

# **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

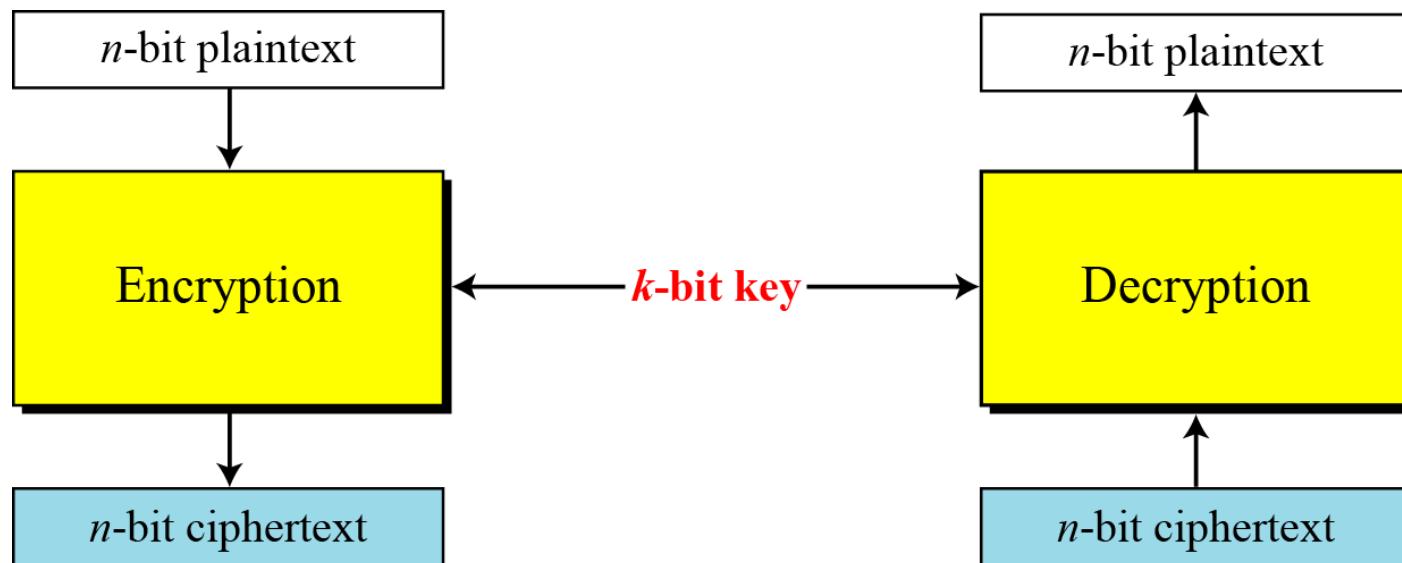
*Mật mã khối hiện đại khóa đối xứng thực hiện mật mã hóa khối **n-bit** **bản rõ** hoặc giải mật mã khối n-bit của bản mật. Các thuật toán mật mã hóa / giải mật mã đều dùng một khóa k-bit.*

## 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

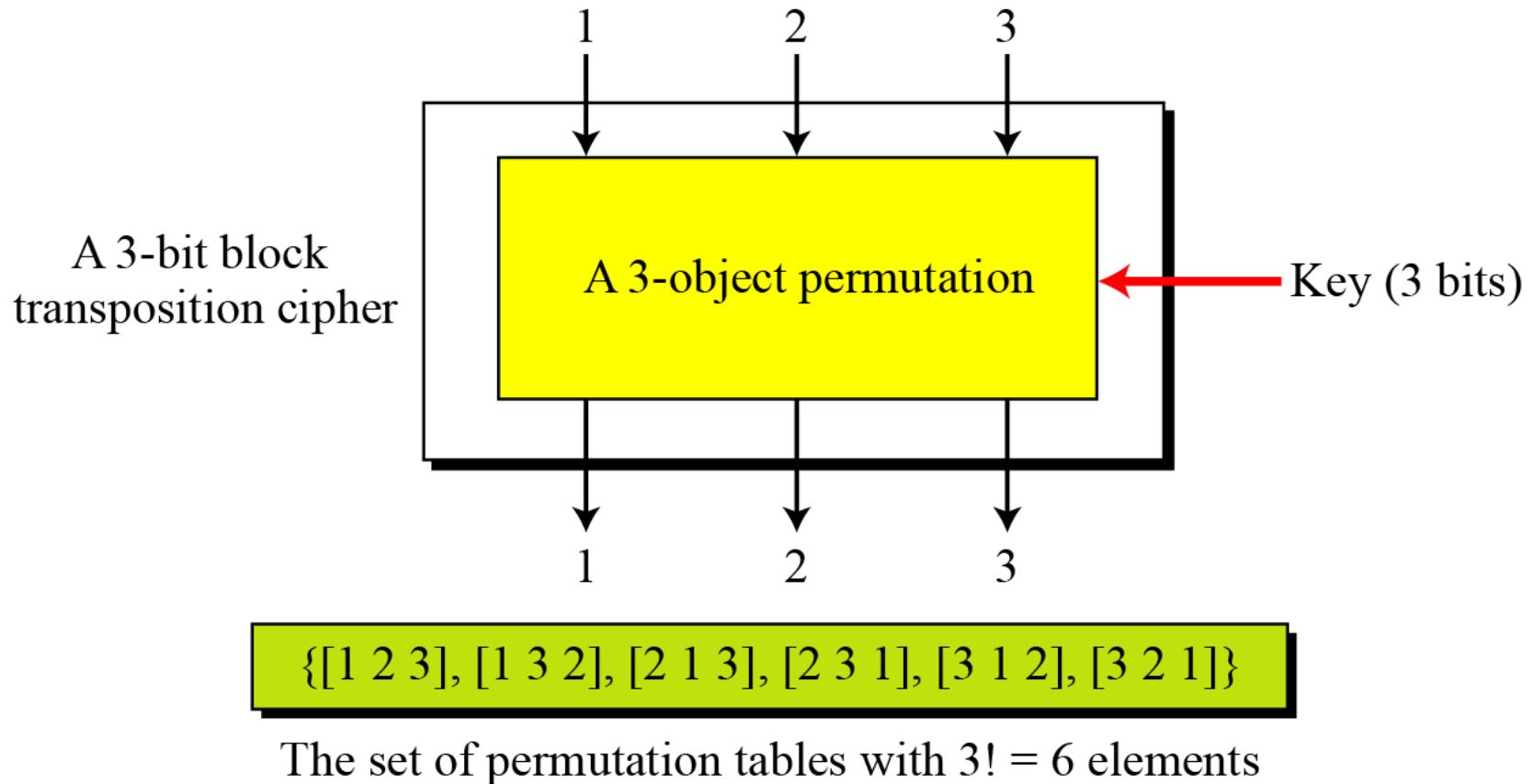
Chỉ ra mô hình và tập các bảng hoán vị cho mã chuyển vị khối 3-bit (mỗi khối có kích thước 3 bits)

### 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*

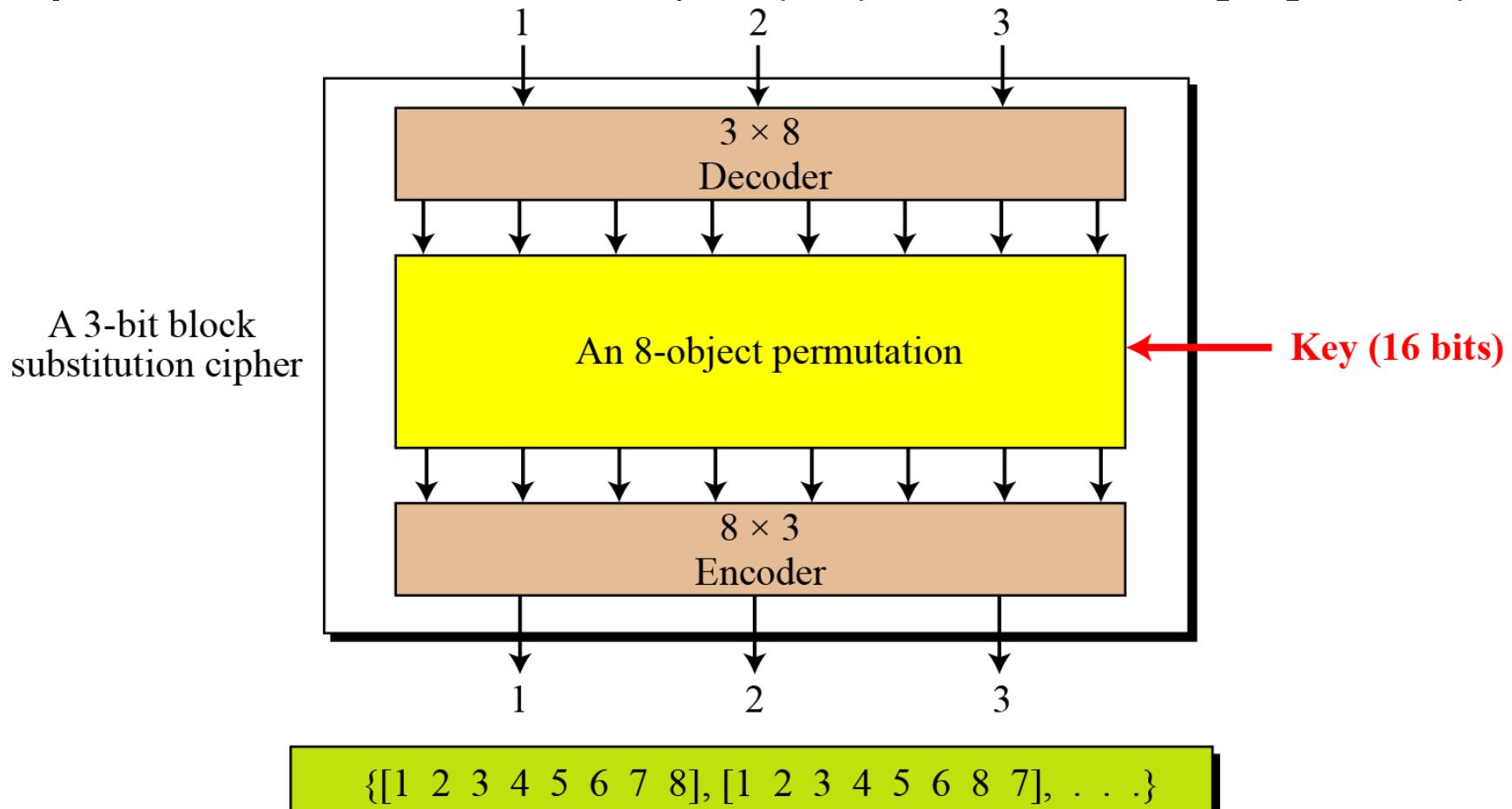
*Mã thay thế khoá kích thước đầy đủ không chuyển đổi các bit; nó thay thế các bit. Chúng ta có thể mô hình hóa mật mã thay thế như một phép hoán vị nếu có thể giải mã đầu vào và mã hoá đầu ra.*

### 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.*

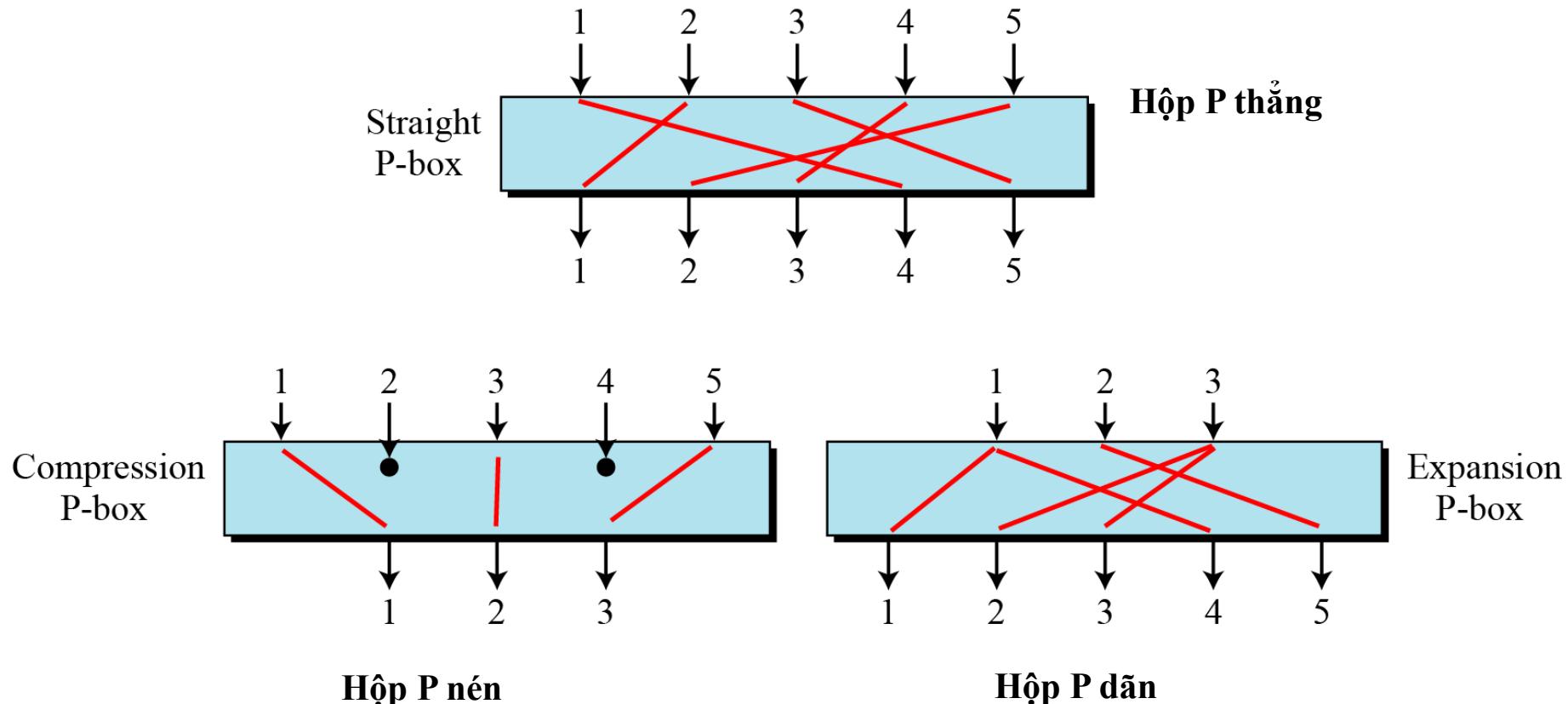
#### P-Boxes

**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

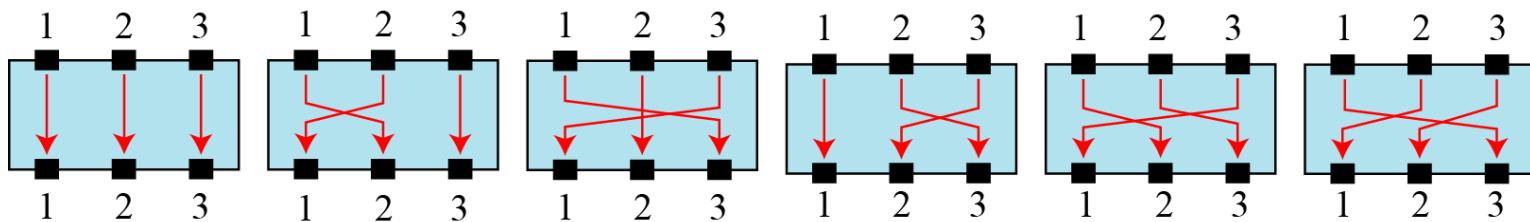


## 5.1.3 Tiếp...

### 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

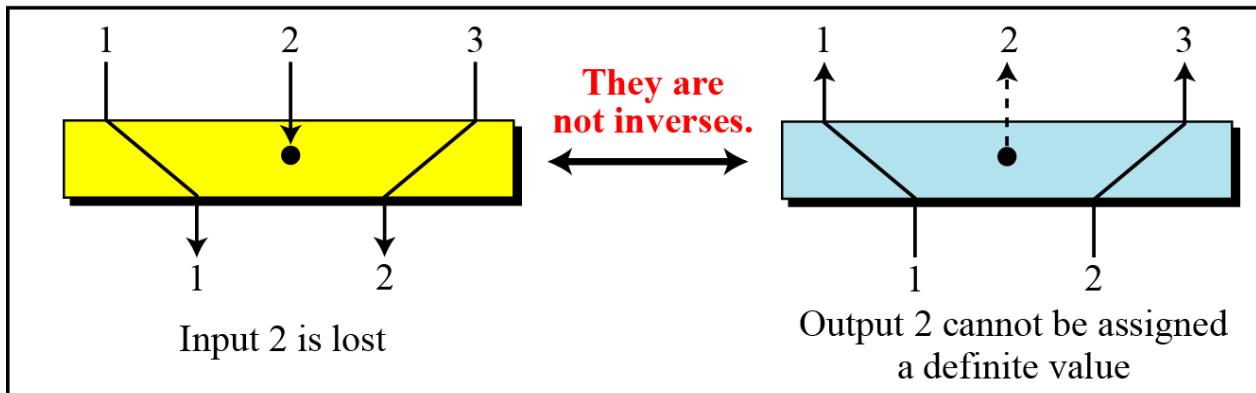
6	5	2	3	4	1
---	---	---	---	---	---

5. Inverted table  
Bảng đảo ngược

## 5.1.3 Tiếp...

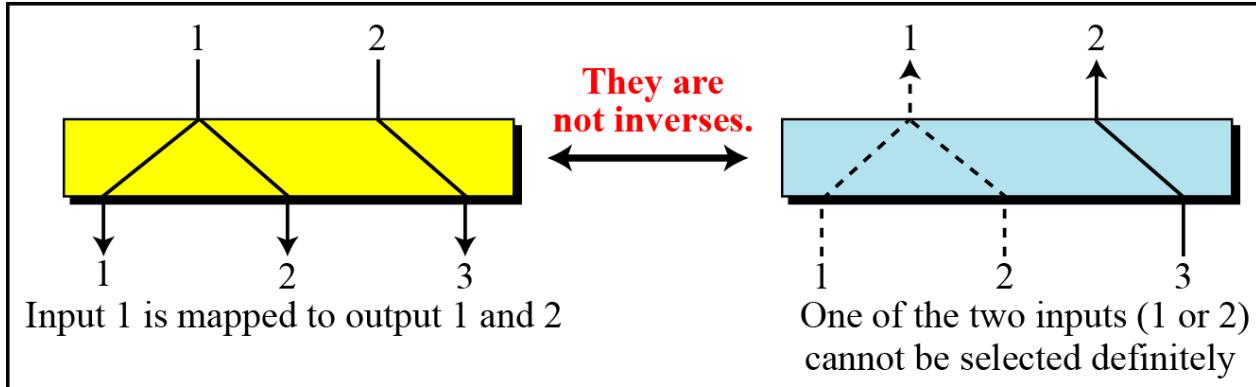
**Hình 5.7 Các hộp nén và mở rộng không có tính đảo ngược**  
**Compression and expansion P-boxes are non-invertible**

Compression P-box



They are  
not inverses.

Expansion P-box



They are  
not inverses.

### 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.*

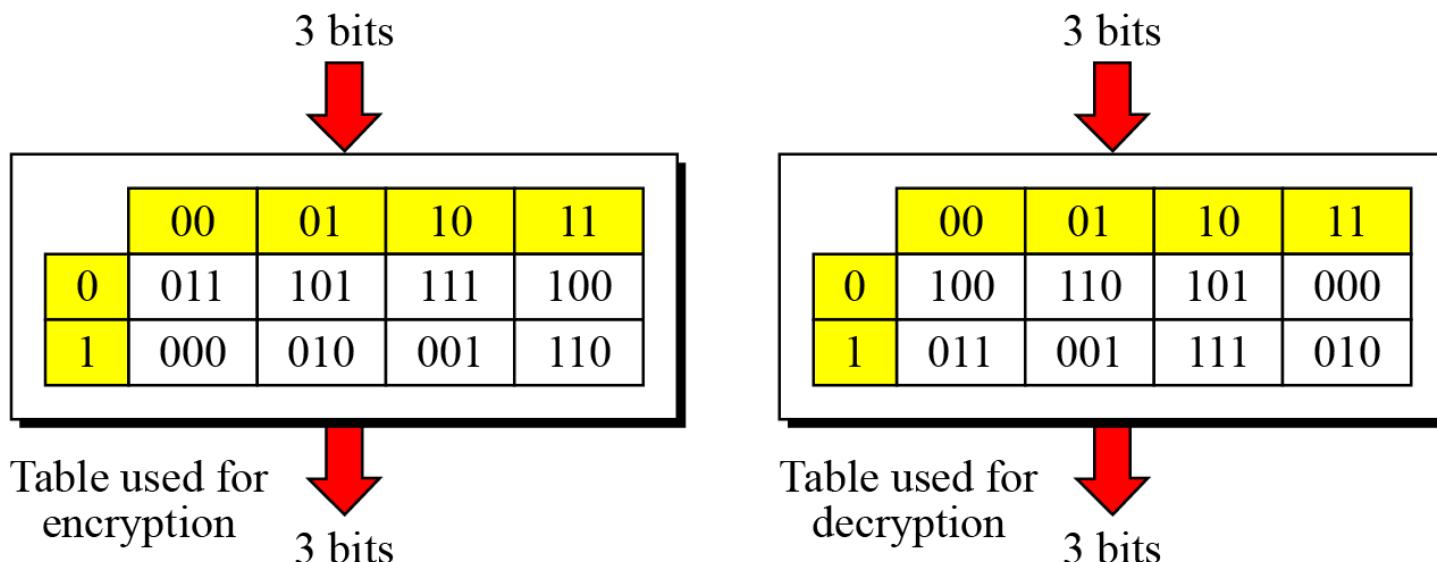
*An S-box may or may not be invertible. In an invertible S-box, the number of input bits should be the same as the number of output bits.*

## 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**

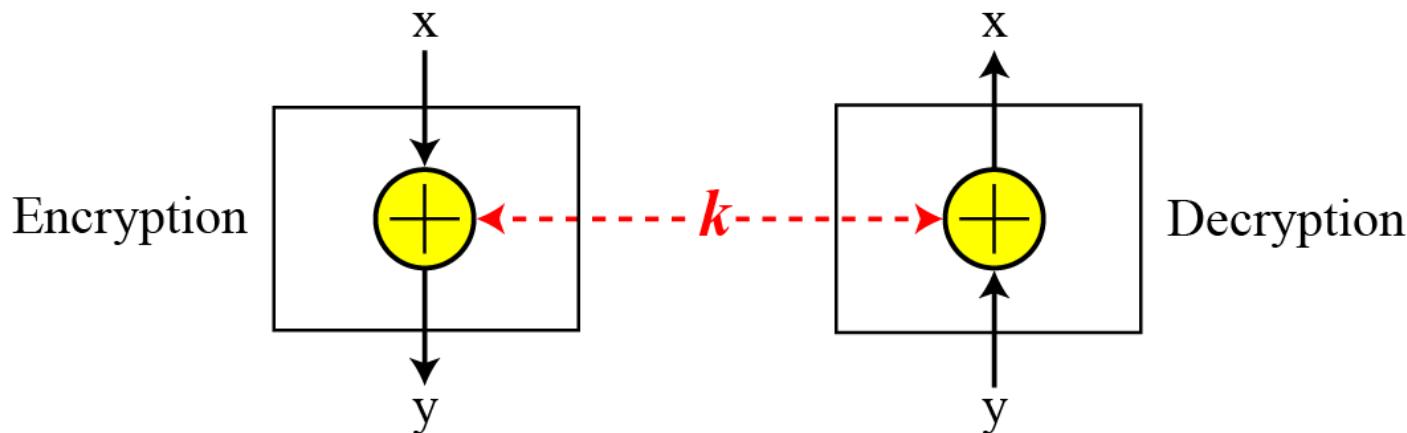


## 5.1.3 Tiếp...

### 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...)

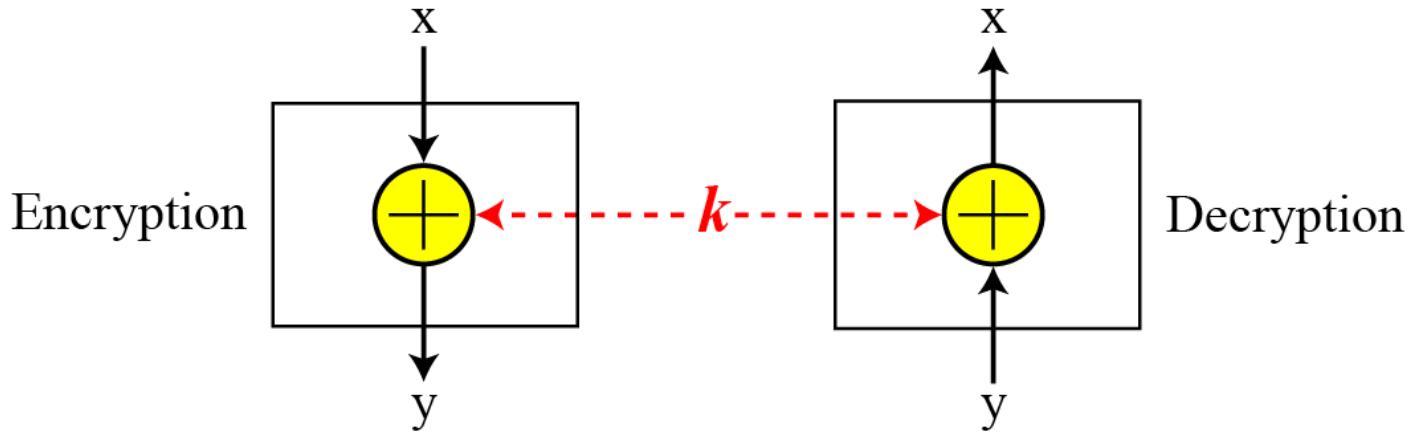
Nghịch đảo một thành phần trong mật mã có ý nghĩa nếu thành phần đại diện cho một phép toán đơn (một đầu vào và một đầu ra). Ví dụ, một hộp không khóa P-box hoặc hộp S-box có thể được thực hiện nghịch đảo bởi chúng có một đầu vào và một đầu ra. Một phép XOR là một toán tử nhị phân. **Phép nghịch đảo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



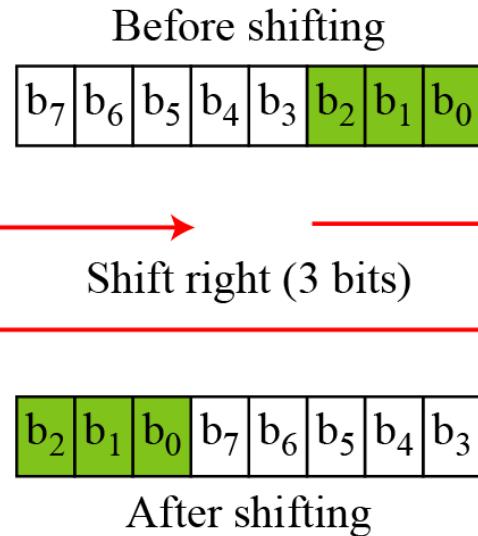
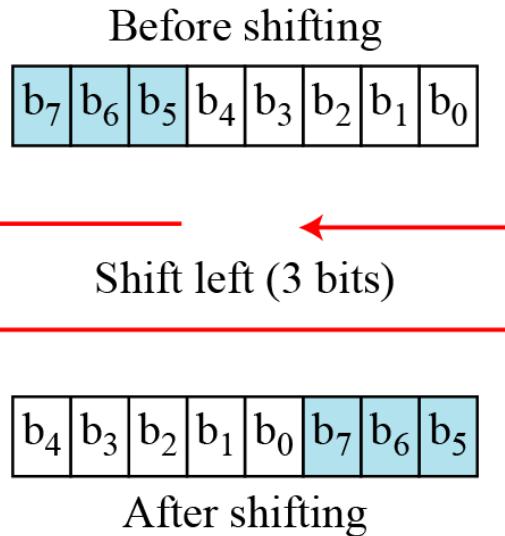
## 5.1.3 Tiếp...

### Dịch vòng tròn - Circular Shift

Một thành phần khác trong một số thuật toán mật mã 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

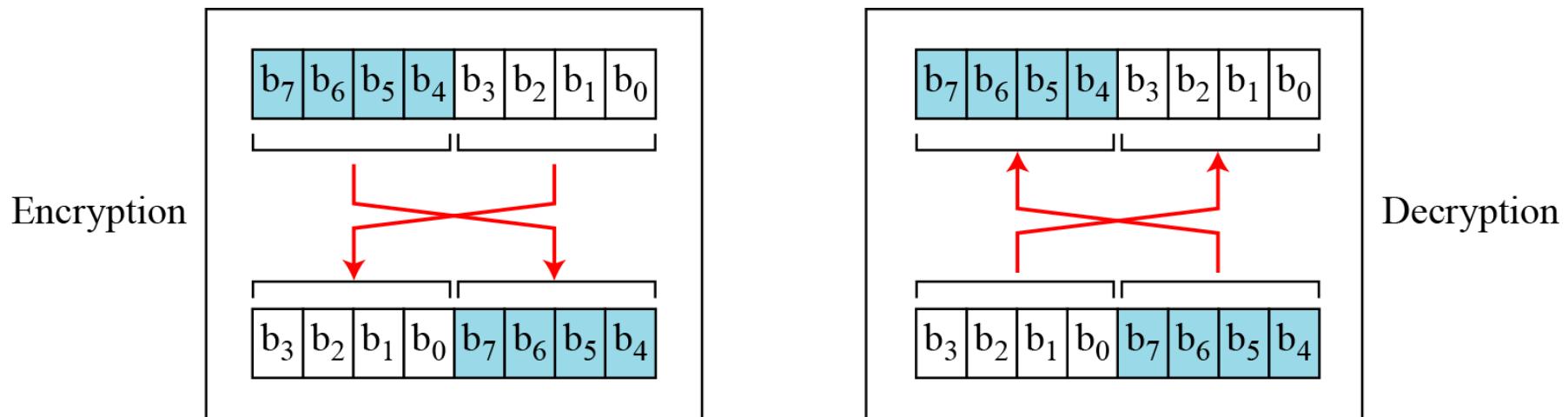
Toán tử dịch  
vòng thực  
hiện trong  
modulo  $n$ . Vì  
nếu  $k=0$  hoặc  
 $k=n$  tức là  
không có phép  
dịch. Nếu  $k > n$   
thì phép dịch  
là  $k \bmod n$  bit

## 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

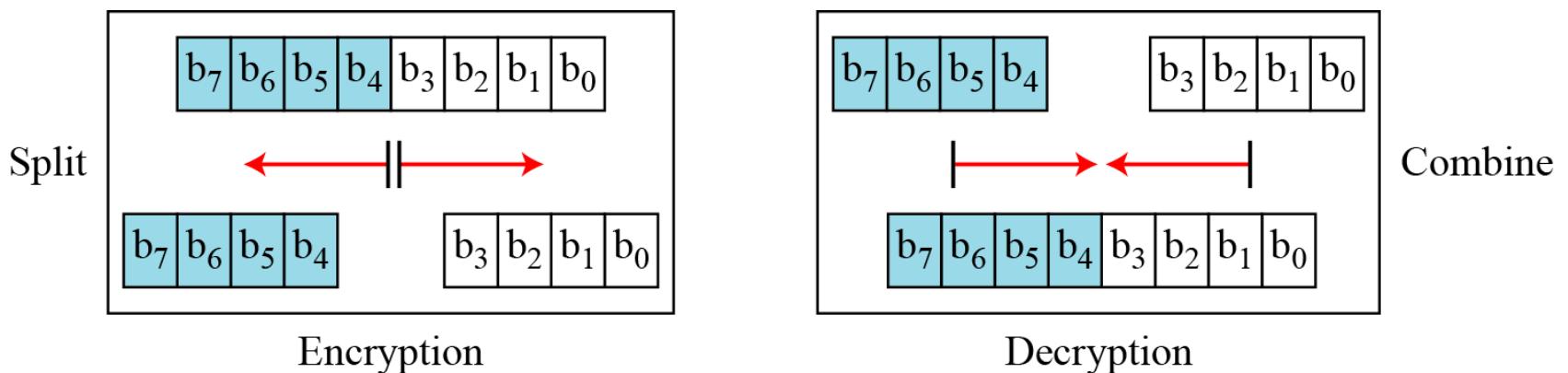


## 5.1.3 Tiếp...

### Chia và Kết hợp - Split and Combine

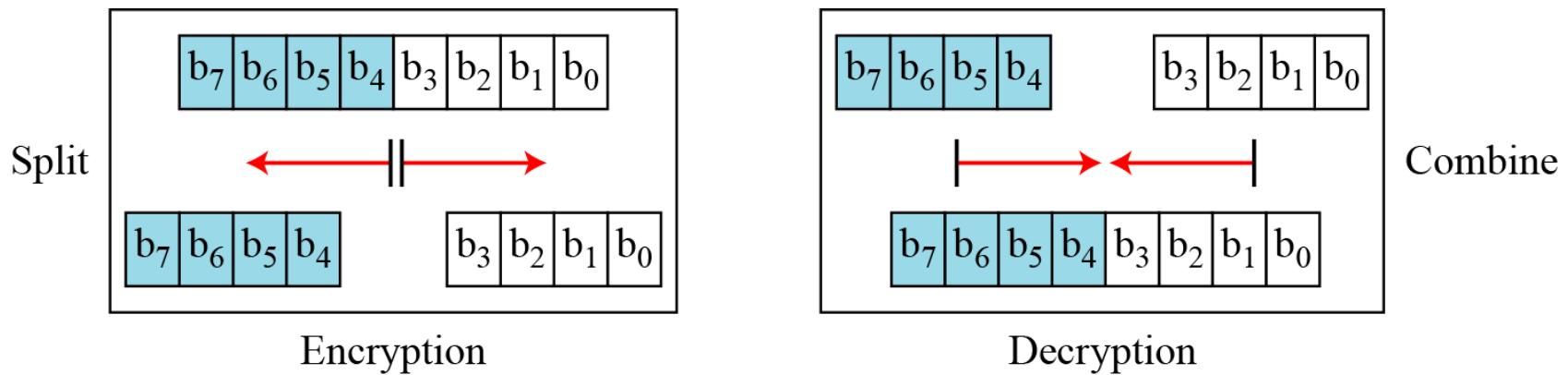
*Hai phép toán khác trong một số mật mã khối là chia và kết hợp. Nếu phép chia dùng cho mã hóa thì phép kết hợp dùng cho giải mã hóa và ngược lại.*

**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 khối 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.*

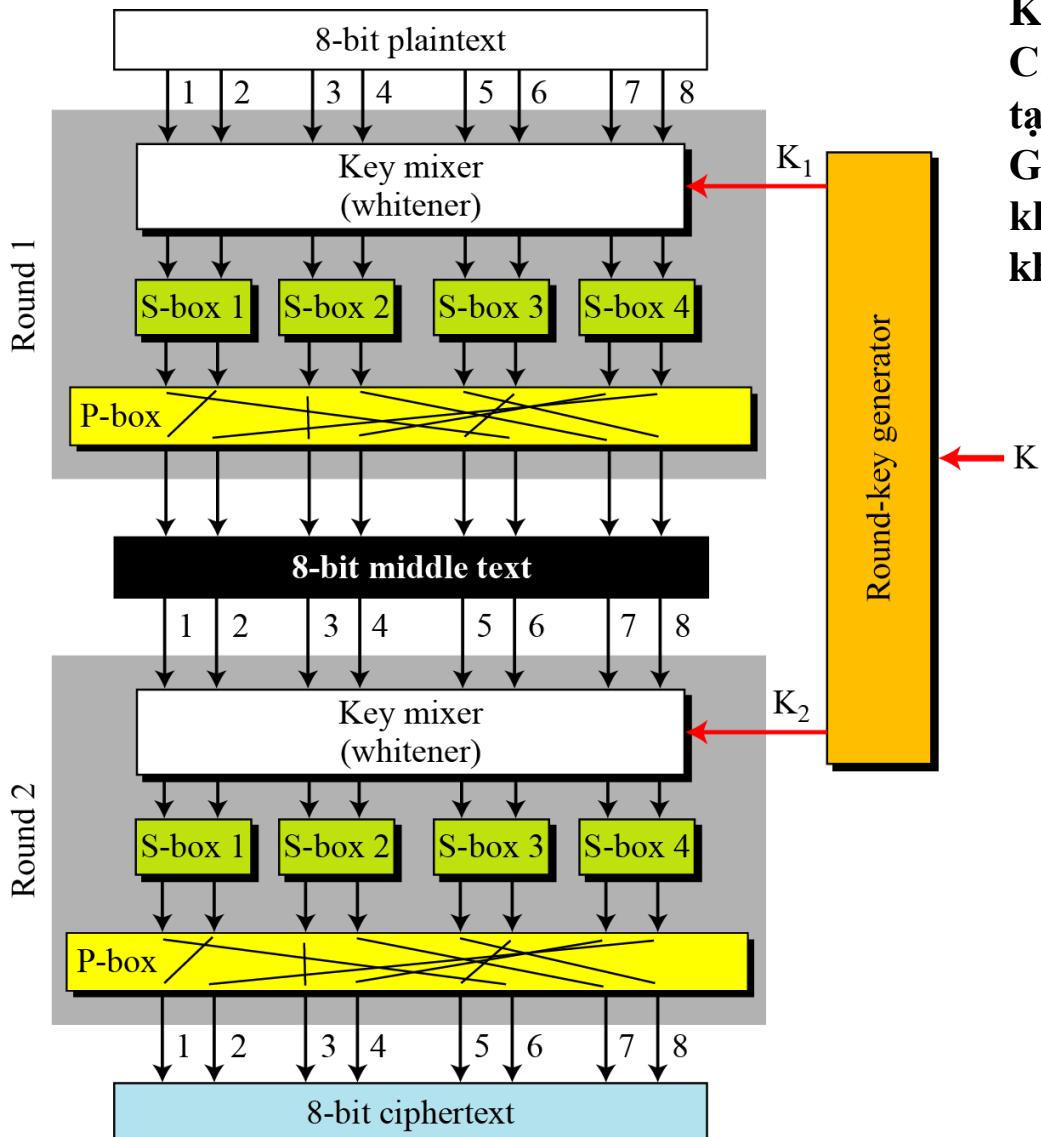
*(Một vòng được thực hiện bằng cách sử dụng kết hợp các phép biến đổi, các thành phần hộp S và hộp P)*

### 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:

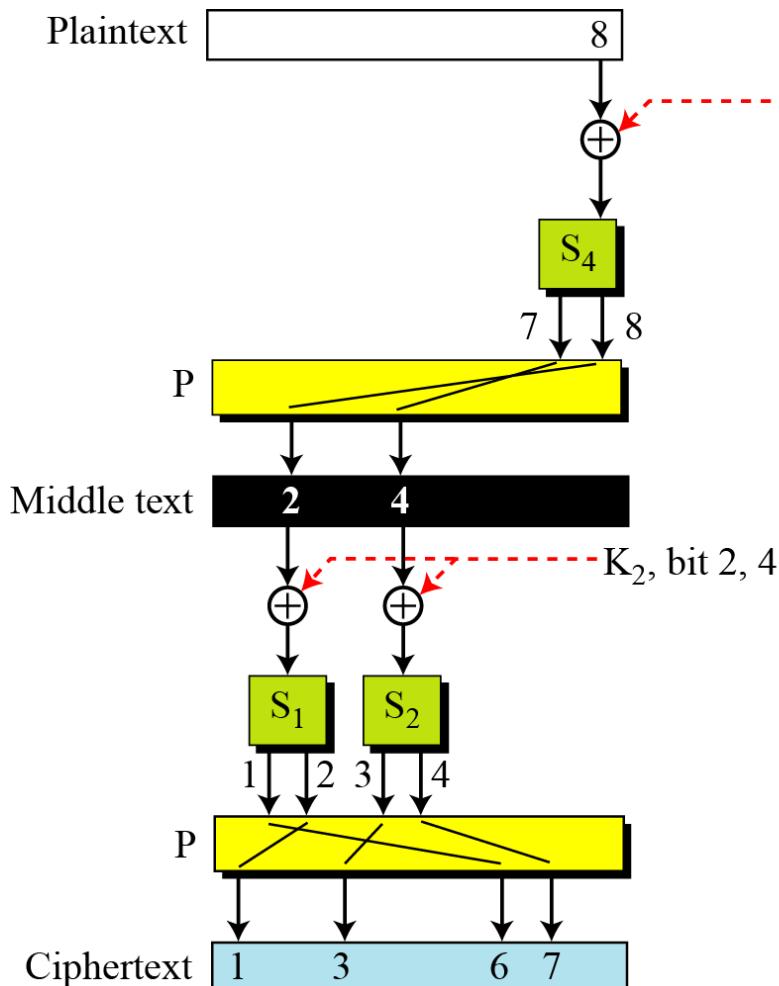
- + 8 bit dữ liệu vào được trộn với khóa k1 thông qua bộ trộn khóa (Key mixer) để thực hiện làm trắng (whiten) bản tin gốc (ẩn giấu các bit vào bởi khóa k1). Thông thường chúng ta sẽ thực hiện toán tử XOR với từ mã 8bit vào và 8 bit khóa k1.

- + 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

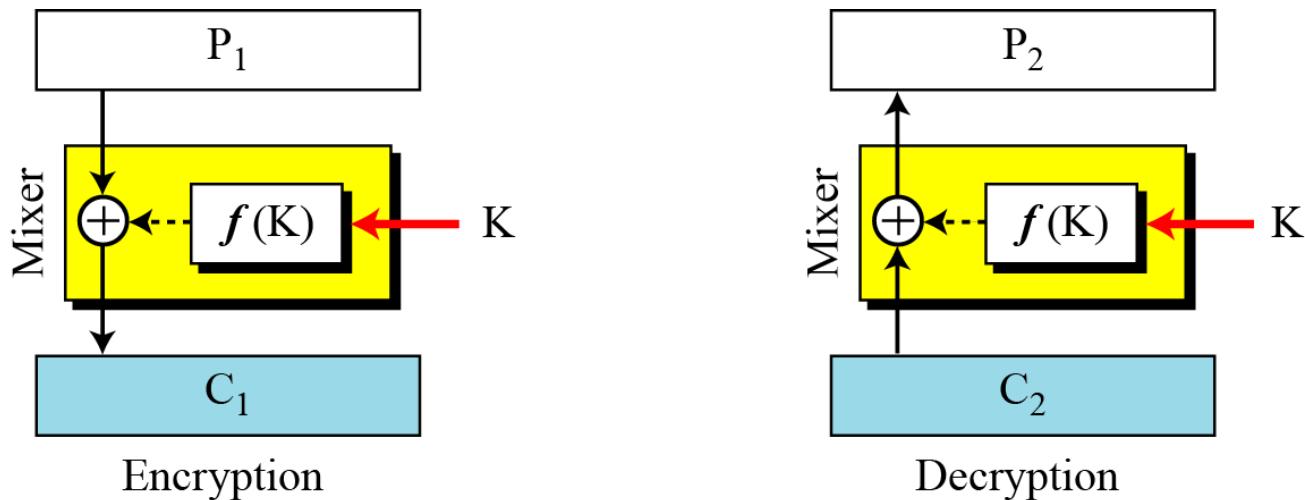
### *Mã Feistel (Feistel Ciphers)*

*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**

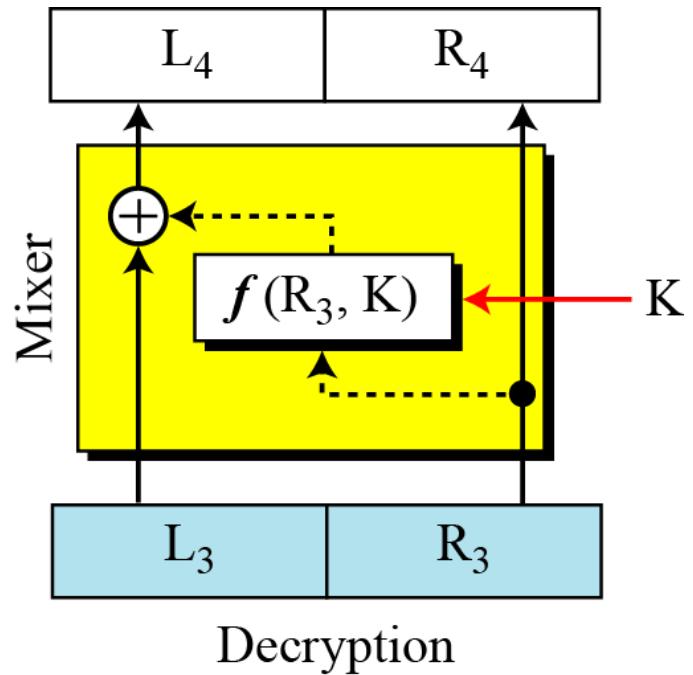
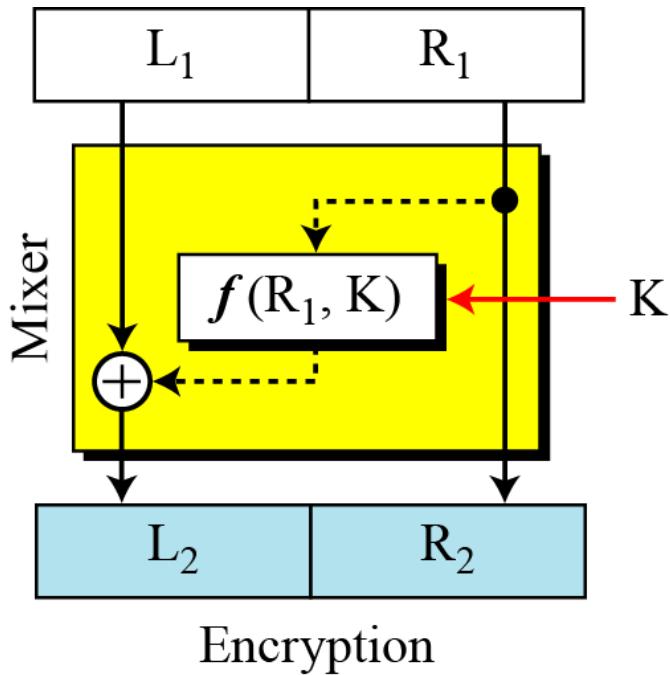
The function extracts the first and second bits to get 11 in binary or 3 in decimal. The result of squaring is 9, which is 1001 in binary.

**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



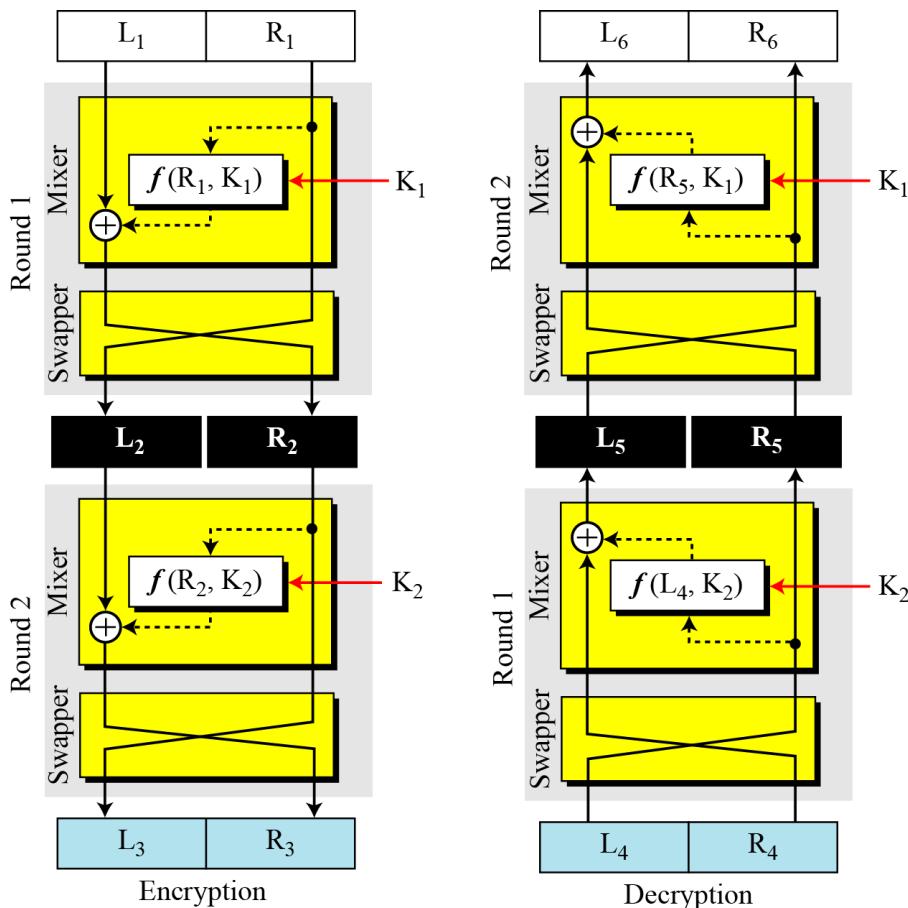
Nếu  $L_3 = L_2$  và  $R_3 = R_2$ .

$R_4 = R_3 = R_2 = R_1$  (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



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

$$R_5 = L_4 = L_3 = R_2$$

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

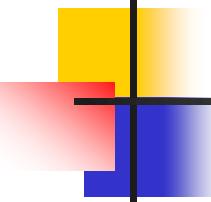
$$R_6 = L_5 = L_2 = R_1$$

## 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  $C_1 \oplus C_2 = P_1 \oplus P_2$ :

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

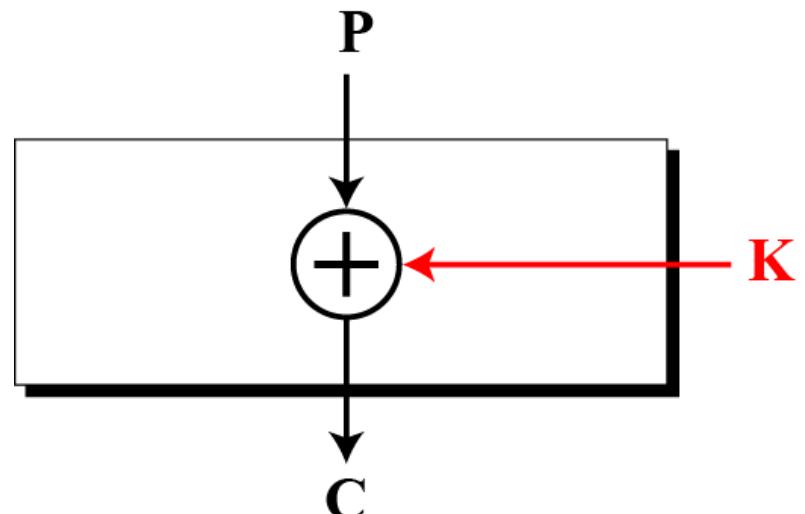


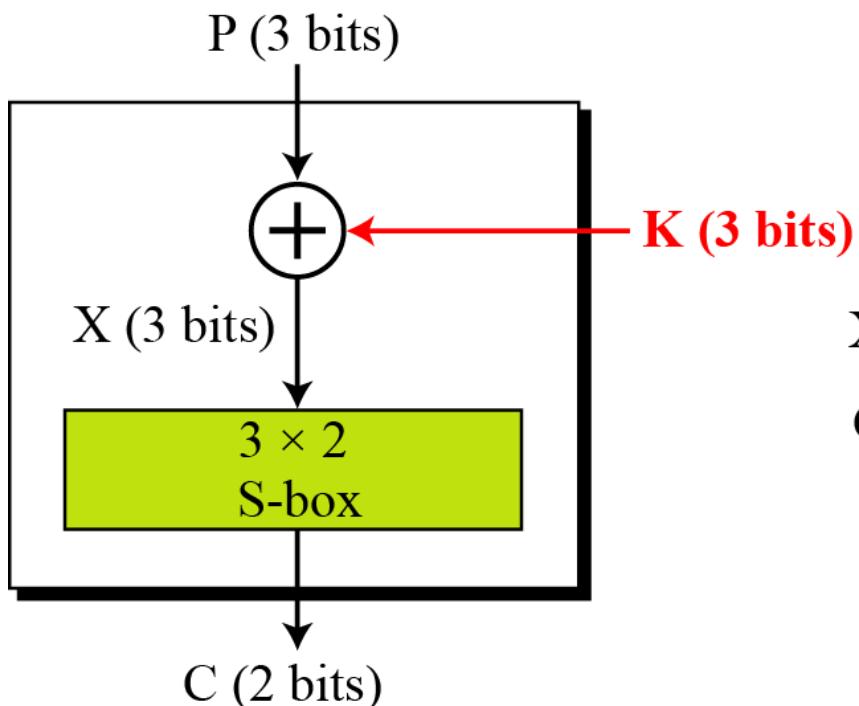
Figure 5.18 *Diagram for Example 5.13*

## 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.**

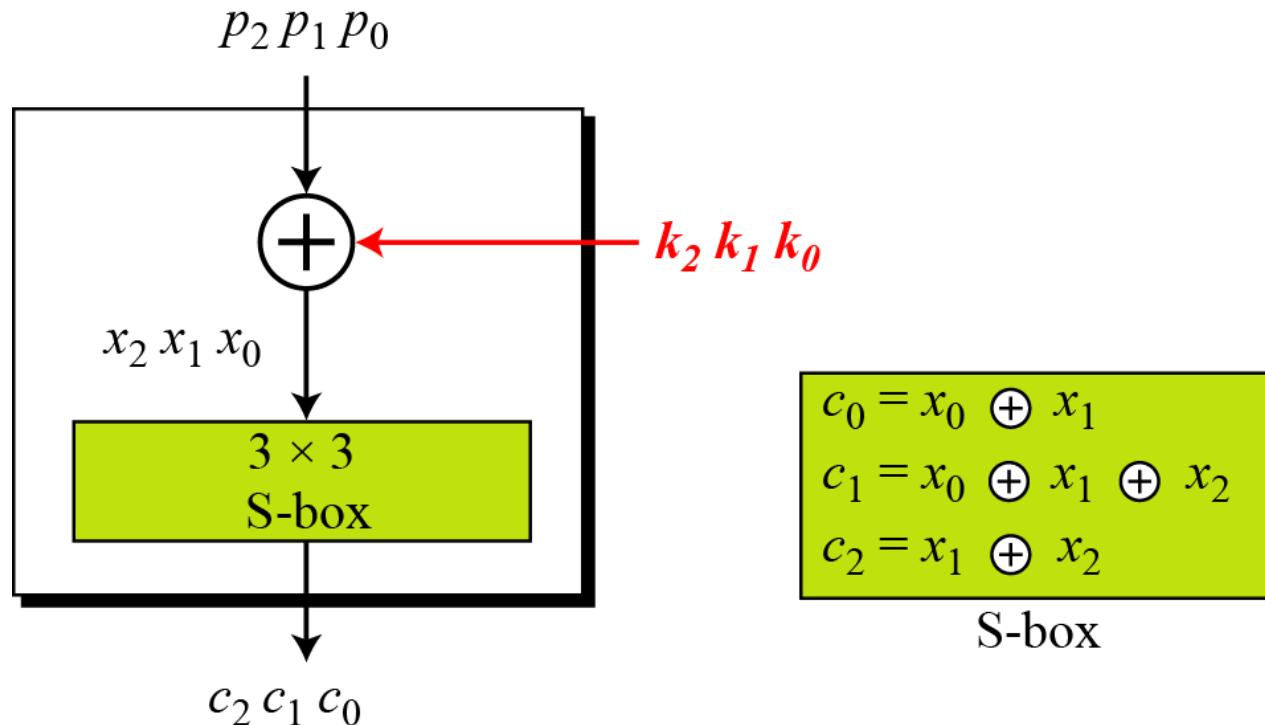
## **5.1.6 Continued**

### *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.*

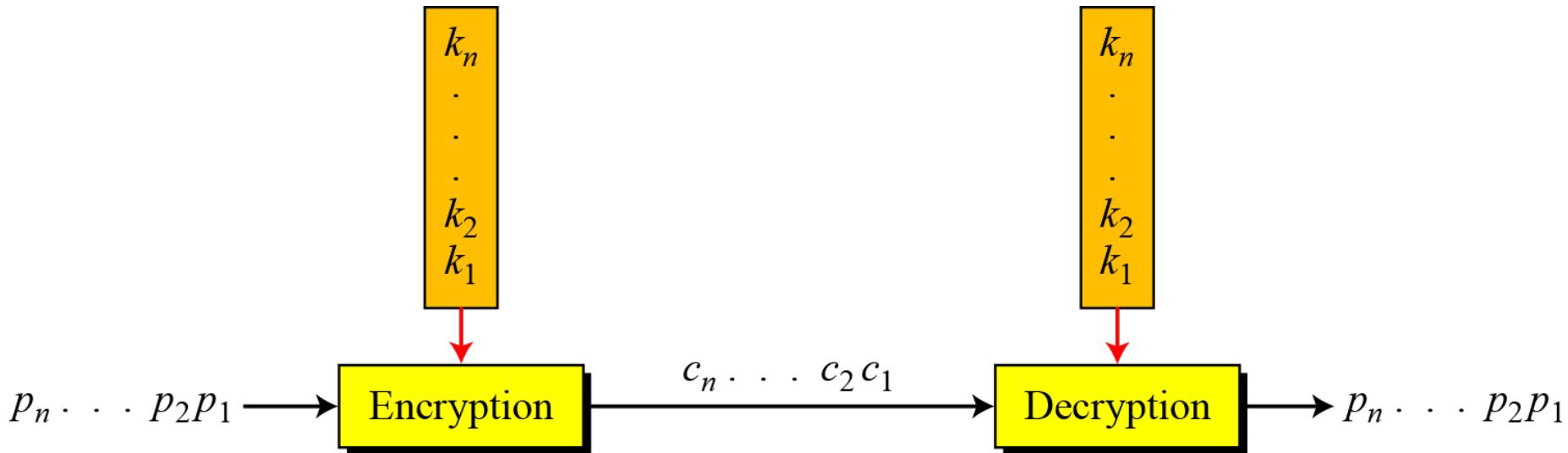
**Topics discussed in this section:**

**5.2.1 Synchronous Stream Ciphers**

**5.2.2 Nonsynchronous Stream Ciphers**

## 5.2 Continued

Figure 5.20 Stream cipher



**Note**

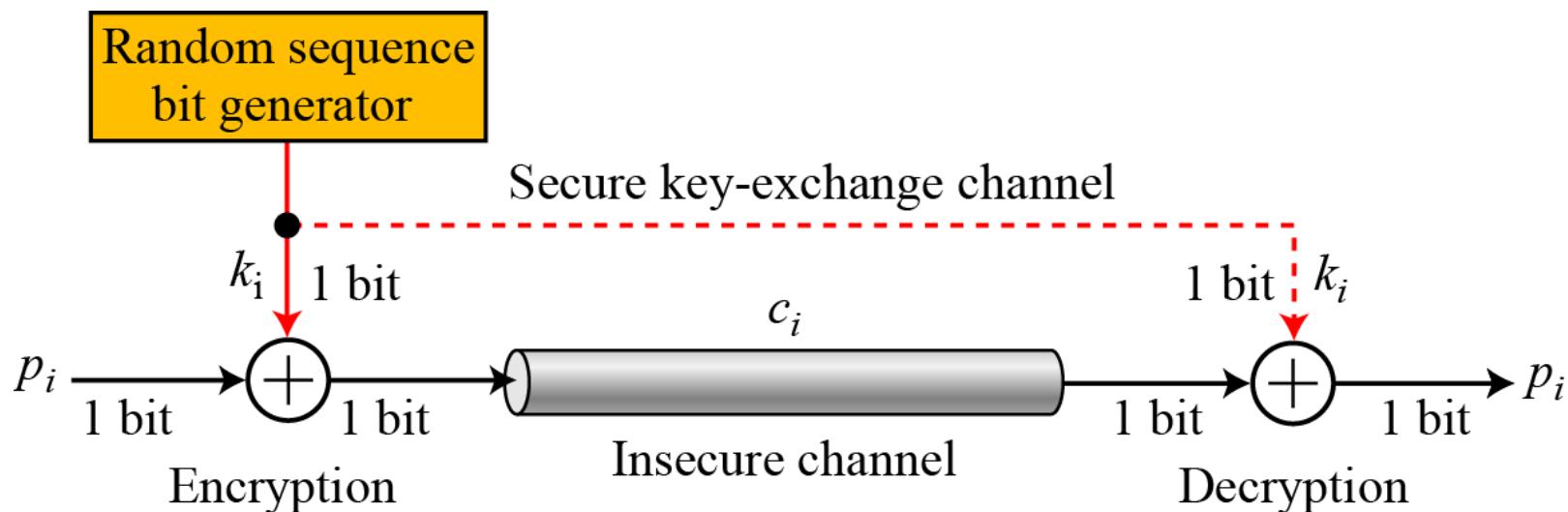
In a modern stream cipher, each  $r$ -bit word in the plaintext stream is enciphered using an  $r$ -bit word in the key stream to create the corresponding  $r$ -bit word in the ciphertext stream.

## 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.

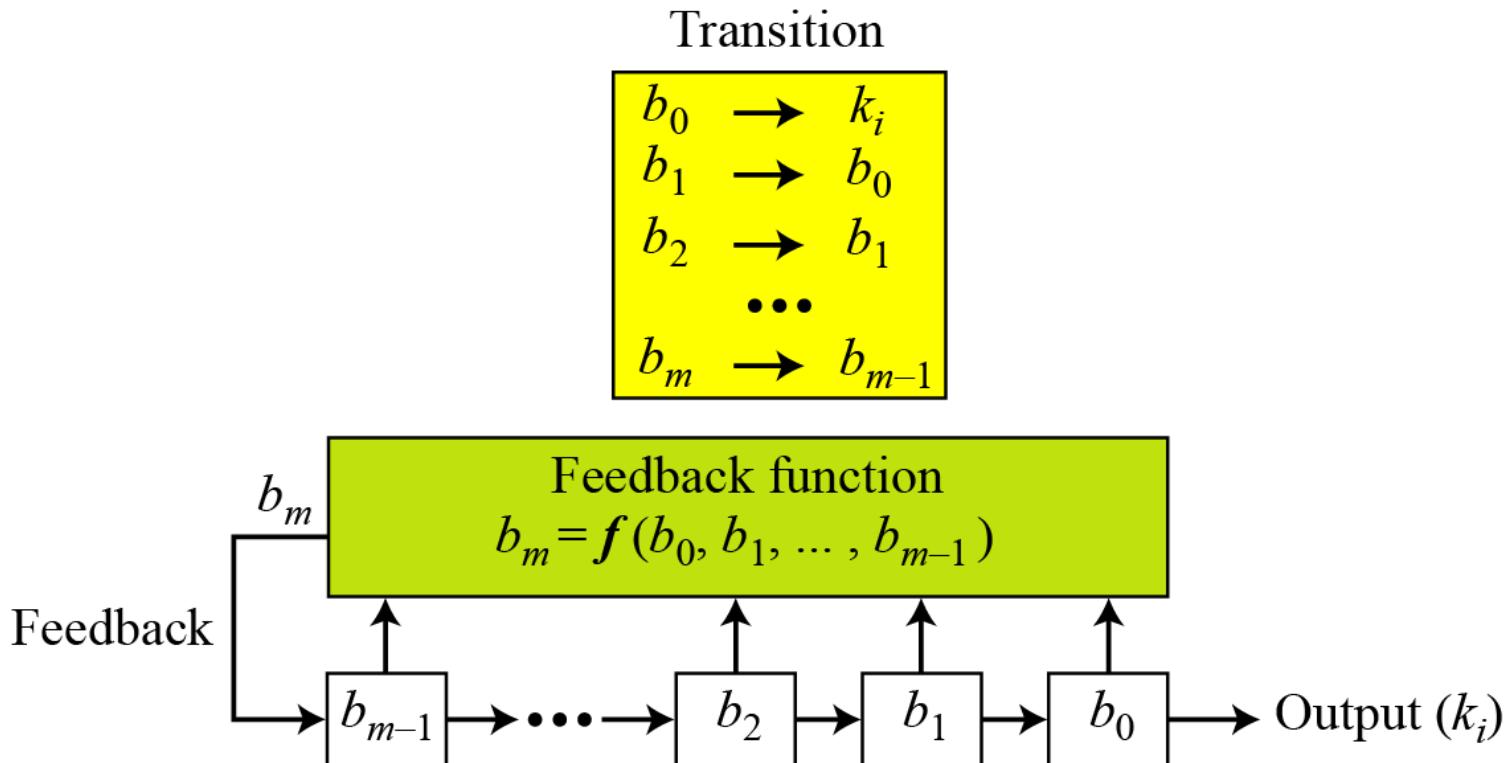
## 5.2.1 *Continued*

### 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

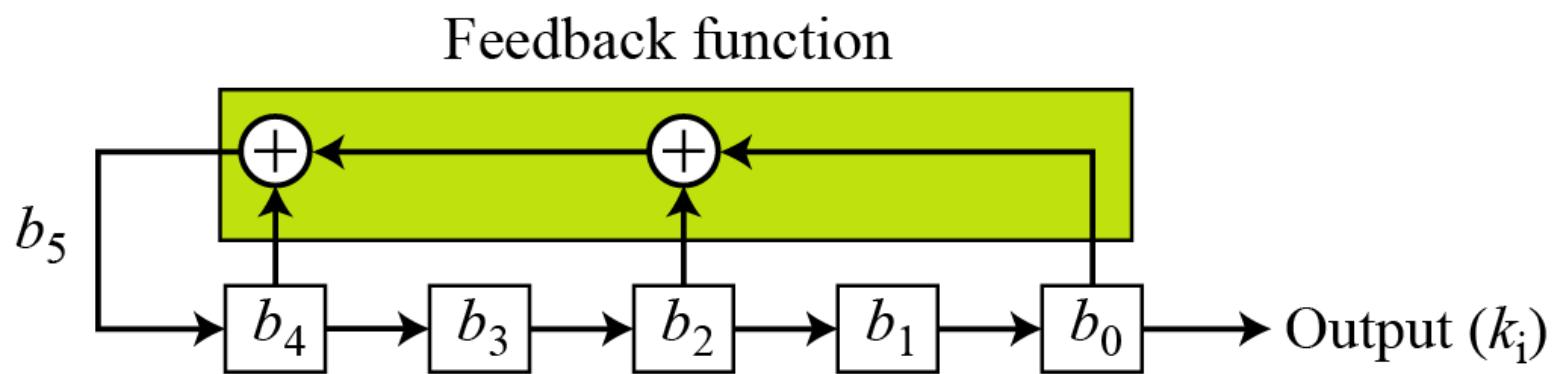
Create a linear feedback shift register with 5 cells in which  $b_5 = b_4 \oplus b_2 \oplus b_0$ .

### 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



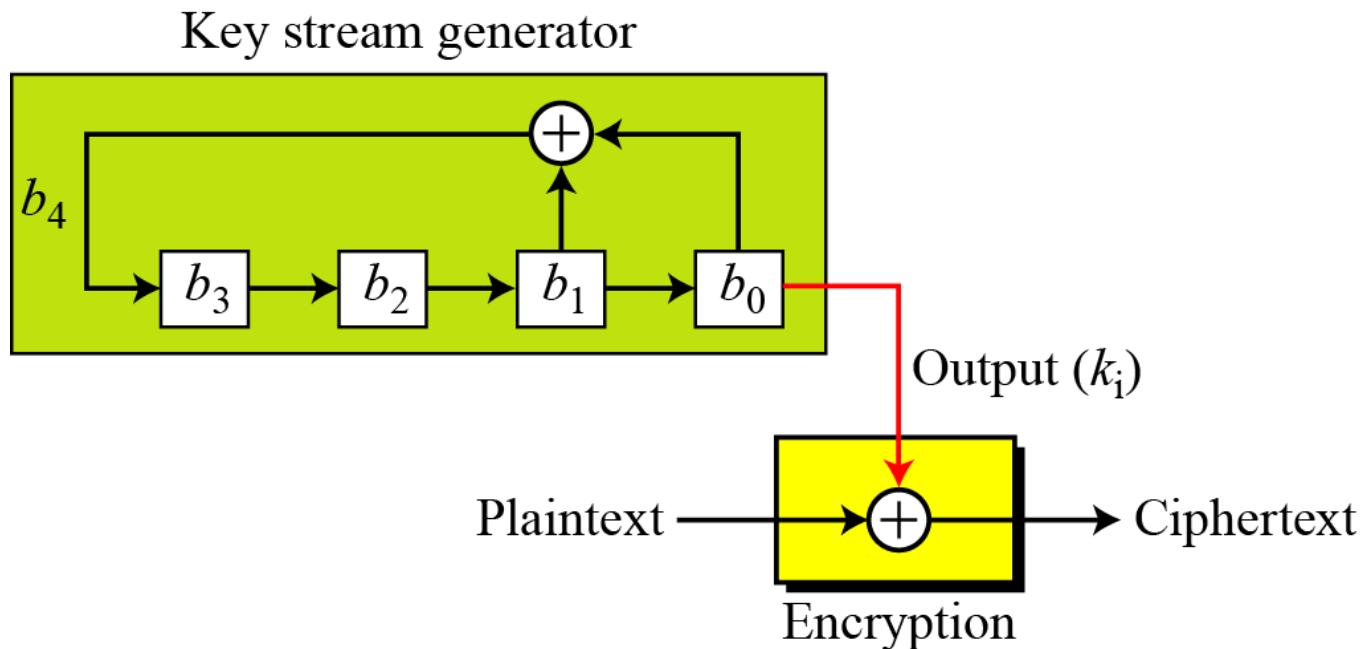
## 5.2.1 *Continued*

### 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.**