Introduction to Modern Symmetric-key Ciphers

Dr. B C Dhara

Department of Information Technology

Jadavpur University

Objectives

- Distinguish between traditional and modern symmetric-key ciphers
- Introduce components of block ciphers such as P-boxes and S-boxes
- Discuss product ciphers and distinguish between Feistel and non-Feistel ciphers

Objectives (contd…)

- Discuss two kinds of attacks particularly designed for modern block ciphers:
  - Differential and linear cryptanalysis
- Introduce stream ciphers and distinguish between synchronous and nonsynchronous stream ciphers
- Discuss linear and nonlinear feedback shift registers for implementing stream ciphers

Introduction

- We have studied so far character-oriented ciphers
- Bit-oriented ciphers are important as information is not just text:
  - It can be number, image, audio, video, etc.

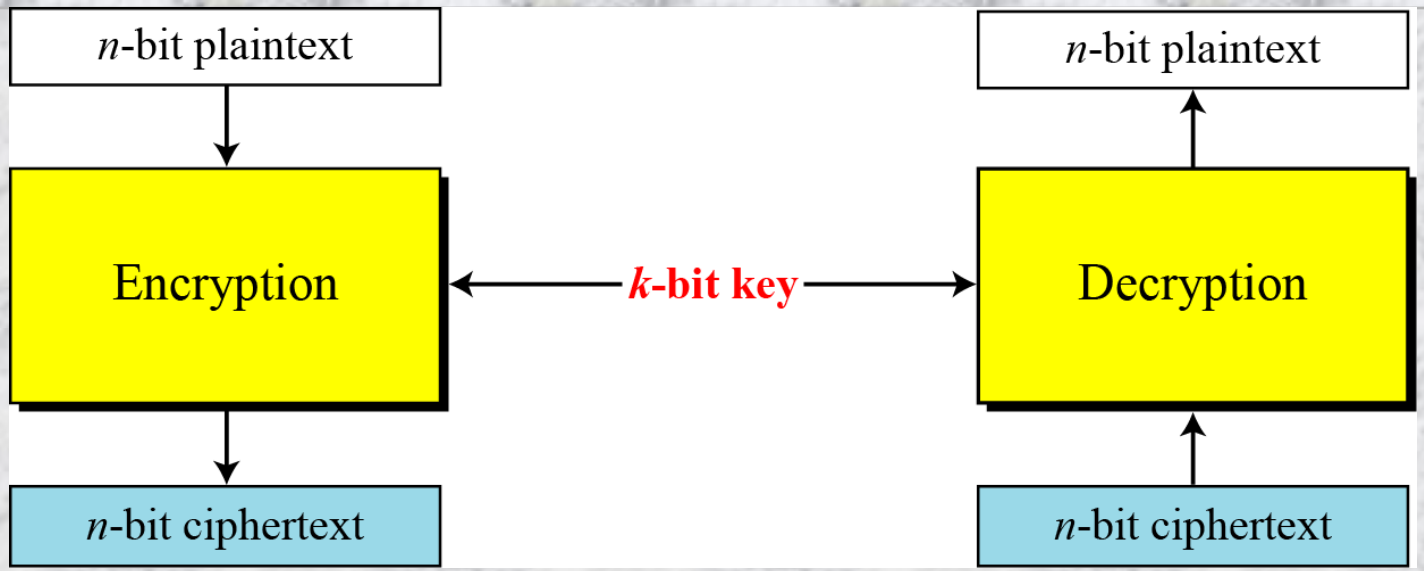
MODERN BLOCK CIPHERS

- *A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key*

Topics to be discussed

- Substitution or Transposition
- Block Ciphers as Permutation Groups
- Components of a Modern Block Cipher
- Product Ciphers
- Two Classes of Product Ciphers
- Attacks on Block Ciphers

A modern block cipher



If message length less than n, extra bits must be padded to make it n-bit block

If message length greater than n, divided into n –bit blocks with appropriate padding to the last block

Substitution or Transposition

- *A modern block cipher can be designed to act as a substitution cipher or a transposition cipher*
- Substitution ciphers change the number of 1's and 0's in ciphertext
- Transposition ciphers same of 1's present in ciphertext
  - which makes cipher vulnerable to exhaustive attack

To be resistant to exhaustive-search attack, a modern block cipher needs to be designed as a substitution cipher.

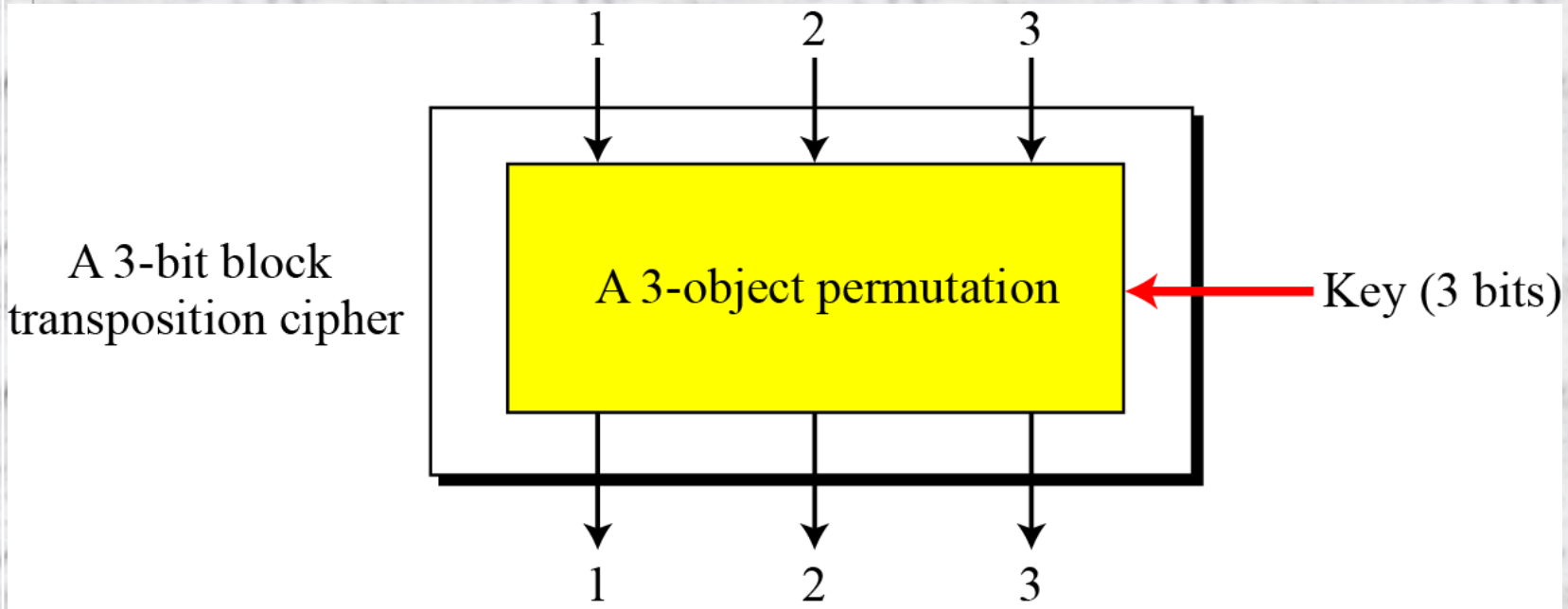
Block Ciphers as Permutation Groups

We consider full-size key ciphers

- *Full-Size Key Transposition Block Ciphers*
  - *For n-bits { i-th bit will mapped to j-th location}, no value change}, i.e., n-object permutation which is n!*
  - *One key represents one permutation, so n! keys and a key is represented by* $\lceil \log_2 n! \rceil$ *bits*

Full-Size Key Transposition Block Ciphers

Show the model and the set of permutation tables for a 3-bit block transposition cipher where the block size is 3 bits.

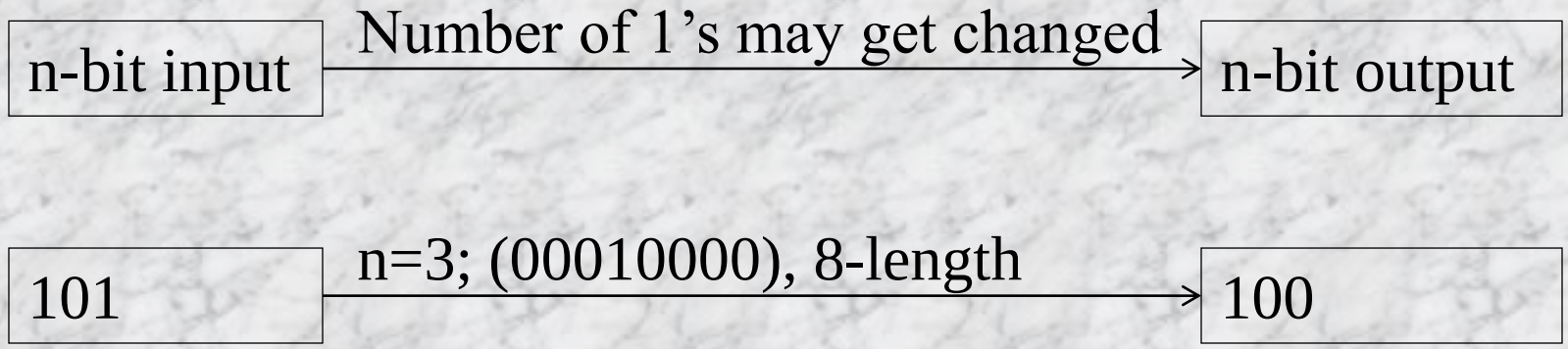


{[1 2 3], [1 3 2], [2 1 3], [2 3 1], [3 1 2], [3 2 1]}

The set of permutation tables with 3! = 6 elements

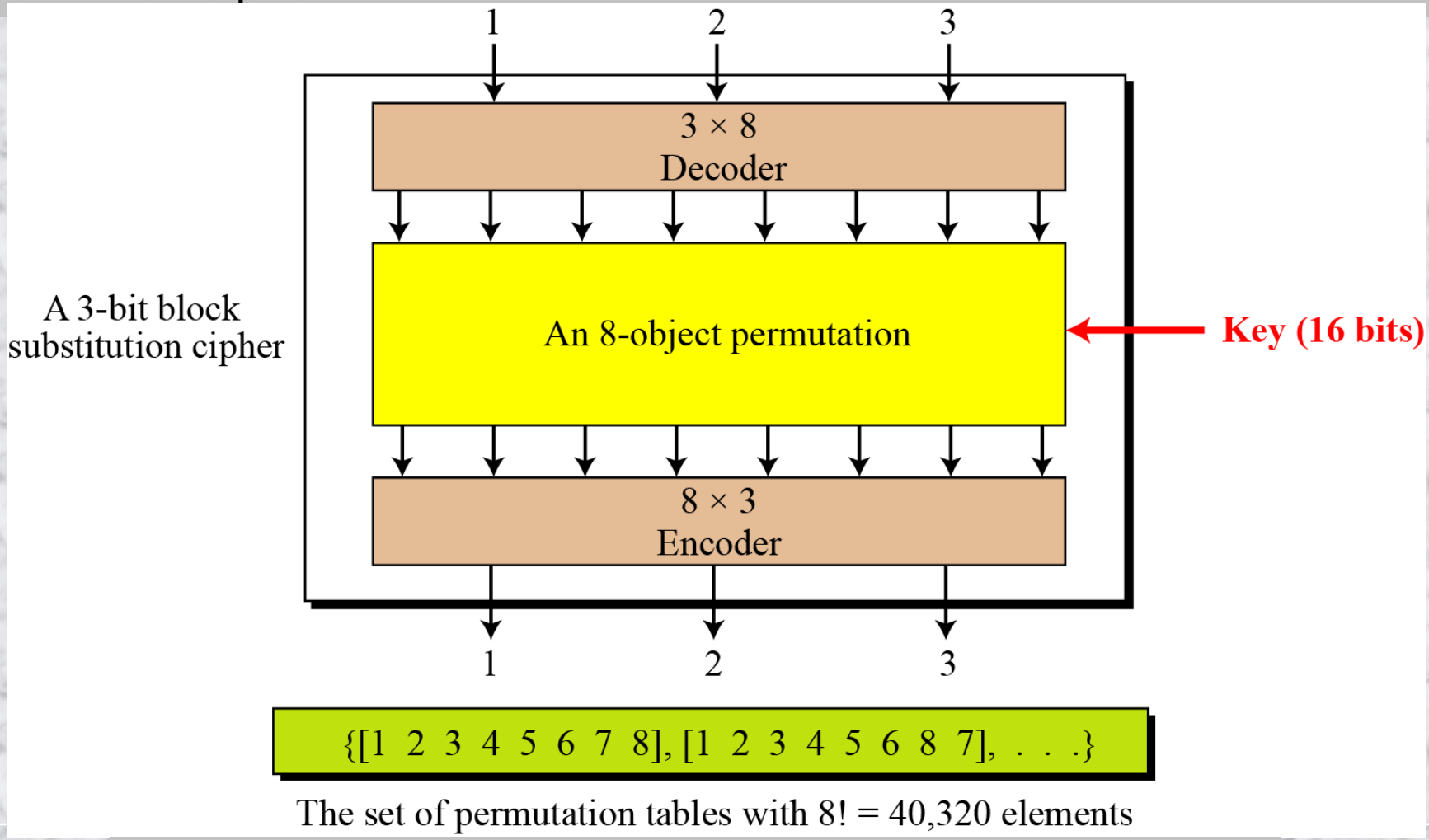
Full-Size Key Substitution Block Ciphers

- *A full-size key substitution cipher does not transpose bits; it substitutes bits.*
  - *We can model the substitution cipher as a permutation if we can decode the input and encode the output.*



Full-Size Key Substitution Block Ciphers (contd…)

Show the model and the set of permutation tables for a 3-bit block substitution cipher



The key is also much longer, $\lceil \log_2 40{,}320 \rceil = 16$ bits

Full-Size Key Block Ciphers (cont…)

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.

Full-Size Key Block Ciphers (cont…)

- Full-size key transposition or substitution cipher is a permutation
- Two or more cascaded permutations in equivalent to a single permutation
- Use of more than one stage of full-size key ciphers is useless
  - Because the effect is the same as having a single stage

Partial-size key ciphers

- In actual ciphers full-size keys cannot use since it is too large (mainly in case of substitution)
- For example,
  - DES uses 64-bit block cipher
  - If full-size key used, then key would be of $\log_2((2^{64})!) = 2^{70}$ bits
  - Key size of DES is 56 bits, a very small fraction of the full-size key
  - DES uses only 2^{56} mappings out of $2^{2^{70}}$ possible mappings

Keyless cipher

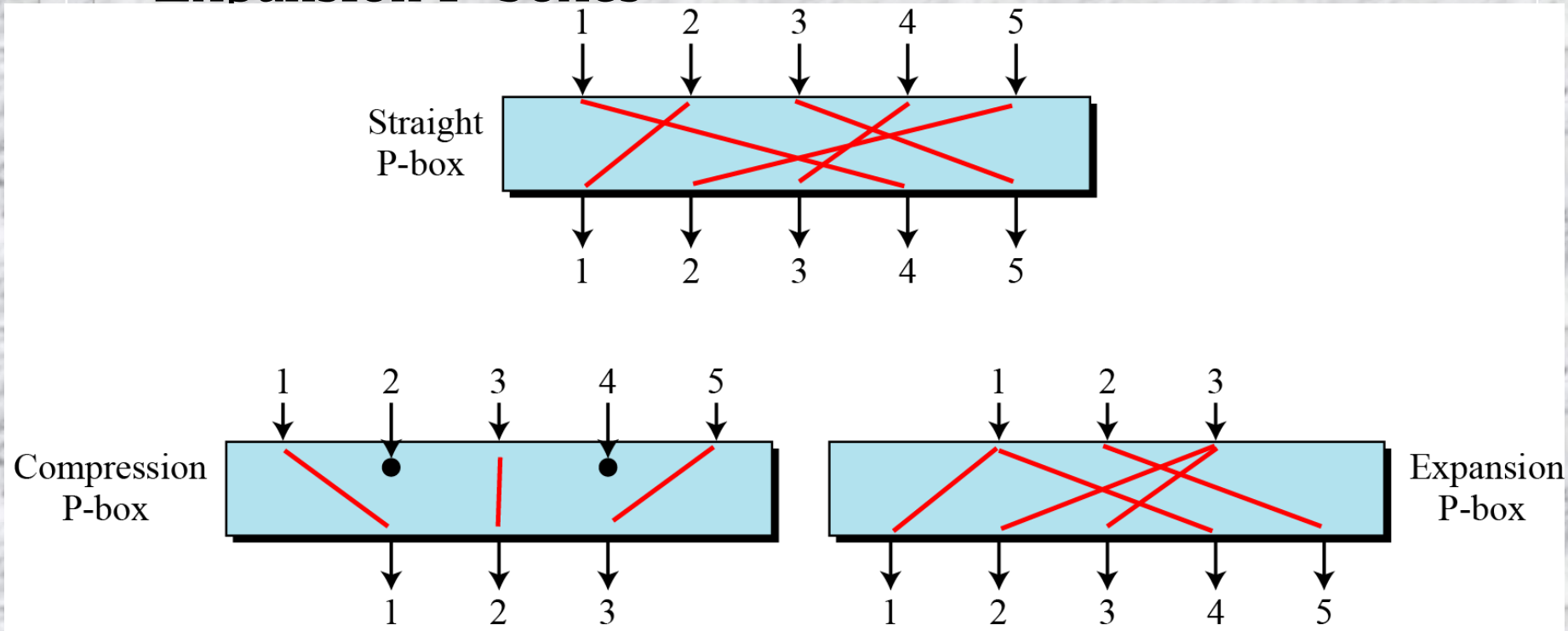
- Independent use of keyless cipher for encryption purpose is useless
- Keyless cipher can be used a components of keyed cipher
- Keyless transposition cipher: P-box
- Keyless substitution cipher: S-box

Components of a Modern Block Cipher

- A modern block cipher is made of a combination of transposition units (P-boxes), substitution units (S-boxes) and some other units

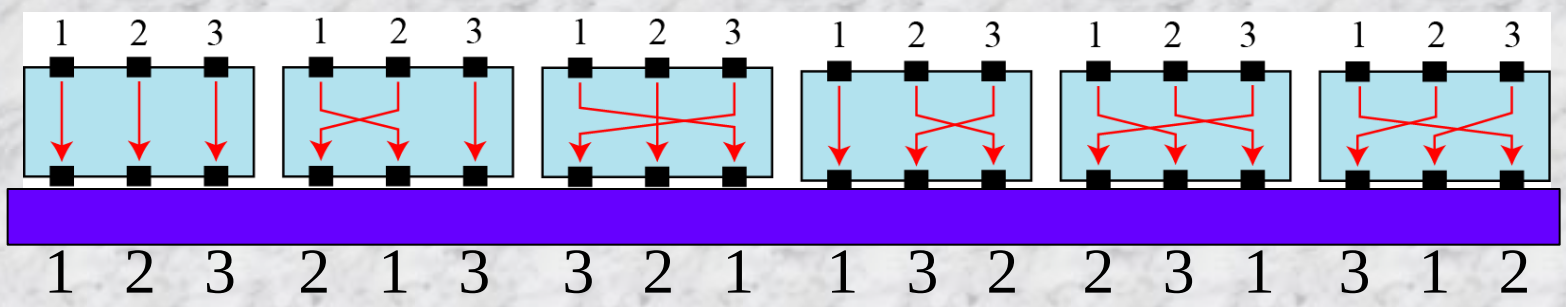
P-Boxes

- We find three types of P-boxes (permutation boxes) in modern block ciphers
  - Straight P-boxes
  - Compression P-boxes
  - Expansion P-boxes



Straight P-Boxes

- With n –inputs and n-outputs: a permutation
  - Key less
- Permutation can be represented by the index of the destination (in tabular form)



Straight P-Boxes (contd…)

- n=64

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

Compression P-Boxes

- *A compression P-box is a P-box with n inputs and m outputs where m < n*
  - *Some inputs are blocked and do not reach to output*

Example of a 32 × 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

Some inputs are blocked: 7, 8, 9, 15, 16, 23, 24 and 25

Expansion P-Boxes

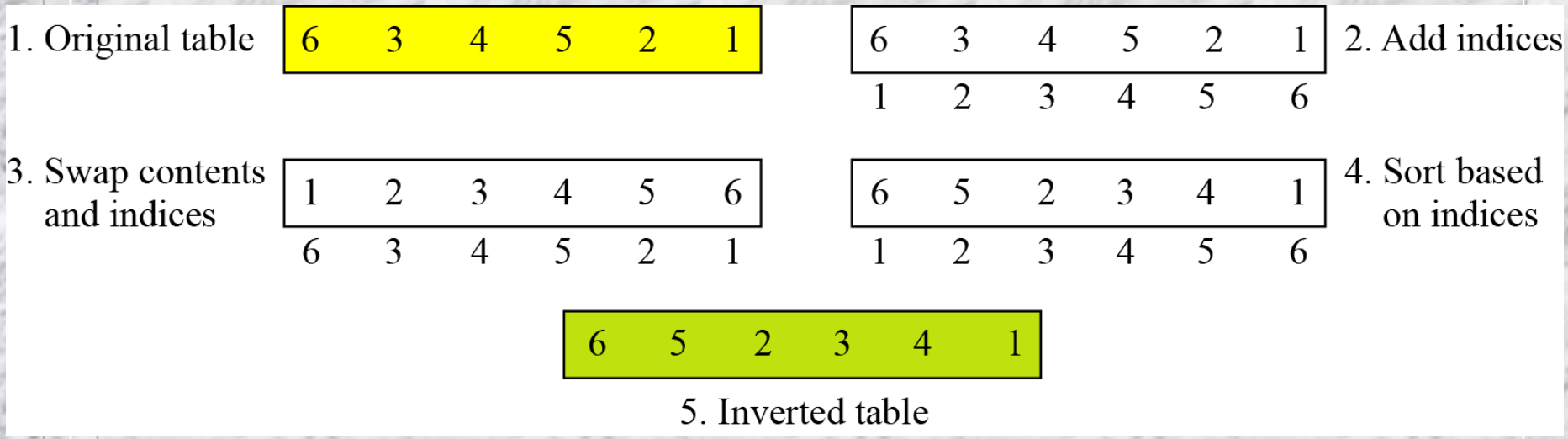
- *An expansion P-box is a P-box with n inputs and m outputs where m > n*
- *Some inputs are connected to more than one output lines*

Example of a 12 × 16 permutation table

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

Invertibility of P-Boxes

- Straight P-Box is invertible
  - Invert a permutation table represented as a one-dimensional table

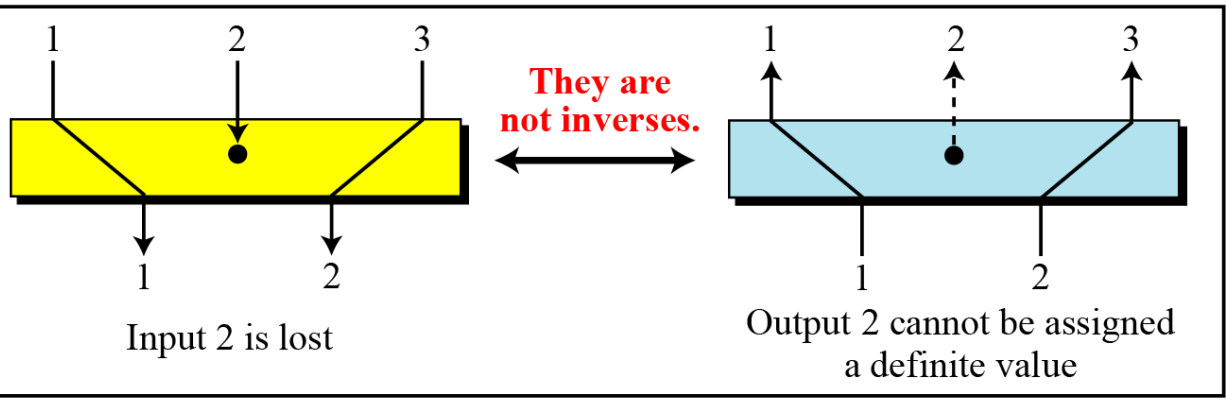


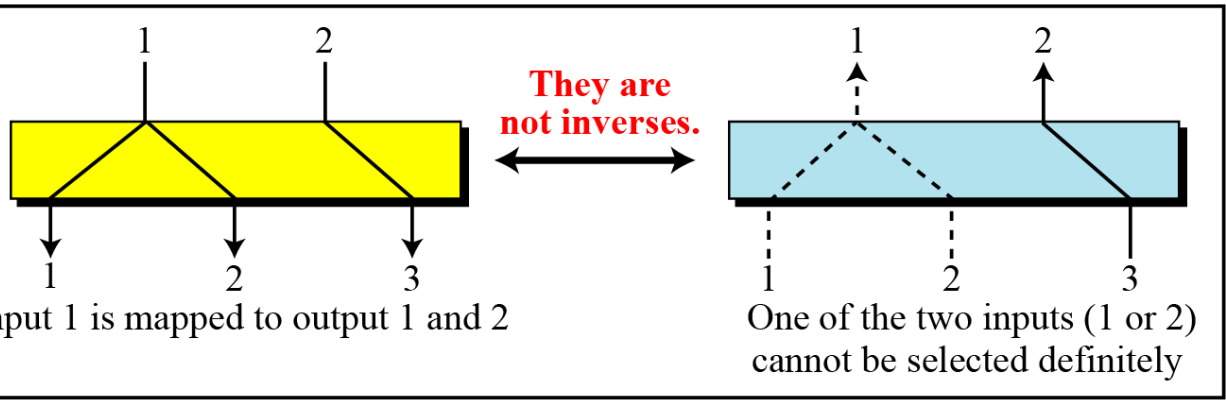
Invertibility of P-Boxes (contd…)

- In compression P-box
  - one input can be dropped during encryption
  - The decryption algorithm does not have any clue how to replace the dropped bit
- In expansion P-box
  - An input may be mapped to more than one output during encryption
  - The decryption algorithm does not have any clue which input mapped to more than one output

Invertibility of P-Boxes (contd…)

Compression P-box





Expansion P-box

A straight P-box is invertible, but compression and expansion P-boxes are not.

S-Boxes

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

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

- *S-box may be ked or keyless*
  - *Modern block cipher normally use keyless S-box*
  - *Mapping from input to output is predetermined*

Linear/non-linear S-boxes

Relationship between input and output of S-box is defined as

$$y_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n)$$
$$y_m = f_m(x_1, x_2, \ldots, x_n)$$

Linear S-boxes (contd…)

$$y_1 = a_{11}x_1 \oplus a_{12}x_2 \oplus a_{13}x_3 \oplus \ldots \oplus a_{1n}x_n$$

$$y_2 = a_{21}x_1 \oplus a_{22}x_2 \oplus a_{23}x_3 \oplus \ldots \oplus a_{2n}x_n$$

$$y_m = a_{m1}x_1 \oplus a_{m2}x_2 \oplus a_{m3}x_3 \oplus \ldots \oplus a_{mn}x_n$$

Linear S-box

In an S-box with three inputs and two outputs, we have

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

The S-box is linear because $a_{1,1} = a_{1,2} = a_{1,3} = a_{2,1} = 1$ and $a_{2,2} = a_{2,3} = 0$. The relationship can be represented by matrices, as shown below:

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

Nonlinear S-box

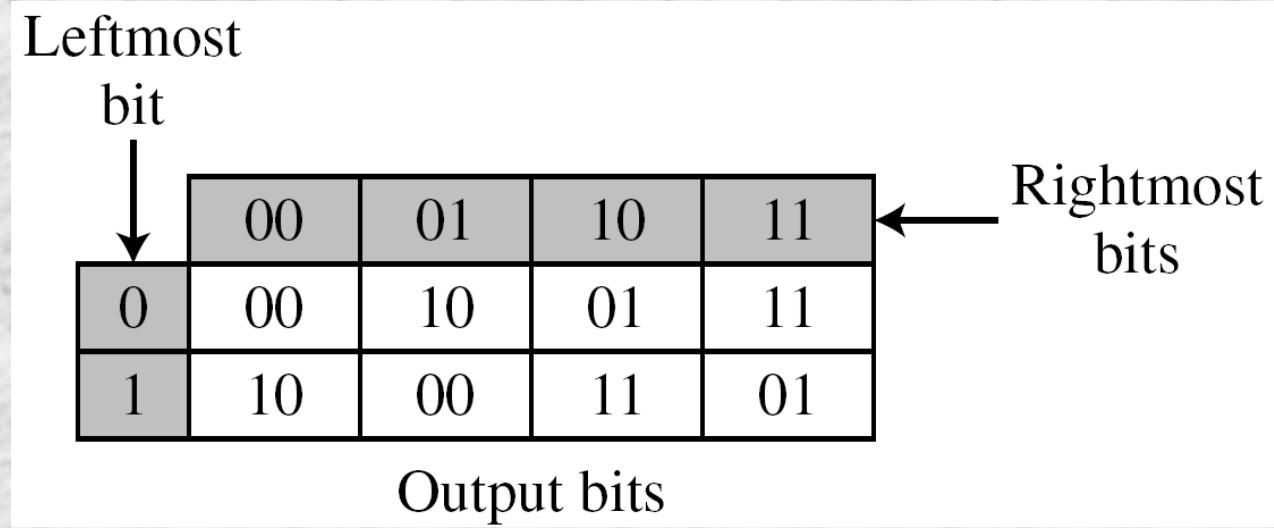
In an S-box with three inputs and two outputs, we have

$$y_1 = (x_1)^3 + x_2 \qquad y_2 = (x_1)^2 + x_1x_2 + x_3$$

where multiplication and addition is in GF(2). The S-box is nonlinear because there is no linear relationship between the inputs and the outputs.

S-box (contd…)

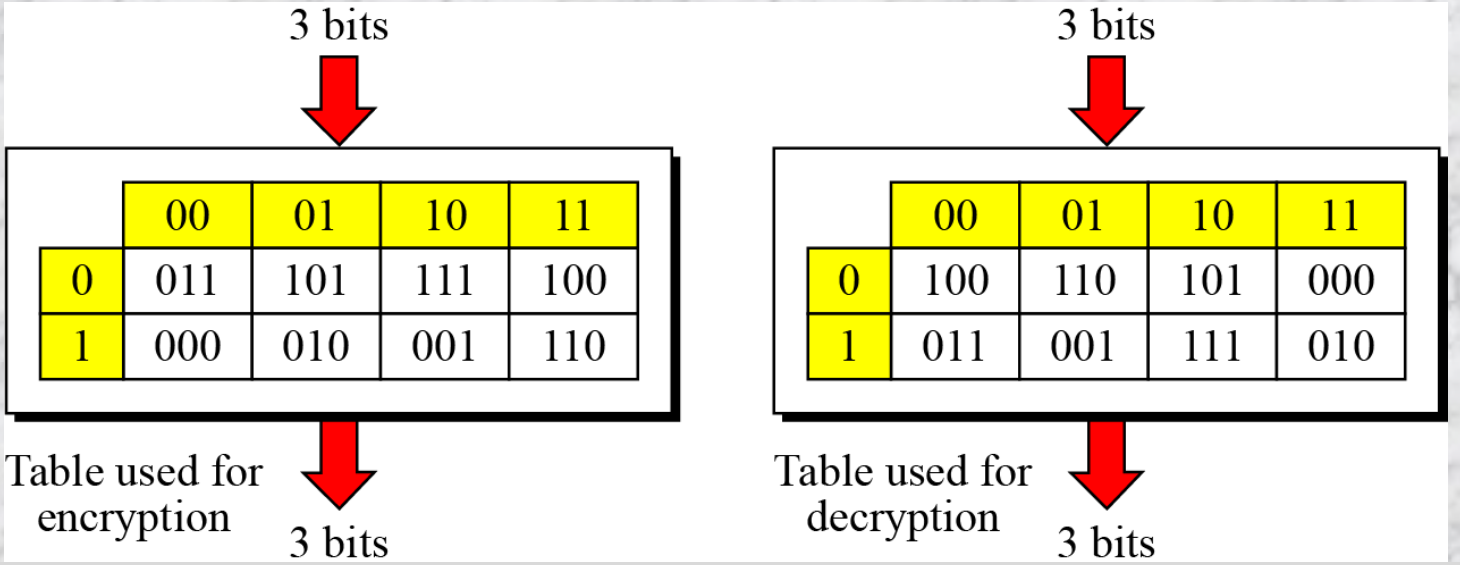
- Input/output relationship of an S-box can be represented by a table as follows (for a 3x2):



Based on the table, an input of 010 yields the output 01. An input of 101 yields the output of 00.

Invertibility of S-box

- 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



Above two tables are inverses of each other. For example, if the input to the left box is 001, the output is 101. The input 101 in the right table creates the output 001

XOR operation

- *An important component in most block ciphers is the exclusive-or operation (XOR)*
- *Addition/subtraction in GF(2^n) is equivalent to XOR operation*

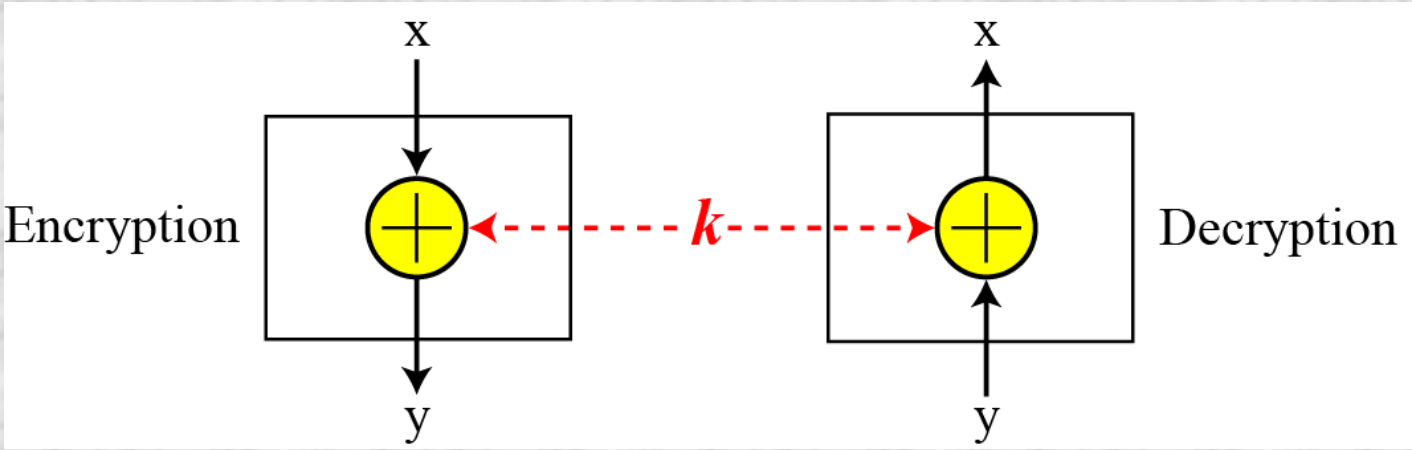
Properties of XOR operation

Closure:	two n-bit words produce one n-bit word
Associativity:	$(x \oplus y) \oplus z = x \oplus (y \oplus z)$
Commutativity:	$x \oplus y = y \oplus x$
Existance of identity:	$x \oplus \mathbf{0} = x$
Existence of inverse:	$x \oplus x = \mathbf{0}$
Complement	$x \oplus \mathbf{1} = x'$

Invertibility of the XOR

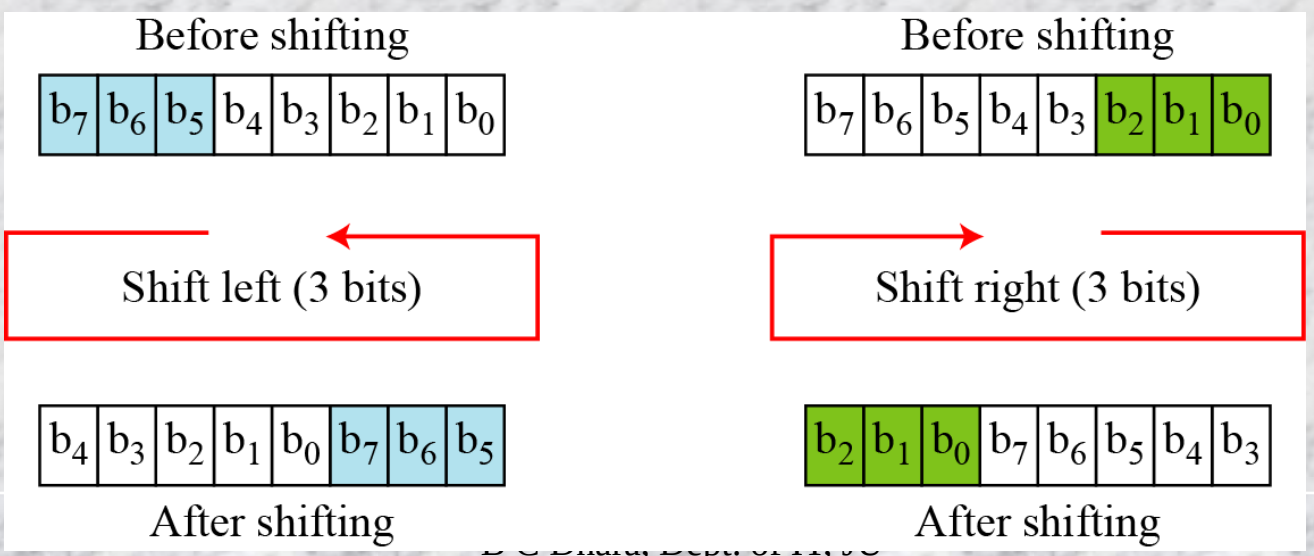
- XOR is self-invertible

$y = x \oplus k \quad \leftrightarrow \quad x = y \oplus k$



Circular shift

- Circular shift is another important component of modern block cipher
  - Circular left/right shift, shifting is modulo n
  - Circular shift mixes bits of a word
    - Hides the pattern in the original word
  - Circular shift normally keyless (amount of shift is fixed)

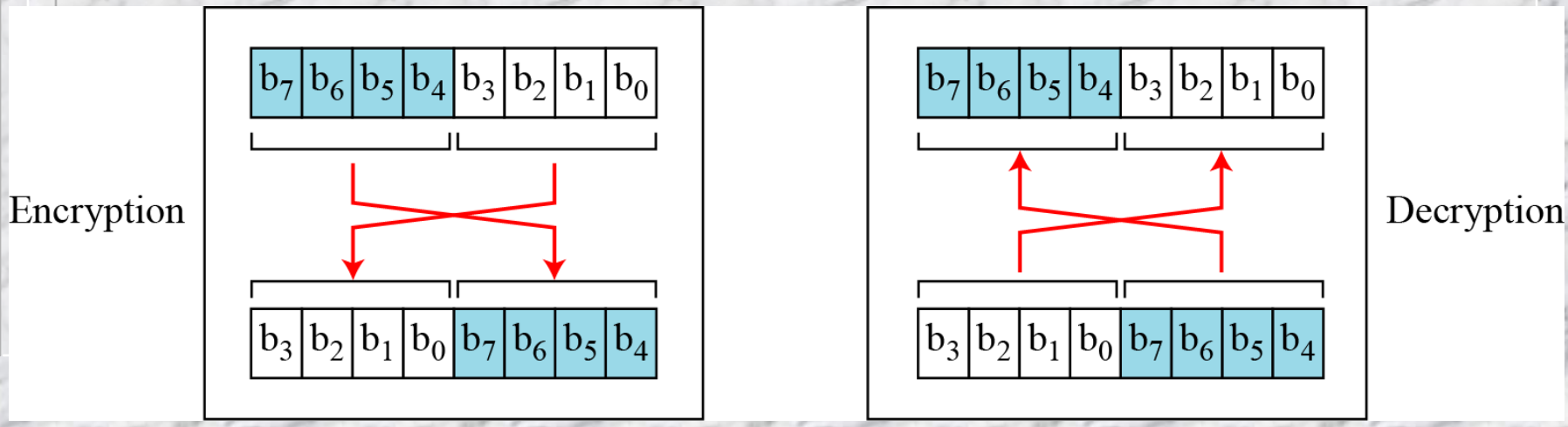


Inverse of circular shift

- Inverse of lest shift is right shift with same amount (and vice versa) OR
- k-bit left shift inverse is (n-k) left shift
  - Similarly for right shift

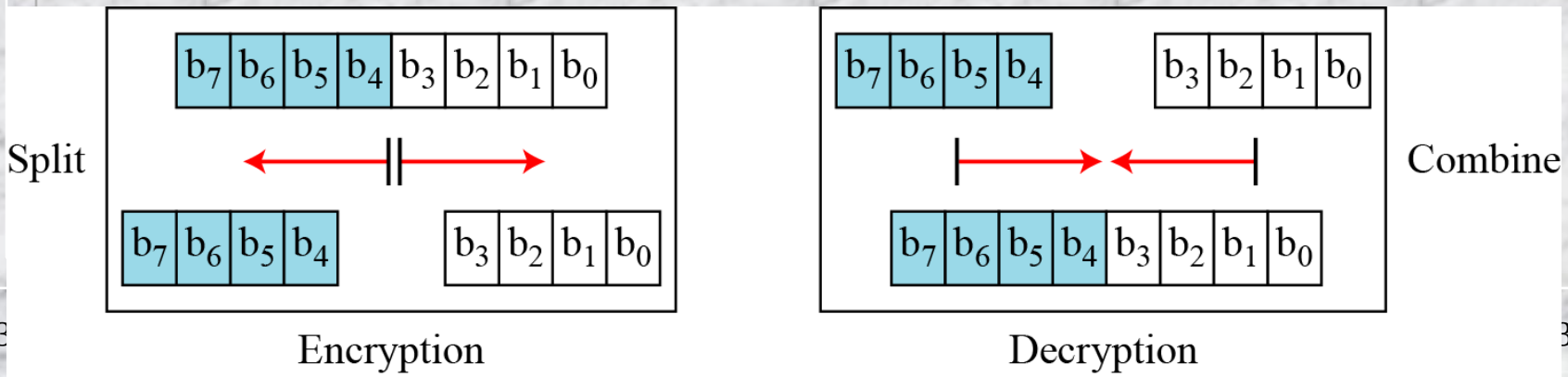
Swap operation

- A special case of circular shift with k=n/2
  - It is defined for even n
- n/2 left shift is equivalent to n/2 right shift
  - This is self-invertible
- A swap operation in encryption can be totally canceled by a swap operation in decryption



Split and Combine

- These two are pair operations
  - One is used in encryption and other is used in decryption
  - These are inverse of each other
- Split: splits an n-bit word in the middle and creates two equal length words
- Combine: concatenates two equal length words to create an n-bit word



Product ciphers

- A complex cipher combining substitution, permutation and other components discussed
- Product ciphers have two different properties: Diffusion and Confusion

Diffusion hides the relationship between ciphertext and plaintext

This frustrate the adversary who uses ciphertext statistics to find the plaintext

Each symbol in ciphertext depends on some or all symbols in the plaintext

If a single symbol in plaintext is changed, several or all symbols in the ciphertext will be changed

Product ciphers (cont…)

- Product ciphers have two different properties: Diffusion and Confusion

Confusion hides the relationship between the ciphertext and the key

This frustrate the adversary who uses ciphertext to find the key

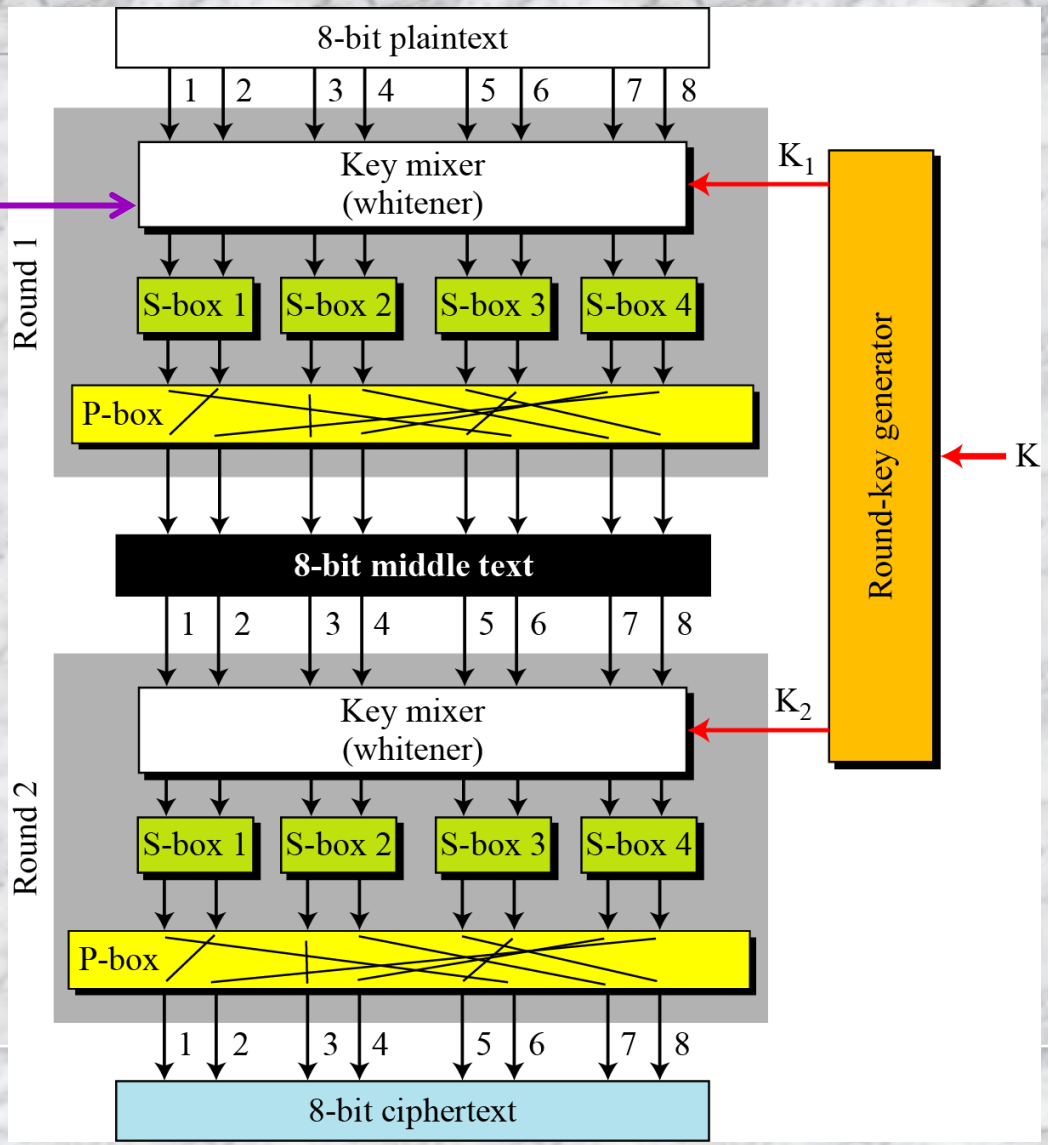
If a single bit in the key is changed, most or all symbols in the ciphertext will be changed

Rounds

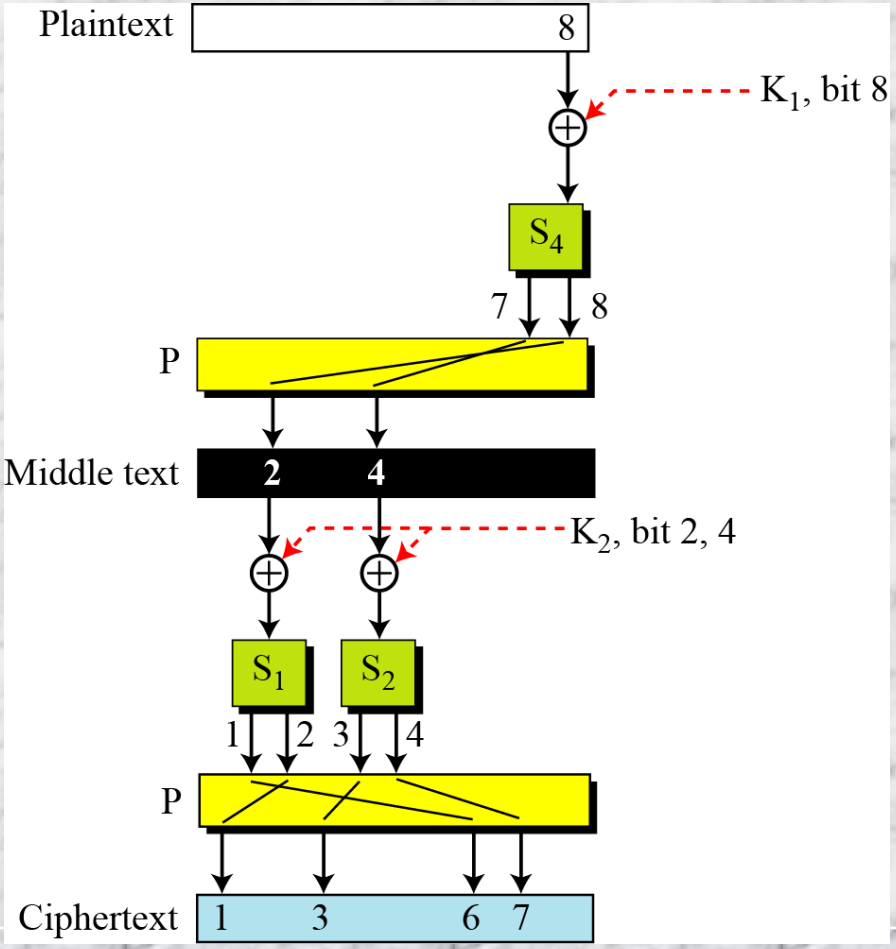
- Diffusion and confusion can be achieved using iterated product ciphers where each iteration is a combination of S-boxes, P-boxes, and other components
- Key schedule/ key generator creates different keys for each round

A product cipher made of two rounds

Normally done by XOR



Diffusion and confusion in the block cipher



Product ciphers (cont…)

- To improve diffusion and confusion:
  - Practical ciphers use larger data blocks, more S-boxes, more rounds, etc.
  - Increasing the number of rounds with more S-boxes may create a better cipher

Two Classes of Product Ciphers

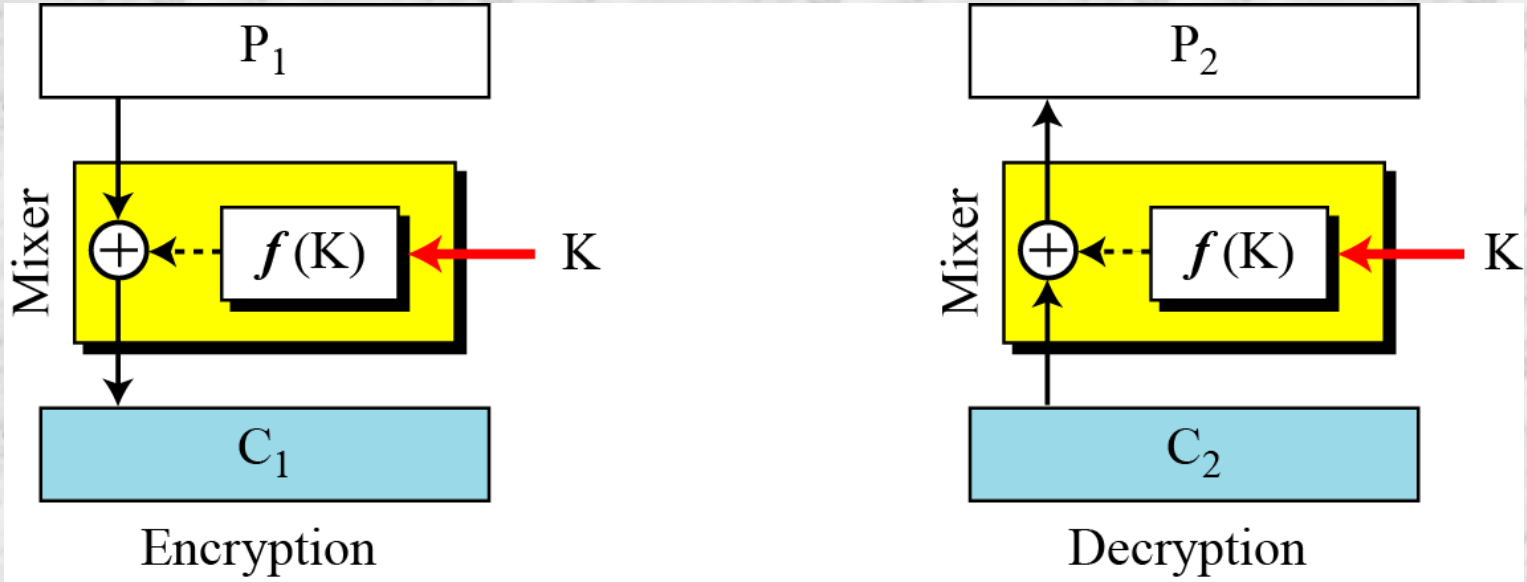
- *Modern block ciphers are all product ciphers, but they are divided into two classes.*
- *Feistel ciphers: includes both invertible and non-invertible components*
- *Non-Feistel ciphers: only invertible components*

Feistel ciphers

- A Feistel cipher can have three types of components:
  - Self-invertible
  - Invertible
  - Noninvertible: all noninvertible components combines in a unit and the same unit uses both in encryption and decryption algorithms
    - How the inverse is ensured?

How the inverse in Feistel is ensured? First thought

- This can be ensured using XOR operation



- Function ‘f’ is noninvertible
- Mixer is self-invertible

Example

The plaintext and ciphertext are each 4 bits long and the key is 3 bits long.

Function ‘f’ in noninvertible: f{111,101}= 1001

Assume that the function takes the first and third bits of the key, interprets these two bits as a decimal number, squares the number, and interprets the result as a 4-bit binary pattern.

The results of encryption and decryption if the original plaintext is 0111 and the key is 101.

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.

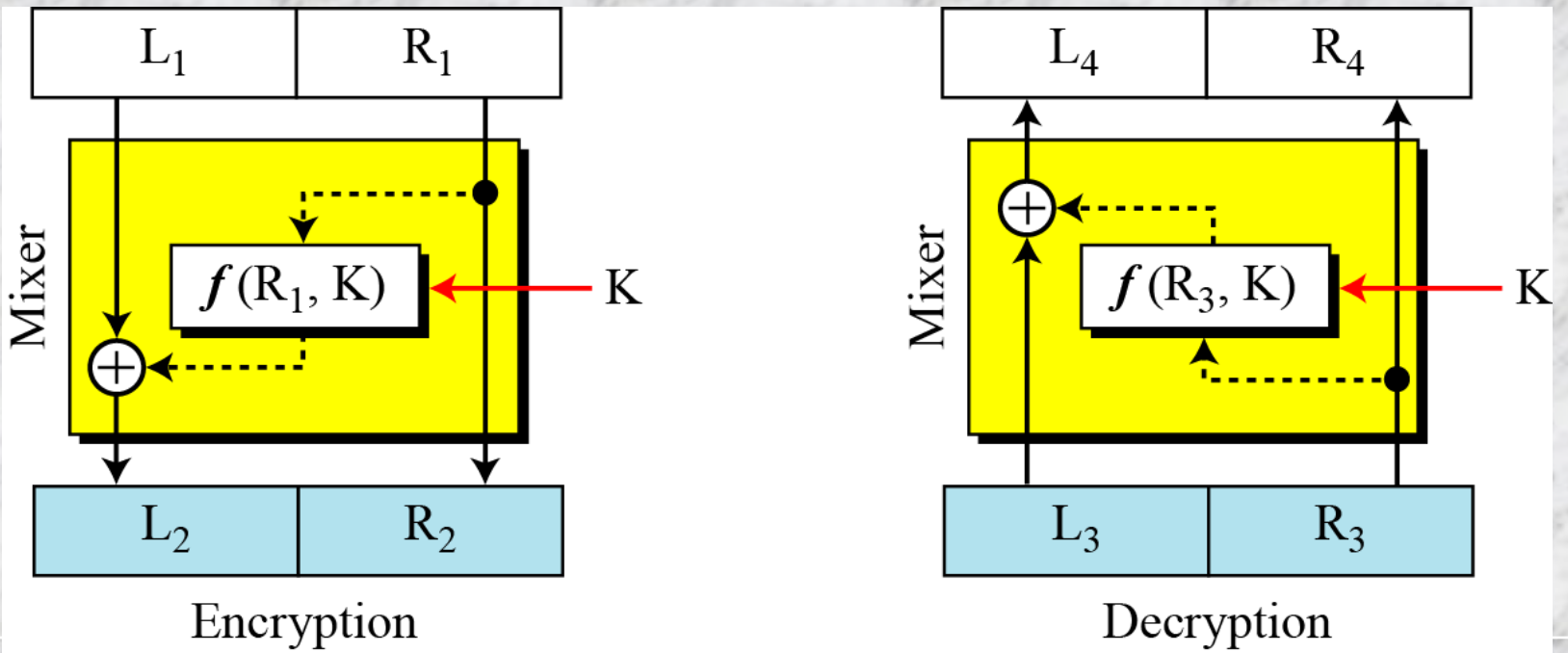
Encryption: $C = P \oplus f(K) = 0111 \oplus 1001 = 1110$

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

Improvement over the previous Feistel design

For non-invertible function same input has to be given

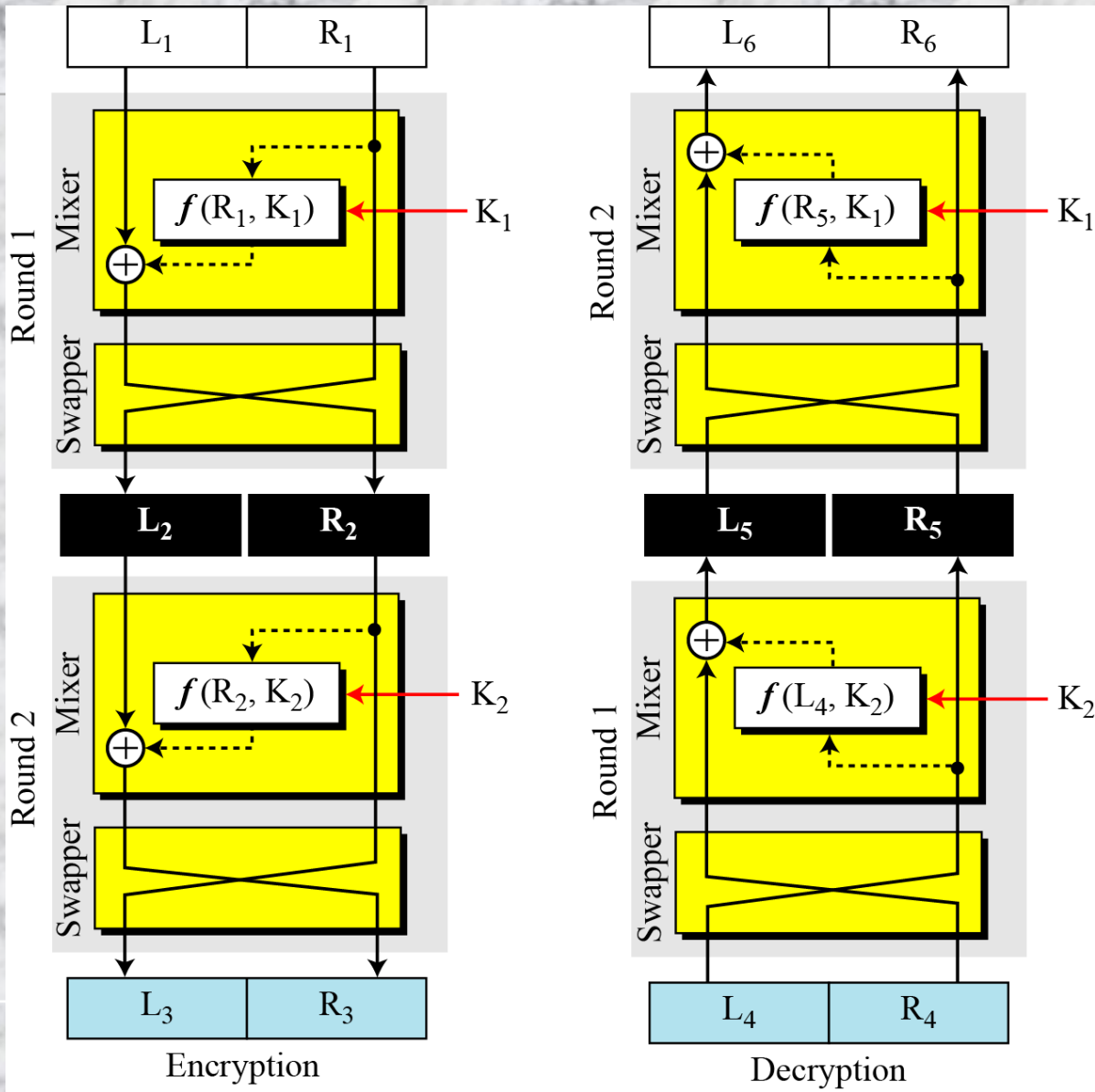
To make this versatile, a part of input from plaintext (or ciphertext)



Disadvantage: right half of the plaintext never changes

Final design of a Feistel cipher with two rounds

- Needs more improvement
- Increase the number of rounds
- Add swaper: swaps left half and right half
  - All part of the text will be changed



Non Feistel ciphers

- Use only invertible components
  - S-boxes with equal number of inputs and outputs
  - Straight P-boxes only

Attacks on block ciphers

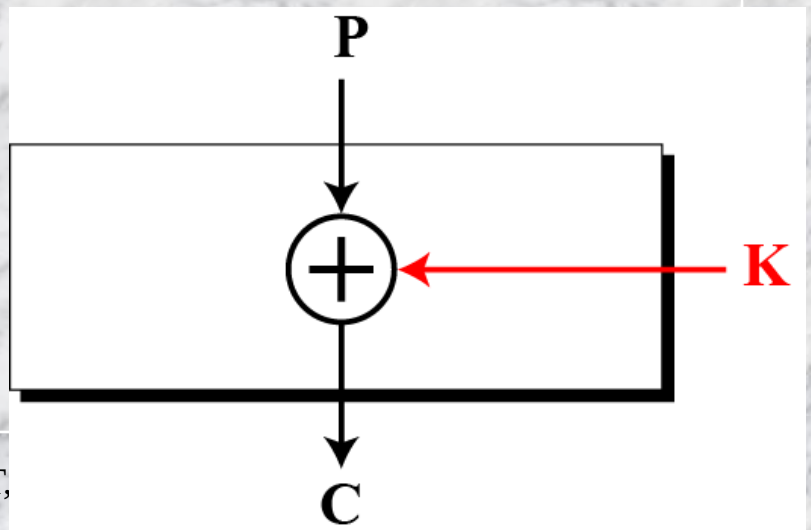
- Modern block ciphers resist most of the tradition attacks
  - Brute-force attack on the key is almost infeasible since key space is normally very large
  - Considering the structure of the modern cipher, new attacks have been devised
    - These attacks use differential and linear cryptanalysis techniques

Differential attacks

- These are mainly based on chosen-plaintext attack
  - Can access the machine of the sender, select some plaintexts and their corresponding ciphertext are computed
  - Goal is to find the cipher key
- Before actual attack, needs to analyze the encryption algorithm to get idea about the plaintext and ciphertext relationship
  - Sometimes relationship between the plaintext difference and ciphertext difference leaking information about the key

Simple attack on modern block cipher

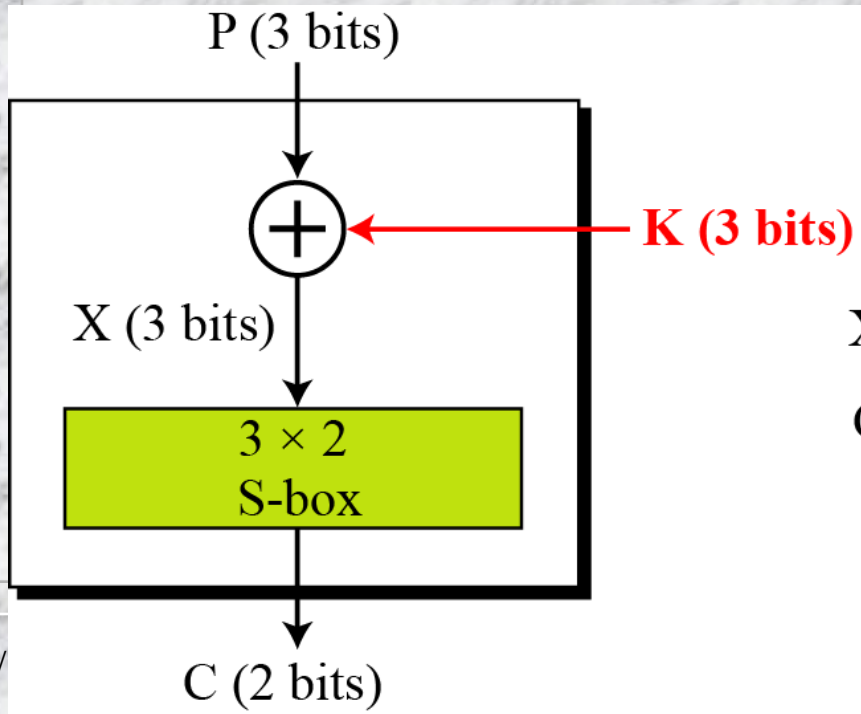
- Assume that the cipher is made only of one exclusive-or operation, as shown in below
- Without knowing the value of the key, the relationship between plaintext differences and ciphertext differences can be identified
- if by plaintext difference we mean P1 ⊕ P2 and by ciphertext difference, we mean C1⊕ C2
  - Prove that C1 ⊕ C2 = P1 ⊕ P2



S-box added then attack on previous cipher

Non-linearity is much needed for the security of the cipher

- $P1 \oplus P2 = X1 \oplus X2$
- Existence of S-box prevents an attacker to get the relationship between plaintext and ciphertext
- A probabilistic relationship can be computed



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

S-box table

$P1 \oplus P2 = 001$; K=101

	$X1 \oplus X2$	$C1 \oplus C2$
001⊕000 →	100⊕101 →	$01 \oplus 00 = 01$
000⊕001 →	101⊕100 →	$00 \oplus 01 = 01$
010⊕011 →	111⊕110 →	$00 \oplus 11 = 11$
100⊕101 →	001⊕000 →	$00 \oplus 11 = 11$
101⊕100 →	000⊕001 →	$11 \oplus 00 = 11$
011⊕010 →	110⊕111 →	$11 \oplus 00 = 11$
110⊕111 →	011⊕010 →	$10 \oplus 10 = 00$
111⊕110 →	010⊕011 →	$10 \oplus 10 = 00$

Probabilistic relationship as shown in Table

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

S-box table

$C_1 \oplus C_2$

$P_1 \oplus P_2$	*00*	*01*	*10*	*11*
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**

Differential distribution table

$P_1 \oplus P_2$ \ $C_1 \oplus C_2$	*00*	*01*	*10*	*11*
000	**1**	**0**	**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**

Differential attack

- if $P_1 \oplus P_2 = 001$, then $C_1 \oplus C_2 = 11$ with the probability of 0.50 (50 percent)
- If assume $C_1 = 00$ 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,

$C_1 = 00 \rightarrow X_1 = 001$ or $X_1 = 111$

If $X_1 = 001 \rightarrow K = X_1 \oplus P_1 = 011$

If $X_1 = 111 \rightarrow K = X_1 \oplus P_1 = 101$

X_1=101 ➔ K= $X_1 \oplus P_1 = 111$

$C_2 = 11 \rightarrow X_2 = 000$ or $X_1 = 110$

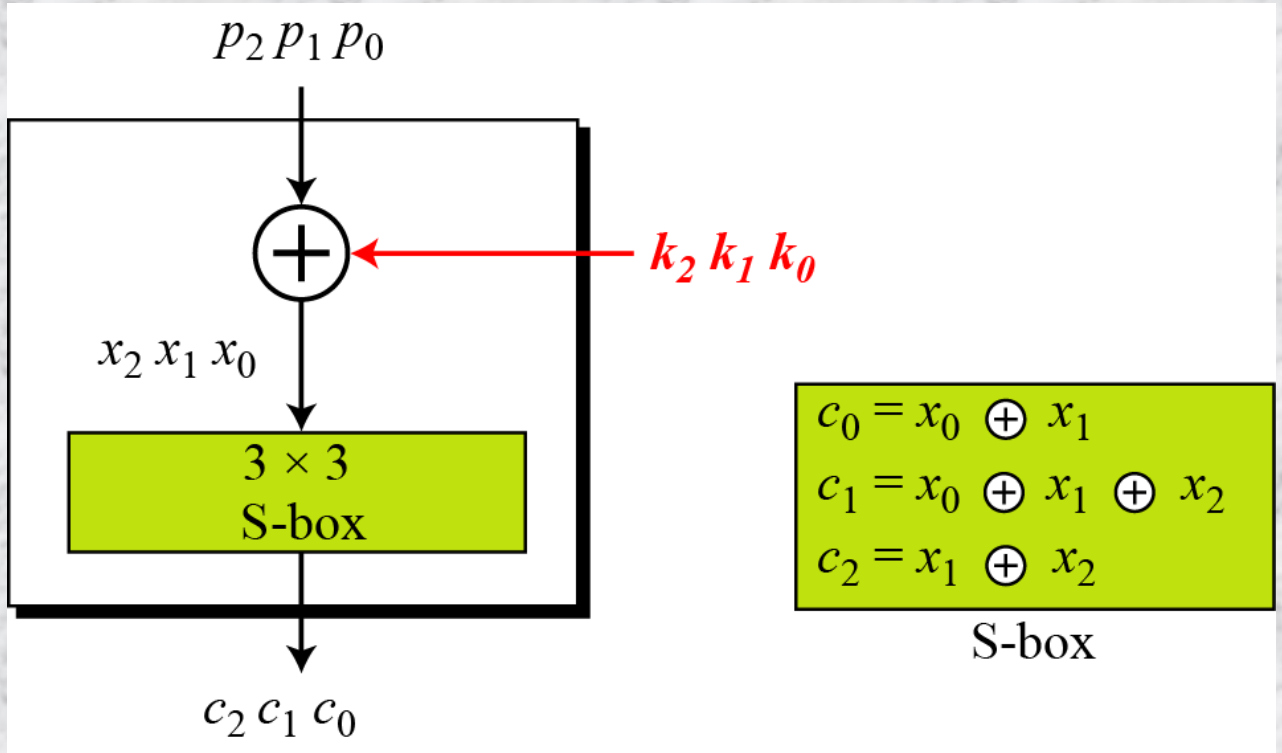
If $X_2 = 000 \rightarrow K = X_2 \oplus P_2 = 011$

If $X_2 = 110 \rightarrow K = X_2 \oplus P_2 = 101$

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

Linear cryptanalysis (contd…)

- A simple cipher with linear S-box



simple cipher with linear S-box (contd…)

$$c_0 = p_0 \oplus k_0 \oplus p_1 \oplus k_1$$
$$c_1 = p_0 \oplus k_0 \oplus p_1 \oplus k_1 \oplus p_2 \oplus k_2$$
$$c_2 = p_1 \oplus k_1 \oplus p_2 \oplus k_2$$

$$k_1 = (p_1) \oplus (c_0 \oplus c_1 \oplus c_2)$$
$$k_2 = (p_2) \oplus (c_0 \oplus c_1)$$
$$k_0 = (p_0) \oplus (c_1 \oplus c_2)$$

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

MODERN STREAM CIPHERS

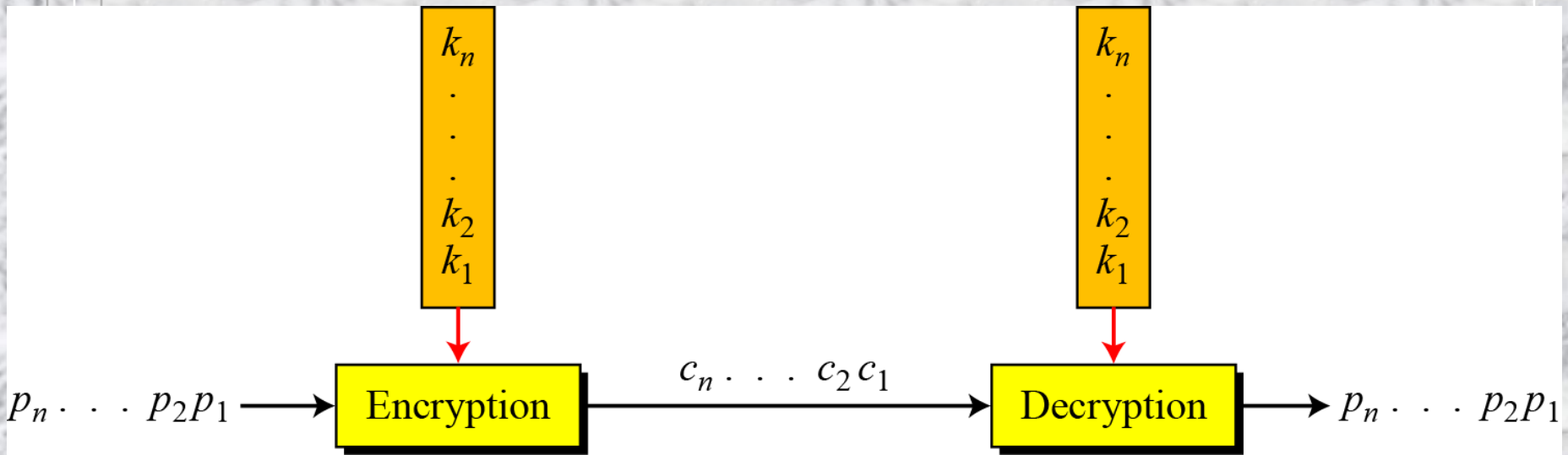
In a modern stream cipher, encryption and decryption are done r bits at a time.

We have a plaintext bit stream $P = p_n \ldots p_2 p_1$, *a ciphertext bit stream* $C = c_n \ldots c_2 c_1$, *and a key bit stream* $K = k_n \ldots k_2 k_1$, *in which* p_i , c_i , *and* k_i *are r-bit words* $c_i = E(k_i, p_i)$ *and* $p_i = D(k_i, c_i)$

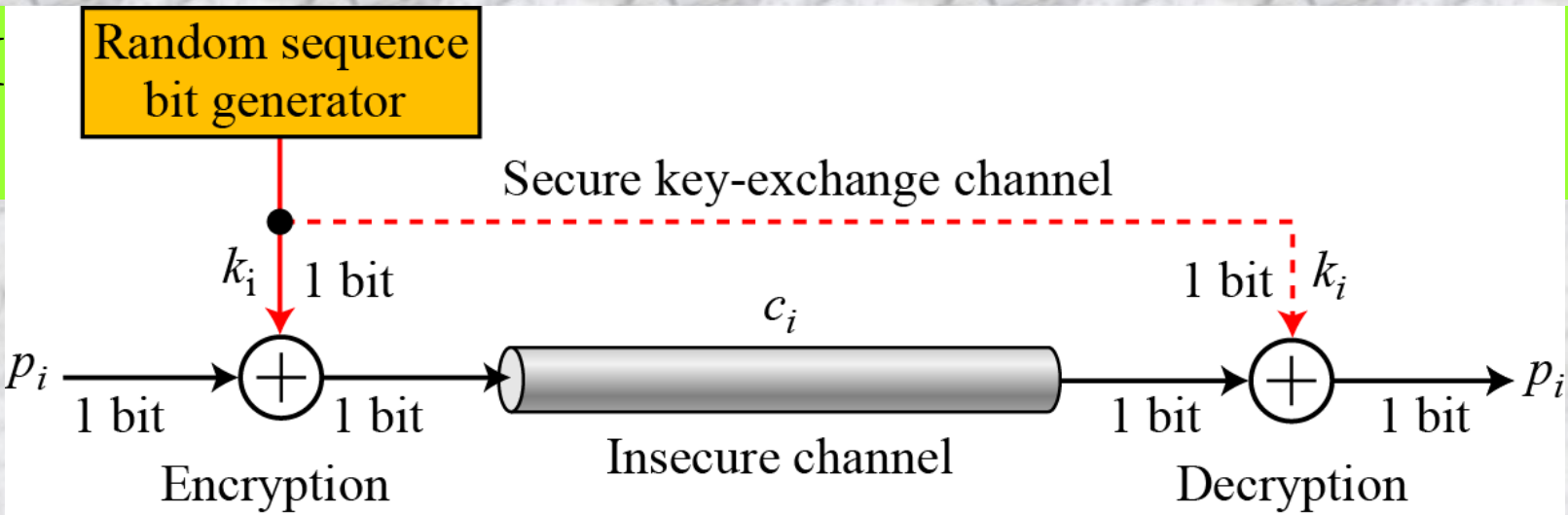
Two important concepts:

Synchronous Stream Ciphers

Nonsynchronous Stream Ciphers



Synchronous Stream Ciphers



- One time pad
  - The simplest and most secure type of synchronous stream cipher is called the one time pad
  - Random key stream is used and both in encryption and decryption 1-bit XOR executed

Notes on one time pad

- An adversary cannot guess the key or the plaintext and ciphertext statistics
- There is no relationship between the plaintext and ciphertext
- Cannot break the cipher unless tries all possible random key stream
  - which is 2^n , if the size of the plaintext is n bits
- Main concern, how to communicate the key each time they want to communicate?
  - Very difficult to achieve

One time pad (*Contd…)*

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.

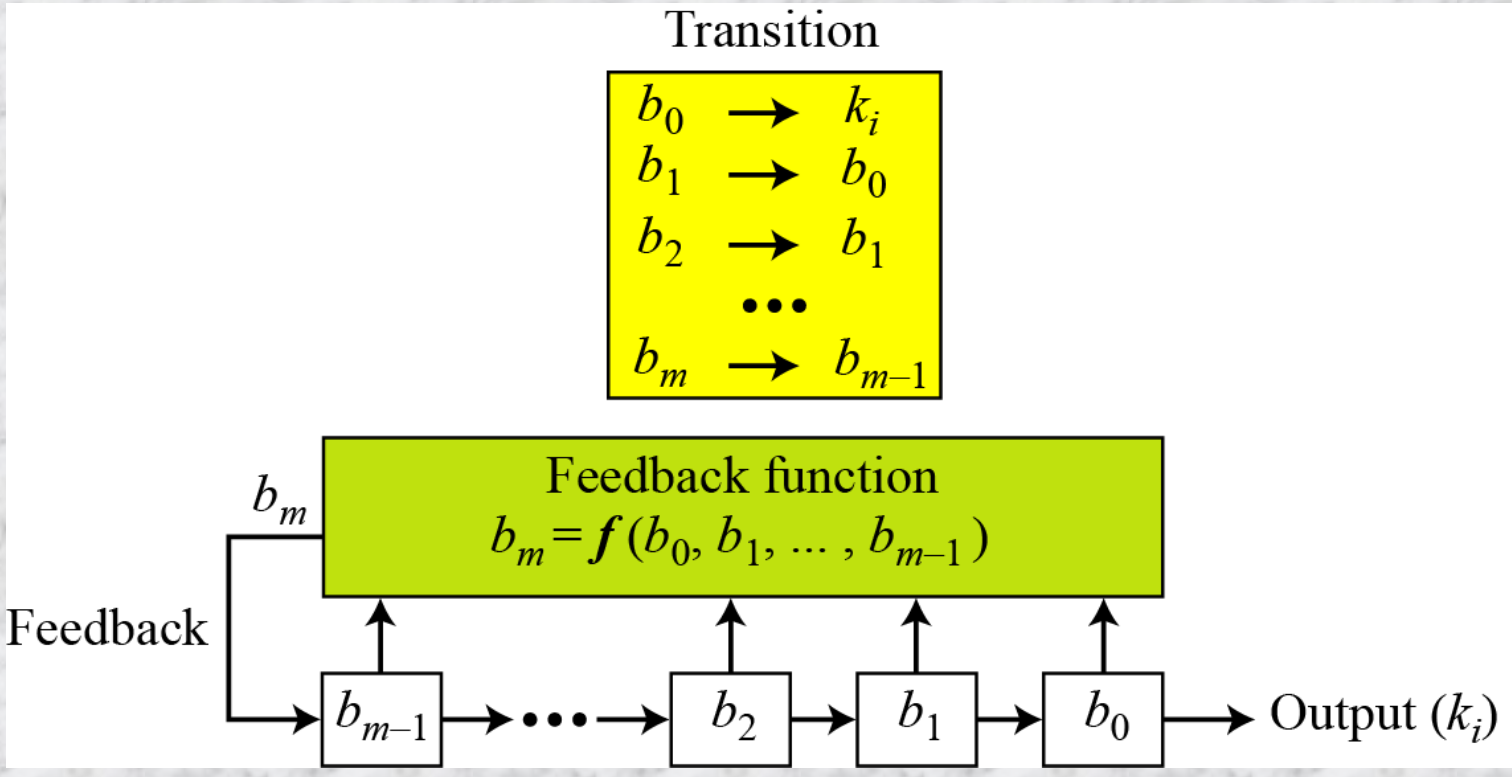
One time pad (*Contd…)*

b. Because $1 \oplus k_i = k_i'$ where 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.

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.

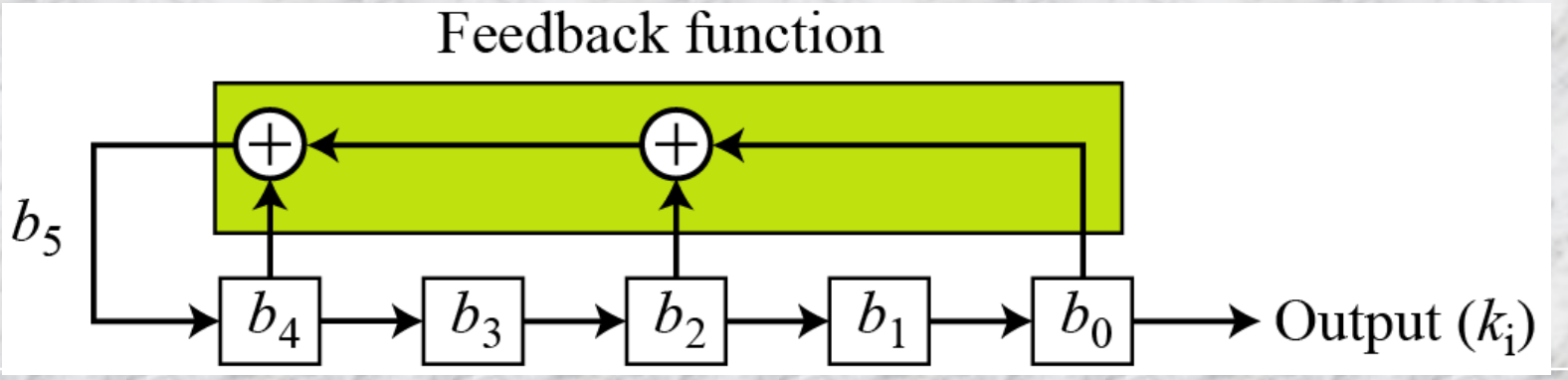
Feedback shift register



Linear feedback shift register

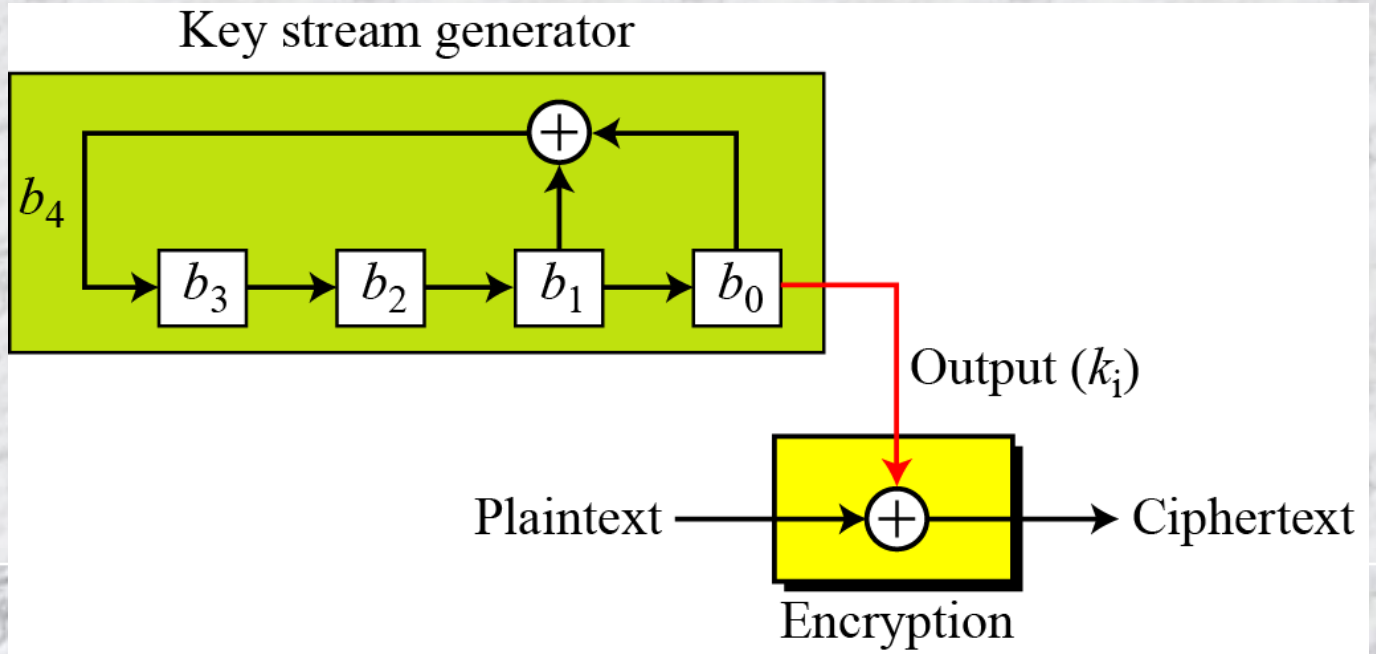
- $b_m = c_{m-1}b_{m-1} + c_{m-2}b_{m-2} + \cdots + c_0 b_0, \; c_0 \neq 0$
- In GF(2)
  - $b_m = c_{m-1}b_{m-1} \oplus c_{m-2}b_{m-2} \oplus \ldots \oplus c_0 b_0, \quad c_0 \neq 0$

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



Linear feedback shift register (contd…)

- 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 $(0001)_2$



Key sequence

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

Key sequence (contd…)

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

Periodicity of LFSR

- 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 maximum period of an LFSR is to $2^m - 1$, m bit seed

Characteristics polynomial

- $x^m = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_0, \ c_0 \neq 0$
- In GF(2)
  - $x^m + c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_0, \ c_0 \neq 0$
- LFSR has maximum period if
  - It has even number of cells
  - The characteristics polynomial is primitive

Attacks on LFSR

- LFSR is simple in structure and vulnerable to attacks. Two simple attacks on LFSR are:
  - If structure is known, intercepting and analyzing one n-bit ciphertext can predict all future ciphertexts
  - If structure is not known, a known plaintext attack of 2n bits to break cipher

Nonlinear Feedback Shift Register

- LFSR is vulnerable to attacks due its simple structure
- A better stream cipher can be achieved using nonlinear feedback shift register
- In NLFSR, b_m is a nonlinear function of b_{m-1}, b_{m-2}, b_1, b_0
  - $b_4 = (b_3 \text{ AND } b_2) \text{ OR } (b_1 \text{ AND } b_0)$
- NLFSR is not common as no mathematical foundation to make NLFSR with maximum period