
$C_1 =$	887297069383528471022570471492275663120260067256562125018188351429 417223599712681114105363661705173051581533189165400973736355080295 736788569060619152881
$C_2 =$	708454333048929944577016012380794999567436021836192446961774506921 244696155165800779455593080345889614402408599525919579209721628879 6813505827795664302950

Bob calculates the plaintext  $P = C_2 \times ((C_1)^d)^{-1} \bmod p = 3200 \bmod p$ .

$P =$	3200
-------	------

# **Cryptographic Hash Functions**

**Hàm băm mật mã**

# Nội dung

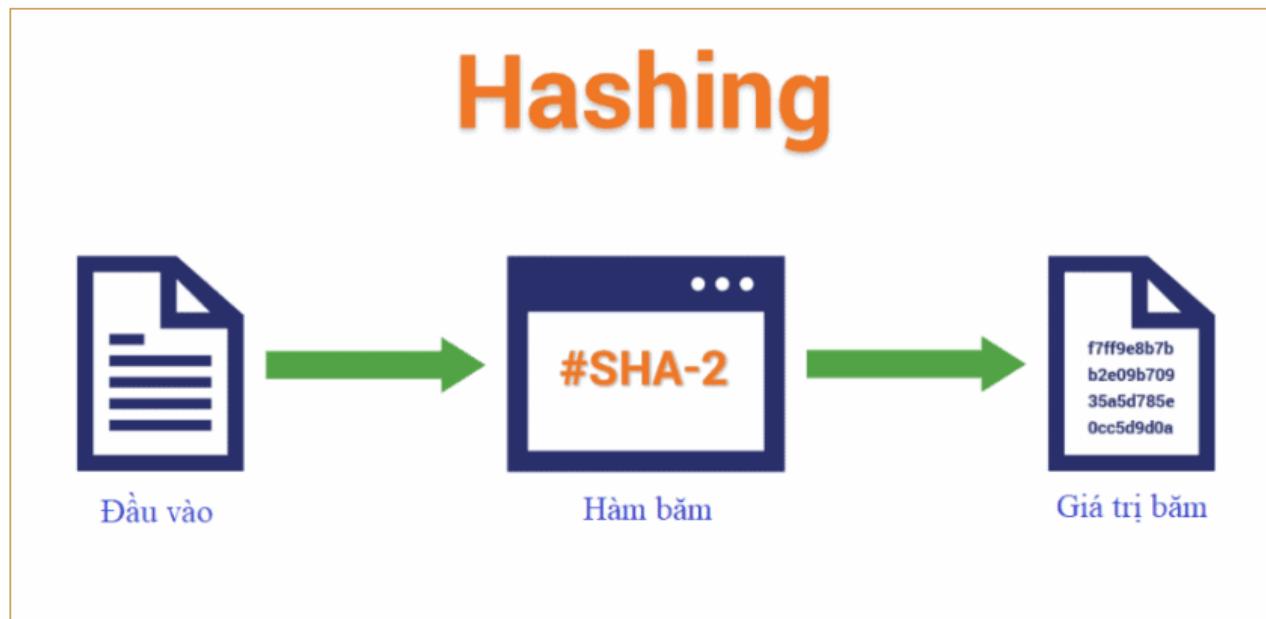
- Giới thiệu những ý tưởng chung đằng sau các hàm băm mật mã
- Thảo luận về lược đồ Merkle-Damgard làm cơ sở cho các hàm băm được lắp lại
- Phân biệt giữa hai loại hàm băm:
  - Cấu trúc của SHA-512.
  - Cấu trúc của Whirlpool.



# 1 GIỚI THIỆU

## Hash là gì?

- ❖ Về cơ bản hashing là quá trình biến một dữ liệu đầu vào có độ dài bất kỳ thành một chuỗi đầu ra đặc trưng có độ dài cố định. Hashing được thực hiện thông qua hàm băm (hash function).
- ❖ Một cách tổng quát hàm băm là bất kỳ hàm nào có thể được sử dụng để ánh xạ dữ liệu có kích thước tùy ý thành các giá trị kích thước cố định. Các giá trị được trả về bởi hàm băm được gọi là giá trị băm, mã băm, thông điệp băm, hoặc đơn giản là “hash”.



# 1 GIỚI THIỆU

- ❖ Ví dụ, khi bạn download một video trên YouTube có dung lượng **50 MB** và thực hiện hashing trên nó bằng **thuật toán băm SHA-256**, thì đầu ra bạn thu được sẽ là một giá trị băm có độ dài **256 bit**. Tương tự, nếu bạn lấy một tin nhắn văn bản có dung lượng **5 KB**, để hashing bằng **SHA-256** thì giá trị băm đầu ra bạn thu được vẫn sẽ là **256 bit**.
- ❖ Trong blockchain, các giao dịch có độ dài khác nhau sẽ được băm thông qua một thuật toán băm nhất định và tất cả đều cho đầu ra có độ dài cố định bất kể độ dài của giao dịch đầu vào là bao nhiêu. Chẳng hạn, Bitcoin sử dụng thuật toán **SHA-256** để băm các giao dịch cho kết quả đầu ra có độ dài cố định là **256 bit (32 byte)** cho dù giao dịch chỉ là một từ hoặc giao dịch phức tạp với lượng dữ liệu khổng lồ.
- ❖ Kỹ thuật hashing thường được sử dụng và có ứng dụng rộng rãi nhất trong việc đảm bảo tính toàn vẹn cho dữ liệu trong blockchain là các hàm băm mật mã (*cryptographic hash function*) chẳng hạn như **SHA-1. SHA-2. SHA-3, SHA-256...** Sỡ dĩ như vậy là do các hàm băm mật mã có một số tính chất quan trọng phù hợp cho việc đảm bảo an toàn dữ liệu.

# 1 GIỚI THIỆU

*Một hàm băm mật mã nhận một thông điệp có độ dài tùy ý và tạo ra một thông báo có độ dài cố định. Mục tiêu của chương này là thảo luận chi tiết về hai thuật toán băm mật mã hứa hẹn nhất – **SHA-512** and **Whirlpool***

**Chủ đề trong phần này gồm:**

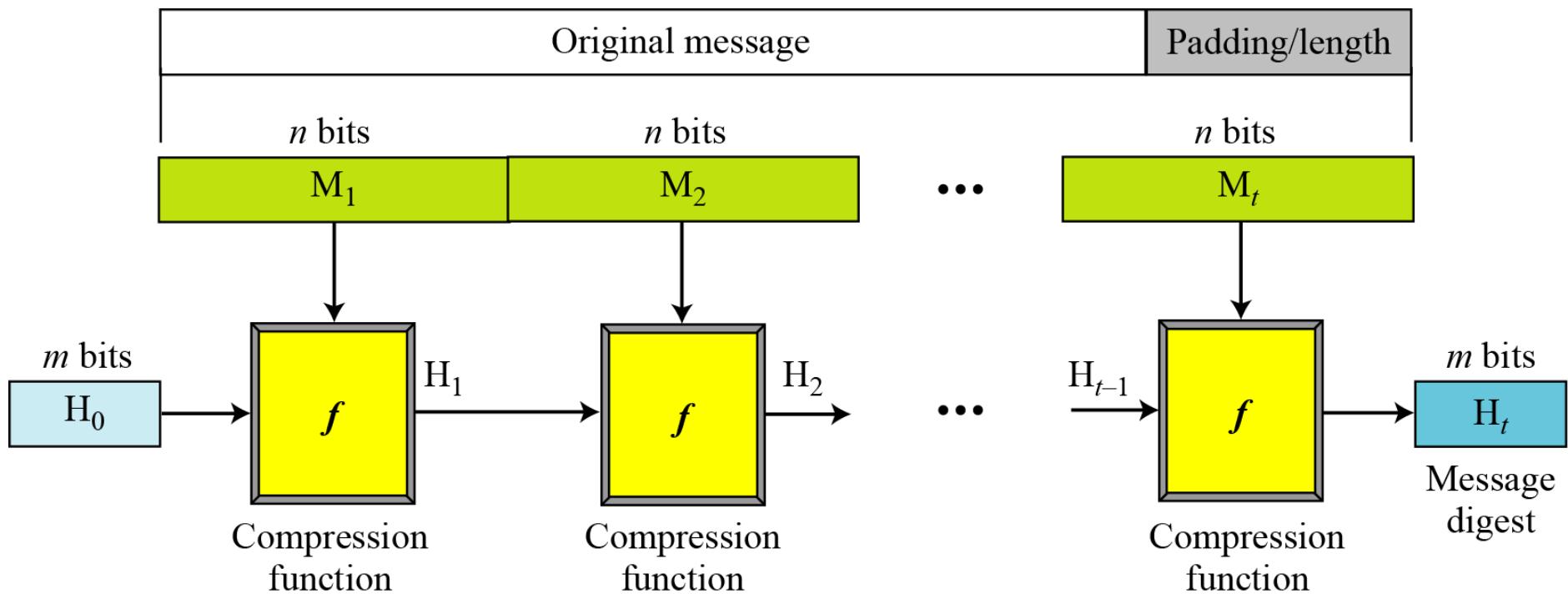
- 1.1 Hàm băm được lặp lại
- 1.2 Hai nhóm chức năng nén

# 1.1 Hàm băm được lắp lại

## Iterated Hash Function

### Lược đồ Merkle-Damgard

Figure 1 Lược đồ Merkle-Damgard



$H_0$ : là giá trị / vec tơ khởi tạo

$H_i = f(H_{i-1}, M_i)$ , trong đó  $f$  là hàm nén.

$H_t$  là hàm băm mã hóa của bản tin gốc, tức là  $h(M)$ .

## *1.2 Hai nhóm hàm nén*

### *Two Groups of Compression Functions*

*1. Chức năng nén được thực hiện từ đầu.*

*Tóm lược thông điệp  
Message Digest (MD)*

*2. Mật mã khôi khóa đối xứng đóng vai trò như một hàm nén.*

*Xoáy nước  
Whirlpool*

## 1.2 Hai nhóm hàm nén

### Two Groups of Compression Functions

1. *Chức năng nén được thực hiện từ đầu.*

**Tóm lược thông điệp Message Digest  
(MD)**

Có nhiều phiên bản MD như MD2, MD4, **MD5** được thiết kế bởi Ron Rivest. MD5 chia bản tin thành các khối 512 bits để tạo thành một digest 128 bit.

SHA (*secure hash algorithm*): *chuẩn băm bảo mật* được NIST công bố theo FIP 180. SHA dựa trên cấu trúc MD5 bao gồm có SHA-1, SHA-224, SHA-256, SHA-384, và SHA-512.

## 1.2 *Continued*

**Table 12.1** *Characteristics of Secure Hash Algorithms (SHAs)*

<i>Characteristics</i>	<i>SHA-1</i>	<i>SHA-224</i>	<i>SHA-256</i>	<i>SHA-384</i>	<i>SHA-512</i>
Maximum Message size	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$	$2^{128} - 1$
Block size	512	512	512	1024	1024
Message digest size	160	224	256	384	512
Number of rounds	80	64	64	80	80
Word size	32	32	32	64	64

**Table 12.8 A Comparison of MD5, SHA-1, and RIPEMD-160**

	<b>MD5</b>	<b>SHA-1</b>	<b>RIPEMD-160</b>
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	$\infty$	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

**Table 12.9 Relative Performance of Several Hash Functions  
(coded in C++ on a 850 MHz Celeron)**

Algorithm	MBps
MD5	26
SHA-1	48
RIPEMD-160	31

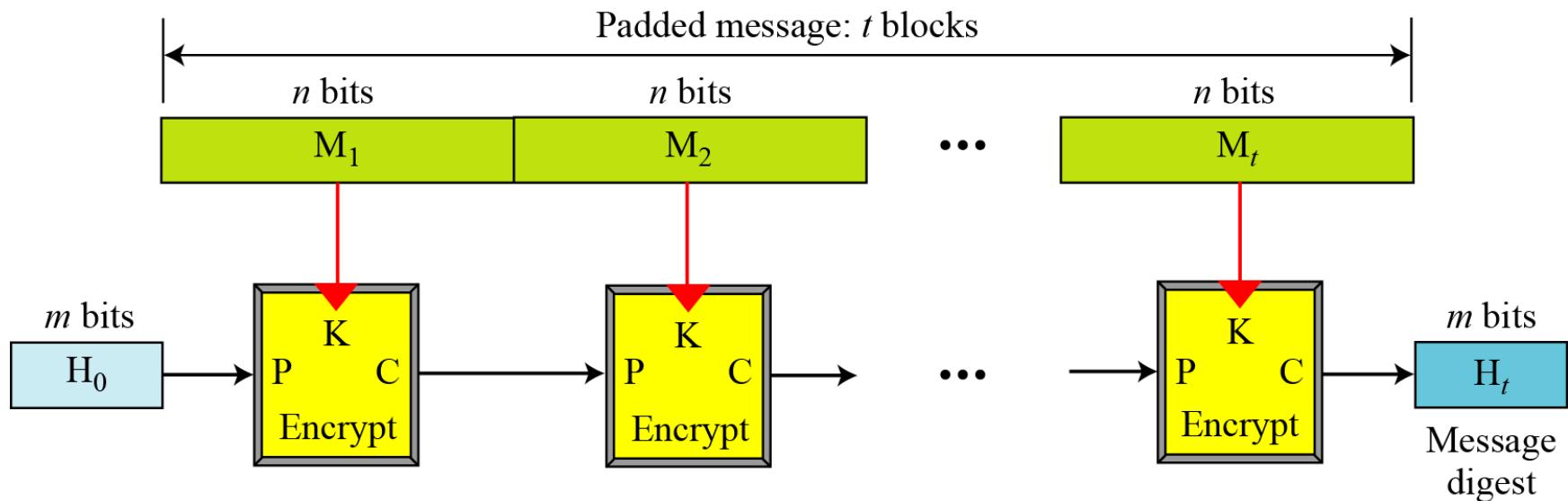
Note: Coded by Wei Dai; results are posted at <http://www.eskimo.com/~weidai/benchmarks.html>

## 1.2 Continued

### Rabin Scheme: Lược đồ Rabin

Lược đồ đơn giản dựa trên Merkle-Damgard: Hàm nén được thay thế bởi một hệ mật mã nào đó, các khối bản tin sử dụng như là Khóa (K), bản tin digest trước mỗi hàm là bản tin gốc (P) của nó. Bản tin C là bản tin digest mới tạo ra bởi hàm đó. Chú ý: kích thước của digest bằng kích thước của mã khối dữ liệu. Ví dụ trong hệ mật DES thì nó (digest) chỉ bằng 64 bit.

Figure 2 Rabin scheme

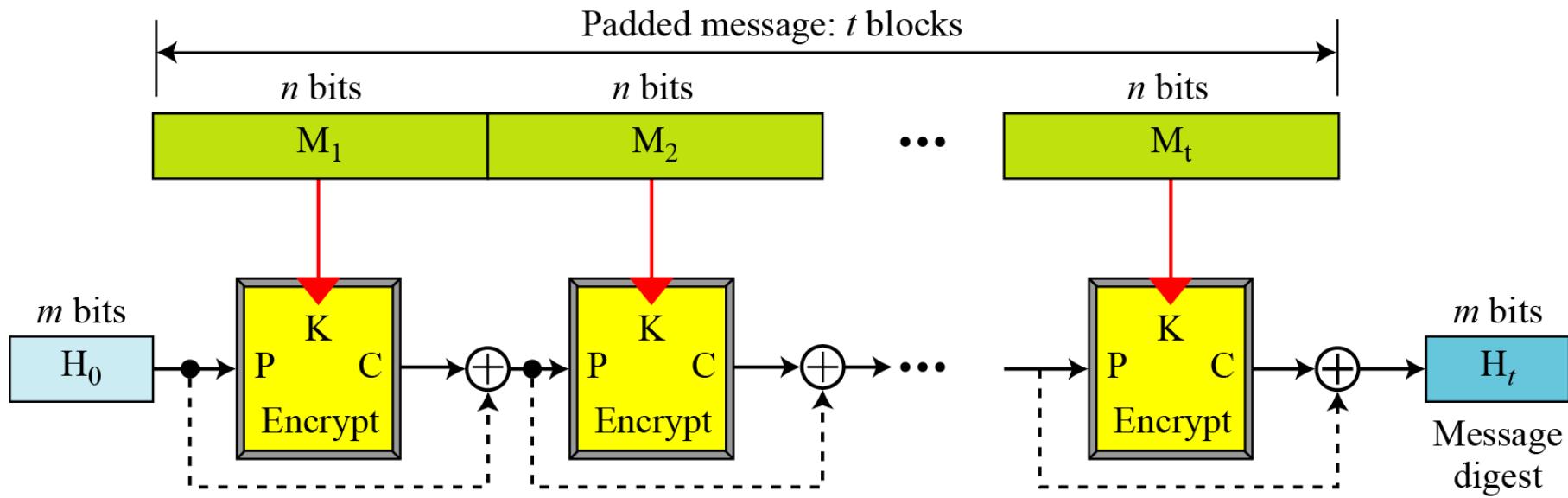


# 1.2 Continued

## Lược đồ Davies-Meyer:

Lược đồ này dựa trên lược đồ Rabin, có thêm phần *XOR* dữ liệu giữa  $C$  và  $P$  sau mỗi hàm băm  $\rightarrow$  nhằm tăng độ bảo mật cho tấn công vào quá trình trung gian (tấn công **meet-in-the-middle**)

Figure 3 Lược đồ Davies-Meyer

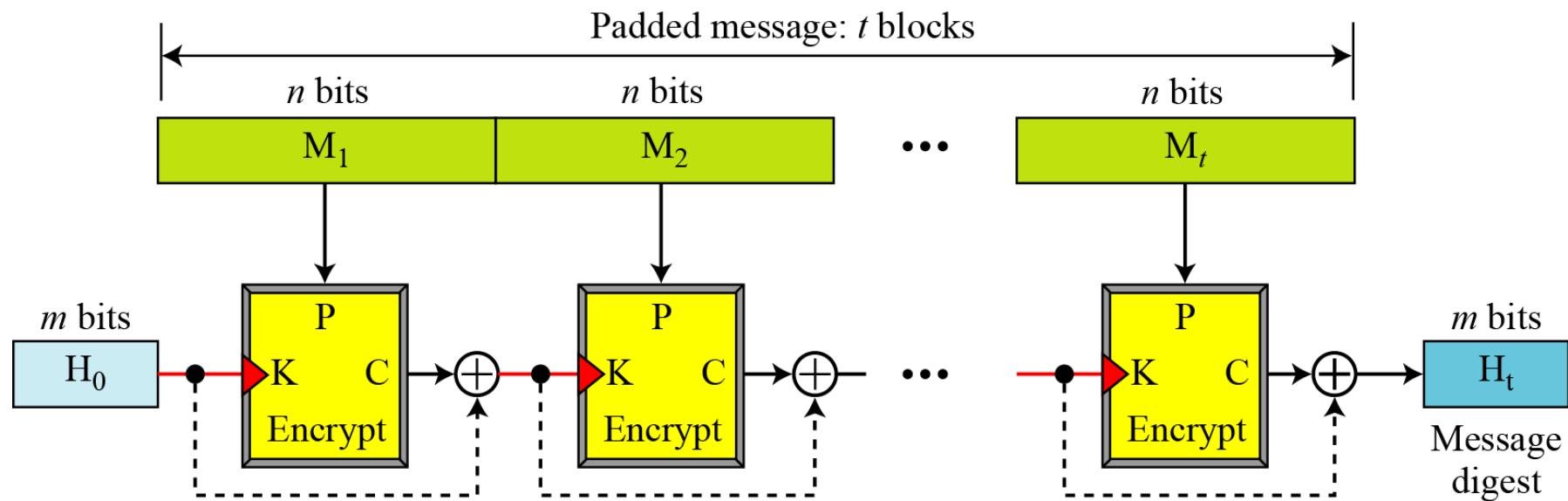


## 1.2 Continued

### Lược đồ Matyas-Meyer-Oseas

Đây là phiên bản kép của Davis-Meyer; các khối bản tin sử dụng cho khóa mật. Lược đồ này sử dụng tốt nhất khi khóa mật và khối bản tin cùng kích thước. Khi đó hệ AES được sử dụng là tốt nhất.

Figure 4 Lược đồ Matyas-Meyer-Oseas scheme

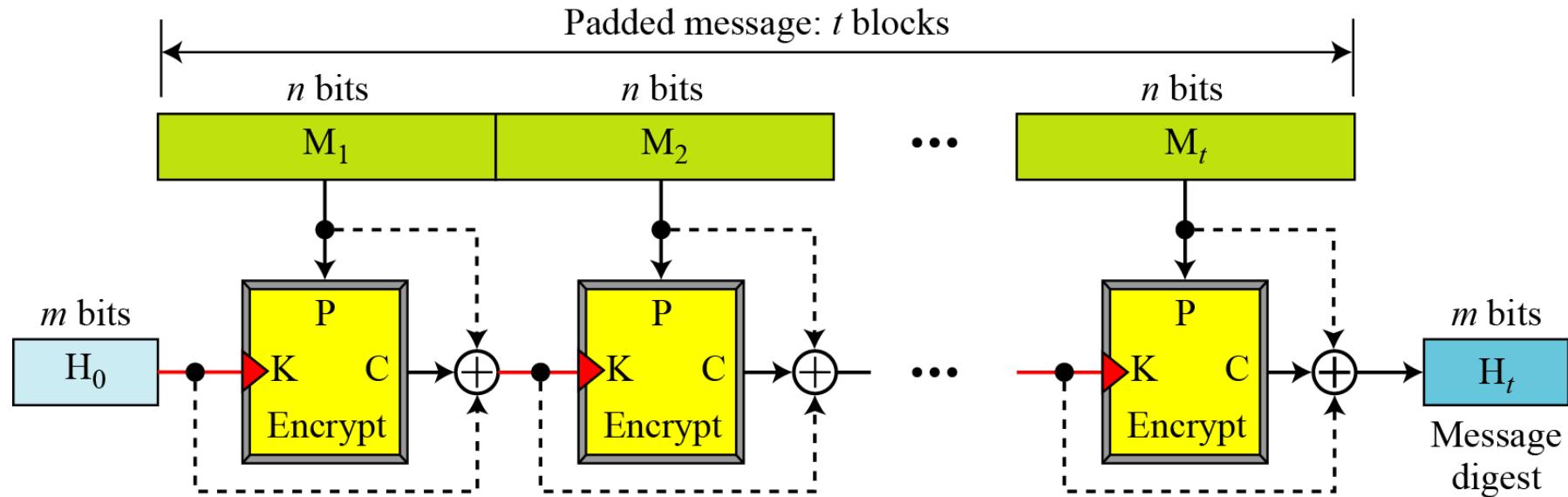


## 1.2 Continued

### Lược đồ Miyaguchi-Preneel:

Là phiên bản mở rộng của lược đồ Matyas-Meyer-Oseas; ở đây phép XOR bởi 3 luồng bản tin  $\{M_t, C, H_{t-1}\}$  tạo ra bản tin digest mới ( $H_t$ )  $\rightarrow$  tăng cường tính bảo mật cho tấn công trung gian.

Figure 5 Lược đồ Miyaguchi-Preneel



## 2 SHA-512

*SHA-512 is the version of SHA with a 512-bit message digest. This version, like the others in the SHA family of algorithms, is based on the Merkle-Damgard scheme.*

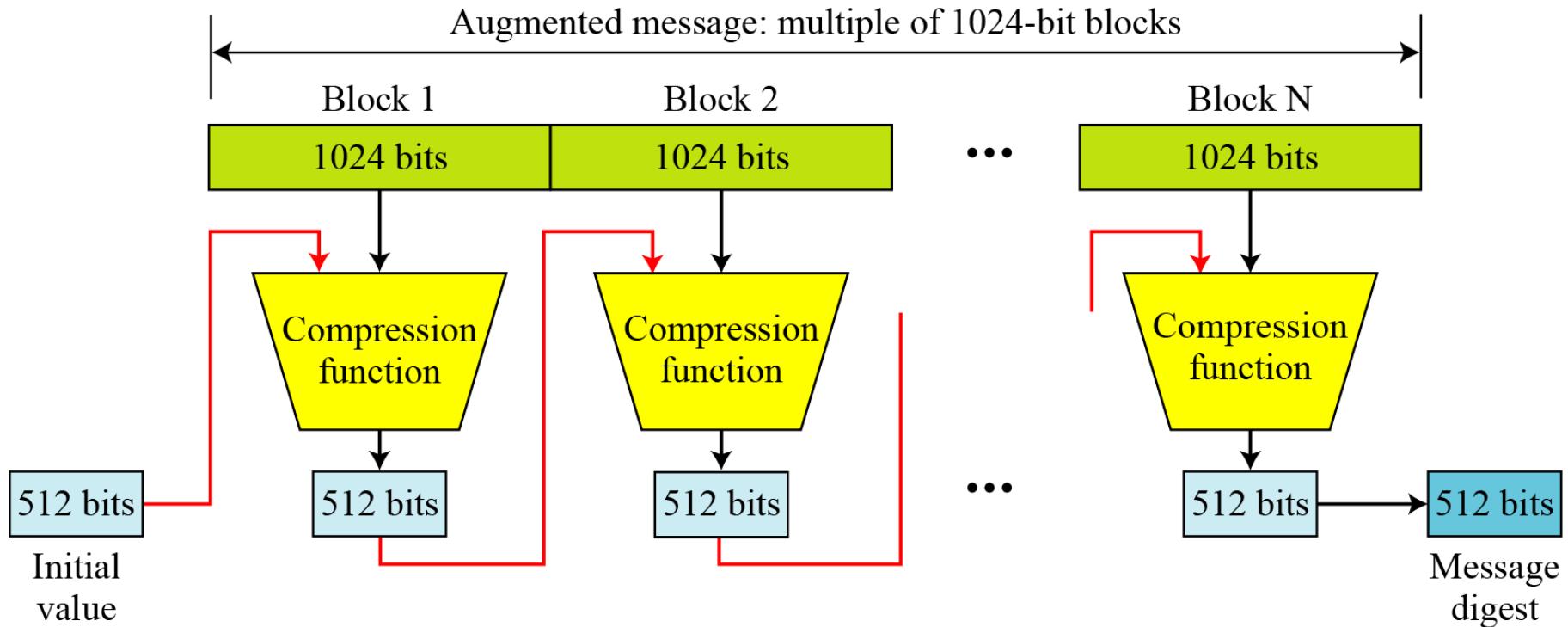
*SHA-512 là phiên bản của SHA với bản tóm tắt thông báo 512 bit. Phiên bản này, giống như các phiên bản khác trong họ thuật toán SHA, dựa trên lược đồ Merkle-Damgard.*

### *Topics discussed in this section:*

- 2.1      Introduction**
- 2.2      Compression Function**
- 2.3      Analysis**

## 2.1 Introduction

**Figure 6 Message digest creation SHA-512**



## 2.1 Continued

### *Message Preparation*

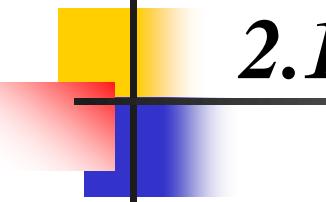
*SHA-512 insists that the length of the original message be less than  $2^{128}$  bits.*

*SHA-512 khẳng định rằng độ dài của bản tin gốc phải nhỏ hơn  $2^{128}$  bit.*

#### **Note**

**SHA-512 creates a 512-bit message digest out of a message less than  $2^{128}$ .**

*SHA-512 tạo bản tóm tắt tin nhắn 512 bit từ một bản tin nhỏ hơn  $2^{128}$ .*



## 2.1 *Continued*

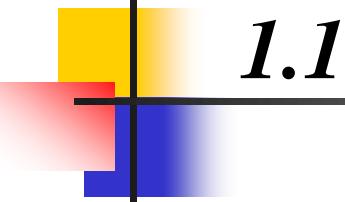
### Example 1

This example shows that the message length limitation of SHA-512 is not a serious problem. Suppose we need to send a message that is  $2^{128}$  bits in length. How long does it take for a communications network with a data rate of  $2^{64}$  bits per second to send this message?

Ví dụ này cho thấy rằng giới hạn độ dài tin nhắn của SHA-512 không phải là một vấn đề nghiêm trọng. Giả sử chúng ta cần gửi một tin nhắn có độ dài  $2^{128}$  bit. Mất bao lâu để một mạng truyền thông với tốc độ dữ liệu  $2^{64}$  bit / giây gửi thông điệp này?

### Solution

A communications network that can send  $2^{64}$  bits per second is not yet available. Even if it were, it would take many years to send this message. This tells us that we do not need to worry about the SHA-512 message length restriction.



## 1.1 *Continued*

### Example 1

This example shows that the message length limitation of SHA-512 is not a serious problem. Suppose we need to send a message that is  $2^{128}$  bits in length. How long does it take for a communications network with a data rate of  $2^{64}$  bits per second to send this message?

Ví dụ: giới hạn độ dài tin nhắn của SHA-512 không phải là một vấn đề nghiêm trọng. Giả sử chúng ta cần gửi một tin nhắn có độ dài  $2^{128}$  bits. Mất bao lâu để một mạng truyền thông với tốc độ dữ liệu  $2^{64}$  bit / giây gửi thông điệp này?

## 2.1 *Continued*

### Example 2

Hỏi cần bao nhiêu pages dùng cho bản tin có độ dài  $2^{128}$  bits

#### Solution

Suppose that a character is 32, or  $2^6$ , bits. Each page is less than 2048, or approximately  $2^{12}$ , characters. So  $2^{128}$  bits need at least  $2^{128} / 2^{18}$ , or  $2^{110}$ , pages. This again shows that we need not worry about the message length restriction.

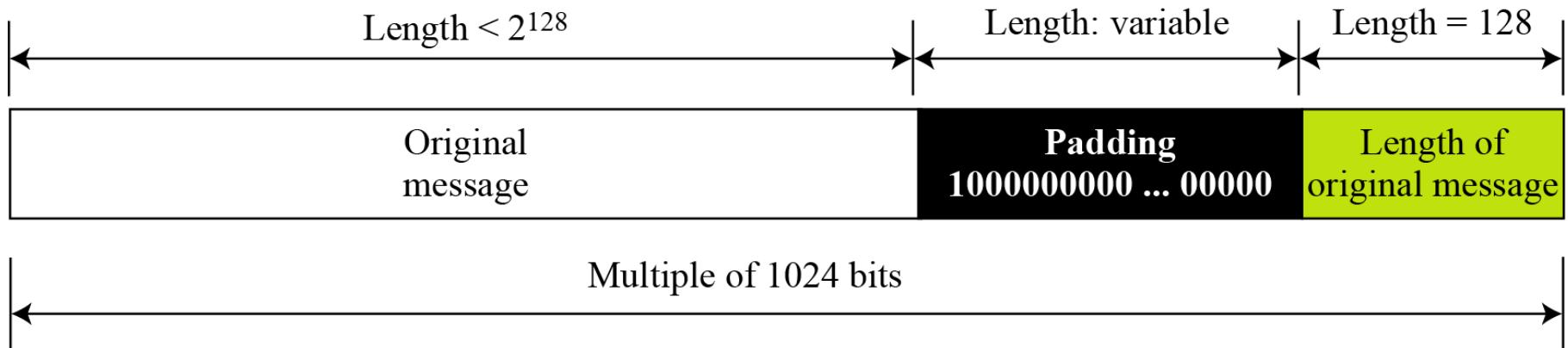
Giả sử rằng một ký tự là 32, hoặc  $2^6$ , bit. Mỗi trang ít hơn 2048, hoặc khoảng  $2^{12}$ , ký tự. Vì vậy,  $2^{128}$  bit cần ít nhất  $2^{128} / 2^{6+12}$  hoặc  $2^{110}$ , trang. Điều này một lần nữa cho thấy rằng chúng ta không cần phải lo lắng về giới hạn độ dài tin nhắn.

## 2.1 Continued

### Trường Padding và length

**Figure 7 Padding and length field in SHA-512**

**Trường đệm và độ dài trong SHA-512**



$$\begin{aligned} & (M+P+128) = 0 \bmod 1024 \\ \rightarrow & P = (-M-128) \bmod 1024 \end{aligned}$$

## 2.1 *Continued*

### Example 3

What is the number of padding bits if the length of the original message is 2590 bits?

*Số bit đệm Padding là bao nhiêu nếu độ dài của bản tin gốc là 2590 bit?*

#### Solution

Chúng ta có thể tính số bit đệm như sau:

$$|P| = (-2590 - 128) \bmod 1024 = -2718 \bmod 1024 = 354$$

The padding consists of one 1 followed by 353 0's.

Phần đệm bao gồm một số 1 sau là 353 số 0.

## 2.1 *Continued*

### Example 4

**Do we need padding if the length of the original message is already a multiple of 1024 bits?**

**Chúng ta có cần đệm không nếu độ dài của tin nhắn gốc đã là bội số của 1024 bit?**

### Solution

**Yes we do, because we need to add the length field. So padding is needed to make the new block a multiple of 1024 bits.**

**Có, bởi vì chúng ta cần thêm trường độ dài. Vì vậy, cần đệm để tạo khối mới là bội số của 1024 bits.**

## 2.1   Continued

### Example 5

What is the minimum and maximum number of padding bits that can be added to a message?

Số bit đệm tối thiểu và tối đa có thể được thêm vào bản tin là bao nhiêu?

### Solution

- a. The minimum length of padding is 0 and it happens when  $(-M - 128) \bmod 1024$  is 0. This means that  $|M| = -128 \bmod 1024 = 896 \bmod 1024$  bits. In other words, the last block in the original message is 896 bits. We add a 128-bit length field to make the block complete.

Độ dài tối thiểu của padding là 0 và nó xảy ra khi  $(-M - 128) \bmod 1024$  bằng 0. Điều này có nghĩa là  $|M| = -128 \bmod 1024 = 896 \bmod 1024$  bit. Nói cách khác, khối cuối cùng trong thông điệp gốc là 896 bit. Chúng ta thêm trường độ dài 128 bit để làm cho khối hoàn chỉnh.

## 2.1 *Continued*

### Example 5

### *Continued*

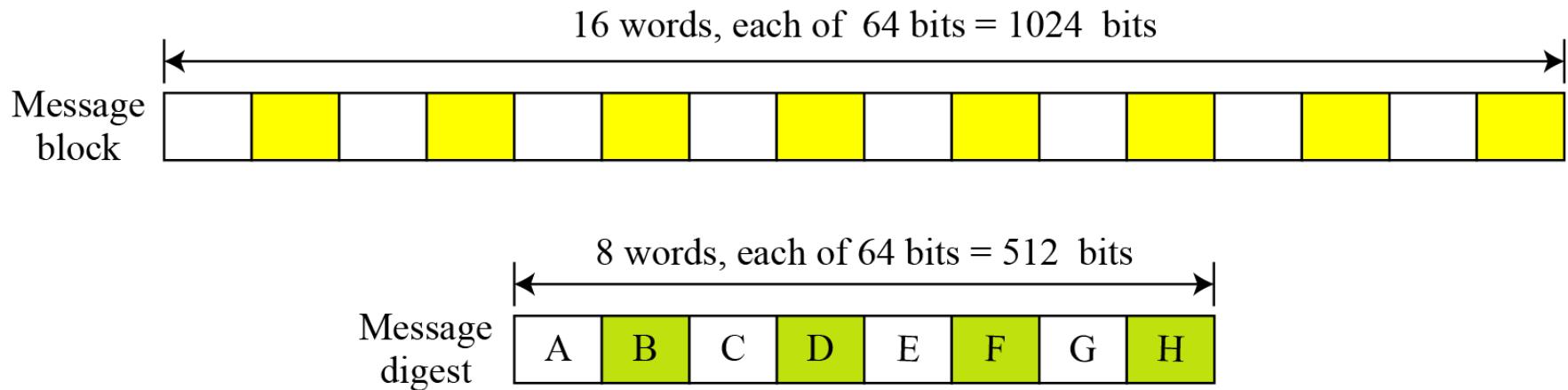
- b) The maximum length of padding is 1023 and it happens when  $(-|M| - 128) = 1023 \bmod 1024$ . This means that the length of the original message is  $|M| = (-128 - 1023) \bmod 1024$  or the length is  $|M| = 897 \bmod 1024$ . In this case, we cannot just add the length field because the length of the last block exceeds one bit more than 1024. So we need to add 897 bits to complete this block and create a second block of 896 bits. Now the length can be added to make this block complete.

*Độ dài tối đa của vùng đệm là 1023 và điều đó xảy ra khi  $(- | M | - 128) = 1023 \bmod 1024$ . Điều này có nghĩa là độ dài của thông điệp gốc là  $| M | = (-128 - 1023) \bmod 1024$  hoặc độ dài là  $| M | = 897 \bmod 1024$ . Trong trường hợp này, chúng ta không thể chỉ thêm trường độ dài vì độ dài của khối cuối cùng vượt quá một bit nhiều hơn 1024. Vì vậy, chúng ta cần thêm 897 bit để hoàn thành khối này và tạo khối thứ hai gồm 896 bit. Bây giờ chiều dài có thể được thêm vào để làm cho khối này hoàn chỉnh.*

## 2.1 Continued

### Words

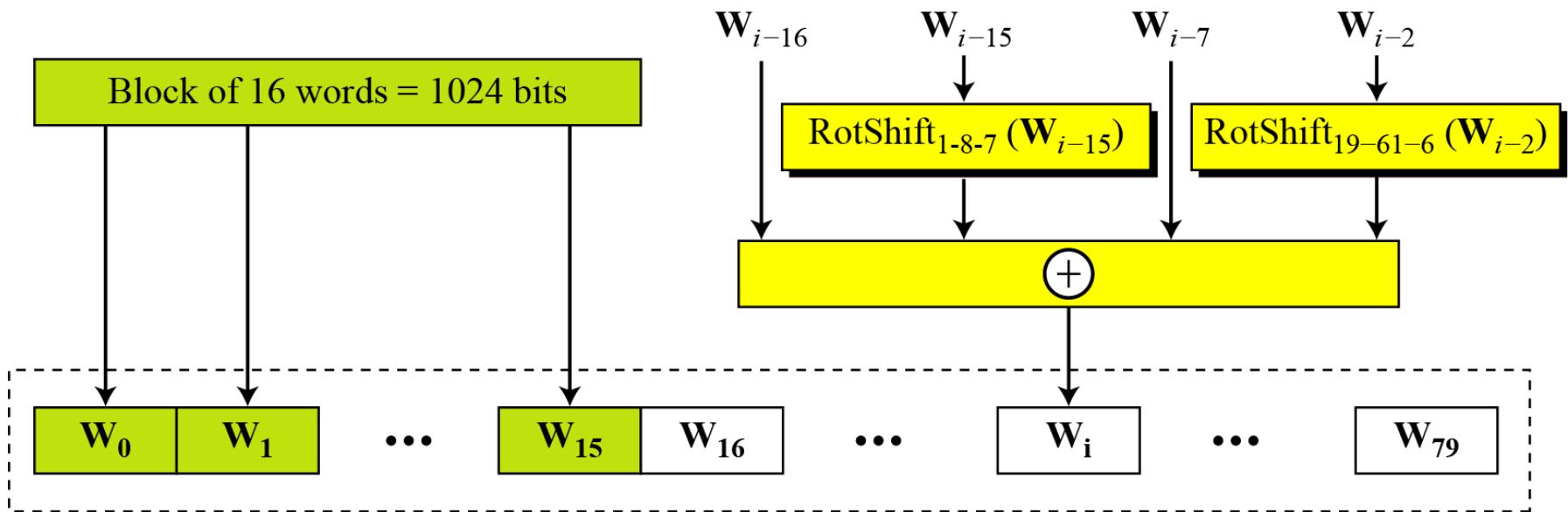
**Figure 12.8 A message block and the digest as words**  
*Một khối bản tin và thông báo dưới dạng các từ*



## 2.1 Continued

### Word Expansion: Mở rộng từ

Figure 9 Word expansion in SHA-512



$\text{RotShift}_{l-m-n}(x)$ :  $\text{RotR}_l(x) \oplus \text{RotR}_m(x) \oplus \text{ShL}_n(x)$

$\text{RotR}_i(x)$ : Right-rotation of the argument  $x$  by  $i$  bits

$\text{ShL}_i(x)$ : Shift-left of the argument  $x$  by  $i$  bits and padding the left by 0's.

## 2.1 *Continued*

### Example 6

Show how W60 is made?

Cho biết W60 được tạo ra như thế nào.

#### Solution

Each word in the range W16 to W79 is made from four previously-made words. W60 is made as

Mỗi từ trong phạm vi W16 đến W79 được tạo từ bốn từ đã tạo trước đó. W60 được làm như

$$W_{60} = W_{44} \oplus \text{RotShift}_{1-8-7}(W_{45}) \oplus W_{53} \oplus \text{RotShift}_{19-61-6}(W_{58})$$

## 2.1 Continued

### Message Digest Initialization

**Table 12.2** Values of constants in message digest initialization of SHA-512

Buffer	Value (in hexadecimal)	Buffer	Value (in hexadecimal)
A <sub>0</sub>	<b>6A09E667F3BCC908</b>	E <sub>0</sub>	<b>510E527FADE682D1</b>
B <sub>0</sub>	<b>BB67AE8584CAA73B</b>	F <sub>0</sub>	<b>9B05688C2B3E6C1F</b>
C <sub>0</sub>	<b>3C6EF372EF94F828</b>	G <sub>0</sub>	<b>1F83D9ABFB41BD6B</b>
D <sub>0</sub>	<b>A54FE53A5F1D36F1</b>	H <sub>0</sub>	<b>5BE0CD19137E2179</b>

Các giá trị khởi tạo trên được tính toán từ 8 số nguyên tố sau :  
2, 3, 5, 7, 11, 13, 17, và 19.

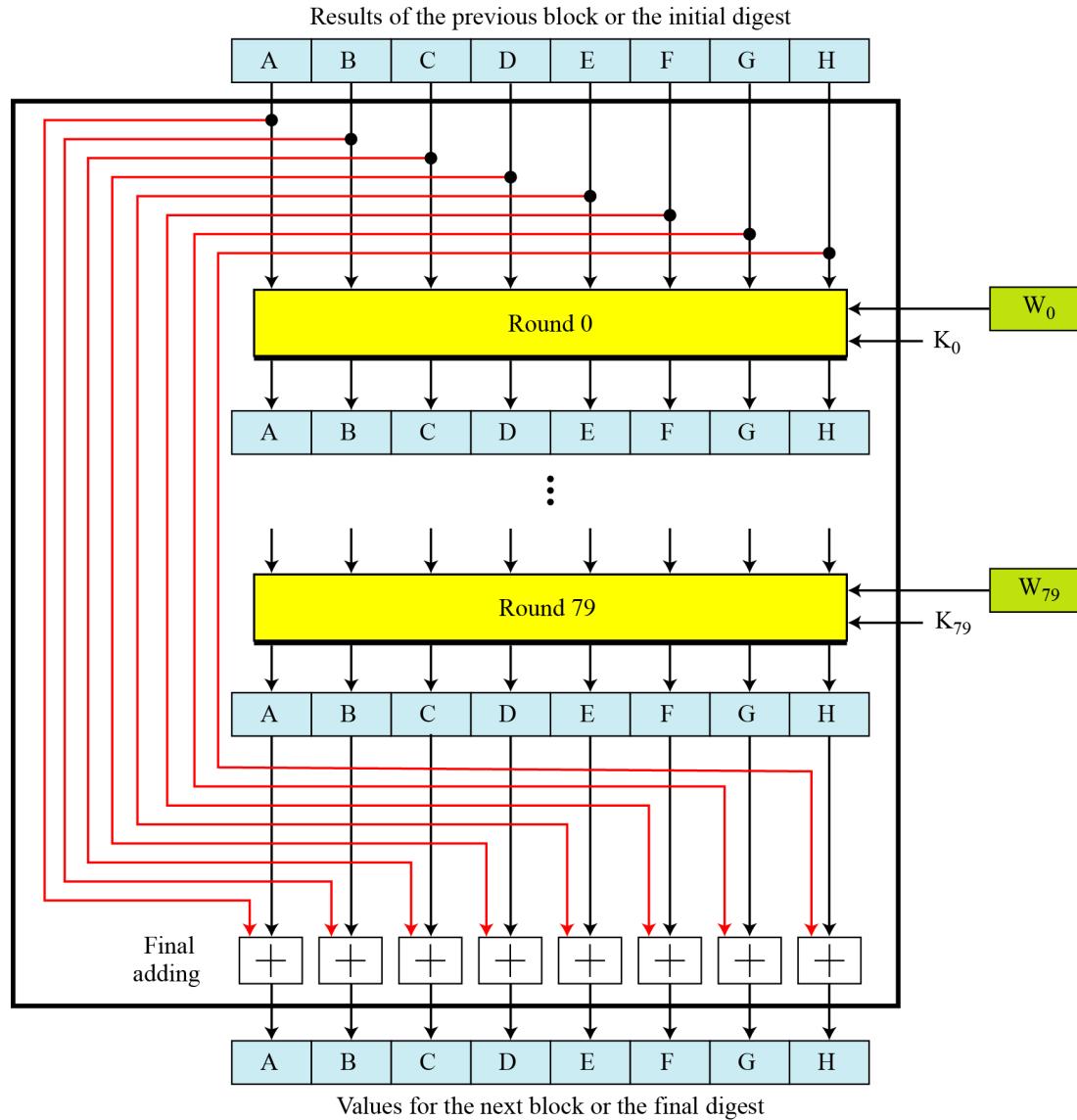
Sau đó **mũ ½** → chuyển đổi thành nhị phân và chỉ giữ lại 64 bits đầu.

Ví dụ:  $\sqrt{19}=4.35889894354\dots$

→0100.0010 0101 ....

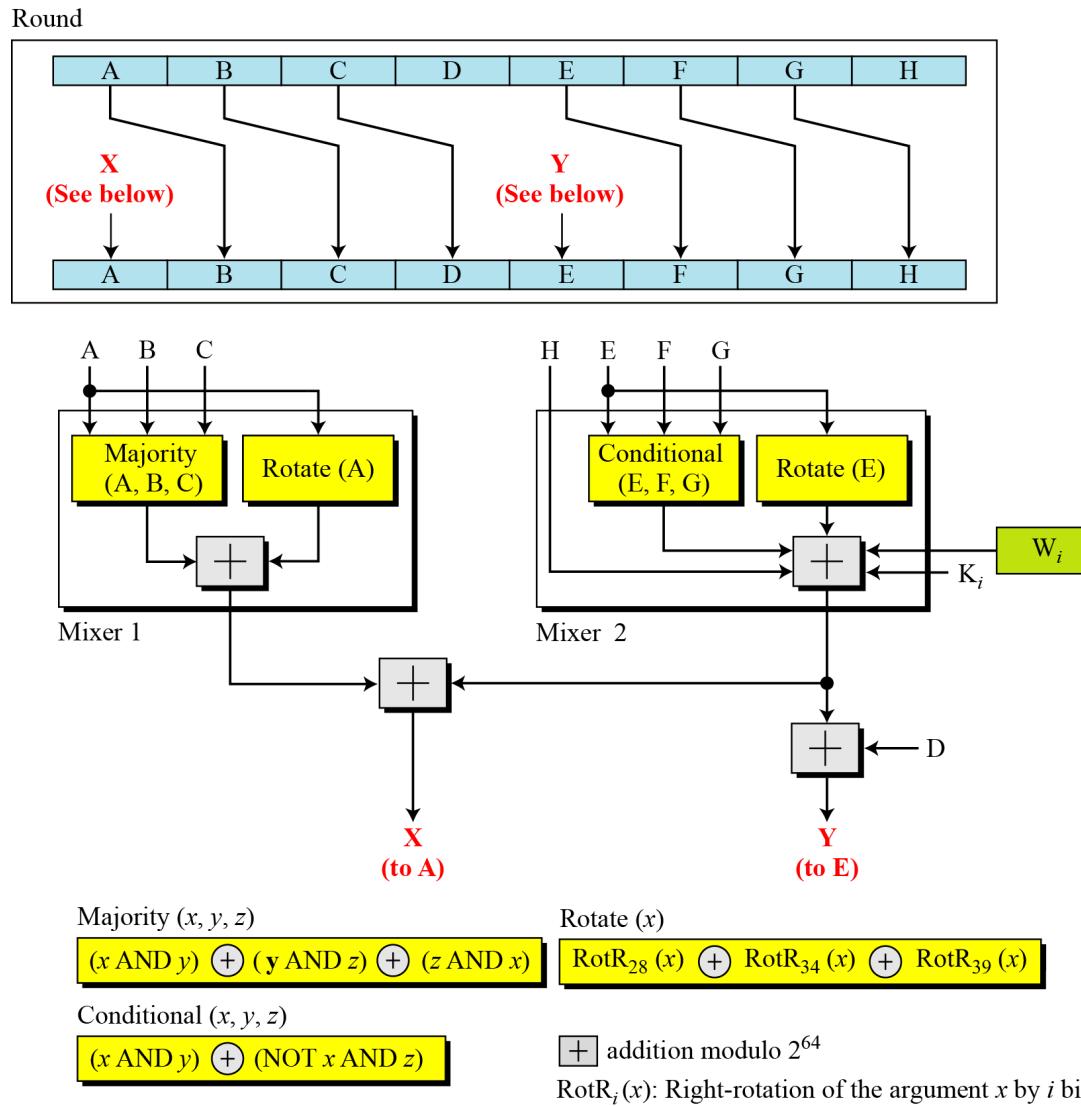
## 2.2 Compression Function: *Chức năng nén*

**Figure 10** *Compression function in SHA-512*  
*Chức năng nén trong SHA-512*



## 2.2 Continued

**Figure 11** Structure of each round in SHA-512



## 2.2 *Continued*

### *Majority Function*

$$(A_j \text{AND } B_j) \oplus (B_j \text{AND } C_j) \oplus (C_j \text{AND } A_j)$$

### *Conditional Function*

$$(E_j \text{AND } F_j) \oplus (\text{NOT } E_j \text{AND } G_j)$$

### *Rotate Functions*

**Rotate (A):**  $\text{RotR}_{28}(A) \oplus \text{RotR}_{34}(A) \oplus \text{RotR}_{29}(A)$

**Rotate (E):**  $\text{RotR}_{28}(E) \oplus \text{RotR}_{34}(E) \oplus \text{RotR}_{29}(E)$

## 2.2 *Continued*

**Table 12.3** Eighty constants used for eighty rounds in SHA-512

428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

## 2.2 Continued

*There are 80 constants,  $K_0$  to  $K_{79}$ , each of 64 bits. Similar These values are calculated from the first 80 prime numbers (2, 3,..., 409). For example, the 80th prime is 409, with the cubic root  $(409)^{1/3} = 7.42291412044\dots$  Converting this number to binary with only 64 bits in the fraction part, we get*

$$(111.0110\ 1100\ 0100\ 0100\ \dots\ 0111)_2 \rightarrow (7.6C44198C4A475817)_{16}$$

*The fraction part:  $(6C44198C4A475817)_{16}$*

## 2.2 *Continued*

### Example 7

We apply the Majority function on buffers A, B, and C. If the leftmost hexadecimal digits of these buffers are 0x7, 0xA, and 0xE, respectively, what is the leftmost digit of the result?

### Solution

The digits in binary are 0111, 1010, and 1110.

- a. The first bits are 0, 1, and 1. The majority is 1.
- b. The second bits are 1, 0, and 1. The majority is 1.
- c. The third bits are 1, 1, and 1. The majority is 1.
- d. The fourth bits are 1, 0, and 0. The majority is 0.

The result is 1110, or 0xE in hexadecimal.

## 2.2 *Continued*

### Example 8

We apply the Conditional function on E, F, and G buffers. If the leftmost hexadecimal digits of these buffers are 0x9, 0xA, and 0xF respectively, what is the leftmost digit of the result?

### Solution

The digits in binary are 1001, 1010, and 1111.

- a. The first bits are 1, 1, and 1. The result is  $F_1$ , which is 1.
- b. The second bits are 0, 0, and 1. The result is  $G_2$ , which is 1.
- c. The third bits are 0, 1, and 1. The result is  $G_3$ , which is 1.
- d. The fourth bits are 1, 0, and 1. The result is  $F_4$ , which is 0.

The result is 1110, or 0xE in hexadecimal.

## 2.3 Analysis

*With a message digest of 512 bits, SHA-512 expected to be resistant to all attacks, including collision attacks.*

*Với bản tóm tắt thông điệp 512 bit, SHA-512 dự kiến có khả năng chống lại tất cả các cuộc tấn công, bao gồm cả các cuộc tấn công va chạm.*

### 3 WHIRLPOOL

*Whirlpool is an iterated cryptographic hash function, based on the Miyaguchi-Preneel scheme, that uses a **symmetric-key block cipher** in place of the **compression function**. The block cipher is a **modified AES** cipher that has been tailored for this purpose.*

*Whirlpool là một hàm băm mật mã được lặp đi lặp lại, dựa trên lược đồ Miyaguchi-Preneel, sử dụng mật mã khối khóa đối xứng thay cho hàm nén. Mật mã khối là một mật mã AES đã được sửa đổi đã được điều chỉnh cho mục đích này. Whirlpool được thiết kế bởi Vincent Rijmen và Paulo S.L.M. Barreto.*

**Topics discussed in this section:**

**12.3.1 Whirlpool Cipher**

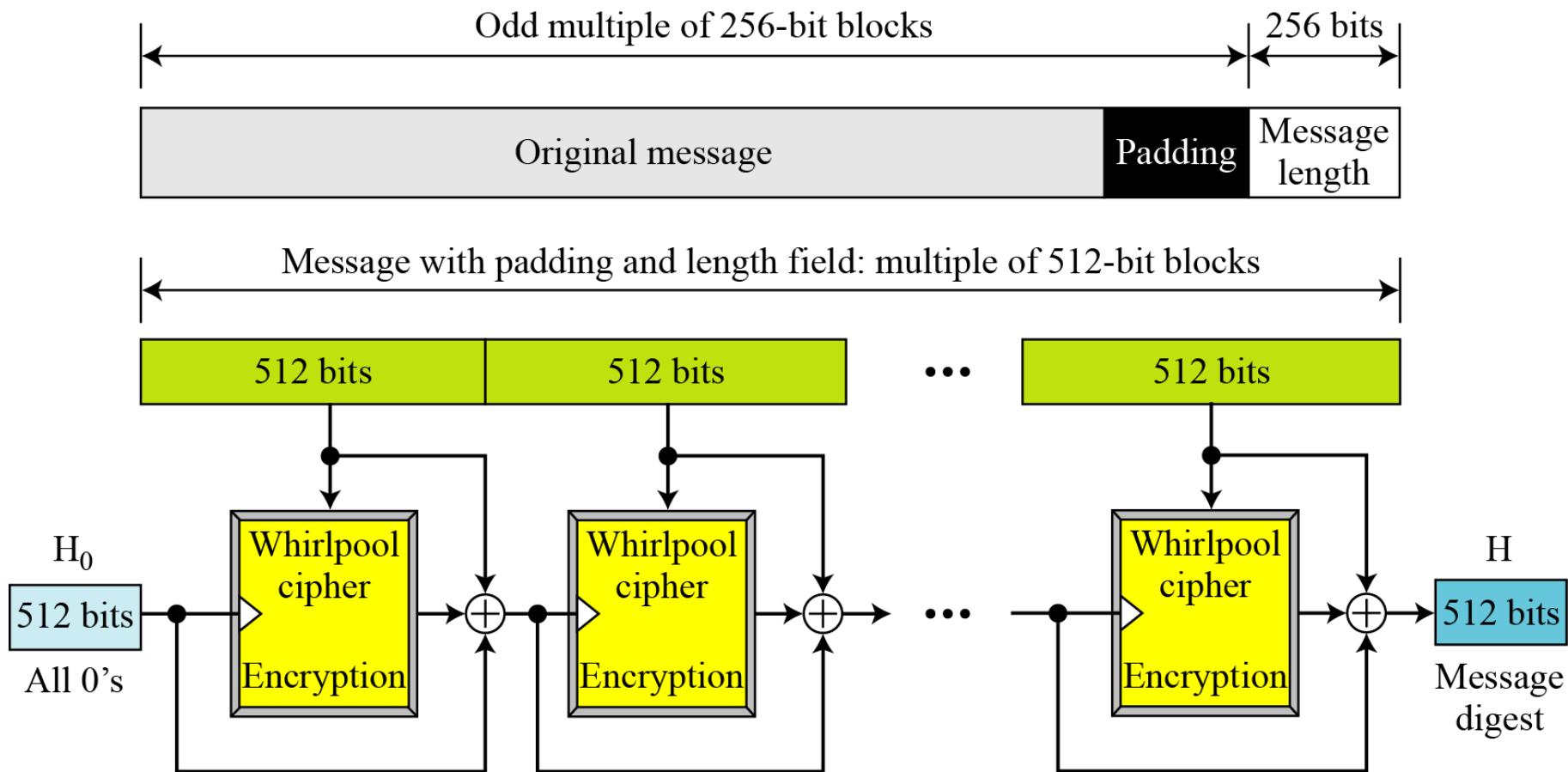
**12.3.2 Summary**

**12.3.3 Analysis**

### 3 Continued

Padding: phần các bit đệm, bắt đầu bằng bit 1, còn lại là các bit 0, cuối cùng là khối 256 bits chỉ độ dài bản tin gốc (Original message).

**Figure 12 Whirlpool hash function  
Hàm băm Whirlpool**

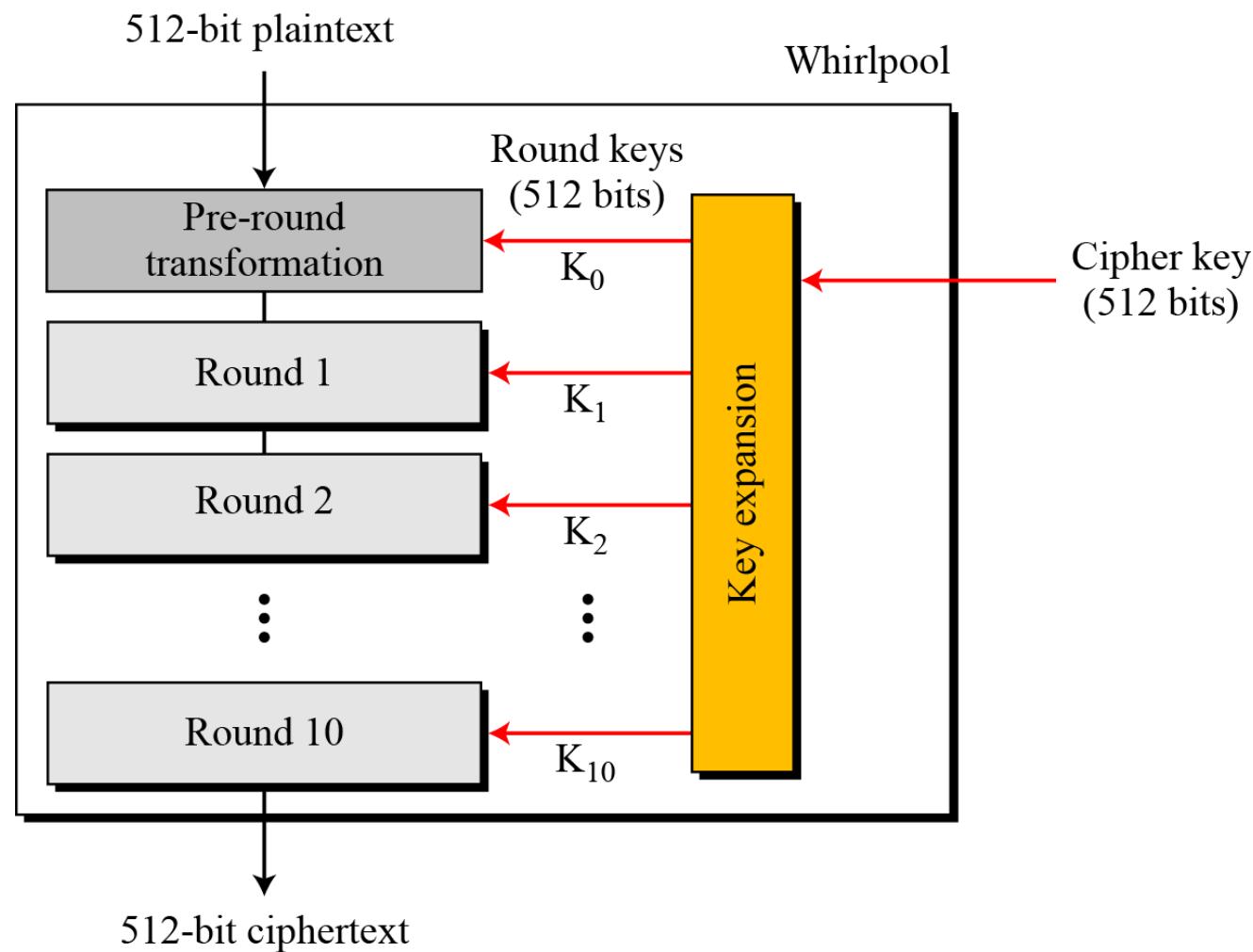


## 3.1 Whirlpool Cipher

Whirlpool dựa trên hệ mật kiều non-Feistel giống như AES để thực hiện phép mã hóa băm. Whirlpool gồm 10 round, mỗi khối có kích thước khối và khóa là 512 bits được đánh số từ K0 đến K10.

**Figure 13 General idea of the Whirlpool cipher**

*Ý tưởng chung về mật mã Xoáy nước*

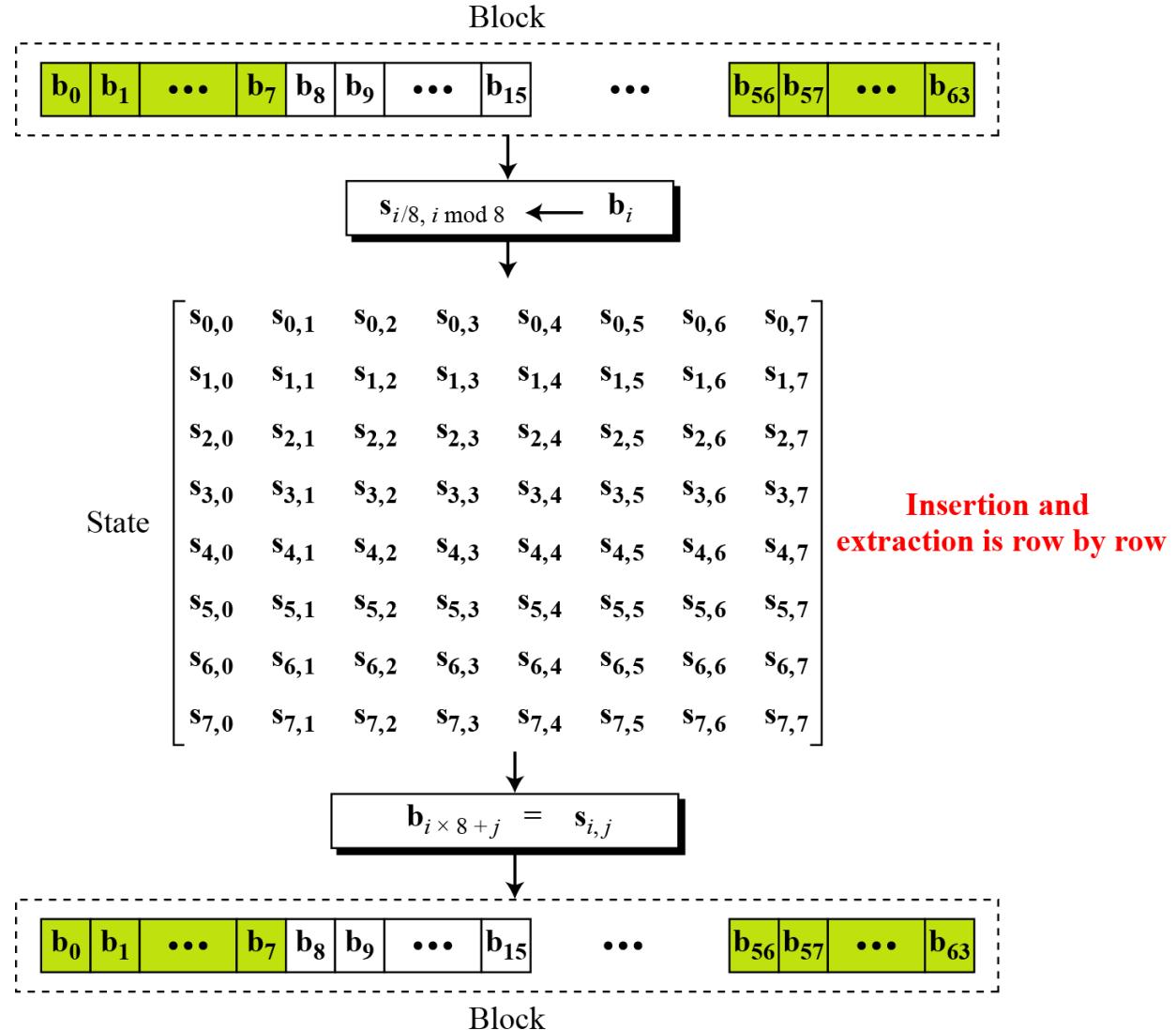


### 3.1 Continued

**Figure 14 Block and state in the Whirlpool cipher**  
*Khối và trạng thái trong mật mã Whirlpool*

Giống như AES, Whirlpool sử dụng các block và state. Một block xem như là 1 ma trận hàng 64 bytes; một state là ma trận vuông kích thước  $8 \times 8$  bytes.

Tuy nhiên, không giống như AES, việc viết block vào state là theo hàng.

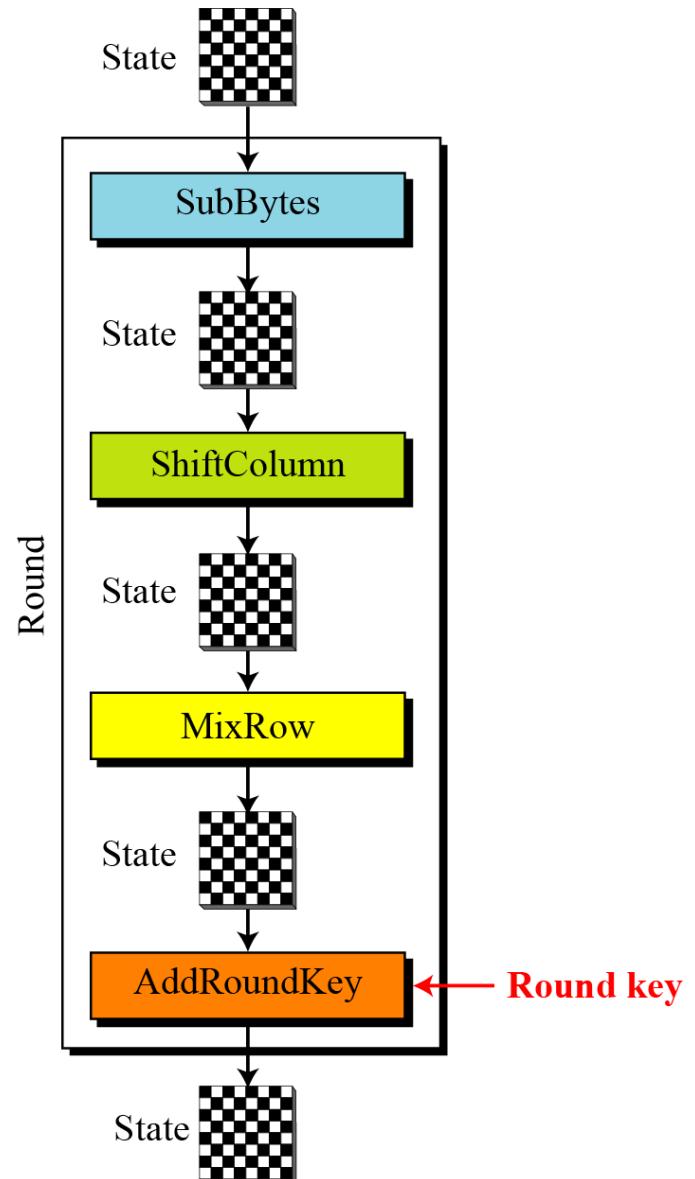


## 3.1 Continued

### *Structure of Each Round*

*Each round uses four transformations.*

*Mỗi vòng sử dụng bốn phép biến đổi.*



**Figure 15** *Structure of each round in the Whirlpool cipher*

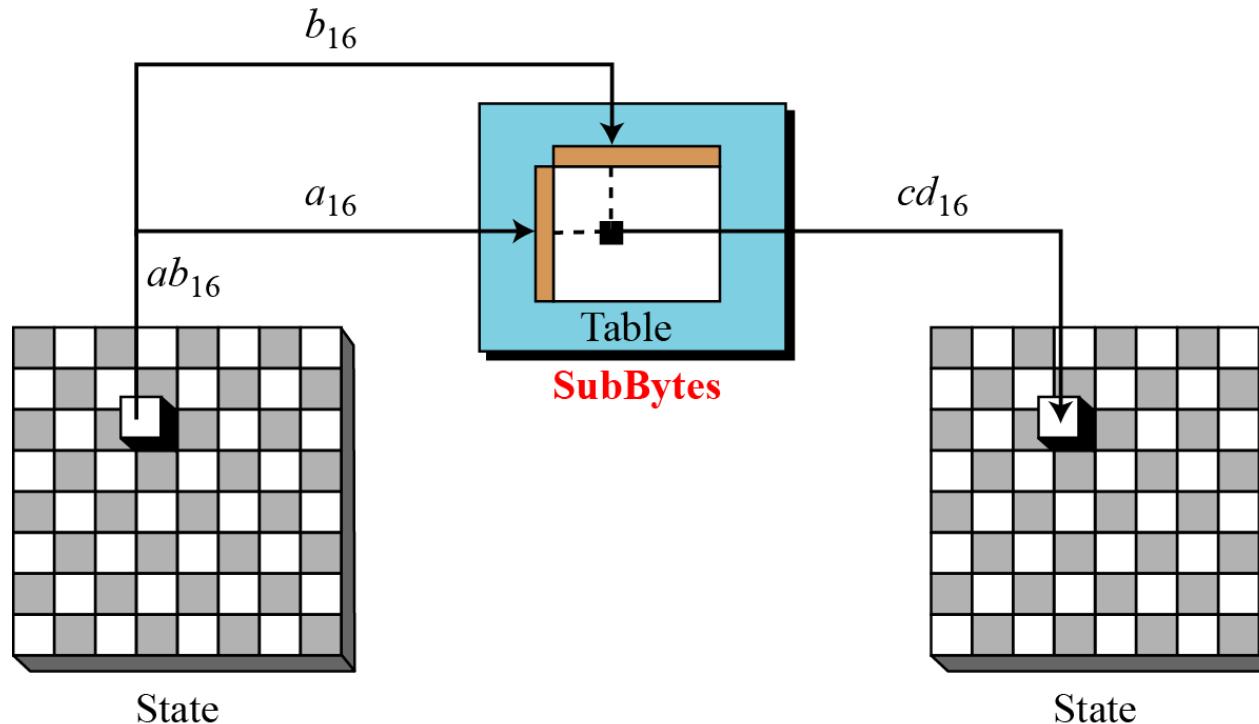
### 3.1 Continued

*SubBytes* Like in AES, SubBytes provide a nonlinear transformation.

*Giống như trong AES, SubBytes cung cấp một phép biến đổi phi tuyến.*

**Figure 16** SubBytes transformations in the Whirlpool cipher

Các phép biến đổi SubBytes trong mật mã Whirlpool



## 12.3.1 Continued

**Table 12.4** SubBytes transformation table (S-Box)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	18	23	C6	E8	87	B8	01	4F	36	A6	D2	F5	79	6F	91	52
1	16	BC	9B	8E	A3	0C	7B	35	1D	E0	D7	C2	2E	4B	FE	57
2	15	77	37	E5	9F	F0	4A	CA	58	C9	29	0A	B1	A0	6B	85
3	BD	5D	10	F4	CB	3E	05	67	E4	27	41	8B	A7	7D	95	C8
4	FB	EF	7C	66	DD	17	47	9E	CA	2D	BF	07	AD	5A	83	33
5	63	02	AA	71	C8	19	49	C9	F2	E3	5B	88	9A	26	32	B0
6	E9	0F	D5	80	BE	CD	34	48	FF	7A	90	5F	20	68	1A	AE
7	B4	54	93	22	64	F1	73	12	40	08	C3	EC	DB	A1	8D	3D
8	97	00	CF	2B	76	82	D6	1B	B5	AF	6A	50	45	F3	30	EF
9	3F	55	A2	EA	65	BA	2F	C0	DE	1C	FD	4D	92	75	06	8A
A	B2	E6	0E	1F	62	D4	A8	96	F9	C5	25	59	84	72	39	4C
B	5E	78	38	8C	C1	A5	E2	61	B3	21	9C	1E	43	C7	FC	04
C	51	99	6D	0D	FA	DF	7E	24	3B	AB	CE	11	8F	4E	B7	EB
D	3C	81	94	F7	9B	13	2C	D3	E7	6E	C4	03	56	44	7E	A9
E	2A	BB	C1	53	DC	0B	9D	6C	31	74	F6	46	AC	89	14	E1
F	16	3A	69	09	70	B6	C0	ED	CC	42	98	A4	28	5C	F8	86

# 3.1 Continued

**Figure 17 SubBytes in the Whirlpool cipher**

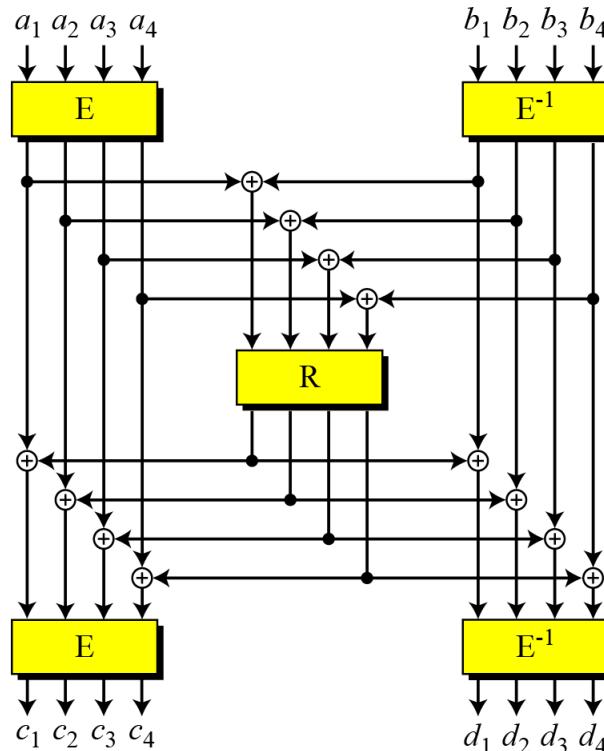
Các phần tử trong bảng 12.4 ở slide trước, bangr SubBytes, có thể tính toán sử dụng trường GF(2<sup>4</sup>) với đa thức tối giản  $x^4 + x + 1$ :

$$E(\text{input}) = (x^3 + x + 1)^{\text{input}} \quad \text{mod}$$

(x<sup>4</sup>+x+1) nếu input khác 0xF

$$E(0xF) = 0.$$

$E^{-1}$  là phép nghịch đảo của E.



Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0
Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6
Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

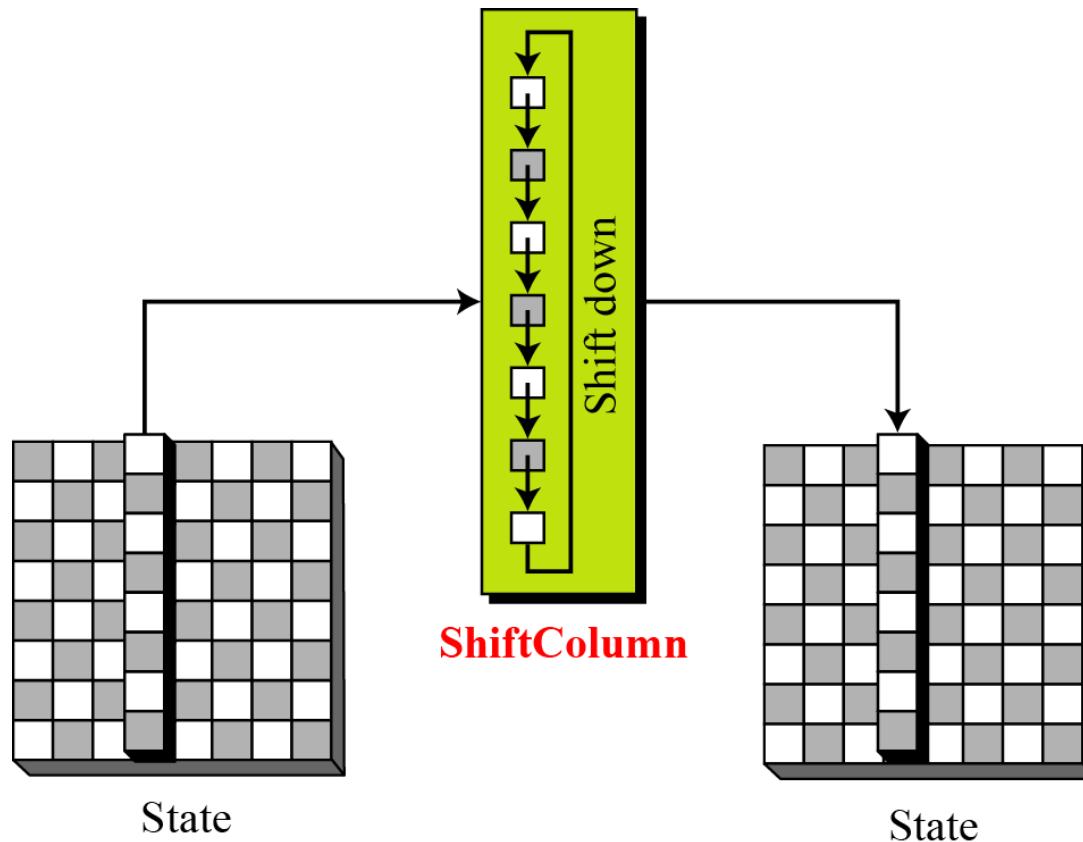
### 3.1 Continued

*ShiftColumns: phép dịch cột*

*Giống phép dịch hàng shiftRows trong AES.*

**Figure 18** ShiftColumns transformation in the Whirlpool cipher

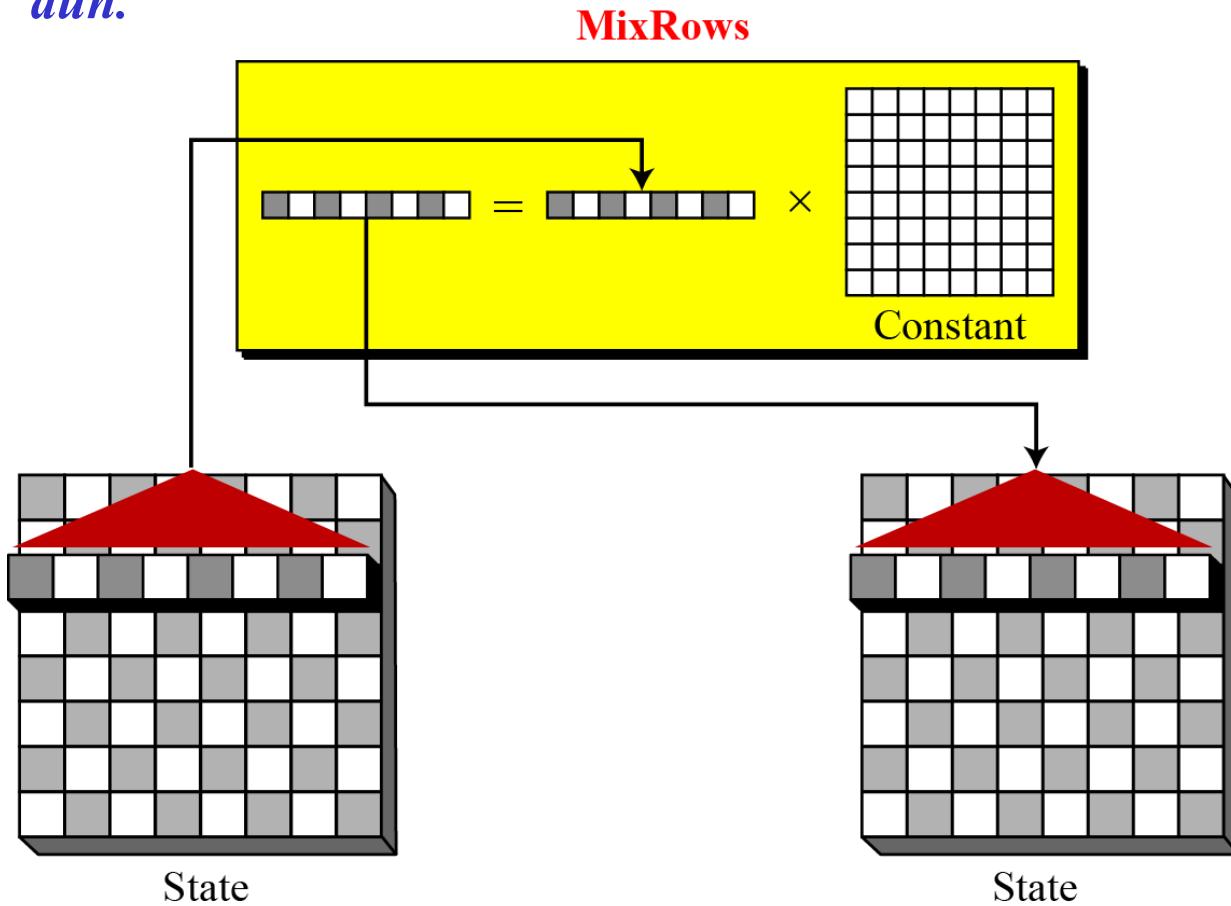
*Chuyển đổi ShiftColumns trong mật mã Xoáy nước*



## 3.1 Continued

### MixRows: phép trộn

Giống như AES, phép nhân byte được thực hiện trong  $GF(2^8)$ , đa thức tối giản  $x^8+x^4+x^3+x^2+1$  trong phép biến đổi mỗ dun.



**Figure 19** MixRows transformation in the Whirlpool cipher

Phép biến đổi MixRows trong mật mã Xoáy nước

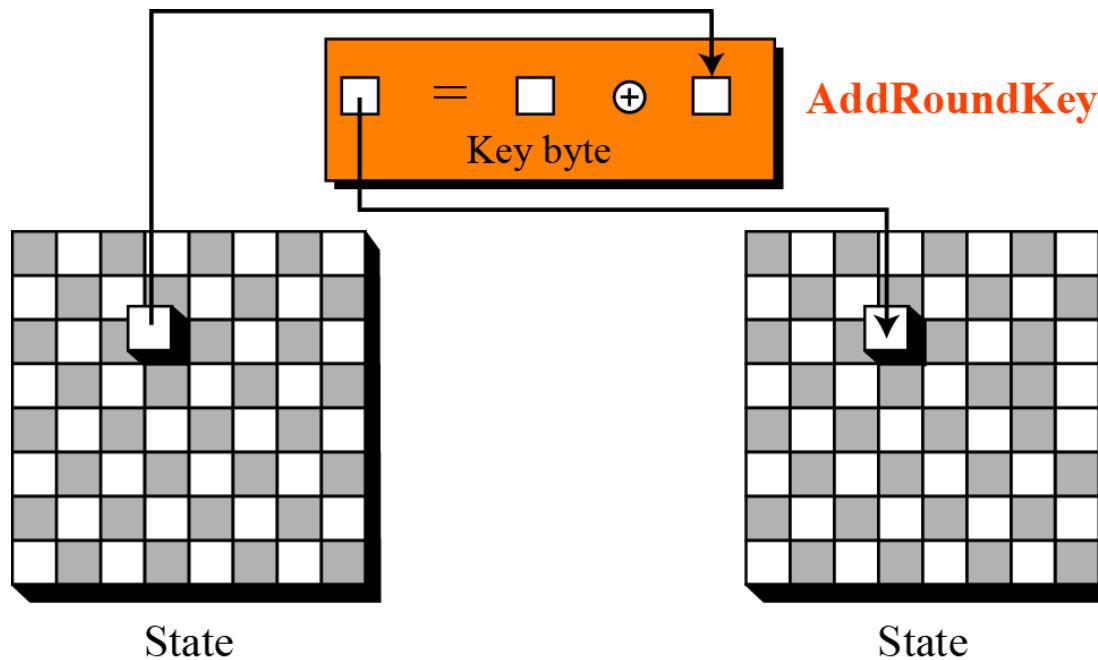
01	01	04	01	08	05	02	09
09	01	01	04	01	08	05	02
02	09	01	01	04	01	08	05
05	02	09	01	01	04	01	08
08	05	02	09	01	01	04	01
01	08	05	02	09	01	01	04
04	01	08	05	02	09	01	01
01	04	01	08	05	02	09	01

Constant matrix

### 3.1 Continued

*AddRoundKey: được thực hiện theo byte trong trường  $GF(2^8)$*

**Figure 20** *AddRoundKey transformation in the Whirlpool cipher*  
*Phép biến đổi AddRoundKey trong mật mã Whirlpool*



# 3.1 Continued

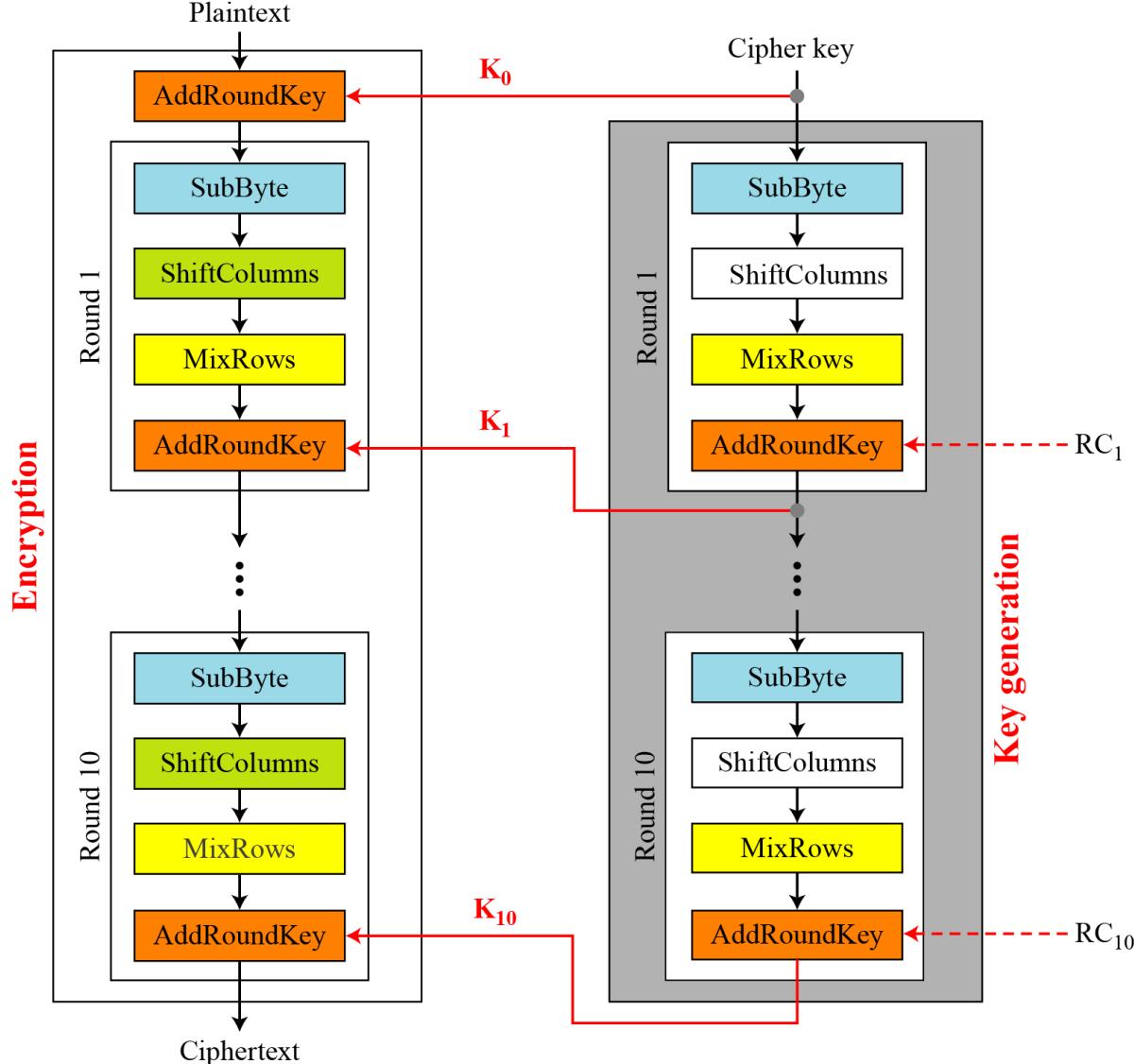
**Figure 21 Key expansion in the Whirlpool cipher**  
**Mở rộng khóa trong mật mã Whirlpool**

**Mở rộng khóa trong mật mã Whirlpool cơ bản khác với mở rộng khóa của AES.**

RC: hằng số vòng (Round Constant)

Thuật toán mở rộng khóa sử dụng các hằng số như là các khóa vòng và thuật toán mã hóa sử dụng đầu ra của mỗi vòng của thuật toán tạo khóa đó.

Thuật toán tạo khóa xem khóa mật như là bản tin plaintext để mã hóa nó.



### **3.1 *Continued***

## *Round constants*

**Figure 22** Round constant for the third round

## **VÍ dụ giá trị RC3 kích thước 8x8**

*Đối với mỗi vòng RCx là ma trận  $8 \times 8$ , trong đó chỉ có hàng đầu tiên có các giá trị khác không, các hàng còn lại bằng 0*

**Giá trị của hàng đầu tiên được thực hiện sử dụng phép biến đổi SubBytes theo bảng 12.4**

**RC\_round** [row, column] = SubBytes (8 (round-1)+column) nếu row =0;  
**RC round** [row, column] = 0, nếu row khác 0.

## 3.2 Summary

**Table 12.5** *Main characteristics of the Whirlpool cipher*

Block size: 512 bits
Cipher key size: 512 bits
Number of rounds: 10
Key expansion: using the cipher itself with round constants as round keys
Substitution: SubBytes transformation
Permutation: ShiftColumns transformation
Mixing: MixRows transformation
Round Constant: cubic roots of the first eighty prime numbers

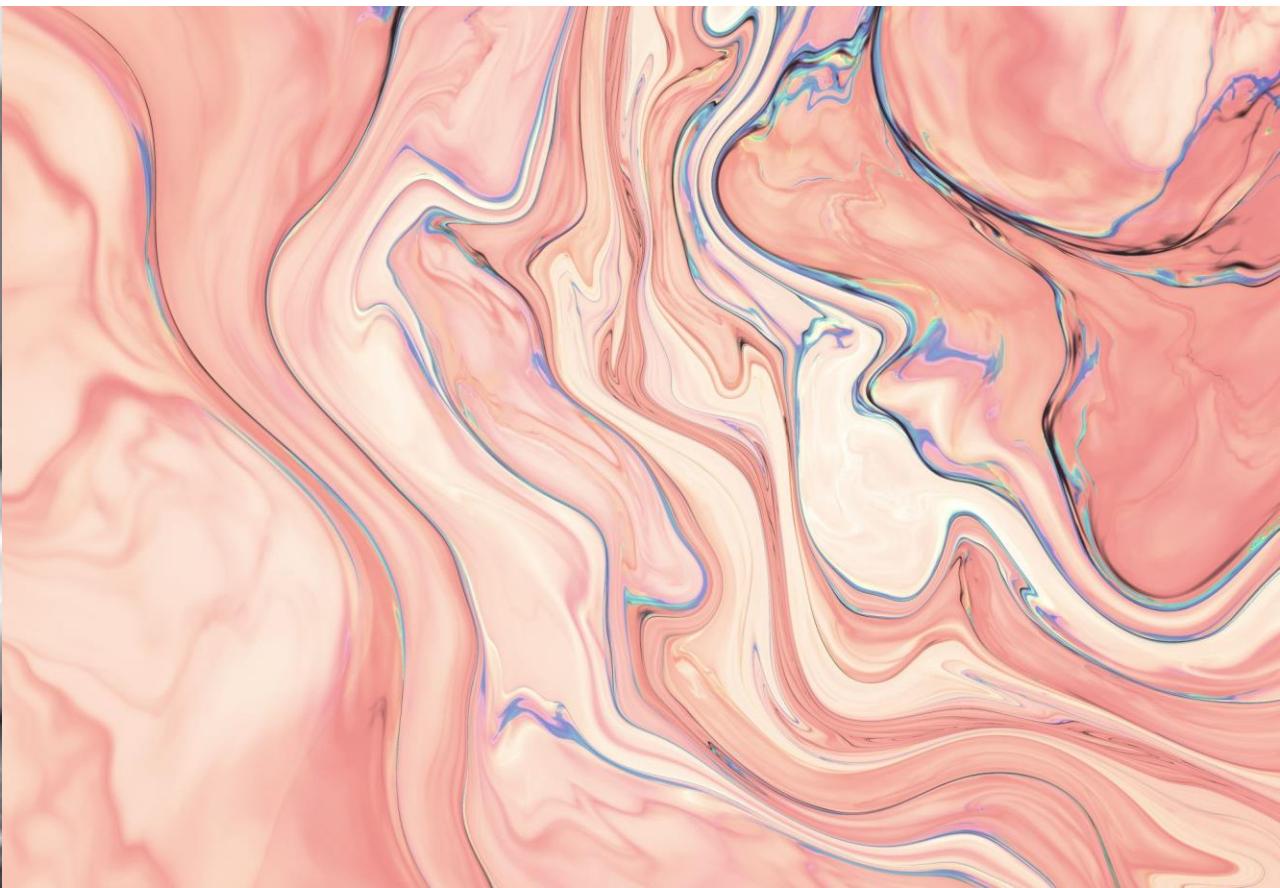
### *3.3 Analysis*

*Although Whirlpool has not been extensively studied or tested, it is based on a robust scheme (Miyaguchi-Preneel), and for a compression function uses a cipher that is based on AES, a cryptosystem that has been proved very resistant to attacks. In addition, the size of the message digest is the same as for SHA-512. Therefore, it is expected to be a very strong cryptographic hash function.*

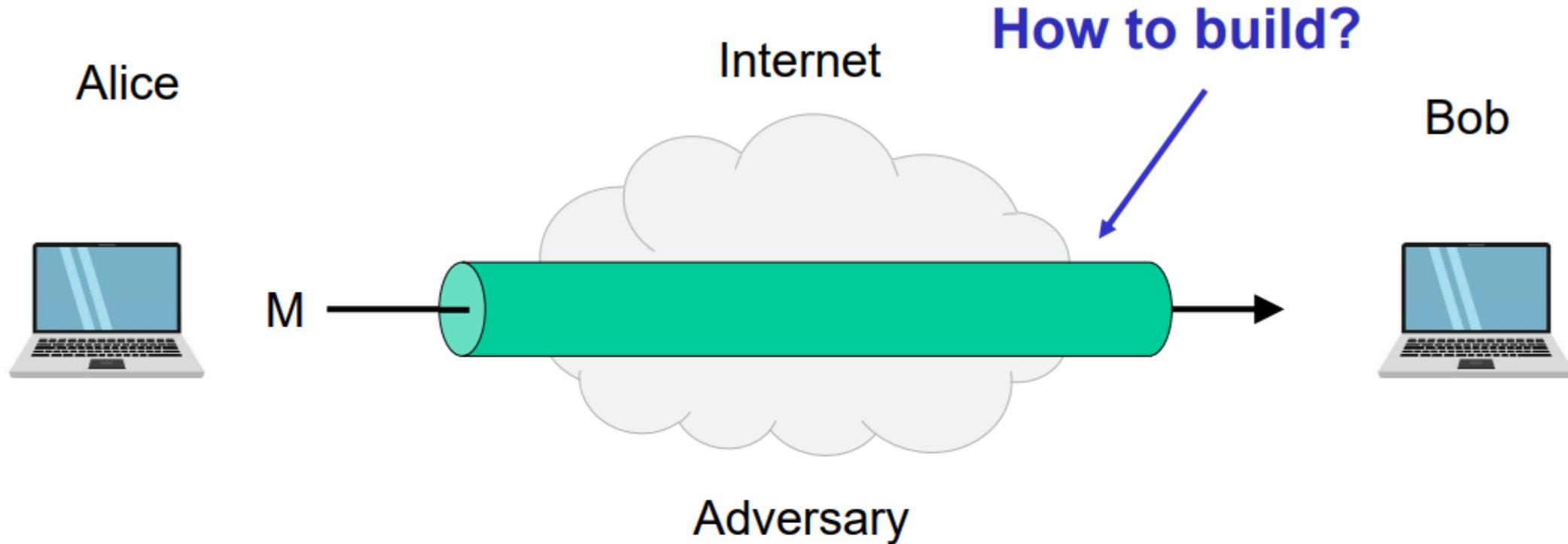
*Mặc dù Whirlpool chưa được nghiên cứu hoặc thử nghiệm rộng rãi, nhưng nó dựa trên một sơ đồ mạnh mẽ (Miyaguchi-Preneel) và đối với chức năng nén sử dụng mật mã dựa trên AES, một hệ thống mật mã đã được chứng minh là có khả năng chống lại các cuộc tấn công. Ngoài ra, kích thước của bản tin digest giống như đối với SHA-512. Do đó, Whirlpool được kỳ vọng là một hàm băm mật mã rất mạnh.*

**DIFFIE HELMAN KEY EXCHANGE**

**PHƯƠNG THỨC TRAO ĐỔI KHÓA DH**



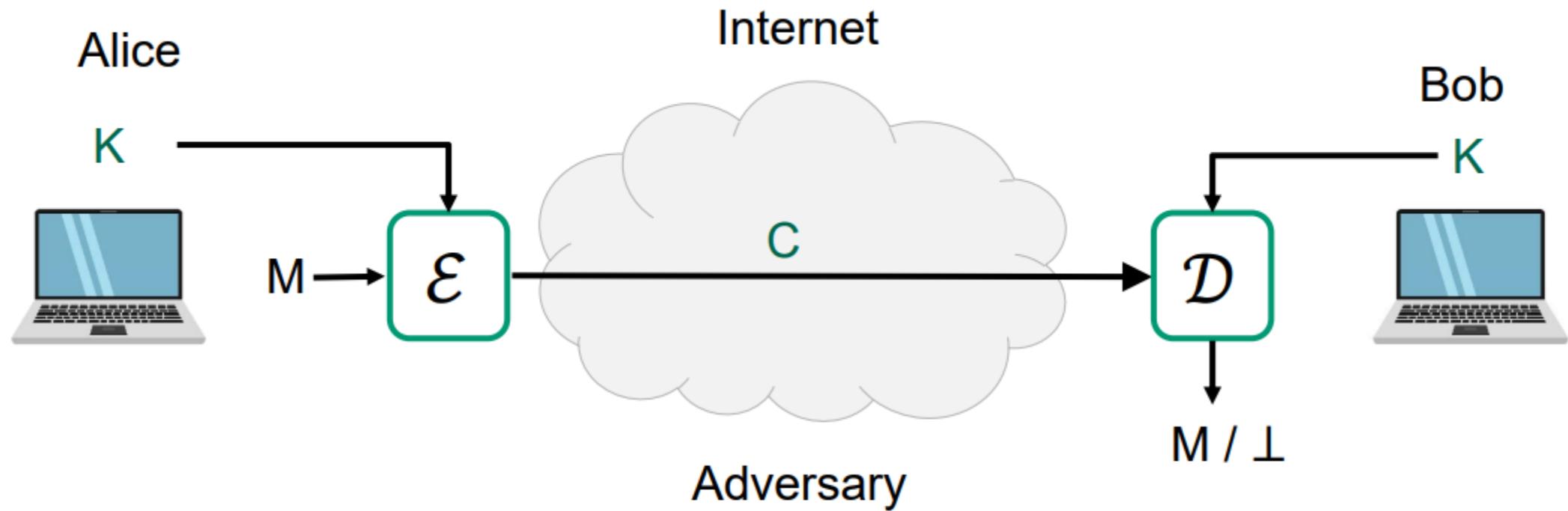
# Diffie-Hellman Key Exchange



## Security goals:

- **Data privacy:** adversary should not be able to read message  $M$  ✓
- **Data integrity:** adversary should not be able to modify message  $M$  ✓
- **Data authenticity:** message  $M$  really originated from Alice ✓

# Diffie-Hellman Key Exchange



$\mathcal{E}$  : encryption algorithm (public)

$K$  : encryption / decryption key (secret)

$\mathcal{D}$  : decryption algorithm (public)

# Diffie-Hellman Key Exchange

- Discovered in the 1970's
- Allows two parties to establish a shared secret without ever having met
- Diffie & Hellman paper also introduced:
  - Public-key encryption
  - Digital signatures



Ralph Merkle

Whitfield Diffie

Martin Hellman

## New Directions in Cryptography

*Invited Paper*

Whitfield Diffie and Martin E. Hellman

**Abstract** Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

### 1 INTRODUCTION

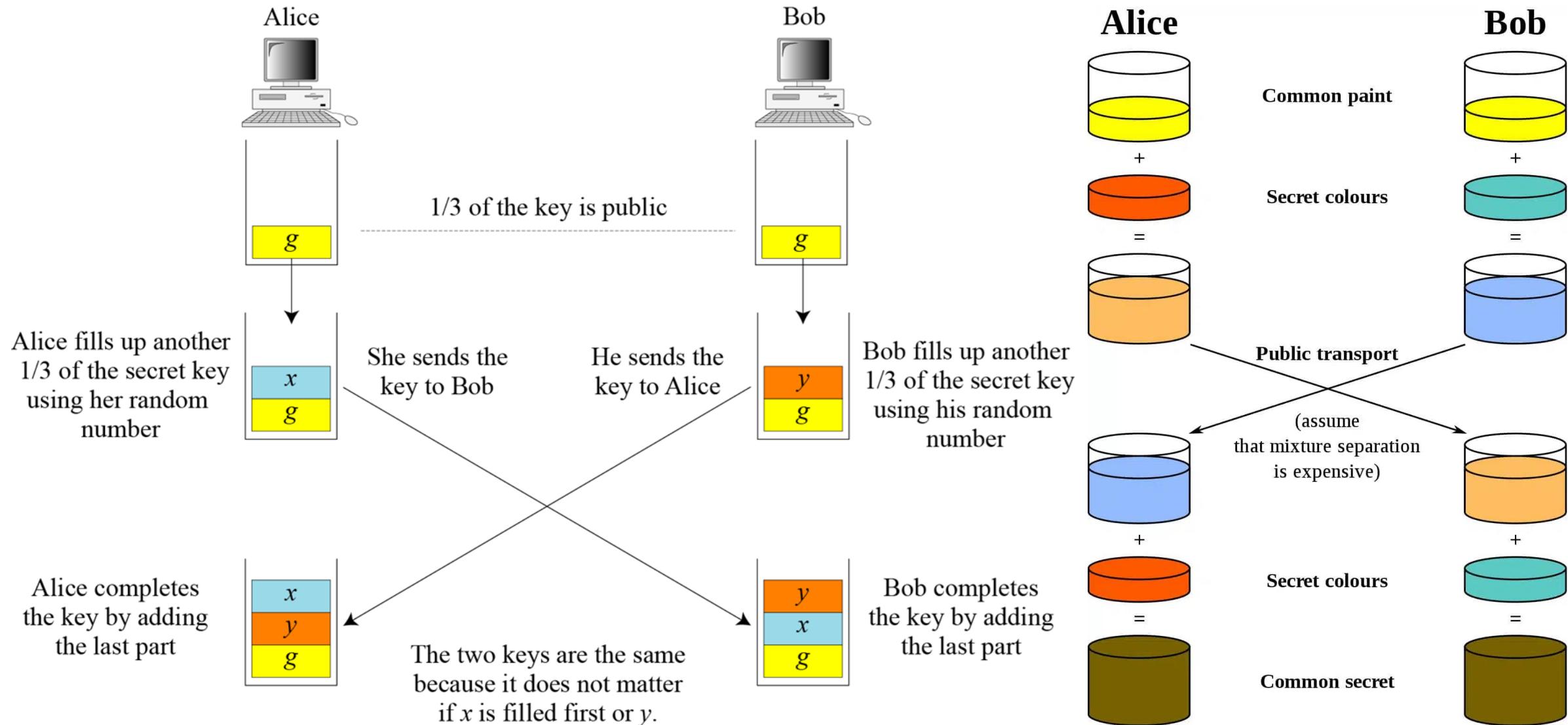
We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication net-

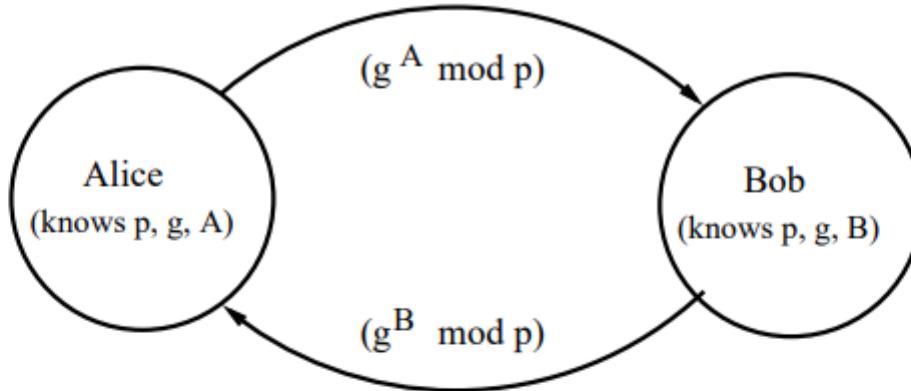
communications over an insecure channel order to use cryptography to insure privacy, however, it currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such a private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channel without compromising the security of the system. In *public key cryptosystem* enciphering and deciphering are governed by distinct keys,  $E$  and  $D$ , such that computing  $D$  from  $E$  is computationally infeasible (e.g., requiring  $10^{100}$  instructions). The enciphering key  $E$  can thus be publicly disclosed without compromising the deciphering key  $D$ . Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is multiple access cipher. A private conversation can therefore be

# Diffie-Hellman Key Exchange



# Diffie-Hellman Key Exchange



Steps in the algorithm:

- ① Alice and Bob agree on a prime number  $p$  and a base  $g$ .
- ② Alice chooses a secret number  $a$ , and sends Bob  $(g^a \text{ mod } p)$ .
- ③ Bob chooses a secret number  $b$ , and sends Alice  $(g^b \text{ mod } p)$ .
- ④ Alice computes  $((g^b \text{ mod } p)^a \text{ mod } p)$ .
- ⑤ Bob computes  $((g^a \text{ mod } p)^b \text{ mod } p)$ .

Both Alice and Bob can use this number as their key. Notice that  $p$  and  $g$  need not be protected.

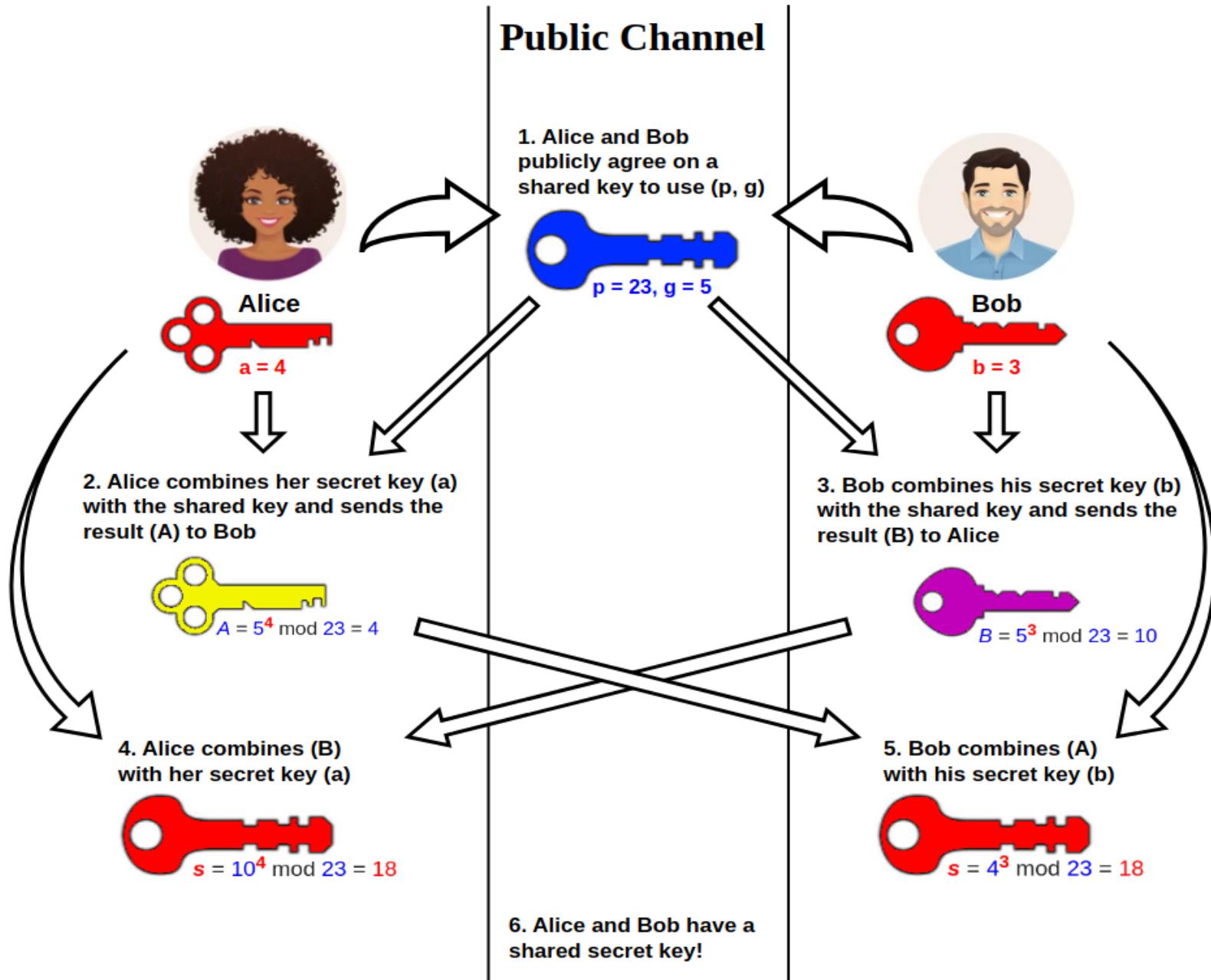
## Diffie-Hellman Key Exchange

- ① Alice and Bob agree on  $p = 23$  and  $g = 5$ .
- ② Alice chooses  $a = 6$  and sends  $5^6 \bmod 23 = 8$ .
- ③ Bob chooses  $b = 15$  and sends  $5^{15} \bmod 23 = 19$ .
- ④ Alice computes  $19^6 \bmod 23 = 2$ .
- ⑤ Bob computes  $8^{15} \bmod 23 = 2$ .

Then 2 is the shared secret.

Clearly, much larger values of  $a$ ,  $b$ , and  $p$  are required. An eavesdropper cannot discover this value even if she knows  $p$  and  $g$  and can obtain each of the messages.

# Diffie-Hellman Key Exchange





Private = 5



$$\begin{aligned}(6^5) \bmod 13 \\ (7776) \bmod 13 \\ \text{Public} = 2\end{aligned}$$



$$\begin{aligned}(9^5) \bmod 13 \\ (59049) \bmod 13 \\ \text{Shared Secret} = 3\end{aligned}$$



Agree upon two numbers:

P Prime Number 13

G Generator of P 6

Randomly generate a Private Key

Calculate Public Key:

$$(G^{\text{Private}}) \bmod P$$

Exchange Public Keys

Calculate the Shared Secret

$$(\text{Shared Public}^{\text{Private}}) \bmod P$$



Private = 4

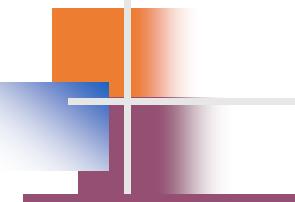


$$\begin{aligned}(6^4) \bmod 13 \\ (1296) \bmod 13 \\ \text{Public} = 9\end{aligned}$$



$$\begin{aligned}(2^4) \bmod 13 \\ (16) \bmod 13 \\ \text{Shared Secret} = 3\end{aligned}$$



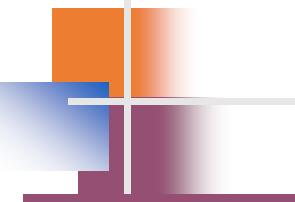


# Diffie-Hellman Key Exchange

## Example

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that  $g = 7$  and  $p = 23$ .





# Diffie-Hellman Key Exchange

## Example

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume that  $g = 7$  and  $p = 23$ . The steps are as follows:

1. Alice chooses  $x = 3$  and calculates  $R_1 = 7^3 \bmod 23 = 21$ .
2. Bob chooses  $y = 6$  and calculates  $R_2 = 7^6 \bmod 23 = 4$ .
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key  $K = 4^3 \bmod 23 = 18$ .
6. Bob calculates the symmetric key  $K = 21^6 \bmod 23 = 18$ .
7. The value of  $K$  is the same for both Alice and Bob;  
 $g^{xy} \bmod p = 7^{18} \bmod 23 = 18$ .

# Diffie-Hellman Key Exchange

## Example

Let us give a more realistic example. We used a program to create a random integer of 512 bits (the ideal is 1024 bits). The integer  $p$  is a 159-digit number. We also choose  $g$ ,  $x$ , and  $y$  as shown below:

Một ví dụ thực tế hơn. Chúng ta đã sử dụng một chương trình để tạo một số nguyên ngẫu nhiên 512 bit (lý tưởng là 1024 bit). Số nguyên  $p$  là một số có 159 chữ số. Chúng ta cũng chọn  $g$ ,  $x$  và  $y$  như hình dưới đây:

$p$	764624298563493572182493765955030507476338096726949748923573772860925 235666660755423637423309661180033338106194730130950414738700999178043 6548785807987581
$g$	2
$x$	557
$y$	273

# Diffie-Hellman Key Exchange

## Example

*The following shows the values of  $R_1$ ,  $R_2$ , and  $K$ .*

<b><math>R_1</math></b>	844920284205665505216172947491035094143433698520012660862863631067673 619959280828586700802131859290945140217500319973312945836083821943065 966020157955354
<b><math>R_2</math></b>	435262838709200379470747114895581627636389116262115557975123379218566 310011435718208390040181876486841753831165342691630263421106721508589 6255201288594143
<b><math>K</math></b>	155638000664522290596225827523270765273218046944423678520320400146406 500887936651204257426776608327911017153038674561252213151610976584200 1204086433617740

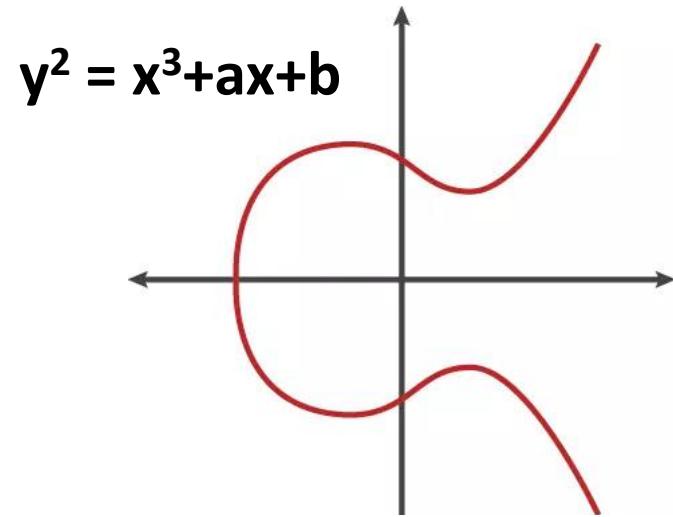
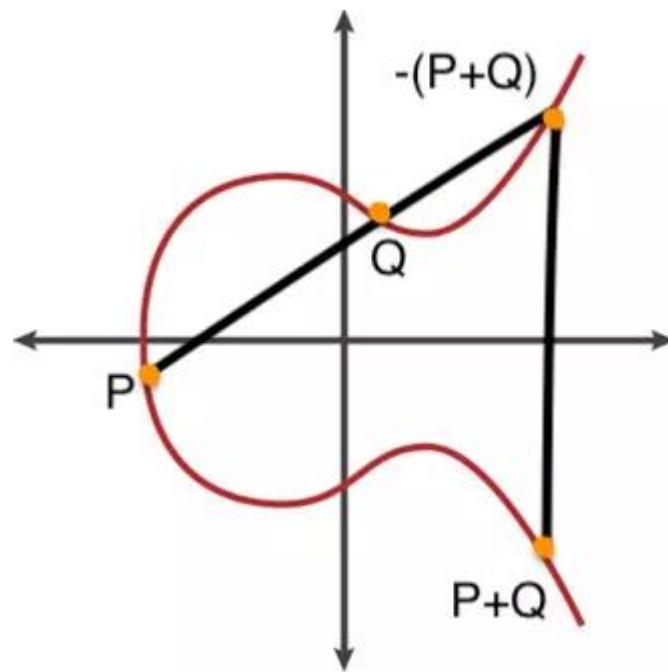
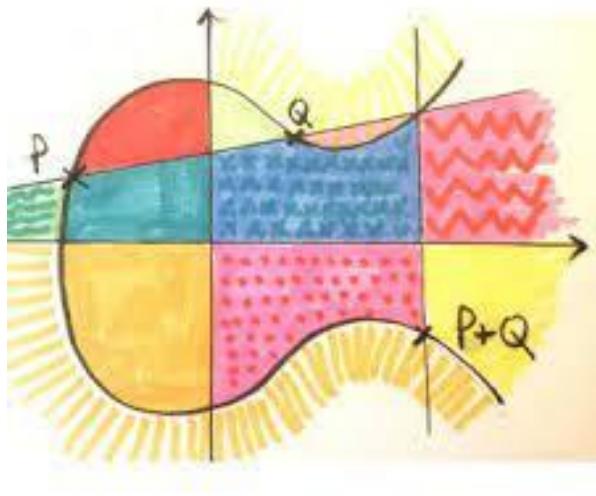
# Diffie-Hellman Key Exchange

- How can two parties agree on a secret value when all of their messages might be overheard by an eavesdropper?
- The Diffie-Hellman algorithm accomplishes this, and is still widely used.
- With sufficiently large inputs, Diffie-Hellman is very secure.

Suppose  $p$  is a prime of around 300 digits, and  $a$  and  $b$  at least 100 digits each.

Discovering the shared secret given  $g$ ,  $p$ ,  $g^a \bmod p$  and  $g^b \bmod p$  would take longer than the lifetime of the universe, using the best known algorithm. This is called the *discrete logarithm problem*.

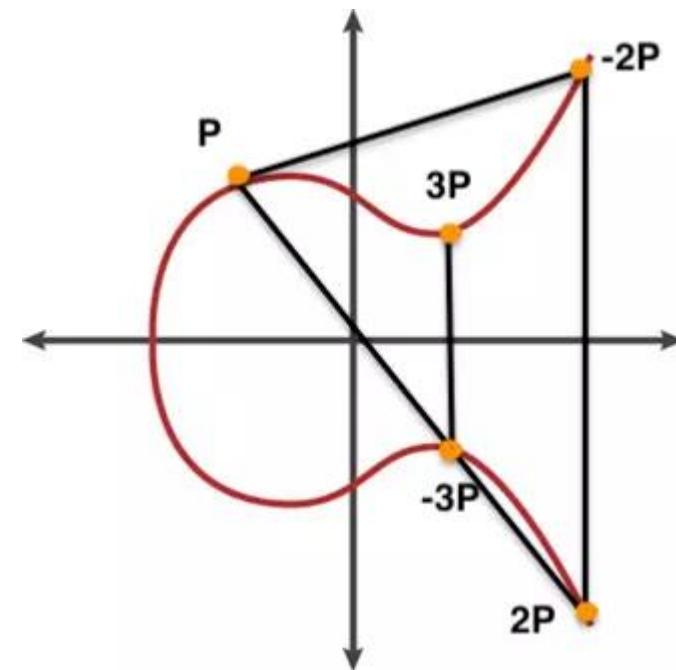
# Diffie-Hellman Key Exchange



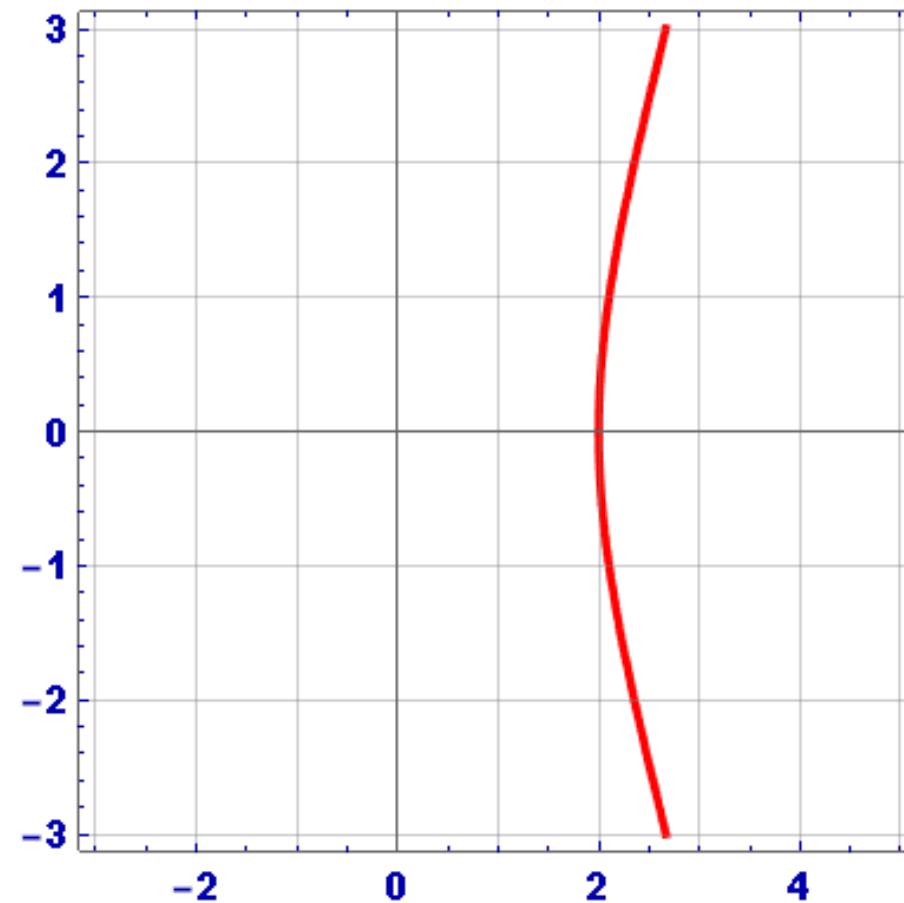
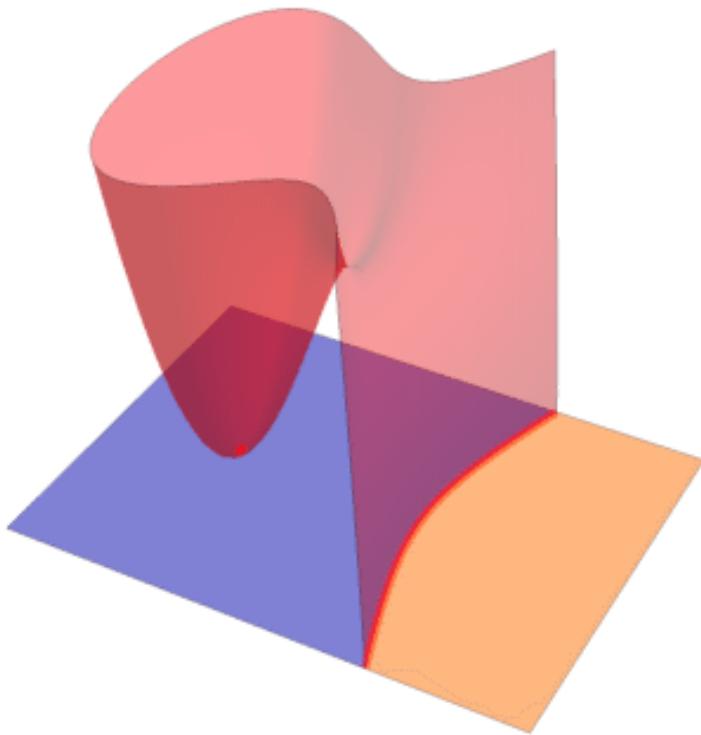
## Elliptic Curves

secp256k1

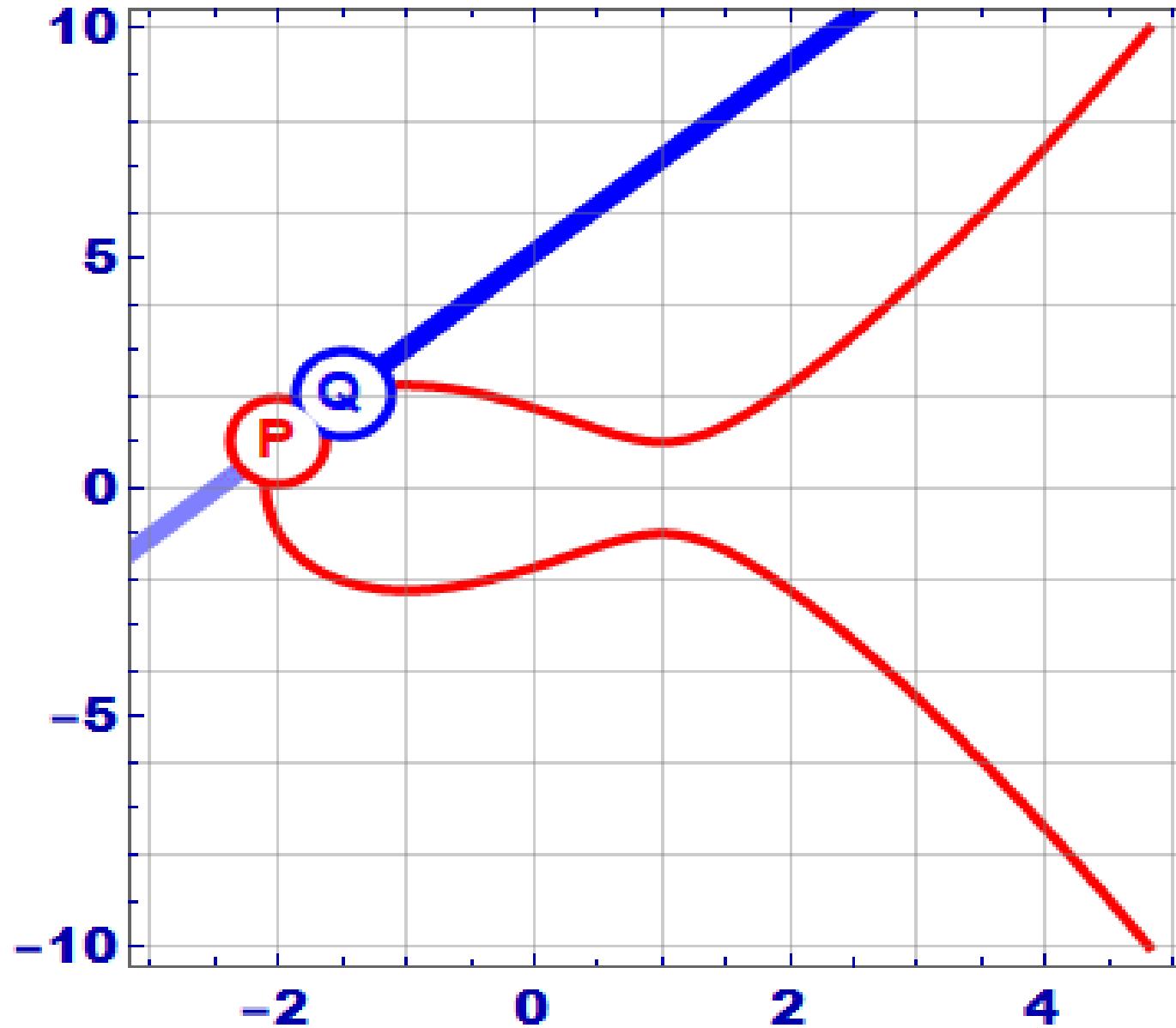
$$y^2 = x^3 + 7$$



# Diffie-Hellman Key Exchange



# Diffie-Hellman Key Exchange



# Diffie-Hellman Key Exchange

1. Alice generates a random ECC key pair: {alicePrivKey, alicePubKey = alicePrivKey \* G}
2. Bob generates a random ECC key pair: {bobPrivKey, bobPubKey = bobPrivKey \* G}
3. Alice and Bob exchange their public keys through the insecure channel (e.g. over Internet)
4. Alice calculates sharedKey = bobPubKey \* alicePrivKey
5. Bob calculates sharedKey = alicePubKey \* bobPrivKey
6. Now both Alice and Bob have the same sharedKey == bobPubKey \* alicePrivKey == alicePubKey \* bobPrivKey

# Diffie-Hellman Key Exchange

Elliptic Curve Diffie Hellman (ECDH) is used to create a shared key. In this example we use secp256k1 (as used in Bitcoin) to generate points on the curve. Its format is:

$$y^2 = x^3 + 7$$

with a prime number ( $p$ ) of 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

and which is  $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

All our operations will be  $(\text{mod } p)$

Bob will generate a public key and a private key by taking a point on the curve. The private key is a random number ( $d_B$ ) and the Bob's public key ( $Q_B$ ) will be:

$$Q_B = d_B \times G$$

Alice will do the same and generate her public key ( $Q_A$ ) from her private key ( $d_A$ ):

$$Q_A = d_A \times G$$

They then exchange their public keys. Alice will then use Bob's public key and her private key to calculate:

$$\text{SharekeyAlice} = d_A \times Q_B$$

This will be the same as:

$$\text{SharekeyAlice} = d_A \times d_B \times G$$

Bob will then use Alice's public key and his private key to determine:

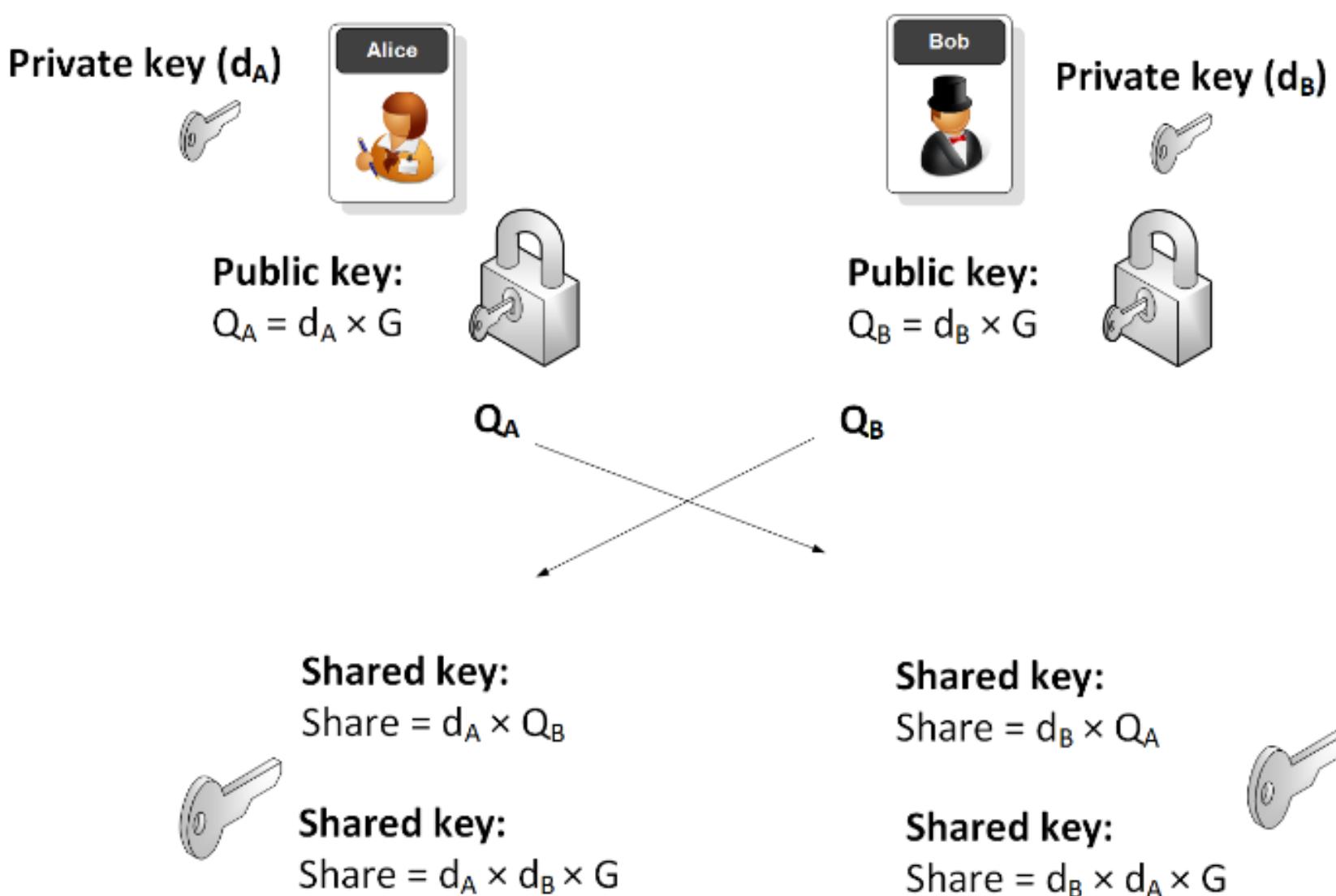
$$\text{SharekeyBob} = d_B \times Q_A$$

This will be the same as:

$$\text{SharekeyBob} = d_B \times d_A \times G$$

And the keys will thus match.

# Diffie-Hellman Key Exchange



# Diffie-Hellman Key Exchange

**ECDH** is a key sharing algorithm, most commonly used to send encrypted messages. ECDH works by multiplying your private key by another's public key to get a shared secret, then using that shared secret to perform symmetric encryption.

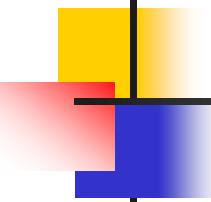
To illustrate why this works:

- Alice and Bob agree on a curve with starting point  $P$
- Alice has a private key  $a$  and public key  $A = a * P$
- Bob has a private key  $b$  and public key  $B = b * P$
- $a * B = a * b * P = b * A$
- So  $a * b * P$  ends up being the shared secret



# **Chữ Ký Số**

# **Digital Signature**



# NỘI DUNG

- Phân biệt chữ ký số và chữ ký thông thường;
- Dịch vụ bảo mật được cung cấp bởi chữ ký số;
- Đánh giá xác định các cuộc tấn công vào chữ ký số;
- Lược đồ chữ ký số, bao gồm RSA, ElGamal;
- Ứng dụng của chữ ký số.

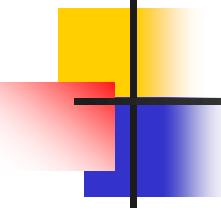
# 1 COMPARISON

*Let us begin by looking at the differences between conventional signatures and digital signatures.*

*Chúng ta hãy bắt đầu bằng cách xem xét sự khác biệt giữa chữ ký thông thường và chữ ký số.*

**Topics discussed in this section:**

- 1.1      Inclusion**
- 1.2      Verification Method**
- 1.3      Relationship**
- 1.4      Duplicity**



## 1.1 Inclusion

*A conventional signature is included in the document; it is part of the document. But when we sign a document digitally, we send the signature as a separate document.*

*Một chữ ký thông thường được bao gồm trong tài liệu; nó là một phần của tài liệu. Nhưng khi chúng ta ký một tài liệu bằng kỹ thuật số, **chúng ta gửi chữ ký như một tài liệu riêng biệt.***

## 1.2 Verification Method

### Phương pháp xác thực

*For a conventional signature, when the recipient receives a document, she compares the signature on the document with the signature on file. For a digital signature, the recipient receives the **message** and the **signature**. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.*

*Đối với chữ ký truyền thống, khi người nhận nhận được tài liệu, cô ấy sẽ so sánh chữ ký trên tài liệu với chữ ký trên hồ sơ. Đối với chữ ký điện tử, người nhận sẽ nhận được thông điệp và chữ ký. Người nhận cần áp dụng kỹ thuật xác minh kết hợp giữa bản tin và chữ ký để **xác minh tính xác thực**.*

## 1.3 Relationship: Mối quan hệ

*For a conventional signature, there is normally a one-to-many relationship between a signature and documents. For a digital signature, there is a one-to-one relationship between a signature and a message.*

*Đối với chữ ký thông thường, thường có mối quan hệ **một-nhiều** giữa chữ ký và các tài liệu. Đối với chữ ký điện tử, có mối quan hệ 1-1 giữa chữ ký và bản tin.*

## **1.4 Duplicity: Sự trùng lặp**

*In conventional signature, a copy of the signed document can be distinguished from the original one on file. In digital signature, there is no such distinction unless there is a factor of time on the document.*

*Trong chữ ký thông thường, bản sao của tài liệu đã ký có thể được phân biệt với bản gốc trong hồ sơ. Trong chữ ký điện tử, không có sự phân biệt như vậy trừ khi có yếu tố thời gian trên tài liệu.*

## 2 PROCESS: QUÁ TRÌNH

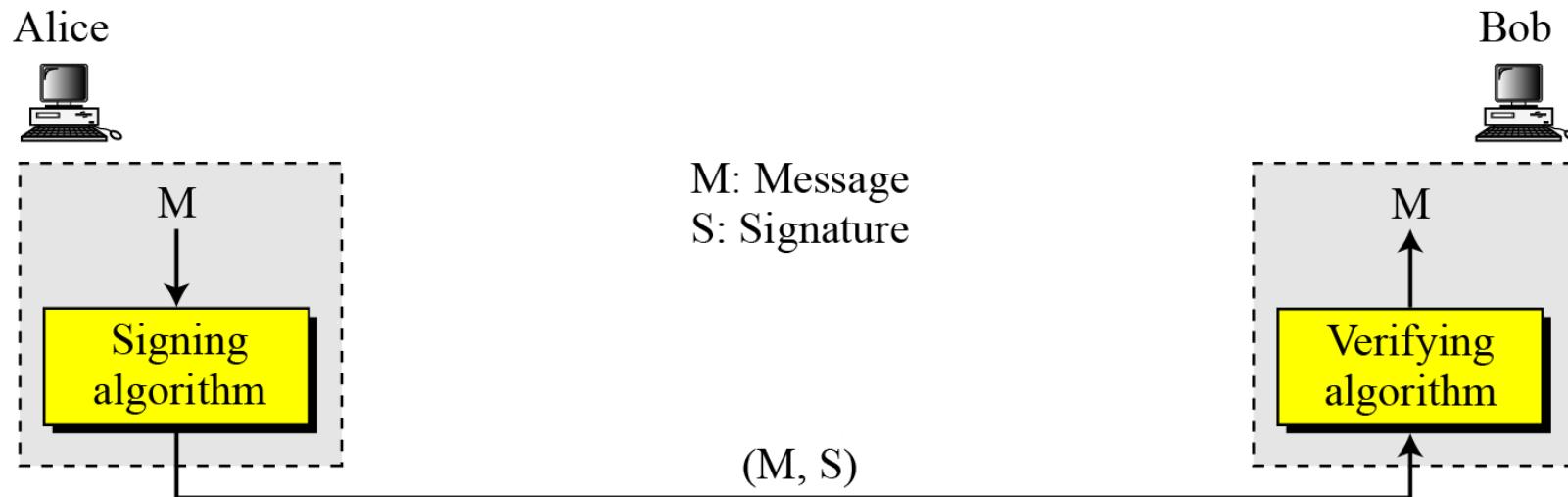
*Figure 1 shows the digital signature process. The sender uses a **signing algorithm** to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the **verifying algorithm to the combination**. If the result is true, the message is accepted; otherwise, it is rejected.*

### **Topics discussed in this section:**

- 2.1      Need for Keys**
- 2.2      Signing the Digest**

## 2 Continued

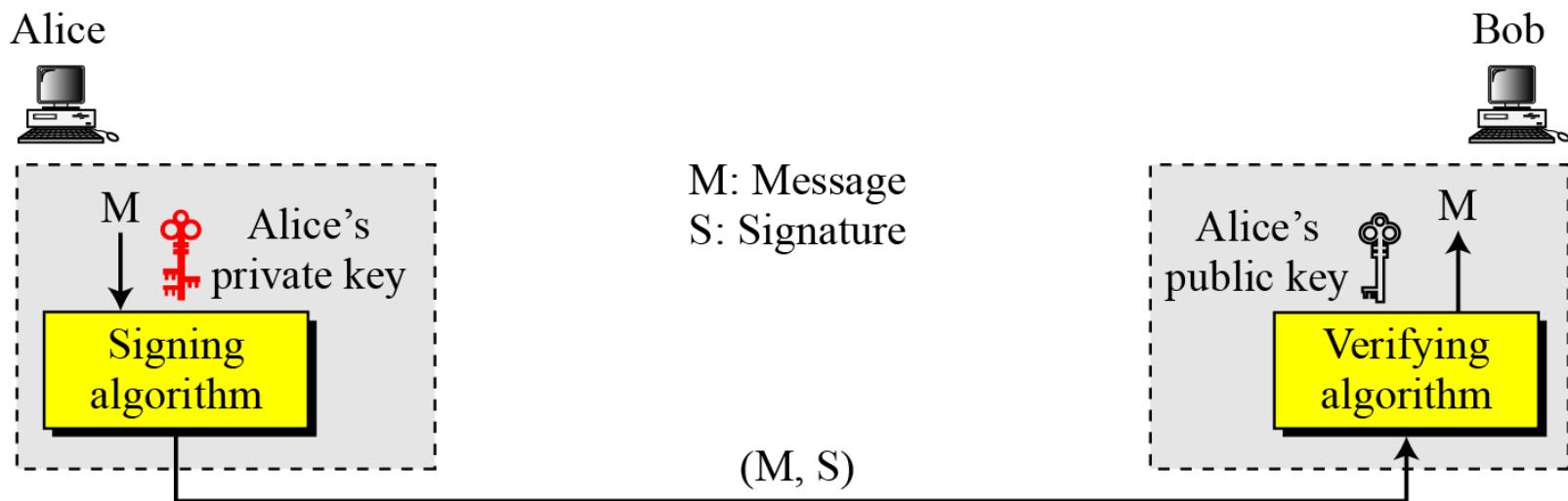
**Figure 1** *Digital signature process*



Hình 1 cho thấy quy trình chữ ký điện tử. Người gửi sử dụng một thuật toán ký để ký vào bản tin. Tin nhắn và chữ ký được gửi đến người nhận. Người nhận nhận được thông điệp và chữ ký và áp dụng thuật toán xác thực cho sự kết hợp. Nếu kết quả là true, thông báo được chấp nhận; nếu không, nó bị từ chối.

## 2.1 Need for Keys: Cần có các khóa

Figure 2 Adding key to the digital signature process



**Note**

Một chữ ký điện tử cần một hệ thống khóa công khai. Người ký bằng khóa riêng của cô ấy; người xác minh xác minh bằng khóa công khai của người ký.

A digital signature needs a public-key system.  
The signer signs with her private key; the verifier  
verifies with the signer's public key.

## 2.1 Continued

Một chữ ký điện tử cần một hệ thống khóa công khai. Người ký gửi bằng khóa riêng của cô ấy; người xác minh xác minh bằng khóa công khai của người ký.

### Note

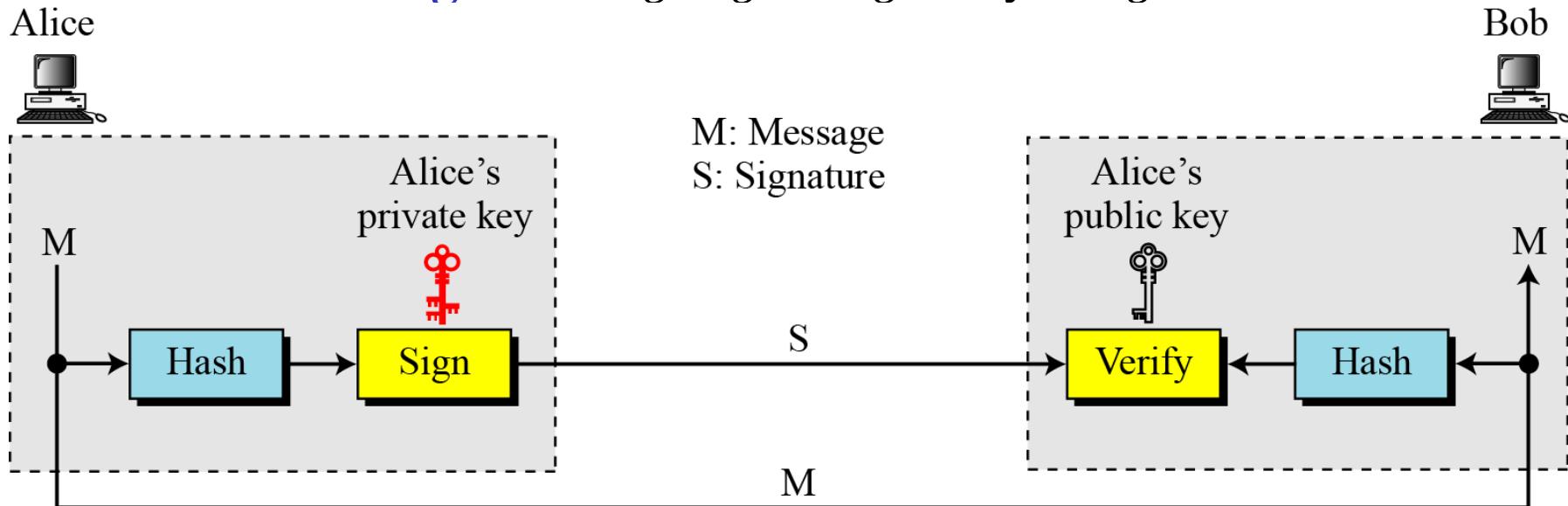
A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.

Chúng ta cần phân biệt khóa bí mật và công khai trong chữ ký số so với khóa công khai và bí mật trong hệ thống mật mã bất đối xứng.

- + Đối với hệ mật bất đối xứng, khóa bí mật và công khai sinh bởi bên người nhận, khóa công khai được sử dụng cho phép mã hóa, khóa bí mật được sử dụng để giải mã hóa.
- + Trong chữ ký số, khóa bí mật và công khai do bên người gửi sử dụng, bên gửi sử dụng khóa bí mật, bên nhận sử dụng khóa công khai của bên gửi.

## 2.2 Signing the Digest: Ký thông báo

Figure 3 Signing the digest: Ký thông báo



Trong hệ thống chữ ký số, các bản tin thường dài, nên giải pháp đưa ra là ký bản tin Digest (có độ dài cố định, ngắn hơn bản tin gốc rất nhiều) của nó. Khi đó cần chú ý việc chọn Digest phải có quan hệ 1-1 với bản tin gốc.

# 3 SERVICES: Các dịch vụ

*We discussed several security services in Chapter 1 including message **confidentiality**, message **authentication**, message **integrity**, and **nonrepudiation**. A digital signature can directly provide the last three; for message confidentiality we still need **encryption/decryption**.*

## **Topics discussed in this section:**

- 3.1 Message Authentication**
- 3.2 Message Integrity**
- 3.3 Nonrepudiation**
- 3.4 Confidentiality**

### 3 SERVICES: Các dịch vụ

*Chúng ta đã thảo luận về một số dịch vụ bảo mật trong Chương 1 bao gồm bí mật thông điệp, xác thực thông báo, tính toàn vẹn của thông điệp và không từ chối. Một chữ ký điện tử có thể cung cấp trực tiếp ba phần cuối cùng; để bảo mật thư, chúng ta vẫn cần mã hóa / giải mã.*

### 3.1 Message Authentication

#### Xác thực bản tin

A secure digital signature scheme, like a secure conventional signature can provide message authentication.

Một lược đồ chữ ký số an toàn, giống như một chữ ký thông thường an toàn có thể cung cấp việc xác thực thông điệp (không dễ bị sao chép).

##### Note

A digital signature provides message authentication.  
Chữ ký điện tử cung cấp xác thực bản tin.

### 3.2 Message Integrity: Toàn vẹn bản tin

*The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.*

Tính toàn vẹn của thông điệp được bảo toàn ngay cả khi chúng ta ký toàn bộ thông điệp vì chúng ta không thể có được chữ ký giống nhau nếu thông điệp bị thay đổi.

Hiện nay, việc sử dụng lược đồ hàm băm để ký và thuật toán xác thực được sử dụng phổ biến.

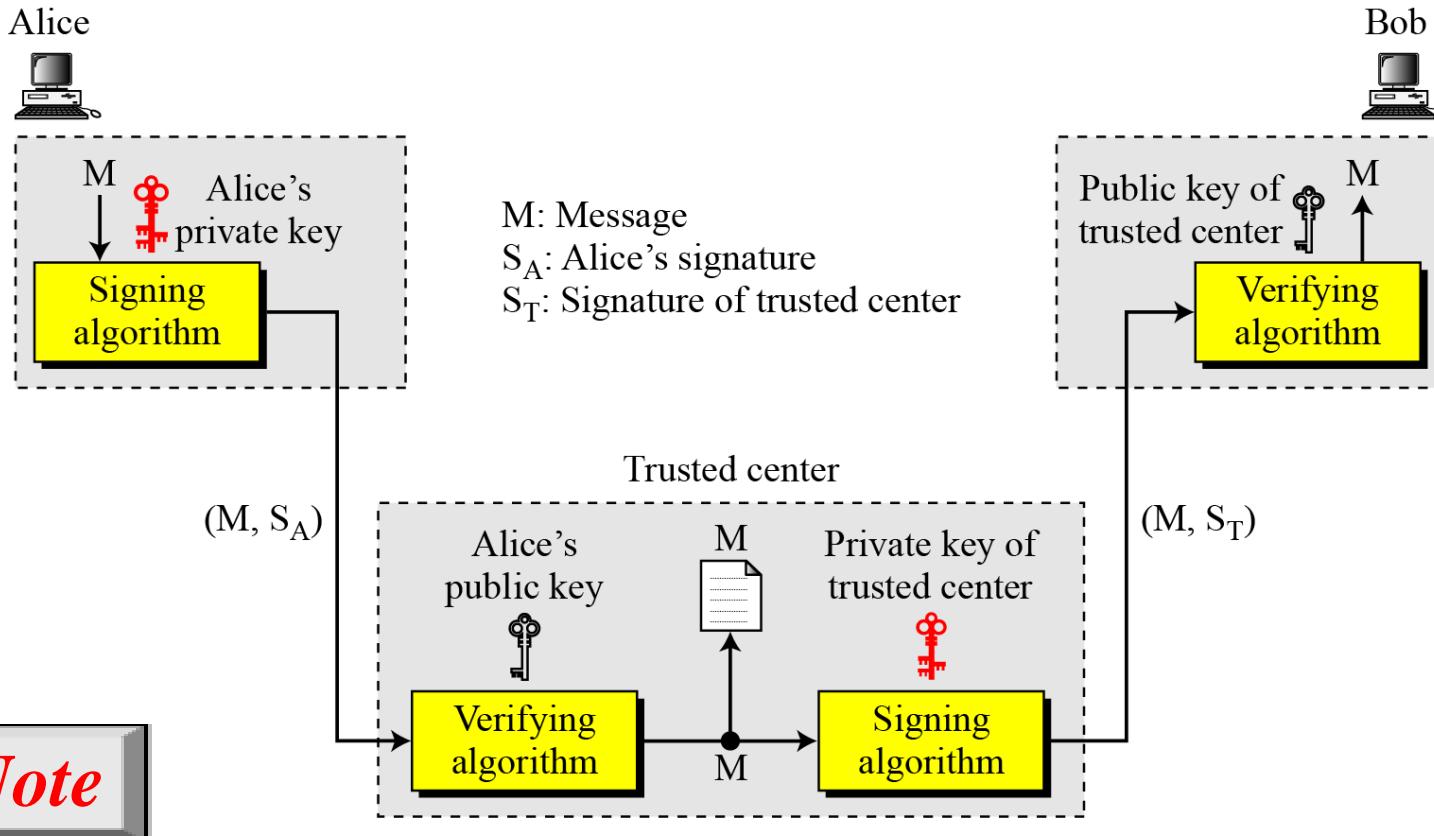
**Note**

A digital signature provides message integrity.

### 3.3 Nonrepudiation: Không bác bỏ

**Figure 4 Using a trusted center for nonrepudiation**

Sử dụng một trung tâm đáng tin cậy để không từ chối



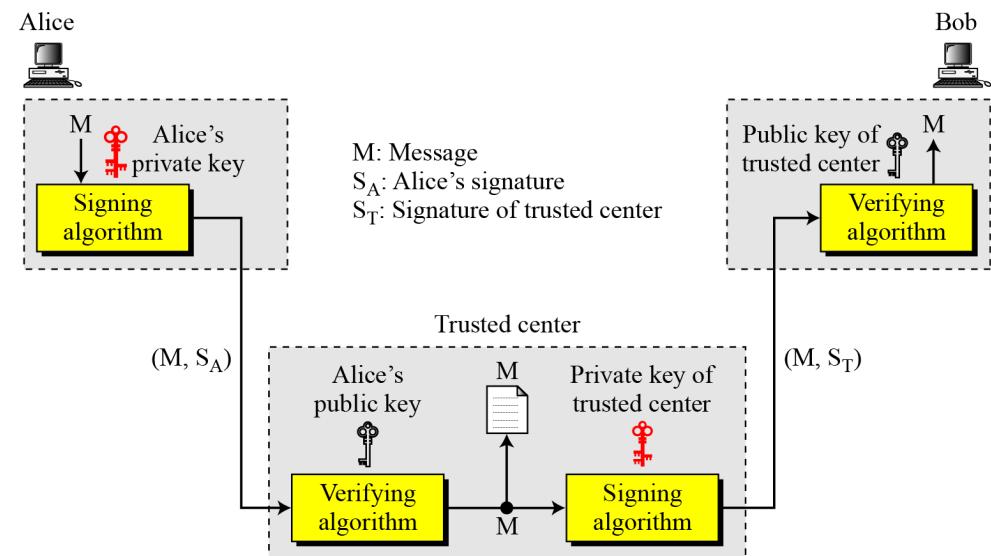
**Nonrepudiation can be provided using a trusted party.**

### 3.3 Nonrepudiation: Không bác bỏ

If Alice signs a message and then denies it, can Bob later prove that Alice actually signed it? For example, if Alice sends a message to a bank (Bob) and asks to transfer \$10,000 from her account to Ted's account, can Alice later deny that she sent this message? With the scheme we have presented so far, Bob might have a problem. Bob must keep the signature on file and later use Alice's public key to create the original message to prove the message in the file and the newly created message are the same. This is not feasible because Alice may have changed her private or public key during this time; she may also claim that the file containing the signature is not authentic.

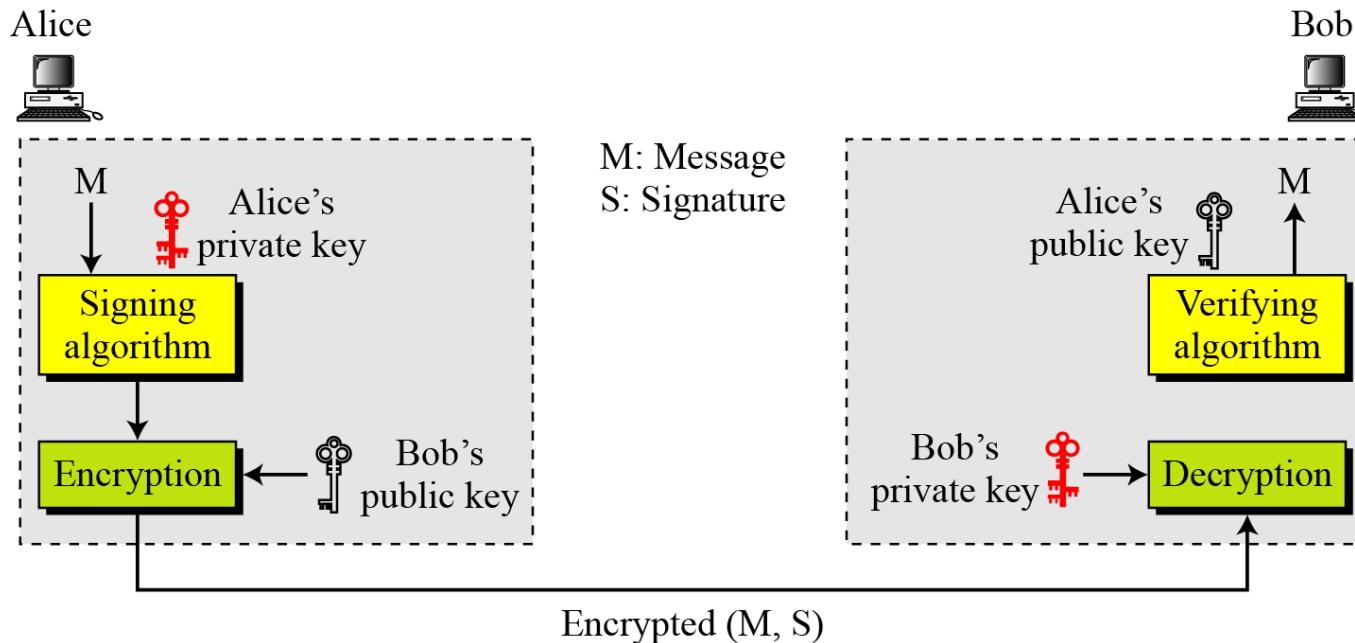
One solution is a trusted third party. People can create an established trusted party among themselves.

**Nonrepudiation can be provided using a trusted party.**



### 3.4 Confidentiality: Bí mật

Figure 5 Adding confidentiality to a digital signature scheme



**Note**

Chữ ký điện tử không cung cấp quyền riêng tư. Nếu có nhu cầu về quyền riêng tư, một lớp mã hóa / giải mã khác phải được áp dụng.

A digital signature does not provide privacy.  
If there is a need for privacy, another layer of encryption/decryption must be applied.

# 4 ATTACKS ON DIGITAL SIGNATURE

*This section describes some attacks on digital signatures and defines the types of forgery.*

*Phần này mô tả một số cuộc tấn công vào chữ ký điện tử và xác định các loại giả mạo.*

## **Topics discussed in this section:**

- 4.1      Attack Types: Các loại tấn công**
- 4.2      Forgery Types: Các loại giả mạo**

## 4.1 Attack Types: Các loại tấn công

### **Key-Only Attack: Tấn công chỉ vào khóa**

Eve tấn công vào thông tin công khai cho bởi Alice. Để tách được bản tin  $M$ , Eve cần tạo chữ ký giả của Alice để thuyết phục Bob cho rằng nó là đến từ Alice.

### **Known-Message Attack: Tấn công bản tin đã biết**

Eve truy cập vào một hoặc nhiều cặp bản tin đã ký của Alice sau đó cố tạo ra một bản tin khác để cho Alice ký vào đó.

### **Chosen-Message Attack: Tấn công bản tin được chọn**

## 4.2 Forgery Types: Các loại giả mạo

*Existential Forgery: Giả mạo tồn tại*

*Selective Forgery: Giả mạo có chọn lọc*

# 5 DIGITAL SIGNATURE SCHEMES

*Several digital signature schemes have evolved during the last few decades. Some of them have been implemented.*

*Một số lược đồ chữ ký điện tử đã phát triển trong vài thập kỷ qua. Một số trong số chúng đã được thực hiện.*

## **Topics discussed in this section:**

- 5.1 RSA Digital Signature Scheme**
- 5.2 ElGamal Digital Signature Scheme**
- 5.3 Schnorr Digital Signature Scheme**
- 5.4 Digital Signature Standard (DSS)**
- 5.5 Elliptic Curve Digital Signature Scheme**

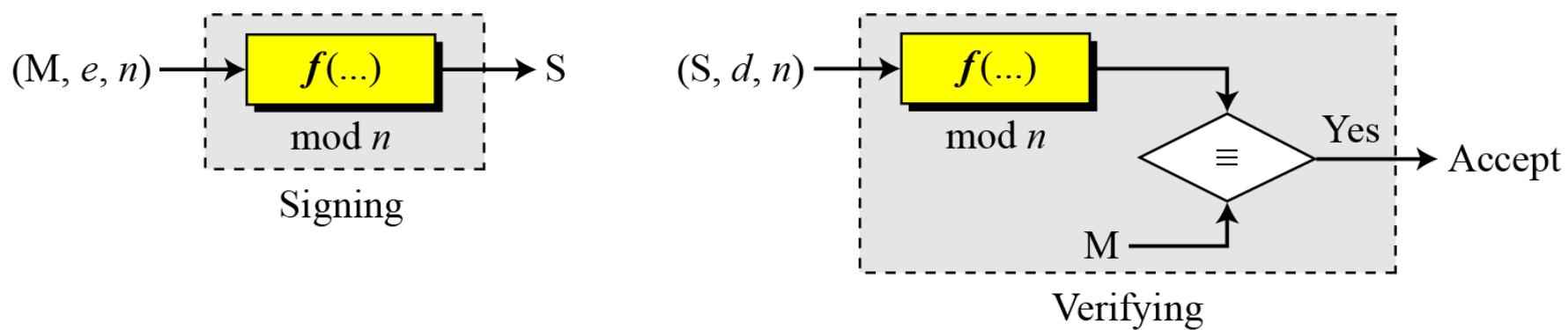
## 5.1 RSA Digital Signature Scheme

### Sơ đồ chữ ký số RSA

**Figure 6 General idea behind the RSA digital signature scheme**  
**Ý tưởng chung đằng sau lược đồ chữ ký số RSA**

M: Message  
S: Signature

$(e, n)$ : Alice's public key  
 $d$ : Alice's private key



- + Sơ đồ chữ ký số RSA thay đổi vai trò của các khóa bí mật và công khai: các khóa bí mật và công khai sử dụng là của bên gửi (không phải bên nhận); bên gửi sử dụng khóa bí mật để ký, bên nhận sử dụng khóa công khai của bên gửi để xác thực.
- + Khóa bí mật đóng vai trò chữ ký của bên gửi, khóa công khai của bên gửi đóng vai trò bản sao của chữ ký. Hàm  $f(\dots)$  dùng chung cho signing và verifying

## 5.1 Continued

### *Key Generation: Tạo khóa*

*Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA*

*Việc tạo khóa trong lược đồ chữ ký số RSA hoàn toàn giống với việc tạo khóa trong RSA*

#### **Note**

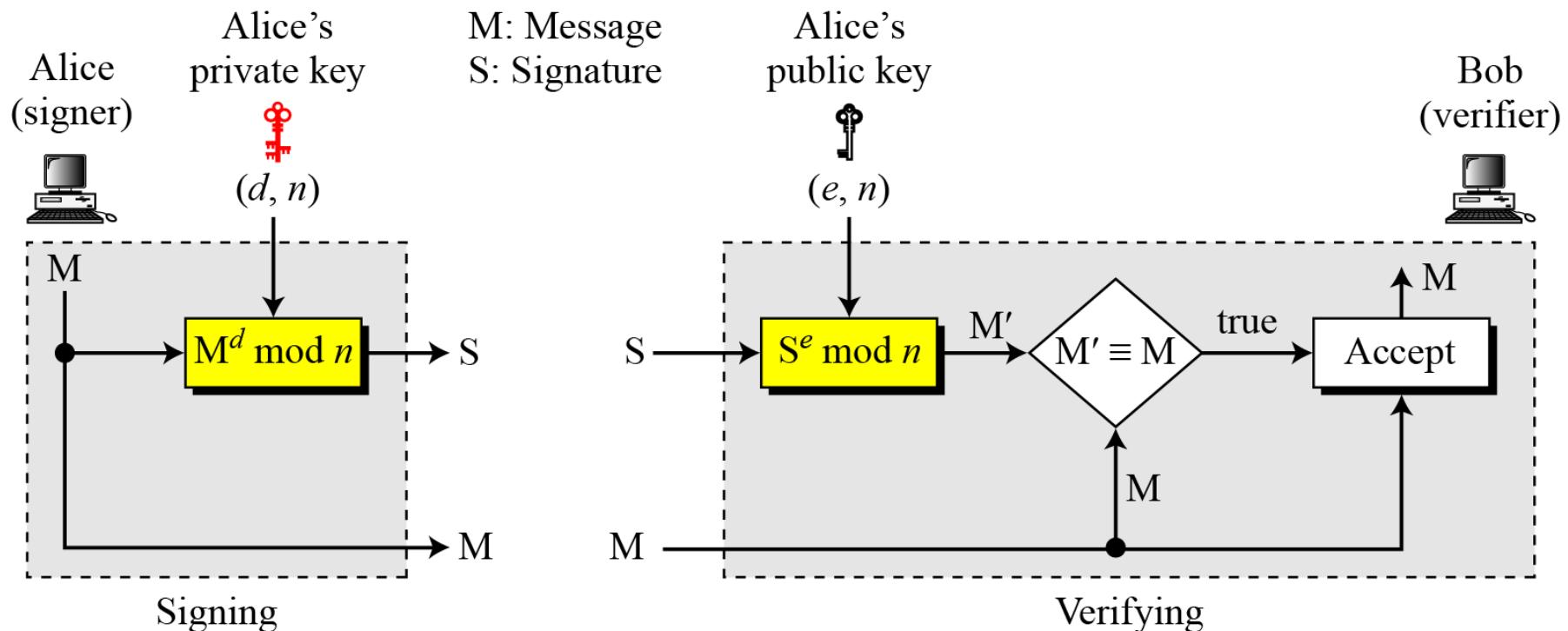
In the RSA digital signature scheme,  $d$  is private;  
 $e$  and  $n$  are public.

# 5.1 Continued

## Signing and Verifying Ký và Xác minh

Chọn hai số nguyên tố  $p, q$ , tính  $n=p \times q$ ,  
 $\Phi(n)=(p-1)(q-1)$ , chọn  $e$ , tính  $d$  thỏa mãn  
 $e \times d = 1 \pmod{\Phi(n)}$

Figure 7 RSA digital signature scheme



## 5.1 *Continued*

### Example 1

As a trivial example, suppose that Alice chooses  $p = 823$  and  $q = 953$ , and calculates  $n = 784319$ . The value of  $\phi(n)$  is **782544**. Now she chooses  $e = 313$  and calculates  $d = 160009$ . At this point key generation is complete. Now imagine that Alice wants to send a message with the value of  $M = 19070$  to Bob. She uses her private exponent, **160009**, to sign the message:

$$M: 19070 \rightarrow S = (19070^{160009}) \bmod 784319 = 210625 \bmod 784319$$

Alice sends the message and the signature to Bob. Bob receives the message and the signature. He calculates

$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \rightarrow M \equiv M' \bmod n$$

**Bob accepts the message because he has verified Alice's signature.**

## 5.1   Continued

### Example 1

Ví dụ đơn giản, giả sử Alice chọn  $p = 823$  và  $q = 953$ , và tính  $n = 784319$ . Giá trị của  $\phi(n)$  là  $782544$ . Bây giờ cô ấy chọn  $e = 313$  và tính  $d = 160009$ . Tại thời điểm này, sinh khóa hoàn tất. Bây giờ, hãy tưởng tượng rằng Alice muốn gửi một tin nhắn có giá trị  $M = 19070$  cho Bob. Cô ấy sử dụng số mū riêng của mình,  $160009$ , để ký tin nhắn:

$$M: 19070 \rightarrow S = (19070^{160009}) \bmod 784319 = 210625 \bmod 784319$$

Alice gửi tin nhắn và chữ ký cho Bob. Bob nhận được tin nhắn và chữ ký và tính toán

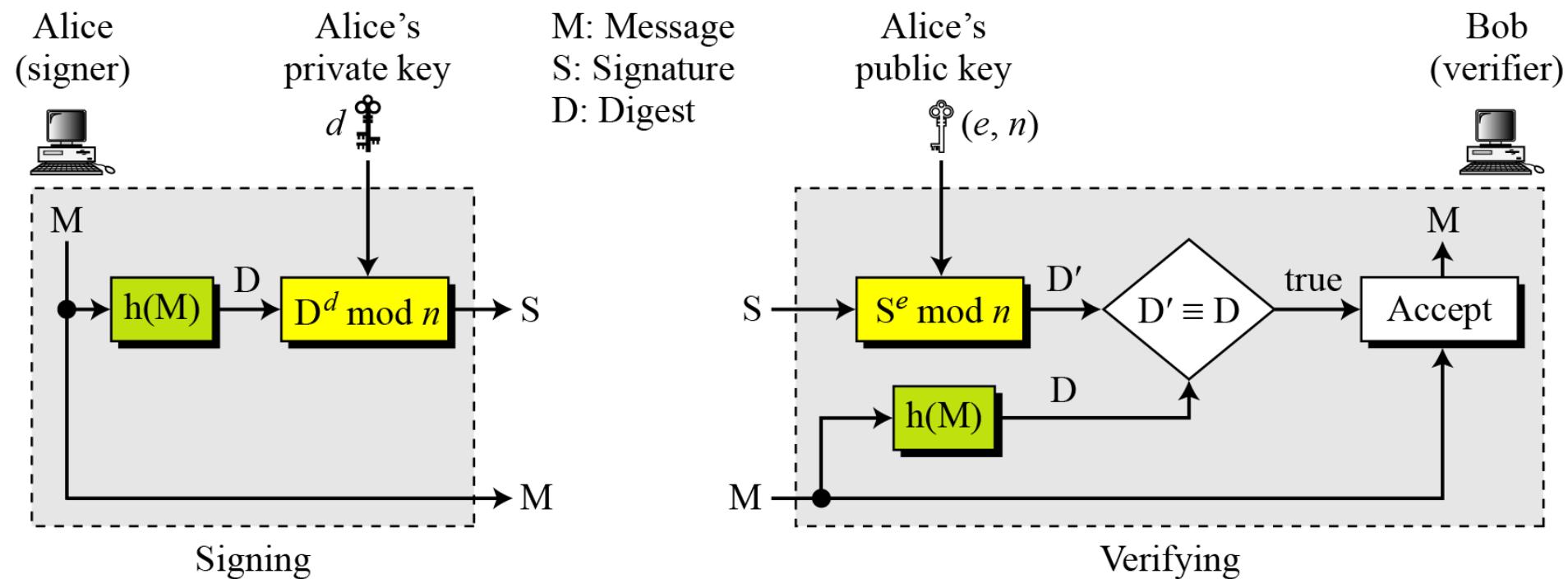
$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \rightarrow M \equiv M' \bmod n$$

Bob chấp nhận tin nhắn vì Bob đã xác minh chữ ký của Alice.

# 5.1 Continued

## RSA Signature on the Message Digest

**Figure 8** The RSA signature on the message digest



## *5.1 Continued*

### **Note**

**When the digest is signed instead of the message itself, the susceptibility of the RSA digital signature scheme depends on the strength of the hash algorithm.**

**Khi Digest được ký thay vì chính thông điệp, tính nhạy cảm của lược đồ chữ ký số RSA phụ thuộc vào độ mạnh của thuật toán băm.**

## 5.2 ElGamal Digital Signature Scheme

### Lược đồ chữ ký số ElGamal

**Figure 9 General idea behind the ElGamal digital signature scheme**  
**Ý tưởng chung**

$S_1, S_2$ : Signatures

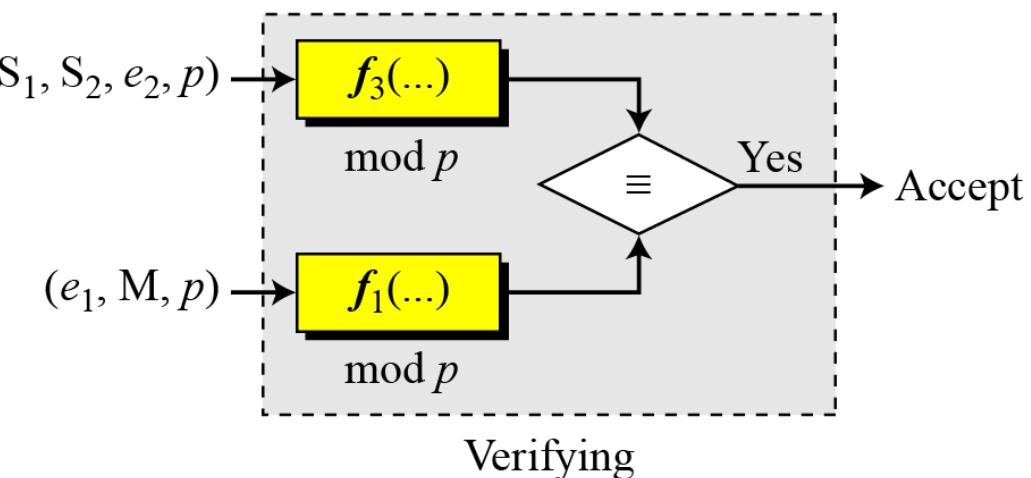
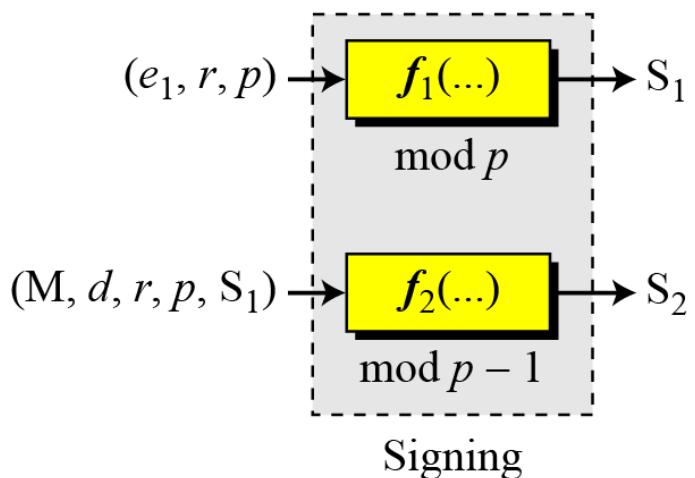
M: Message

$(e_1, e_2, p)$ : Alice's public key

$d$ : Alice's private key

$r$ : Random secret

$$e_2 = e_1^d$$



## 5.2 Continued

### **Key Generation: Tạo Khóa**

*The key generation procedure here is exactly the same as the one used in the cryptosystem.*

*Quy trình tạo khóa ở đây hoàn toàn giống với quy trình được sử dụng trong hệ thống mật mã*

#### **Note**

In ElGamal digital signature scheme,  $(e_1, e_2, p)$  is Alice's public key;  $d$  is her private key.

## 5.2 Continued

### Verifying and Signing: Xác minh và ký

**Figure 10 ElGamal digital signature scheme**  
**Lược đồ chữ ký số ElGamal**

M: Message

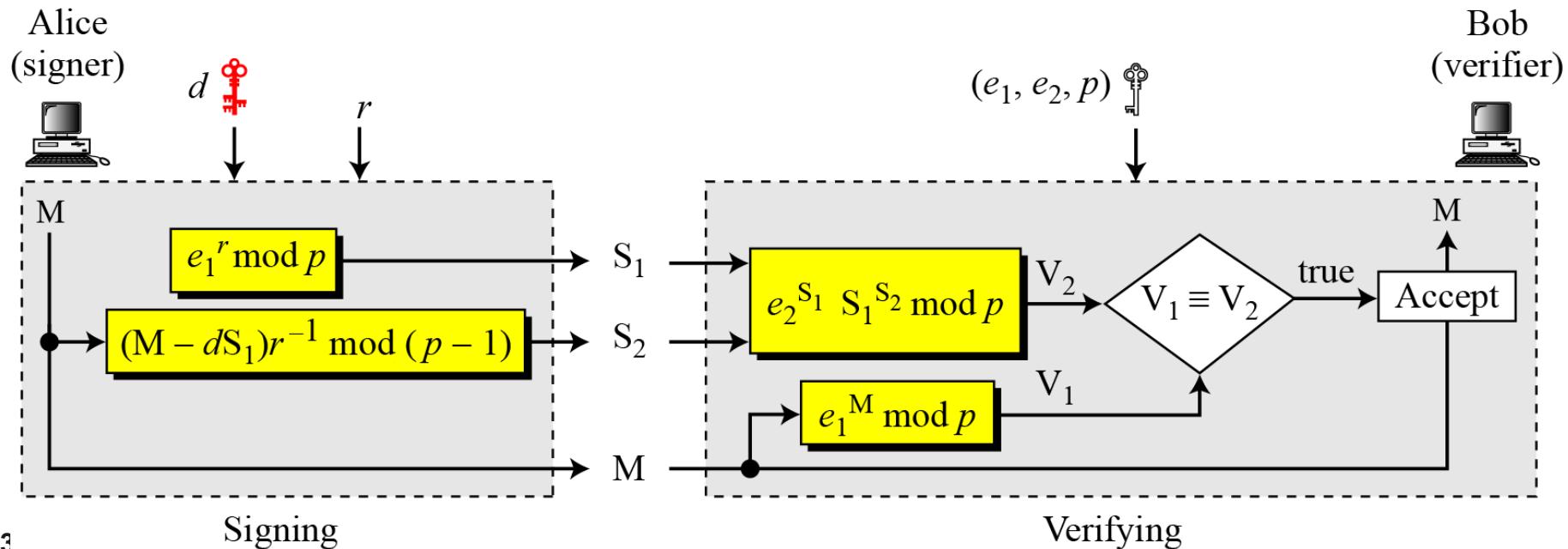
S<sub>1</sub>, S<sub>2</sub>: Signatures

V<sub>1</sub>, V<sub>2</sub>: Verifications

r: Random secret

d: Alice's private key

(e<sub>1</sub>, e<sub>2</sub>, p): Alice's public key



## 5.1 *Continued*

### Example 2

Here is a trivial example. Alice chooses  $p = 3119$ ,  $e_1 = 2$ ,  $d = 127$  and calculates  $e_2 = 2^{127} \bmod 3119 = 1702$ . She also chooses  $r$  to be 307. She announces  $e1$ ,  $e2$ , and  $p$  publicly; she keeps  $d$  secret. The following shows how Alice can sign a message.

$$M = 320$$

$$S_1 = e_1^r = 2^{307} = 2083 \bmod 3119$$

$$S_2 = (M - d \times S_1) \times r^{-1} = (320 - 127 \times 2083) \times 307^{-1} = 2105 \bmod 3118$$

Alice sends  $M$ ,  $S_1$ , and  $S_2$  to Bob. Bob uses the public key to calculate  $V_1$  and  $V_2$ .

$$V_1 = e_1^M = 2^{320} = 3006 \bmod 3119$$

$$V_2 = d^{S_1} \times S_1^{S_2} = 1702^{2083} \times 2083^{2105} = 3006 \bmod 3119$$

## 5.1 *Continued*

### Example 2

Đây là một ví dụ nhỏ. Alice chọn  $p = 3119$ ,  $e_1 = 2$ ,  $d = 127$  và tính  $e_2 = 2^{127} \text{ mod } 3119 = 1702$ . Cô ấy cũng chọn  $r$  là 307. Cô ấy công bố  $e_1$ ,  $e_2$  và  $p$ ; cô ấy giữ bí mật  $d$ . Sau đây là cách Alice có thể ký một tin nhắn.

$$M = 320$$

$$S_1 = e_1^r = 2^{307} = 2083 \text{ mod } 3119$$

$$S_2 = (M - d \times S_1) \times r^{-1} = (320 - 127 \times 2083) \times 307^{-1} = 2105 \text{ mod } 3118$$

Alice gửi  $M$ ,  $S_1$  và  $S_2$  cho Bob. Bob sử dụng khóa công khai để tính  $V_1$  và  $V_2$ .

$$V_1 = e_1^M = 2^{320} = 3006 \text{ mod } 3119$$

$$V_2 = d^{S_1} \times S_1^{S_2} = 1702^{2083} \times 2083^{2105} = 3006 \text{ mod } 3119$$

## 5.1 *Continued*

### Example 13.3

Now imagine that Alice wants to send another message,  $M = 3000$ , to Ted. She chooses a new  $r$ , 107. Alice sends  $M$ ,  $S_1$ , and  $S_2$  to Ted. Ted uses the public keys to calculate  $V_1$  and  $V_2$ .

Bây giờ, hãy tưởng tượng rằng Alice muốn gửi một tin nhắn khác,  $M = 3000$ , cho Ted. Cô ấy chọn một  $r$  mới, 107. Alice gửi  $M$ ,  $S_1$  và  $S_2$  cho Ted. Ted sử dụng các khóa công khai để tính  $V_1$  và  $V_2$ .

$$M = 3000$$

$$S_1 = e_1^r = 2^{107} = 2732 \text{ mod } 3119$$

$$S_2 = (M - d \times S_1) r^{-1} = (3000 - 127 \times 2083) \times 107^{-1} = 2526 \text{ mod } 3118$$

$$V_1 = e_1^M = 2^{3000} = 704 \text{ mod } 3119$$

$$V_2 = d^{S_1} \times S_1^S = 1702^{2732} \times 2083^{2526} = 704 \text{ mod } 3119$$

# Bài Tập Chữ ký số RSA, ElGamal

1. Sử dụng lược đồ RSA: cho  $p=809$ ,  $q=751$  và  $d=23$ , tính khóa công khai  $e$  sau đó thực hiện:
  - a. Ký và xác thực bản tin  $M_1=100$ , kết quả là  $S_1$
  - b. Ký và xác thực bản tin  $M_2=50$ , kết quả là  $S_2$
  - c. Chỉ ra nếu  $M=M_1 \times M_2=50000$  thì  $S=S_1 \times S_2$ ?
  
2. Sử dụng lược đồ ElGamal, cho  $p=881$  và  $d=700$ , tìm giá trị  $e_1$  và  $e_2$ , chọn  $r=17$ , hãy tìm giá trị  $S_1$  và  $S_2$  nếu cho  $M=400$ ?

## 5.3 Schnorr Digital Signature Scheme

### Sơ đồ chữ ký số Schnorr

**Figure 11** General idea behind the Schnorr digital signature scheme  
Ý tưởng chung dựa trên lược đồ ElGamal

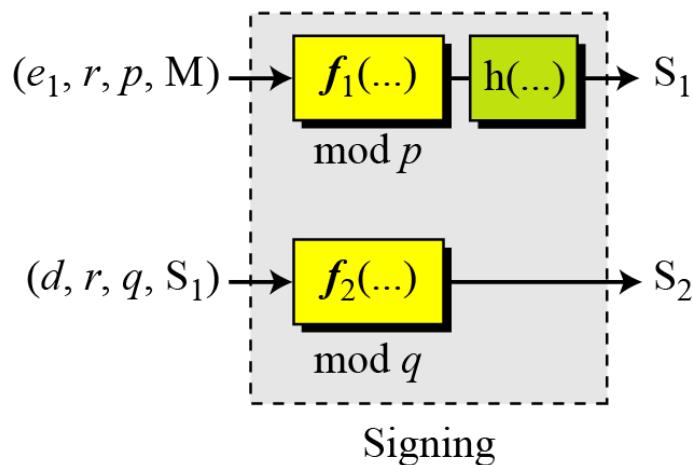
$S_1, S_2$ : Signatures

( $d$ ): Alice's private key

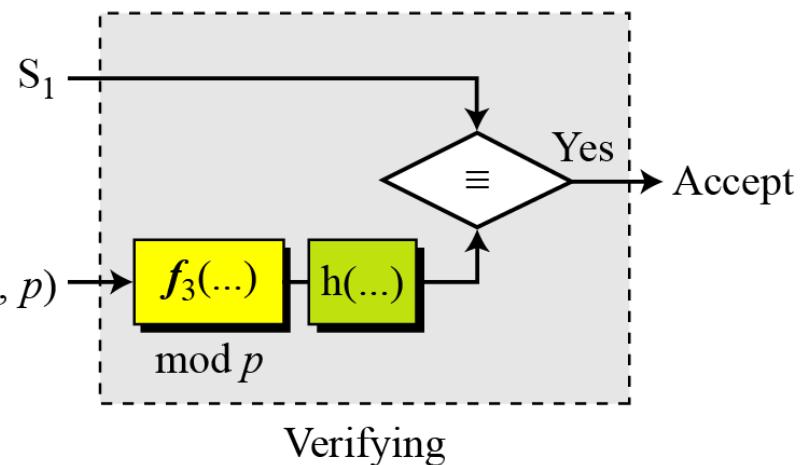
M: Message

r: Random secret

( $e_1, e_2, p, q$ ): Alice's public key



$(S_1, S_2, M, e_1, e_2, p)$



## 5.3 Continued

### Key Generation: Tạo khóa

- 1) Alice selects a prime  $p$ , which is usually 1024 bits in length.
- 2) Alice selects another prime  $q$  (thường cùng kích thước digest của hàm  $h(\dots)$ )
- 3) Alice chooses  $e_1$  to be the  $q$ th root of 1 modulo  $p$ .  $e_1 = e_0^{(p-1)q} \text{ mod } p$ ,  **$e_0$  là giá trị khởi tạo cho bảng.**
- 4) Alice chooses an integer,  $d$ , as her private key.
- 5) Alice calculates  $e_2 = e_1^d \text{ mod } p$ .
- 6) Alice's public key is  $(e_1, e_2, p, q)$ ; her private key is  $(d)$ .

**Note**

In the Schnorr digital signature scheme, Alice's public key is  $(e_1, e_2, p, q)$ ; her private key  $(d)$ .

## 5.3 Continued

### Key Generation: Tạo khóa

- 1) Alice chọn một số nguyên tố  $p$ , thường có độ dài 1024 bit.
- 2) Alice chọn  $q$  nguyên tố khác.
- 3) Alice chọn  $e_1$  là căn thứ  $q$  của 1 módun  $p$ .
- 4) Alice chọn một số nguyên,  $d$ , làm khóa riêng của cô ấy.
- 5) Alice tính  $e_2 = e_1^d \text{ mod } p$ .
- 6) Khóa công khai của Alice là  $(e_1, e_2, p, q)$ ; khóa riêng của cô ấy là  $(d)$ .

**Note**

In the Schnorr digital signature scheme, Alice's public key is  $(e_1, e_2, p, q)$ ; her private key  $(d)$ .

# 5.3 Continued

## *Signing and Verifying: Ký và Xác minh*

**Figure 12 Schnorr digital signature scheme**

M: Message

S<sub>1</sub>, S<sub>2</sub>: Signatures

V: Verification

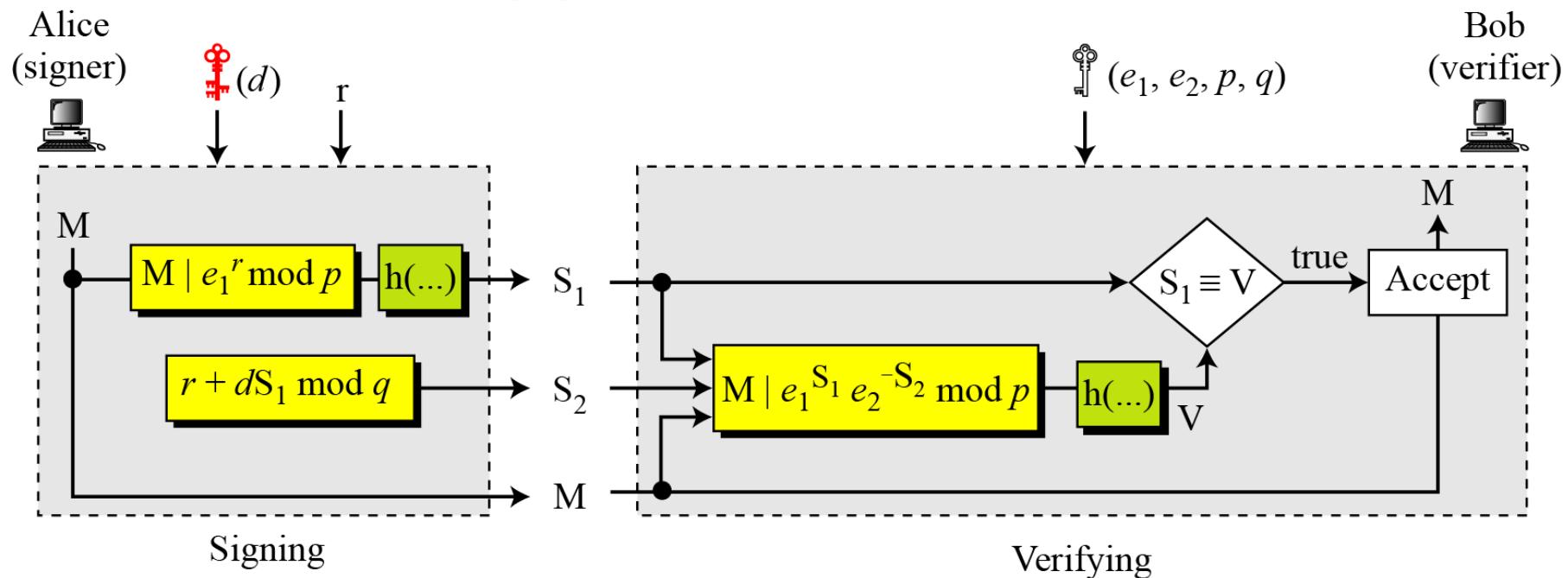
r: Random secret

(d): Alice's private key

(e<sub>1</sub>, e<sub>2</sub>, p, q): Alice's public key

| : Concatenation

h(...): Hash algorithm



## 5.3 Continued

### *Signing*

1. *Alice chooses a random number  $r$ ,  $1 < r < q$ .*
2. *Alice calculates  $S_1 = h(M|e_1^r \bmod p)$ .*
3. *Alice calculates  $S_2 = r + d \times S_1 \bmod q$ .*
4. *Alice sends  $M$ ,  $S_1$ , and  $S_2$ .*

### *Verifying Message*

1. *Bob calculates  $V = h(M | e_1^{S_2} e_2^{-S_1} \bmod p)$ .*
2. *If  $S_1$  is congruent to  $V$  modulo  $p$ , the message is accepted;*

## 5.1   Continued

### Example 4

Here is a trivial example. Suppose we choose  $q = 103$  and  $p = 2267$ . Note that  $p = 22 \times q + 1$ . We choose  $e_0 = 2$ , which is a primitive in  $\mathbb{Z}_{2267}^*$ . Then  $(p - 1) / q = 22$ , so we have  $e_1 = 2^{22} \bmod 2267 = 354$ . We choose  $d = 30$ , so  $e_2 = 354^{30} \bmod 2267 = 1206$ . Alice's private key is now  $(d)$ ; her public key is  $(e_1, e_2, p, q)$ .

Alice wants to send a message  $M$ . She chooses  $r = 11$  and calculates  $e_2^r = 354^{11} = 630 \bmod 2267$ . Assume that the message is 1000 and concatenation means 1000630. Also assume that the hash of this value gives the digest  $h(1000630) = 200$ . This means  $S_1 = 200$ . Alice calculates  $S_2 = r + d \times S_1 \bmod q = 11 + 1026 \times 200 \bmod 103 = 35$ . Alice sends the message  $M = 1000$ ,  $S_1 = 200$ , and  $S_2 = 35$ . The verification is left as an exercise.

## 5.4 Digital Signature Standard (DSS)

### Chuẩn Chữ Ký Số

Figure 13 General idea behind DSS scheme

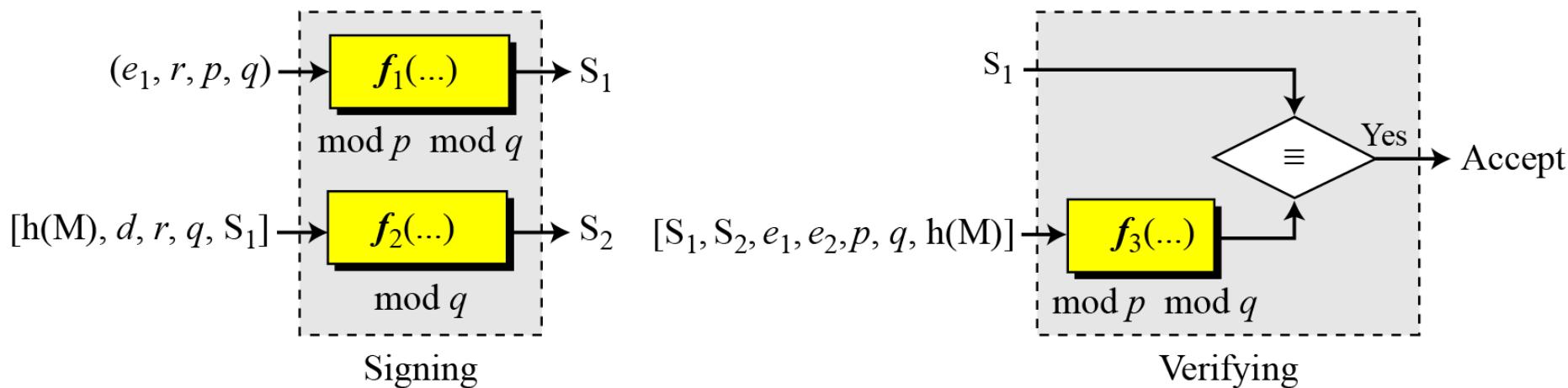
$S_1, S_2$ : Signatures

$d$ : Alice's private key

M: Message

$r$ : Random secret

$(e_1, e_2, p, q)$ : Alice's public key



*Chuẩn này được đưa ra bởi NIST năm 1994, DSS sử dụng thuật toán chữ ký số DSA dựa trên lược đồ ElGamal cùng với ý tưởng của lược đồ Schnorr.*

## 5.4 Continued

### **Key Generation: Tạo khóa**

- 1) Alice chooses primes  $p$  and  $q$ ,  $p = 512 - 1024$  bits,  $p =$  multiple of 64;  $q$  divides  $(p-1)$
- 2) Alice uses  $\langle \mathbb{Z}_p^*, \times \rangle$  and  $\langle \mathbb{Z}_q^*, \times \rangle$ .
- 3) Alice creates  $e_1$  to be the  $q$ th root of 1 modulo  $p$ ,  $e_1^p = 1 \bmod p$ .
- 4) Alice chooses  $d$  as private key and calculates  $e_2 = e_1^d$ .
- 5) Alice's public key is  $(e_1, e_2, p, q)$ ; her private key is  $(d)$ .

## 5.4 Continued

### Verifying and Signing

Figure 14 DSS scheme

M: Message

$S_1, S_2$ : Signatures

V: Verification

r: Random secret

d: Alice's private key

$(e_1, e_2, p, q)$ : Alice's public key

$h(M)$ : Message digest

Alice  
(signer)



M

$$(e_1^r \bmod p) \bmod q$$

$$(h(M) + dS_1)r^{-1} \bmod q$$

$S_1$

M

$S_2$

M

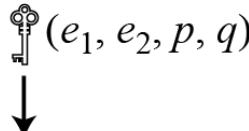


Bob  
(verifier)



M

Accept



V

Verifying

Signing

Verifying

## 5.1 Continued

### Example 5

Alice chooses  $q = 101$  and  $p = 8081$ . Alice selects  $e_0 = 3$  and calculates  $e_1 = e_0^{(p-1)/q} \bmod p = 6968$ . Alice chooses  $d = 61$  as the private key and calculates  $e_2 = e_1^d \bmod p = 2038$ . Now Alice can send a message to Bob. Assume that  $h(M) = 5000$  and Alice chooses  $r = 61$ :

$$h(M) = 5000 \quad r = 61$$

$$S_1 = (e_1^r \bmod p) \bmod q = 54$$

$$S_2 = ((h(M) + d S_1) r^{-1}) \bmod q = 40$$

Alice sends  $M$ ,  $S_1$ , and  $S_2$  to Bob. Bob uses the public keys to calculate  $V$ .

$$S_2^{-1} = 48 \bmod 101$$

$$V = [(6968^{5000 \times 48} \times 2038^{54 \times 48}) \bmod 8081] \bmod 101 = 54$$

## 5.4 Continued

### *DSS Versus RSA: DSS so với RSA*

*Computation of DSS signatures is faster than computation of RSA signatures when using the same p.*

*Tính toán chữ ký DSS nhanh hơn tính toán chữ ký RSA khi sử dụng cùng một p.*

### *DSS Versus ElGamal: DSS so với ElGamal*

*DSS signatures are smaller than ElGamal signatures because q is smaller than p.*

*Chữ ký DSS nhỏ hơn chữ ký ElGamal vì q nhỏ hơn p.*

## 5.5 Elliptic Curve Digital Signature Scheme

**Figure 15** General idea behind the ECDSS scheme

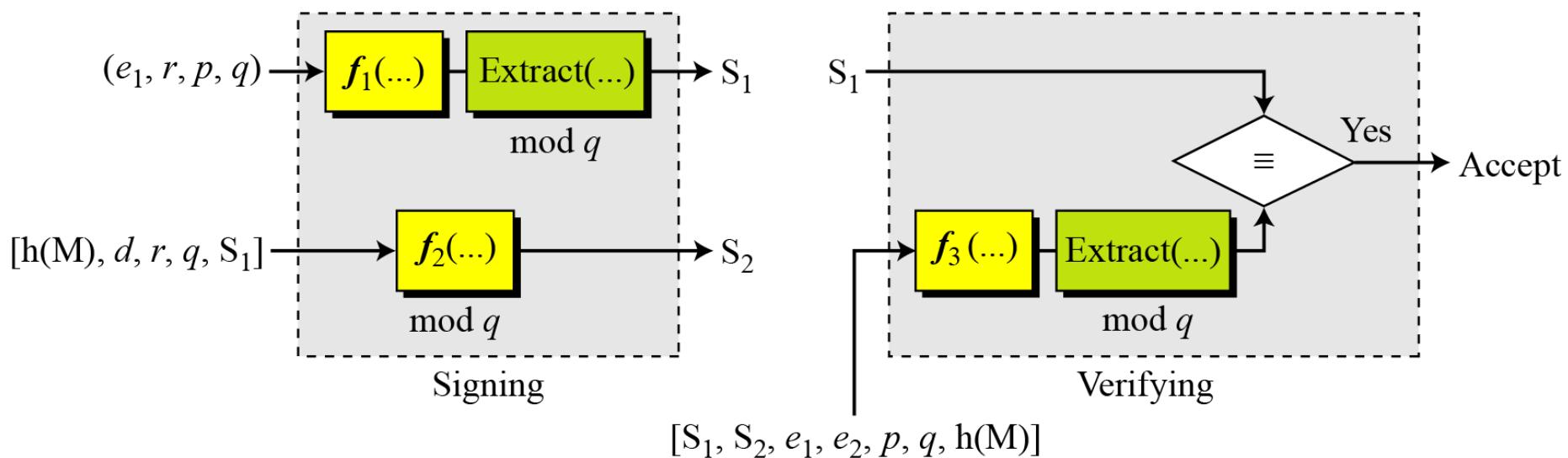
$S_1, S_2$ : Signatures

M: Message

$(a, b, p, q, e_1, e_2)$ : Alice's public key

$d$ : Alice's private key

$r$ : Random secret



## 5.5 Continued

### Key Generation

*Key generation follows these steps:*

- 1) *Alice chooses an elliptic curve  $E_p(a, b)$ .*
- 2) *Alice chooses another prime  $q$  the private key  $d$ .*
- 3) *Alice chooses  $e_1(\dots, \dots)$ , a point on the curve.*
- 4) *Alice calculates  $e_2(\dots, \dots) = d \times e_1(\dots, \dots)$ .*
- 5) *Alice's public key is  $(a, b, p, q, e1, e2)$ ; her private key is  $d$ .*

# 5.5 Continued

## Signing and Verifying

**Figure 13.16 The ECDSS scheme**

M: Message

$S_1, S_2$ : Signatures

V: Verification

r: Random secret

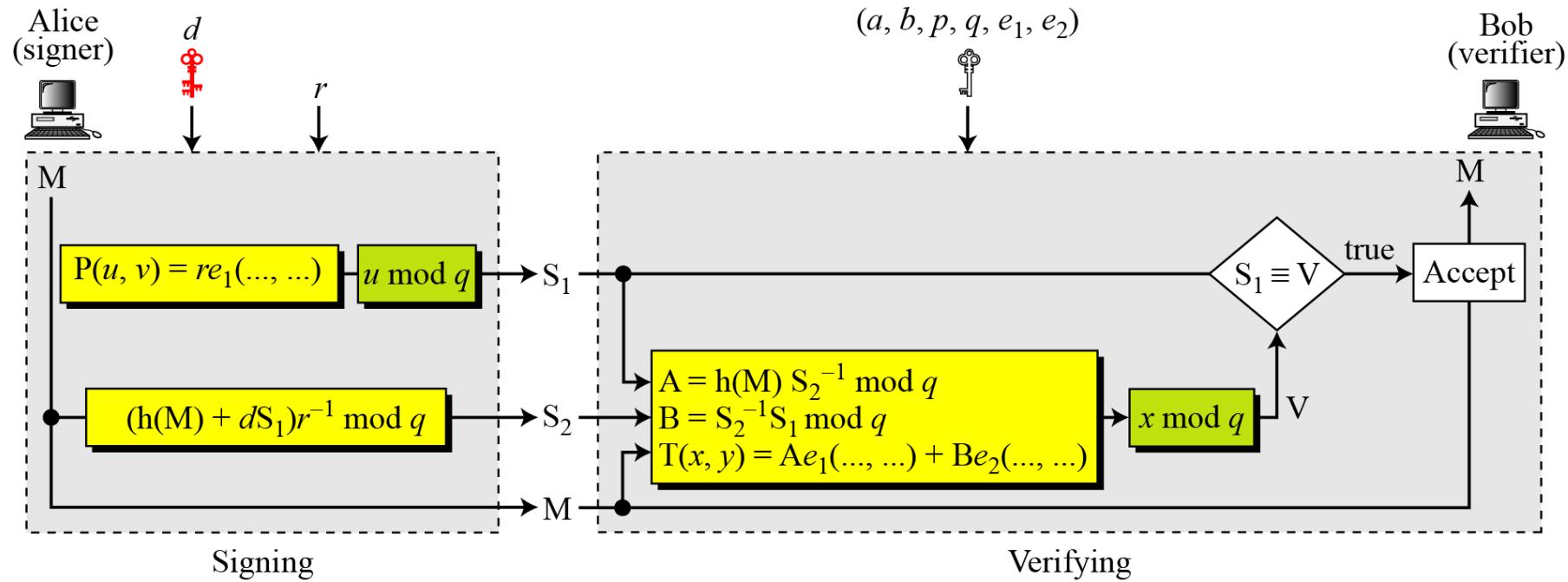
d: Alice's private key

( $a, b, p, q, e_1, e_2$ ): Alice's public key

$P(u, v), T(x, y)$ : Points on the curve

$h(M)$ : Message digest

A, B: Intermediate results



# 6 VARIATIONS AND APPLICATIONS

*This section briefly discusses variations and applications for digital signatures.*

*Phần này thảo luận ngắn gọn về các biến thể và ứng dụng cho chữ ký điện tử.*

## **Topics discussed in this section:**

**6.1 Variations**

**6.2 Applications**

## 6.1 Variations

### Time Stamped Signatures: Chữ ký đóng dấu thời gian

Sometimes a signed document needs to be time stamped to prevent it from being replayed by an adversary. This is called time-stamped digital signature scheme.

Đôi khi, một tài liệu đã ký cần được đóng dấu thời gian để tránh bị kẻ thù phát lại. Đây được gọi là lược đồ chữ ký điện tử có dấu thời gian.

### Blind Signatures: Chữ ký mù

Sometimes we have a document that we want to get signed without revealing the contents of the document to the signer.

Đôi khi chúng ta có một tài liệu mà chúng ta muốn ký mà không tiết lộ nội dung của tài liệu đó cho người ký.