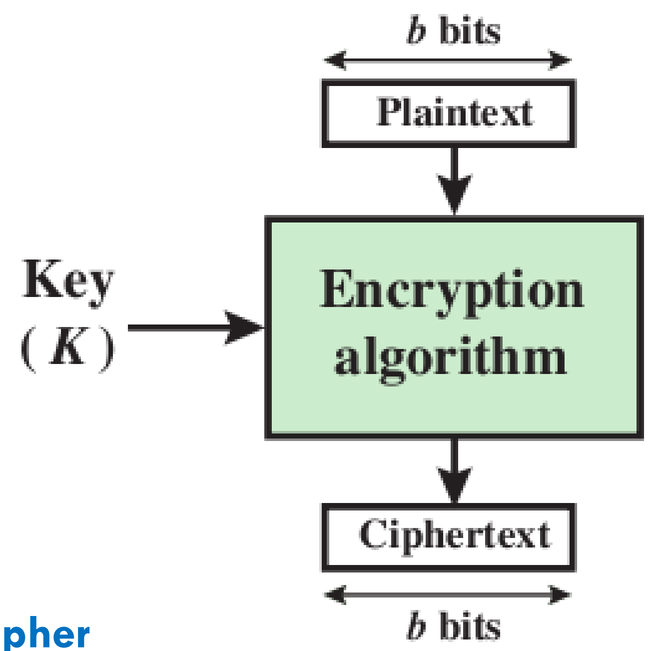
BLOCK CIPHERS

Introduction

- Many symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher
- For that reason, it is important to examine the design principles of the Feistel cipher.
- A comparison of stream ciphers and block ciphers will be made

Block Ciphers

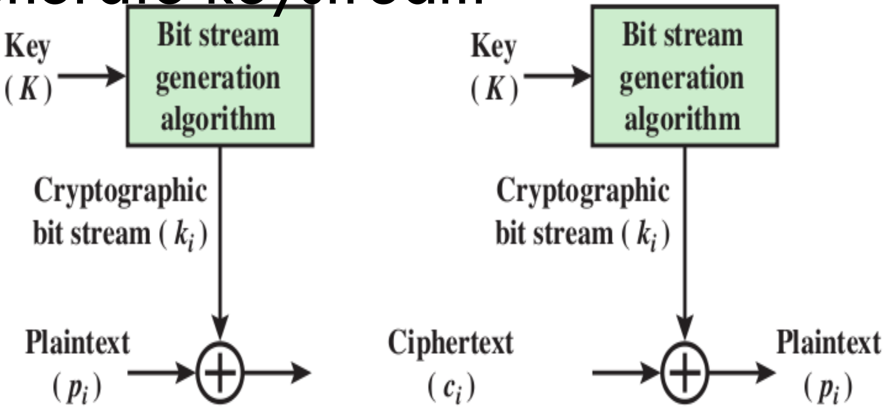
- Encrypt a block of plaintext as a whole to produce same sized cipher text
- Typical block sizes are 64 or 128 bits
- As with a stream cipher, the two users share a symmetric encryption key
- Using some modes of operation block ciphe the same effect as a stream cipher.
- applicable to a broader range of
- applications than stream ciphers.



Block cipher

Stream Ciphers

- Encrypts a digital data stream one bit or one byte at a time
- One time pad is example; but has practical limitations
- Typical approach for stream cipher:
  - Key (K) used as input to bit-stream generator algorithm
  - Algorithm generates cryptographic bit stream (k_i) used to encrypt plaintext
  - Users share a key; use it to generate keystream



Stream cipher using algorithmic bit-stream generator

Motivation for the Feistel Cipher Structure : Reversible and irreversible Mappings

- n-bit block cipher takes n bit plaintext and produces n bit ciphertext
- In n bits, 2n possible different plaintext blocks
- Encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext
- For n = 2,

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

- If we limit ourselves to reversible mappings, the number of different transformations is $(2^n)!$.

Ideal Block Cipher

- n-bit input maps to 2^n possible input states
- Substitution used to produce 2^n output states
- Output states map to n-bit output
- Feistel refers to this as Ideal block cipher because it allows maximum number of possible encryption mappings from plaintext block
- Problems with ideal block cipher:
  - Small block size: equivalent to classical substitution cipher; cryptanalysis based on statistical characteristics feasible
  - Large block size: key must be very large; performance/implementation problems

Ideal block cipher example

P	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12
00	00	00	00	00	00	00	01	01	10	10	11	11
01	01	01	10	10	11	11	00	00	00	00	00	00
10	10	11	01	11	01	10	10	11	01	11	01	10
11	11	10	11	01	10	01	11	10	11	01	10	01

P	K13	K14	K15	K16	K17	K18	K19	K20	K21	K22	K23	K24
00	01	01	10	10	11	11	01	01	10	10	11	11
01	10	11	01	11	01	10	10	11	01	11	01	10
10	00	00	00	00	00	00	11	10	11	01	10	01
11	11	10	11	01	10	01	00	00	00	00	00	00

2 bit block, $2^2=4$ mappings

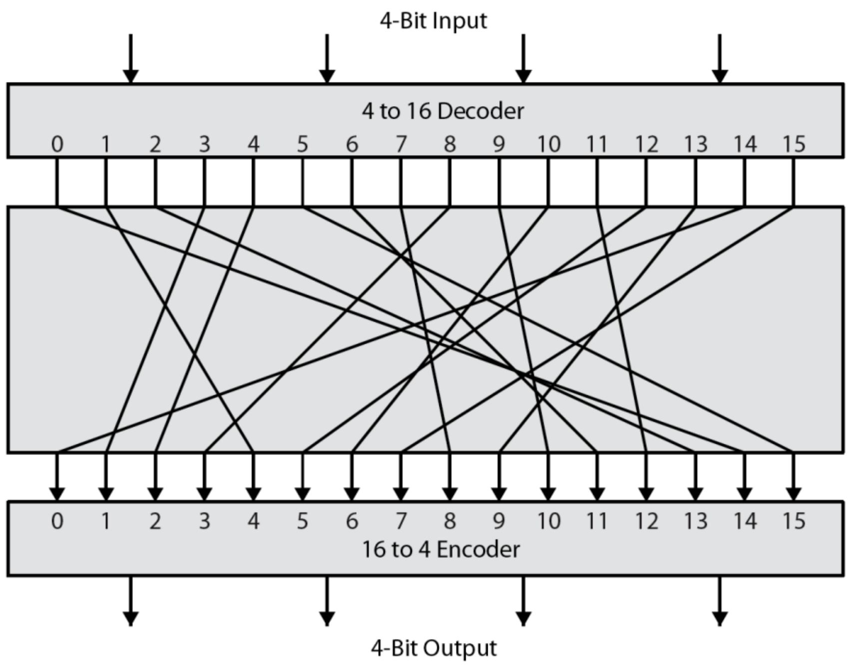
Input 01
Output 01 if K17 is used, as
K17=11 01 00 10

- Ideal : n-bit block, 2^n Mappings.
- Total $2^n!$ mappings
- Any Key length (to represent any mapping) n. 2^n bits (each mapping contains n bits)
- Fiestel: n-bit block, 2^K mappings, key length K

Substitution/Block cipher

- 4-bit input produces one of 16 input states
- What is the possible number of different transformations?
- which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits.
- This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext.

Figure illustrates the logic of a general substitution cipher for $n = 4$.



Encryption and Decryption Tables for Substitution Cipher

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Substitution-permutation (S-P) networks

Claude Shannon and Substitution-Permutation Ciphers

- Claude Shannon introduced idea of substitution-permutation (S-P) networks in 1949 paper
- This idea is the basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- Provide *confusion & diffusion* of message & key

Diffusion and Confusion

Diffusion

- Dissipates statistical structure of plaintext over bulk of ciphertext
- E.g. A plaintext letter affects the value of many ciphertext letters
- How: repeatedly apply permutation (transposition) to data, and then apply function

Confusion

- Makes relationship between ciphertext and key as complex as possible
- Even if attacker can find some statistical characteristics of ciphertext, still hard to find key
- How: apply complex (non-linear) substitution algorithm

Diffusion

- How to achieve this?
  - Develop a many-to-many mapping between plain-ciphertext
  - Having each plaintext digit affect the value of many ciphertext digits; generally
  - this is equivalent to having each ciphertext digit be affected by many plaintext digits.
  - An example: encrypt a message of characters with an averaging operation:
  - adding k successive letters to get a ciphertext letter y_n.
  - One can show that the statistical structure of the plaintext has been dissipated

$$M = m_1, m_2, m_3, \ldots$$

$$y_n = \left(\sum_{i=1}^{k} m_{n+i}\right) \bmod 26$$

Confusion

- How to achieve this?
- Achieved by the use of a complex substitution algorithm.
- In contrast, a simple linear substitution function would add little confusion.

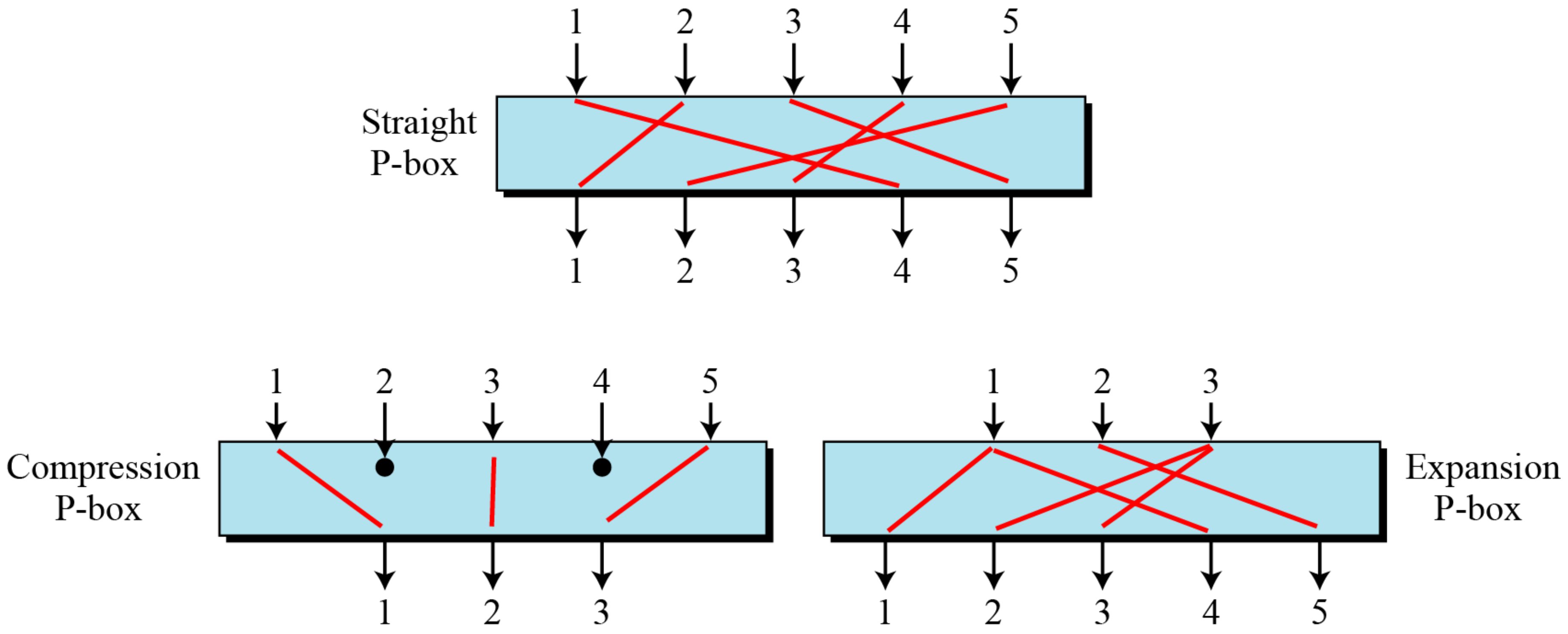
Components of a Modern Block Cipher

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

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

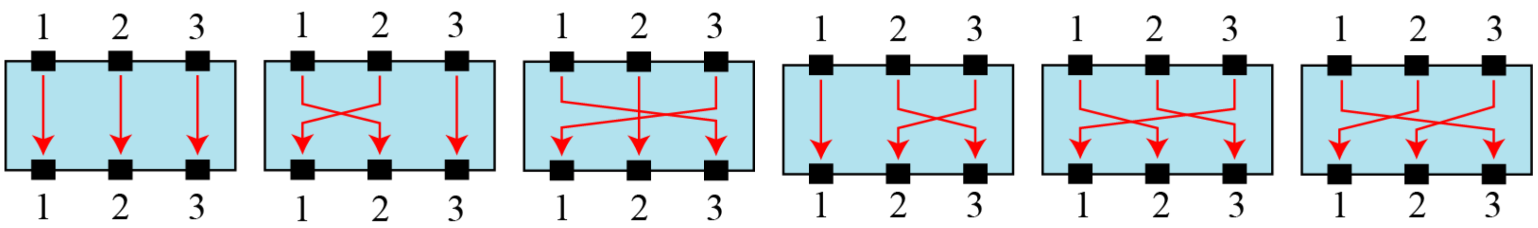
Three types of P-boxes



Example

Figure shows all 6 possible mappings of a 3 × 3 P-box.

The possible mappings of a 3 × 3 P-box



Continued

Straight P-Boxes

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

Continued

Example

Design an 8 × 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.

Solution

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.

Continued

Compression P-Boxes

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

Table ***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

Expansion P-Box *Continued*

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

Table ***Example of a 12 × 16 permutation table***

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

P-Boxes: Invertibility

Continued

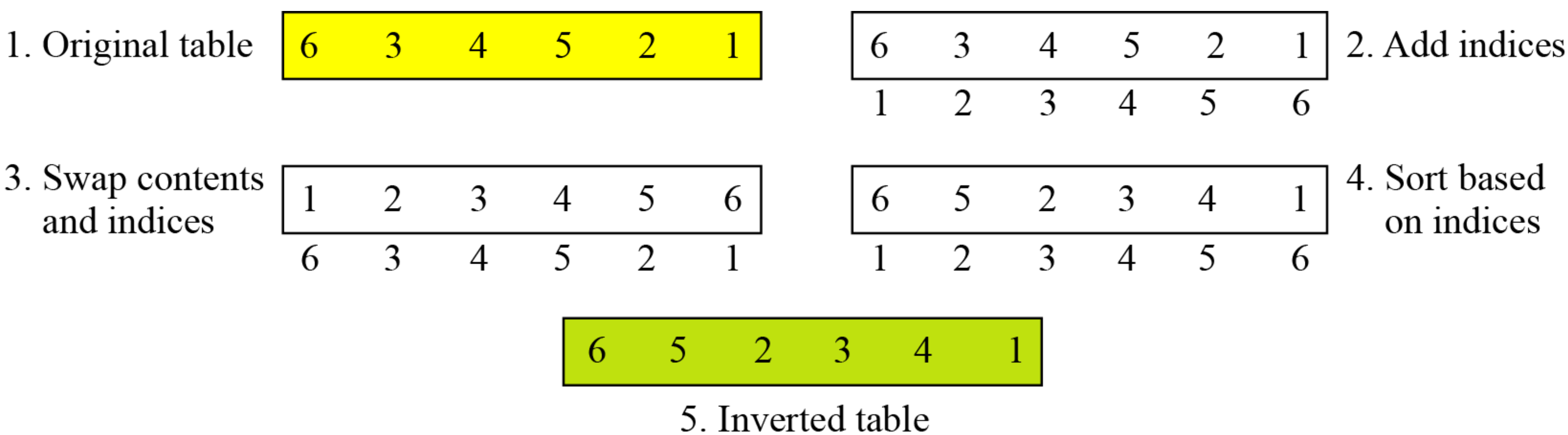
A straight P-box is invertible, but compression and expansion P-boxes are not.

Continued

Example

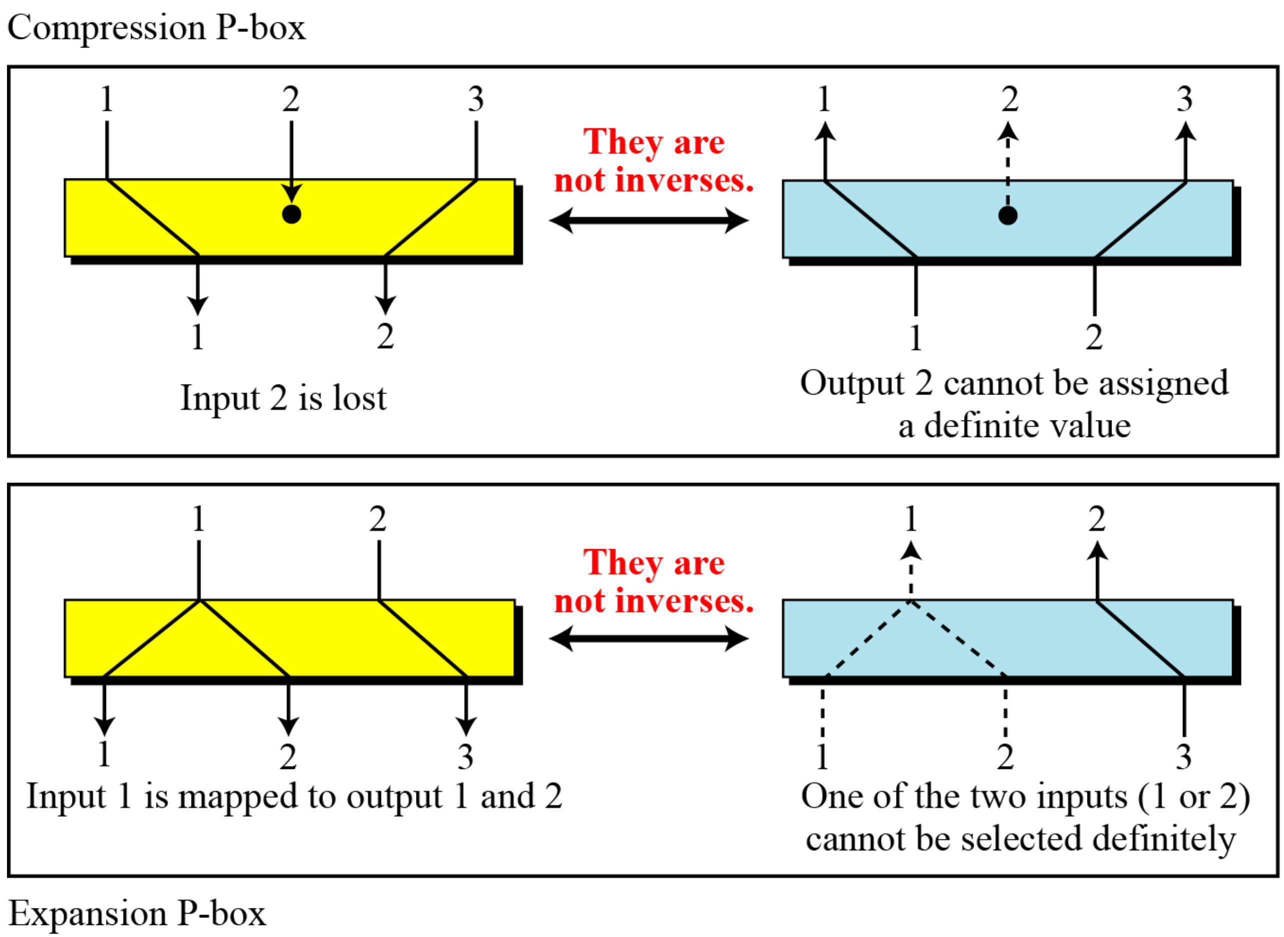
Figure shows how to invert a permutation table represented as a one-dimensional table.

Figure Inverting a permutation table



Continued

Figure ***Compression and expansion P-boxes are non-invertible***



Continued

S-Box

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.

Example

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:

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

Example ***Continued***

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.

Example *Continued*

The following table defines the input/output relationship for an S-box of size 3 × 2. The leftmost bit of the input defines the row; the two rightmost bits of the input define the column. The two output bits are values on the cross section of the selected row and column.

Leftmost bit

Rightmost bits

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

Output bits

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

Continued

Example

Figure shows an example of an invertible S-box. 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, which shows that the two tables are inverses of each other.

Figure ***S-box tables for Example***

3 bits

	00	01	10	11
0	011	101	111	100
1	000	010	001	110

Table used for encryption

3 bits

3 bits

	00	01	10	11
0	100	110	101	000
1	011	001	111	010

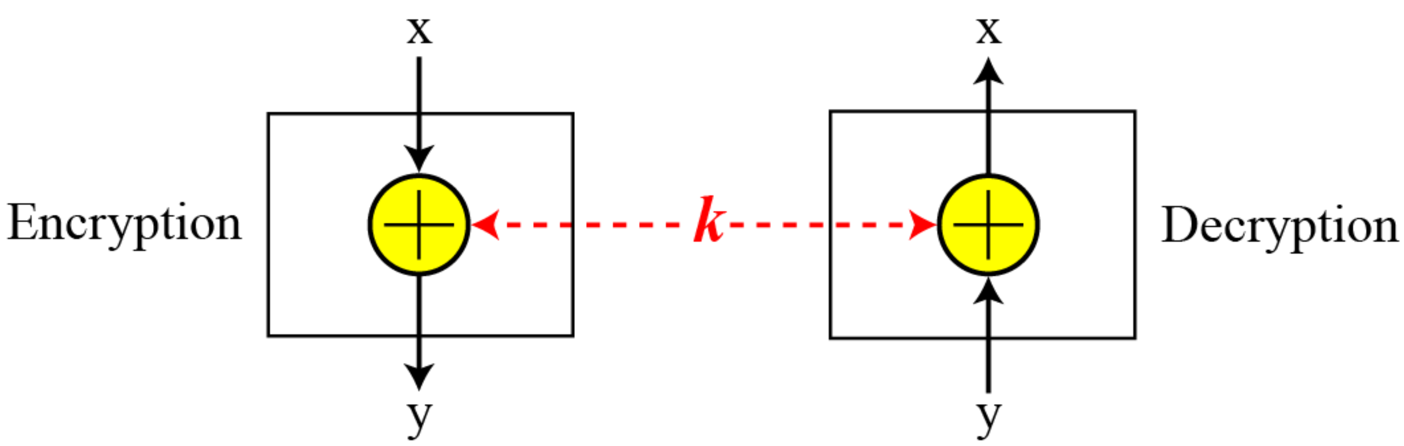
Table used for decryption

3 bits

Exclusive-Or

An important component in most block ciphers is the exclusive-or operation.

Figure ***Invertibility of the exclusive-or operation***



Continued

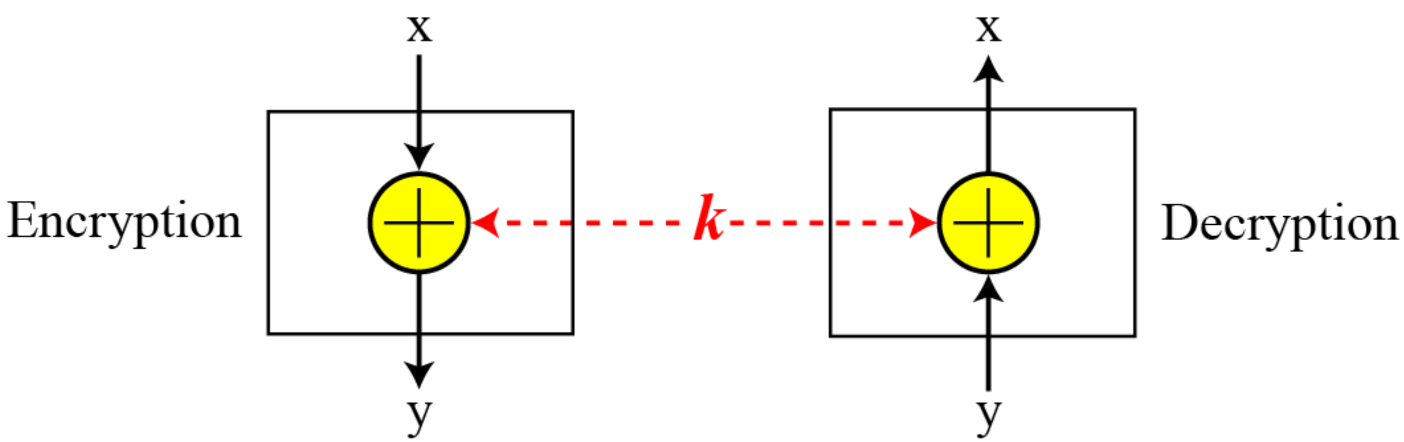
Exclusive-Or (Continued)

An important component in most block ciphers is the exclusive-or operation. Addition and subtraction operations in the $GF(2^n)$ field are performed by a single operation called the exclusive-or (XOR).

The five properties of the exclusive-or operation in the $GF(2^n)$ field makes this operation a very interesting component for use in a block cipher: closure, associativity, commutativity, existence of identity, and existence of inverse.

Continued

Figure Invertibility of the exclusive-or operation

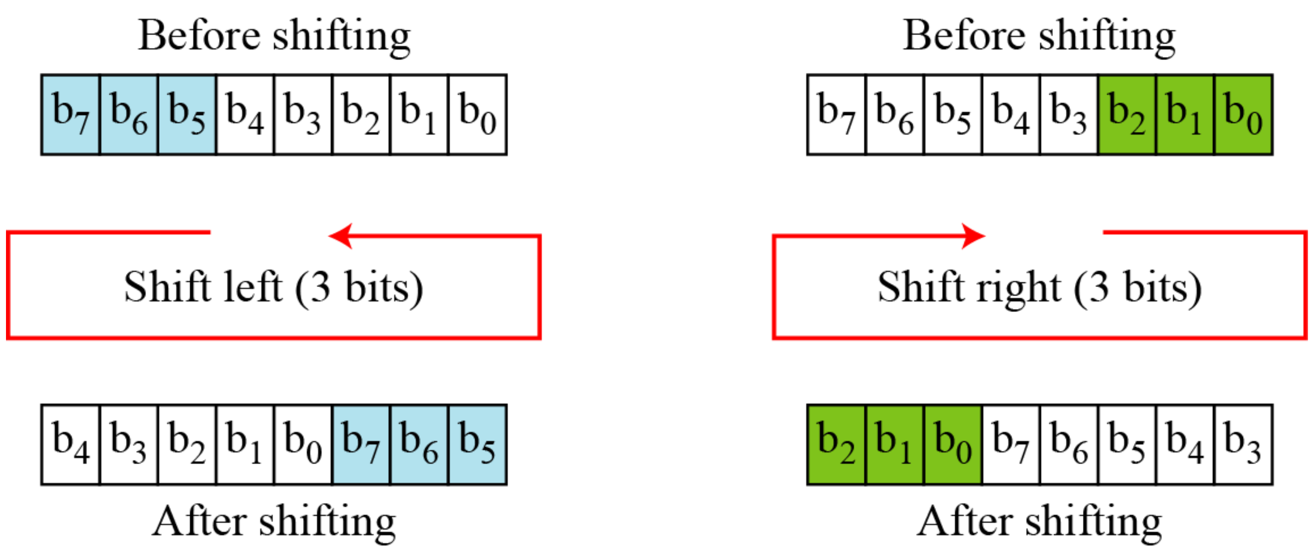


Continued

Circular Shift

Another component found in some modern block ciphers is the circular shift operation.

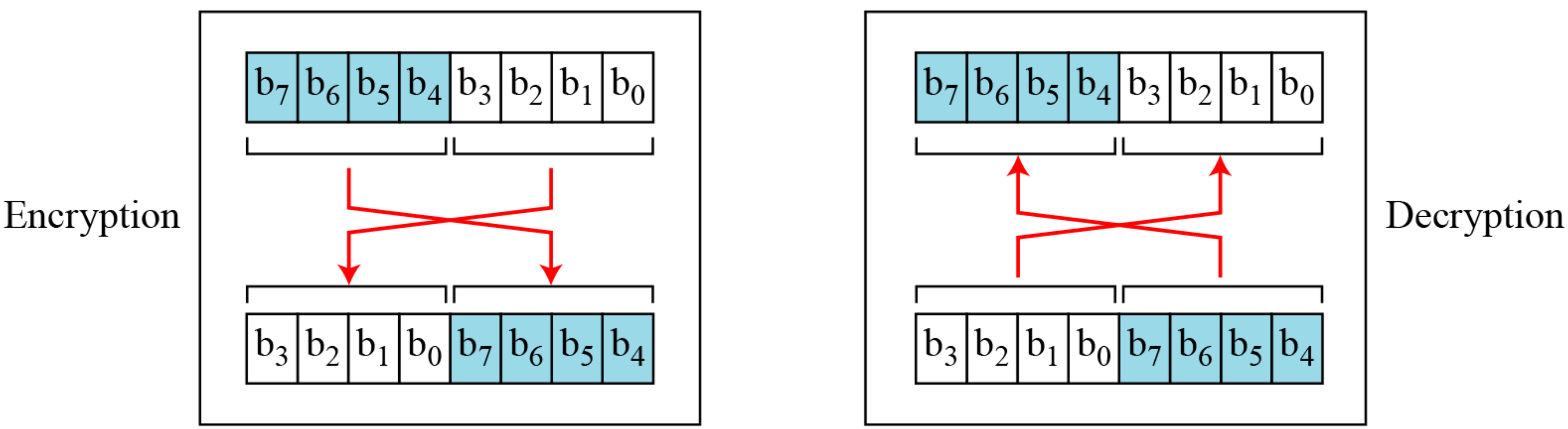
Figure ***Circular shifting an 8-bit word to the left or right***



Swap

The swap operation is a special case of the circular shift operation where $k = n/2$.

Figure *Swap operation on an 8-bit word*

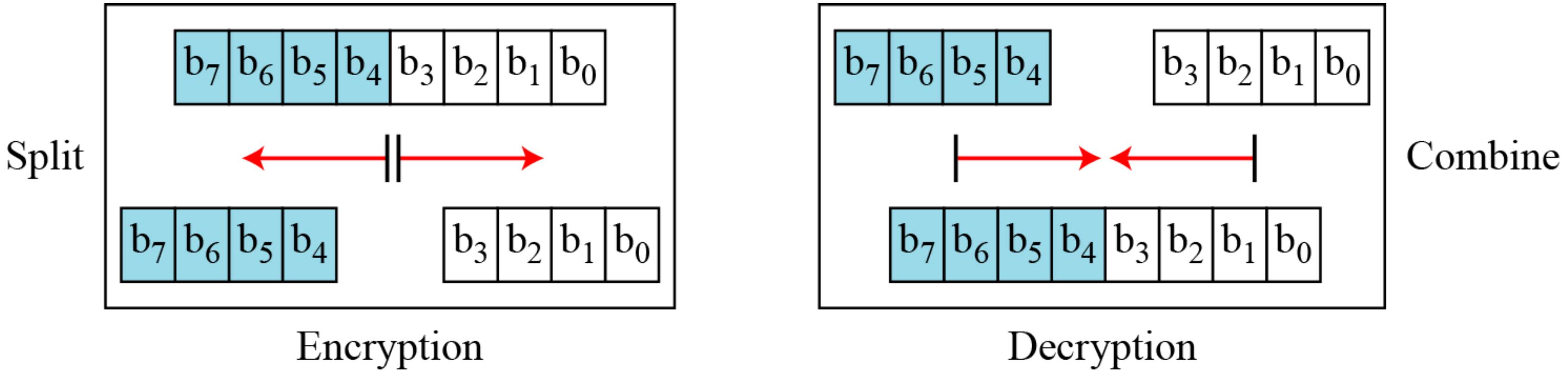


Split and Combine

Continued

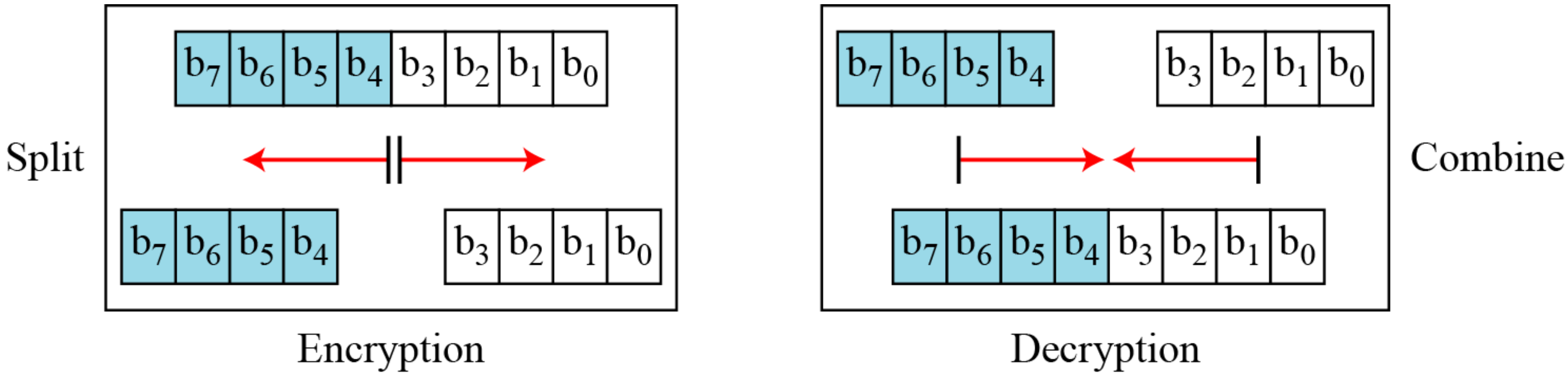
Two other operations found in some block ciphers are split and combine.

Figure 5.12 ***Split and combine operations on an 8-bit word***



Continued

Figure ***Split and combine operations on an 8-bit word***



Product Ciphers

Shannon introduced the concept of a product cipher. A product cipher is a complex cipher combining substitution, permutation, and other components.

Continued

Diffusion

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

Diffusion hides the relationship between the ciphertext and the plaintext.

Continued

Confusion

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

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

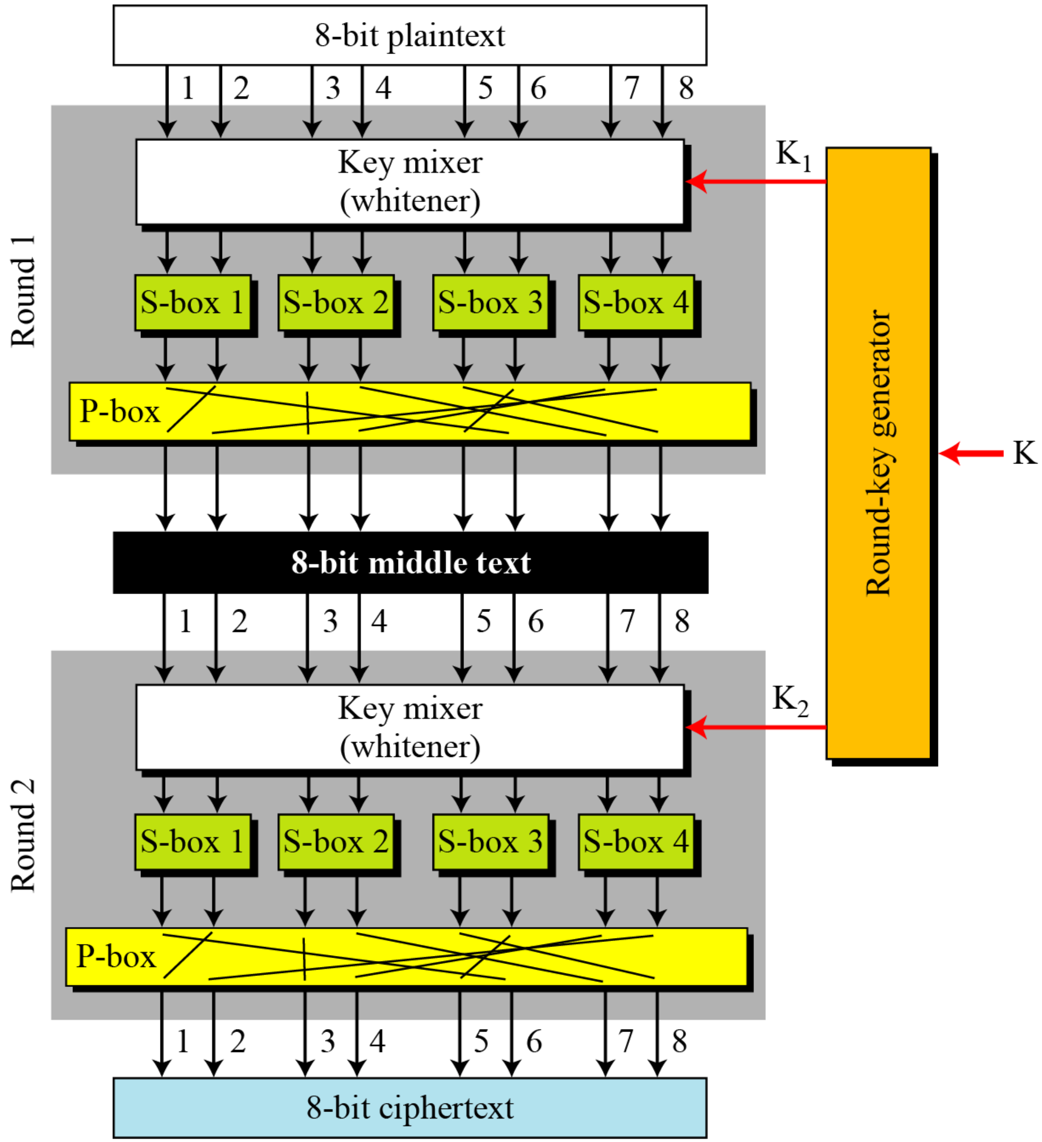
Continued

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.

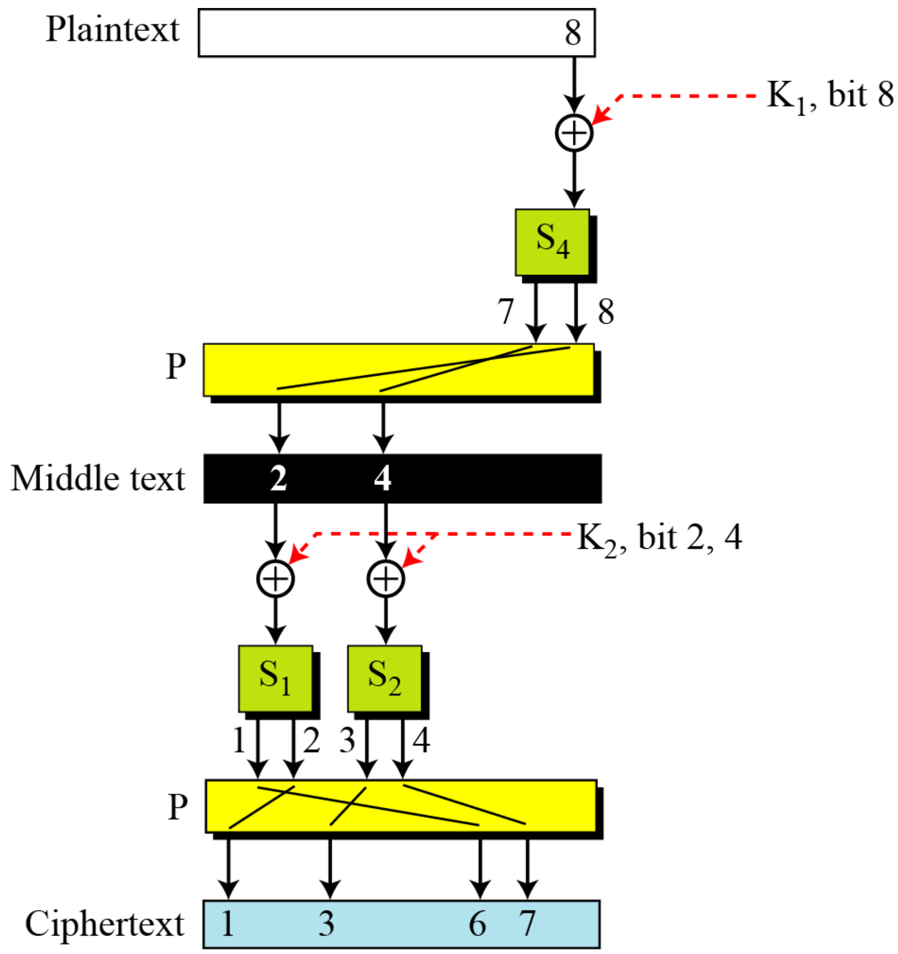
Continued

Figure *A product cipher made of two rounds*



Continued

Figure ***Diffusion and confusion in a block cipher***



Two Classes of Product Ciphers

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

1. Feistel ciphers

2. Non-Feistel ciphers

Two Classes of Product Ciphers (cont.)

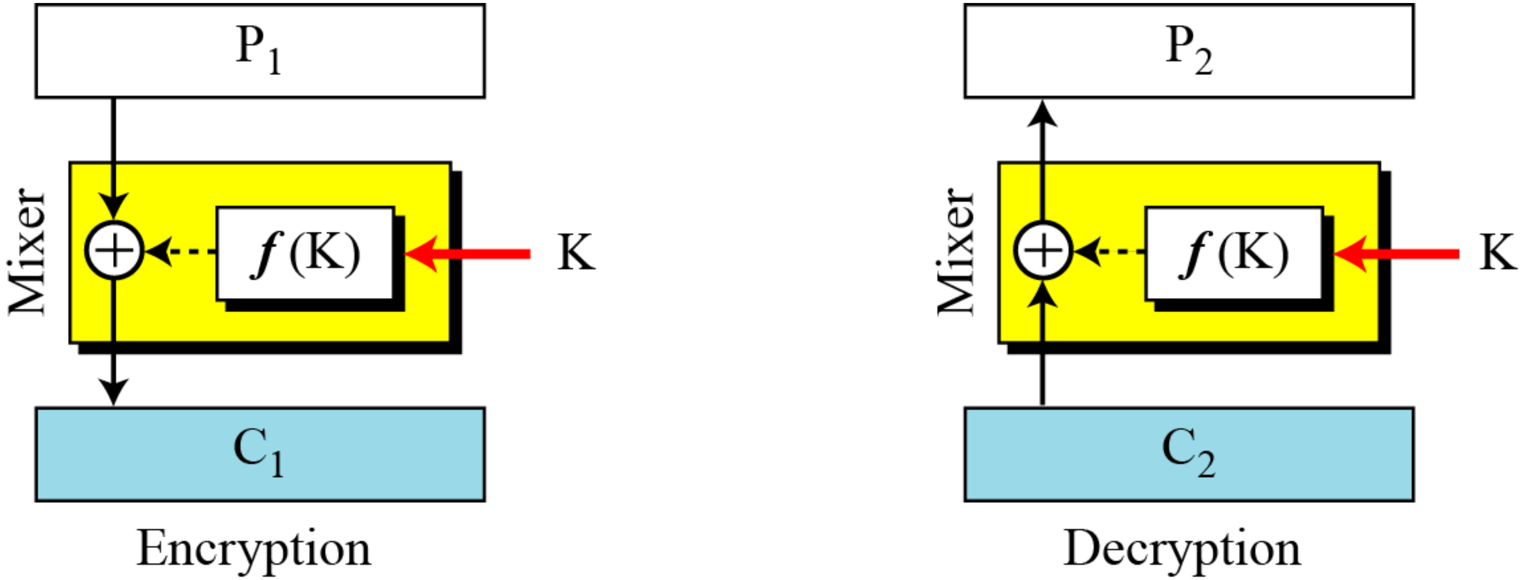
Feistel Ciphers

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.

Continued

The first thought in Feistel cipher design

Non-invertible elements cancels out when X-ored



Diffusion hides the relationship between the ciphertext and the plaintext.

Two algorithms are inverses of each other: If C2=C1 then P2=P1

Encryption: $C_1 = P_1 \oplus f(K)$

Decryption: $P_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = P_1 \oplus f(K) \oplus f(K) = P_1 \oplus (00\ldots0) = P_1$

The mixer in the Feistel design is self-invertible.

Continued

Example

This is a trivial example. The plaintext and ciphertext are each 4 bits long and the key is 3 bits long. 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. Show the results of encryption and decryption if the original plaintext is 0111 and the key is 101.

Solution

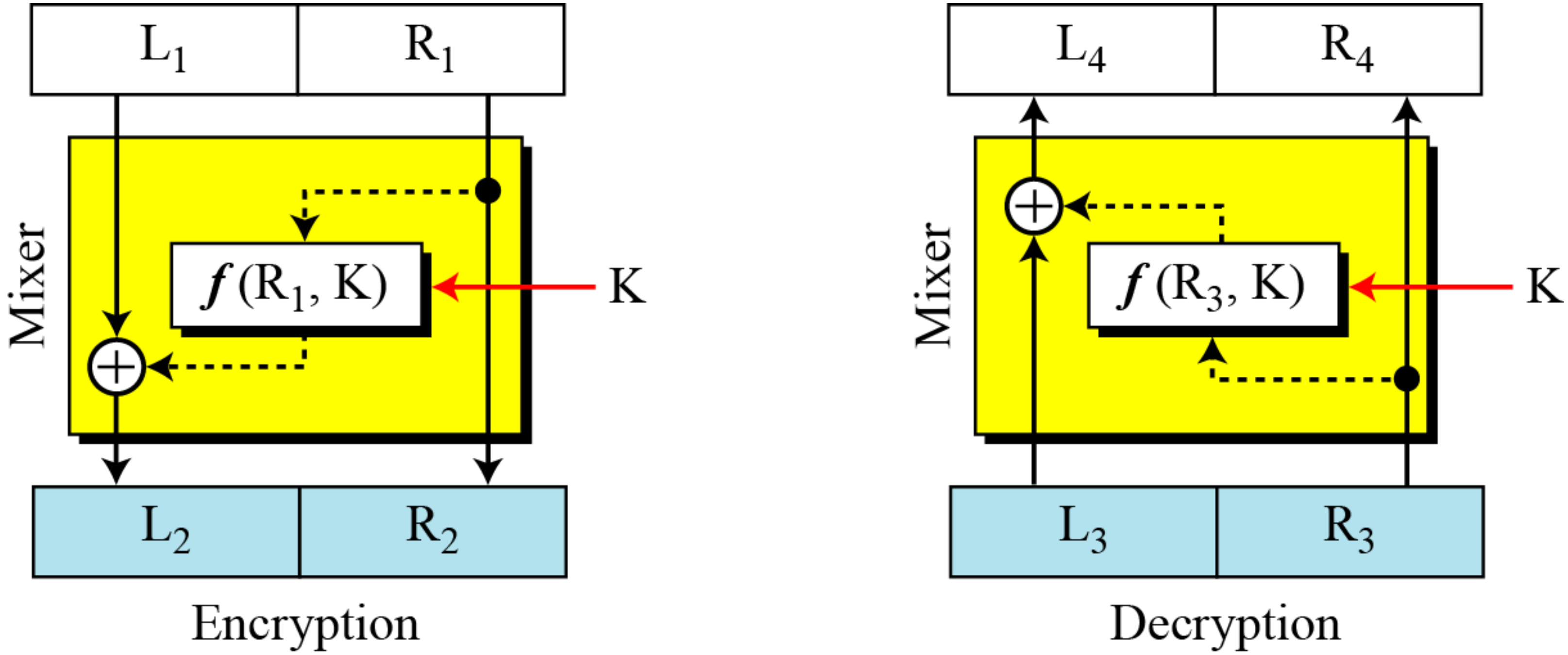
The function extracts the first and third bits to get 11 in binary or 3 in decimal. The result of squaring is 9, which is 1001 in binary.

$$\textbf{Encryption: } C = P \oplus f\,(K) = 0111 \oplus 1001 = 1110$$

$$\textbf{Decryption: } P = C \oplus f\,(K) = 1110 \oplus 1001 = 0111$$

Continued

The improvement in Feistel cipher design

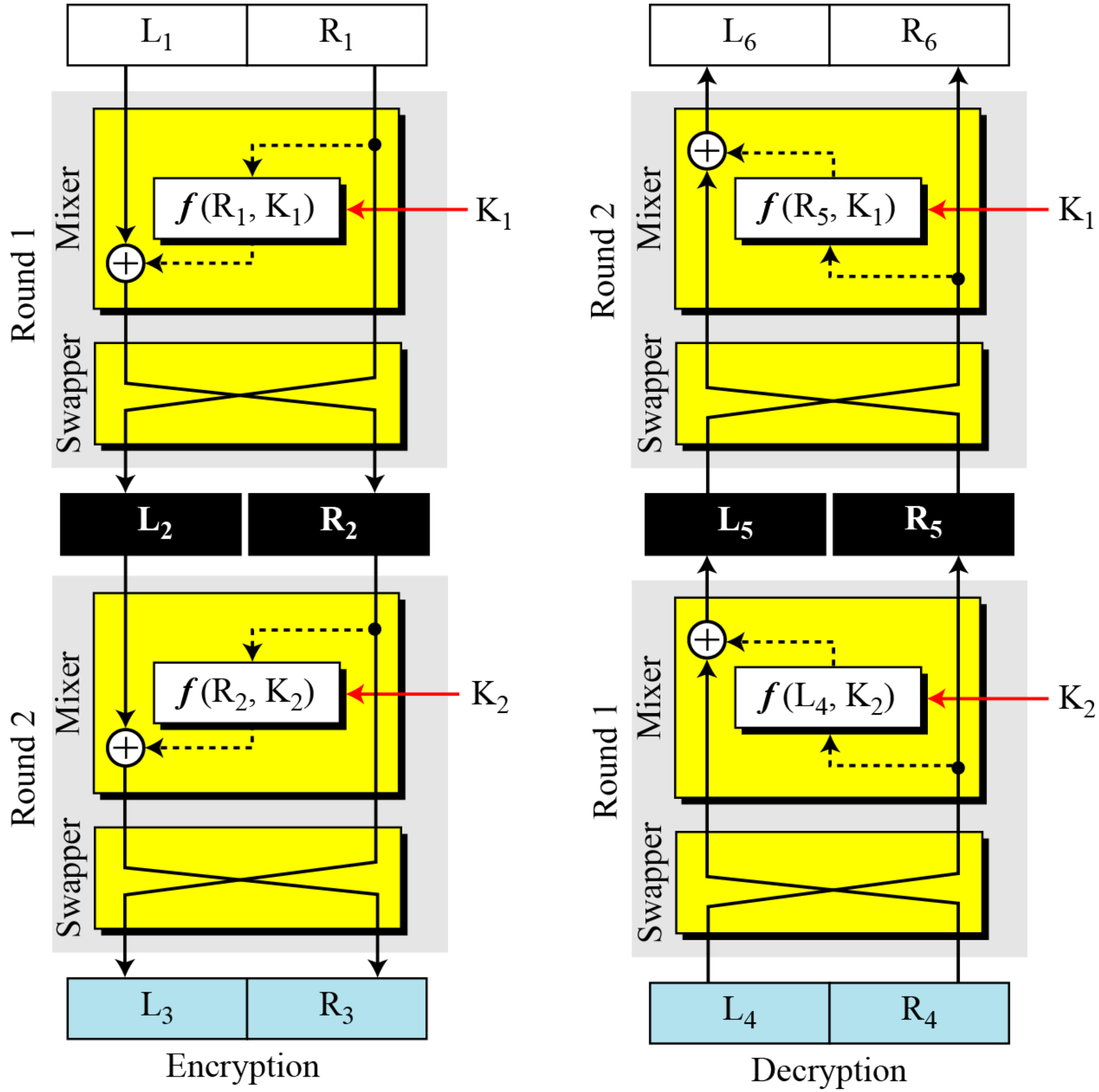


Two algorithms are inverses of each other: If L3=L2 and R3=R2

$$R_4 = R_3 = R_2 = R_1$$

$$L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_2, K) = L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1$$

The final design of Feistel cipher *Continued*



Final design
Flaw: no change in Right half.
Inc: rounds
Add: swapper

Two algorithms are inverses of each other: If L6=L1 and R6=R1 assuming that
L4=L3 and R4=R3

$$L_5 = R_4 \oplus f(L_4, K_2) = R_3 \oplus f(R_2, K_2) = L_2 \oplus f(R_2, K_2) \oplus f(R_2, K_2) = L_2$$
$$R_5 = L_4 = L_3 = R_2$$

Then it is easy to prove that the holds for two plaintext blocks

$$L_6 = R_5 \oplus f(L_5, K_1) = R_2 \oplus f(L_2, K_1) = L_1 \oplus f(R_1, K_1) \oplus f(R_1, K_1) = L_1$$
$$R_6 = L_5 = L_2 = R_1$$

Non-Feistel Ciphers

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

Feistel Structure for Block Ciphers

- Feistel proposed applying two or more simple ciphers in sequence so final result is cryptographically stronger than component ciphers
- n-bit block length; k-bit key length; 2^k transformations
- Feistel cipher alternates: substitutions, transpositions (permutations)
- Applies concepts of diffusion and confusion
- Applied in many ciphers today

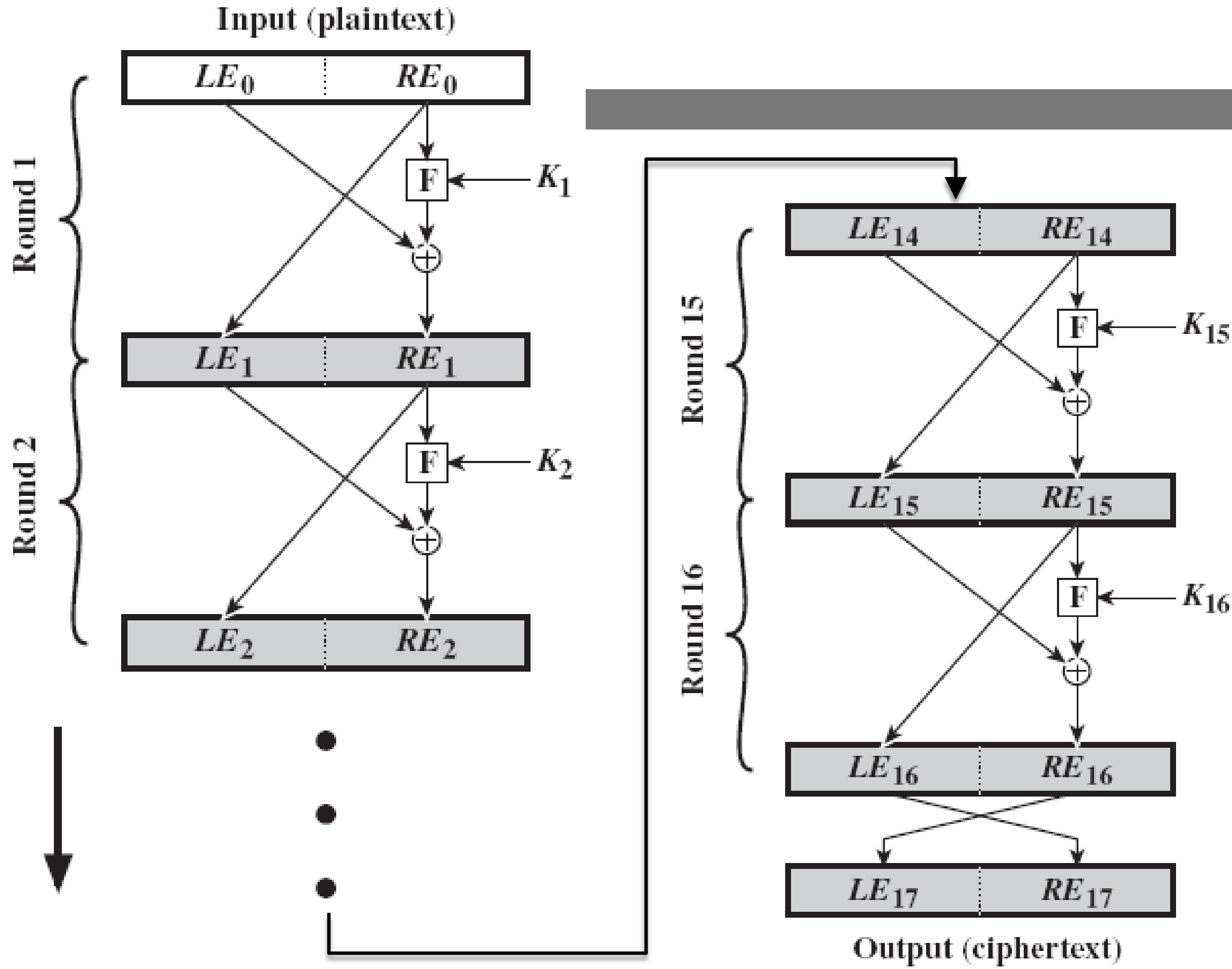
Feistel Cipher Structure

- Horst Feistel devised the **Feistel cipher**
  - based on concept of **invertible product cipher**
- Partitions input block into two halves
  - Subkeys (or round keys) generated from key
  - Round function, F, applied to right half $F(RE_i, K_{i+1})$
  - Apply substitution on left half using XOR
  - Apply permutation: interchange to halves
- Implements Shannon's S-P net concept

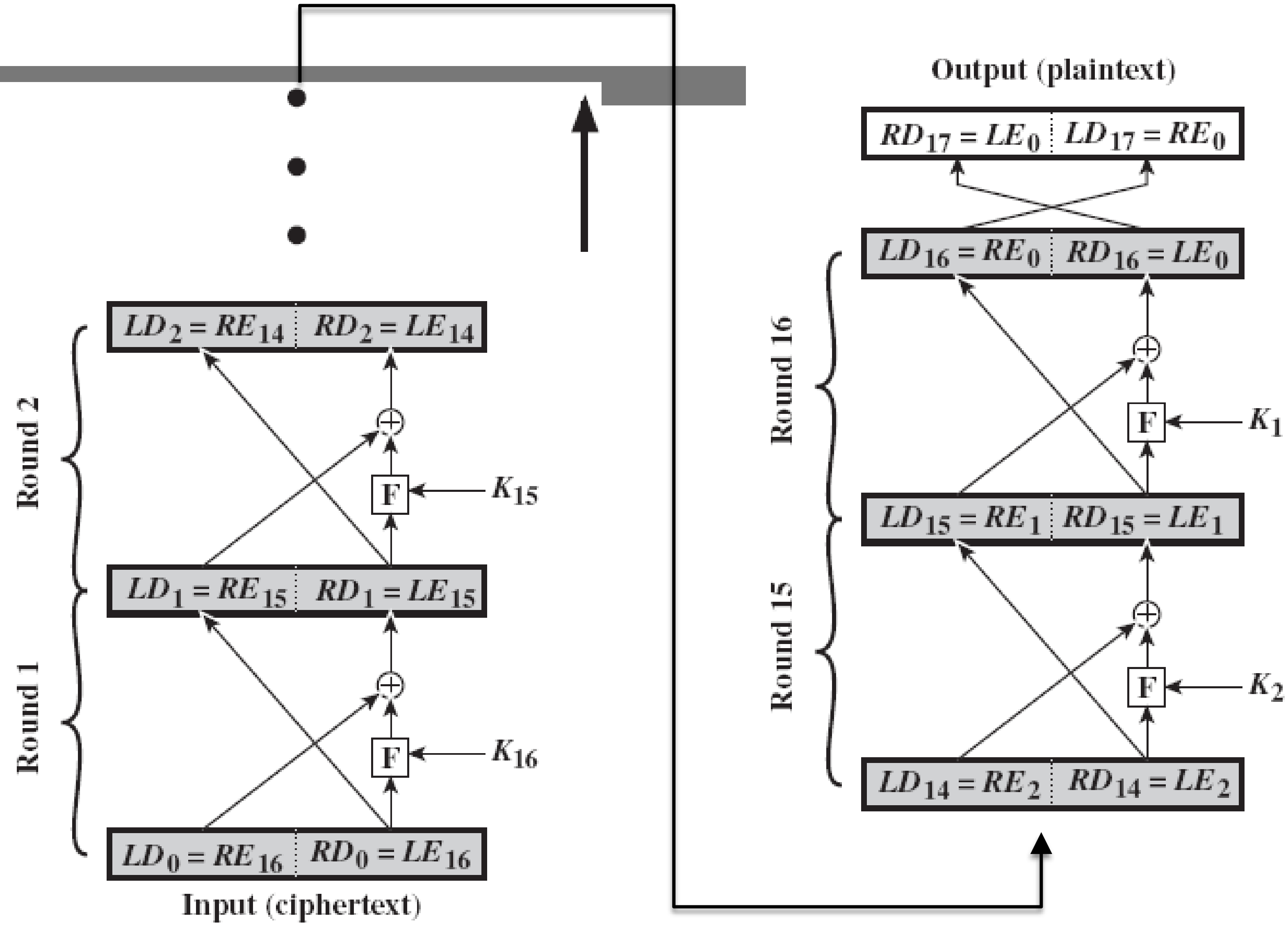
Using the Feistel Structure

- Exact implementation depends on various design features
  - Block size, e.g. 64, 128 bits: larger values leads to more diffusion
  - Key size, e.g. 128 bits: larger values leads to more confusion, resistance against brute force
  - Number of rounds, e.g. 16 rounds
  - Subkey generation algorithm: should be complex
  - Round function F: should be complex
- Other factors include fast encryption in software and ease of analysis
- Trade-off: security vs. performance

Feistel Cipher Structure Encryption



Feistel Cipher Structure Decryption



General Formula for Encryption/Decryption

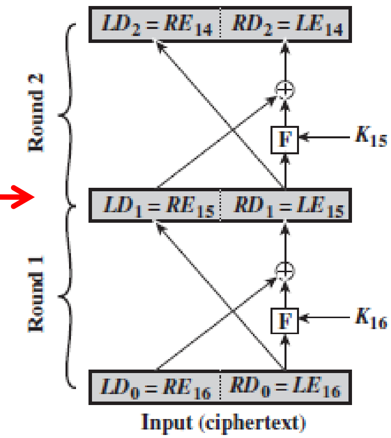
- For the ith iteration of the encryption algorithm

$$
\begin{aligned}
LE_i &= RE_{i-1} \\
RE_i &= LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, K_i)
\end{aligned}
$$

- Rearranging terms gives the decryption:

$$
\begin{aligned}
RE_{i-1} &= LE_i \\
LE_{i-1} &= RE_i \oplus \mathrm{F}(RE_{i-1}, K_i) = RE_i \oplus \mathrm{F}(LE_i, K_i)
\end{aligned}
$$

Relation between output and input



- Show that the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth ro encryption process.

$$LE_{16} = RE_{15}$$

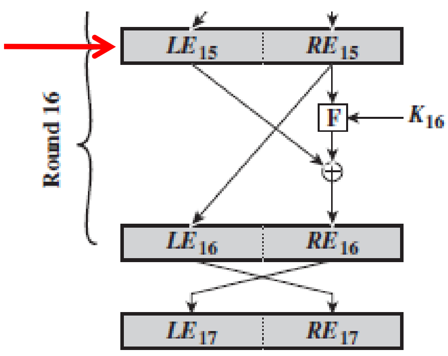
$$RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$$

- consider the encryption

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

- decryption side

$$\begin{aligned} RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$



- Thus, we have $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$
- Therefore, the output of the first round of the decryption process is $RE_{15} \| LE_{15}$, which is the 32-bit swap of the input to the sixteenth round of the encryption

Feistel Cipher Design Elements Discussions

- Block size
  - Larger block sizes mean greater security
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speed
- Number of rounds
  - a single round offers inadequate security but that multiple rounds offer increasing security

Feistel Cipher Design Elements Discussions

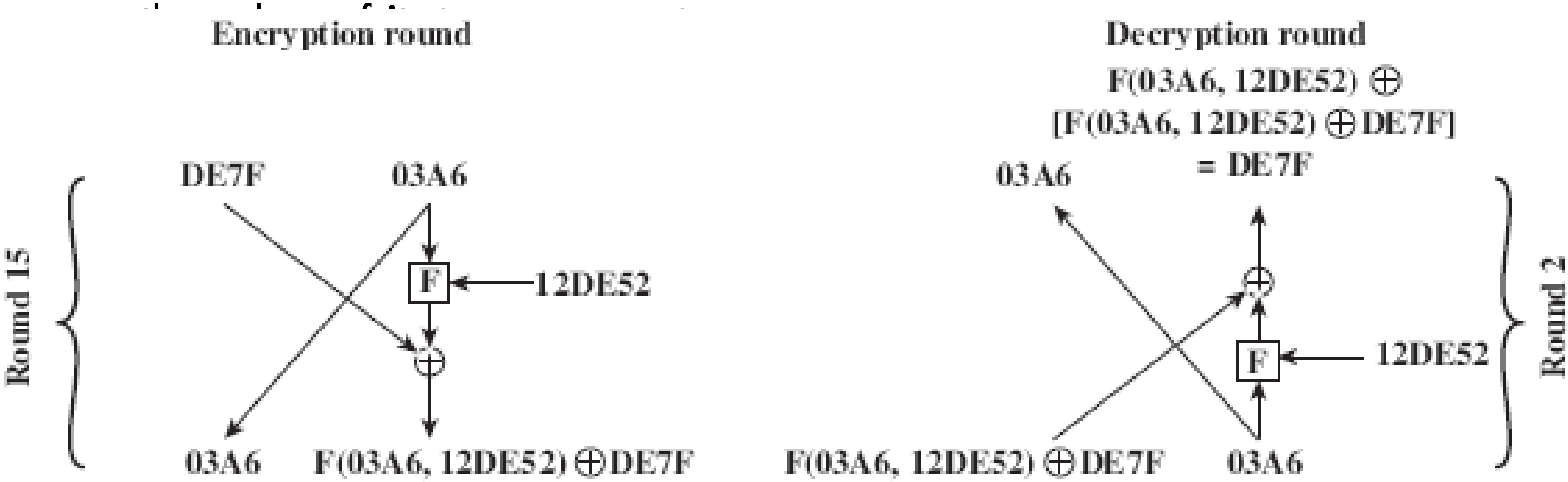
- Subkey generation algorithm
  - Greater complexity leads to greater difficulty of cryptanalysis
- Round function
  - Same as subkey gen.

Feistel Cipher Design Elements Discussions

- Fast software en/decryption
  - the speed of execution of the algorithm becomes a concern
- Ease of analysis
  - if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

Dependency on function F

- The derivation does not require that F be a reversible function.
- For example, F produces a constant output (e.g., all ones) regardless of

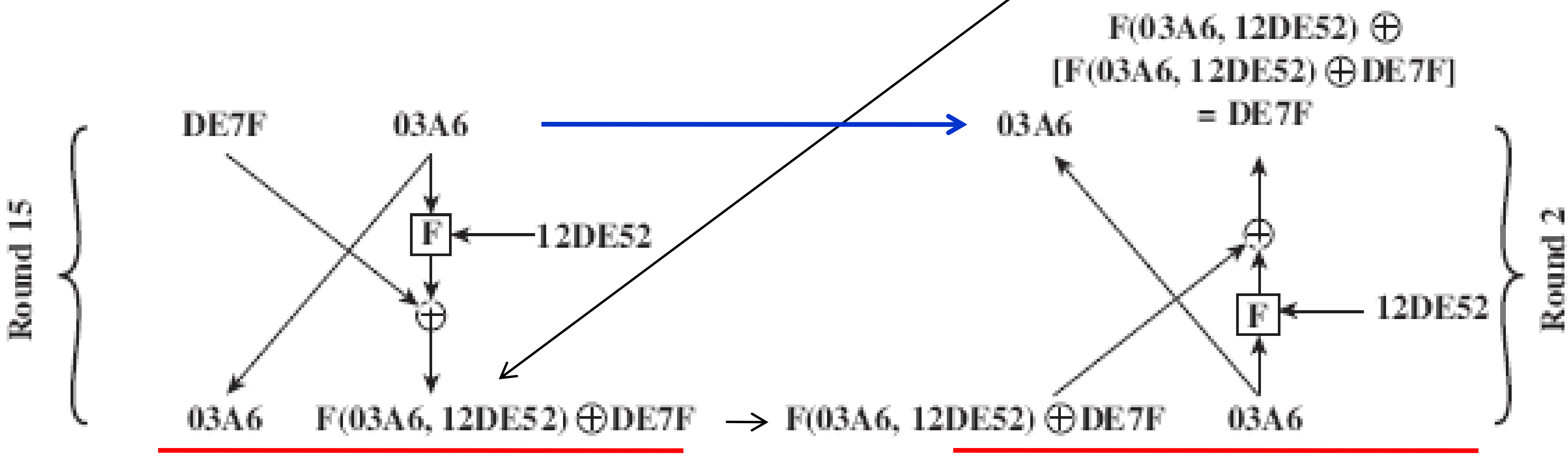


- 15^{th} round of encryption corresponds to 2^{nd} round of decryption
- Block size is 32 bits (two 16-bit halves) and key size is 24 bits

Dependency on function F

the key size is 24 bits. Suppose that at the end of encryption round fourteen, the value of the intermediate block (in hexadecimal) is DE7F03A6. Then LE_{14} = DE7F and RE_{14} = 03A6. Also assume that the value of K_{15} is 12DE52. After round 15, we have LE_{15} = 03A6 and RE_{15} = F(03A6, 12DE52) ⊕ DE7F.

Now let's look at the decryption. We assume that $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$, as shown in Figure 3.3, and we want to demonstrate that $LD_2 = RE_{14}$ and $RD_2 = LE_{14}$. So, we start with LD_1 = F(03A6, 12DE52) ⊕ DE7F and RD_1 = 03A6. Then, from Figure 3.3, LD_2 = 03A6 = RE_{14} and RD_2 = F(03A6, 12DE52) ⊕ [F(03A6, 12DE52) ⊕ DE7F]= DE7F = LE_{14}.



Symmetric Block Cipher Algorithms

- DES (Data Encryption Standard)
- 3DES (Triple DES)
- AES (Advanced Encryption Standard)

Data Encryption Standard

- Symmetric block cipher
  - 56-bit key, 64-bit input block, 64-bit output block
- One of most used encryption systems in world
  - Developed in 1977 by NBS/NIST
  - Designed by IBM (Lucifer) with input from NSA
  - Principles used in other ciphers, e.g. 3DES, IDEA
- Simplied DES (S-DES)
  - Cipher using principles of DES
  - Developed for education (not real world use)

Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has considerable controversy over its security

DES History

- IBM developed Lucifer cipher
  - by team led by Feistel in late 60's
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

Triple DES

- Triple DES (3DES) was first standardized for use in financial applications in ANSI standard X9.17 in 1985.
- 3DES was incorporated as part of the Data Encryption Standard in 1999 with the publication of FIPS 46-3.

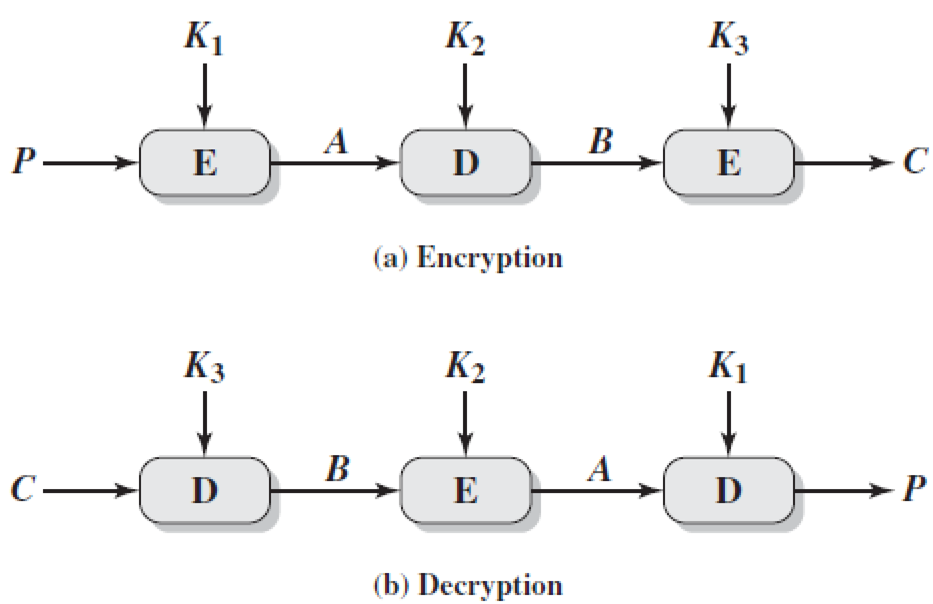


Figure 2.4 Triple DES

Triple DES

3DES uses three keys and three executions of the DES algorithm.The function follows an encrypt-decrypt-encrypt (EDE) sequence

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

where

$$\begin{aligned} C &= \text{ciphertext} \\ P &= \text{plaintext} \\ \mathrm{E}[K, X] &= \text{encryption of } X \text{ using key } K \\ \mathrm{D}[K, Y] &= \text{decryption of } Y \text{ using key } K \end{aligned}$$

$$P = \mathrm{D}(K_1, \mathrm{E}(K_2, \mathrm{D}(K_3, C)))$$

There is no cryptographic significance to the use of decryption for the second stage of 3DES encryption.

Triple DES comments

- 3DES is the FIPS approved symmetric encryption algorithm of choice.
- The original DES, which uses a single 56-bit key, is permitted under the standard for legacy systems only. New procurements should support 3DES.
- Government organizations with legacy DES systems are encouraged to transition to 3DES.
- It is anticipated that 3DES and the Advanced Encryption Standard (AES) will coexist as FIPS-approved algorithms, allowing for a gradual transition to AES.

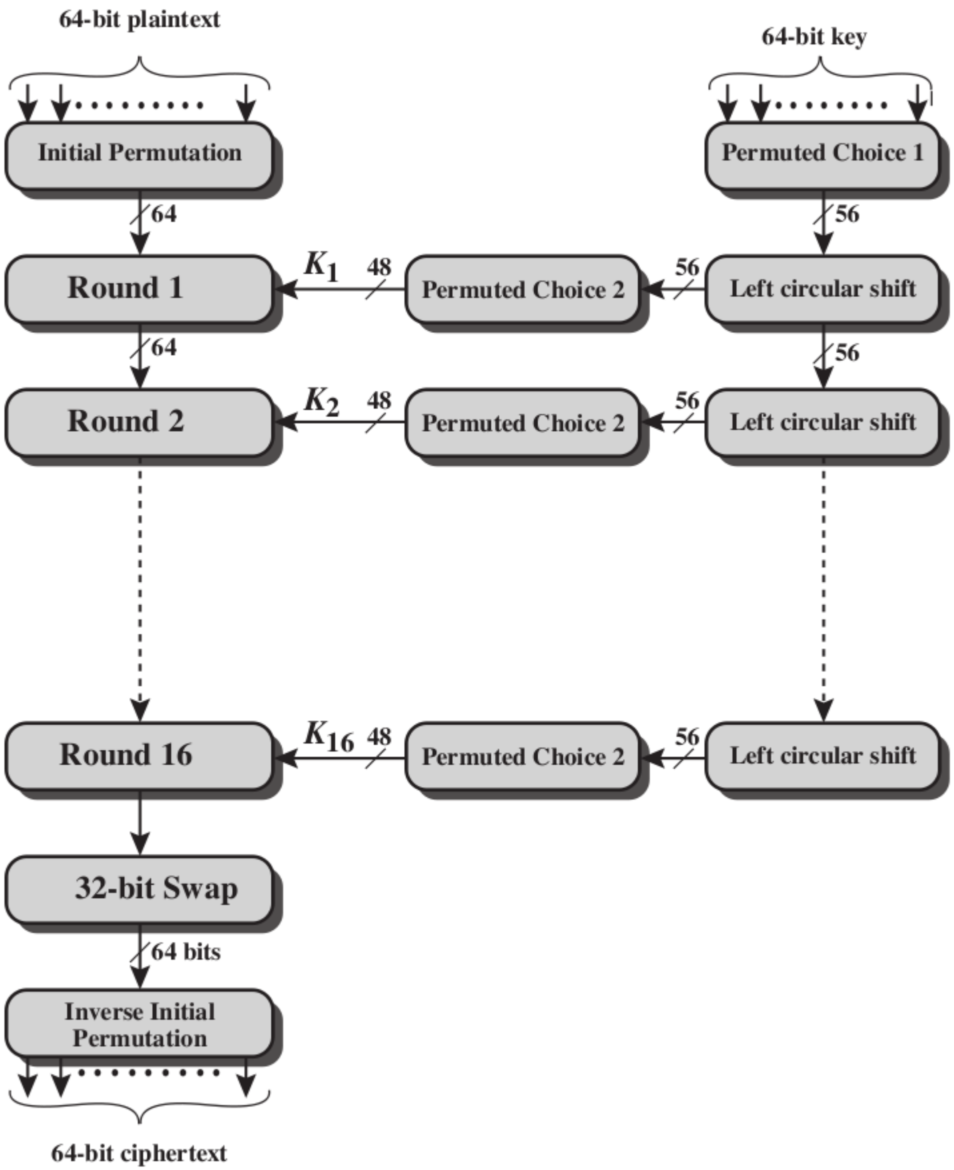
Triple DES comments

- **FIPS: Federal Information Processing Standards**
- The purpose of FIPS is to ensure that all federal government and agencies adhere to the same guidelines regarding security and communication.

DES

- For DEA, data are encrypted in 64-bit blocks using a
- 56-bit key.
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output.
- The same steps, with the same key, are used to reverse the encryption.
- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

General DES Encryption Algorithm



DES Encryption

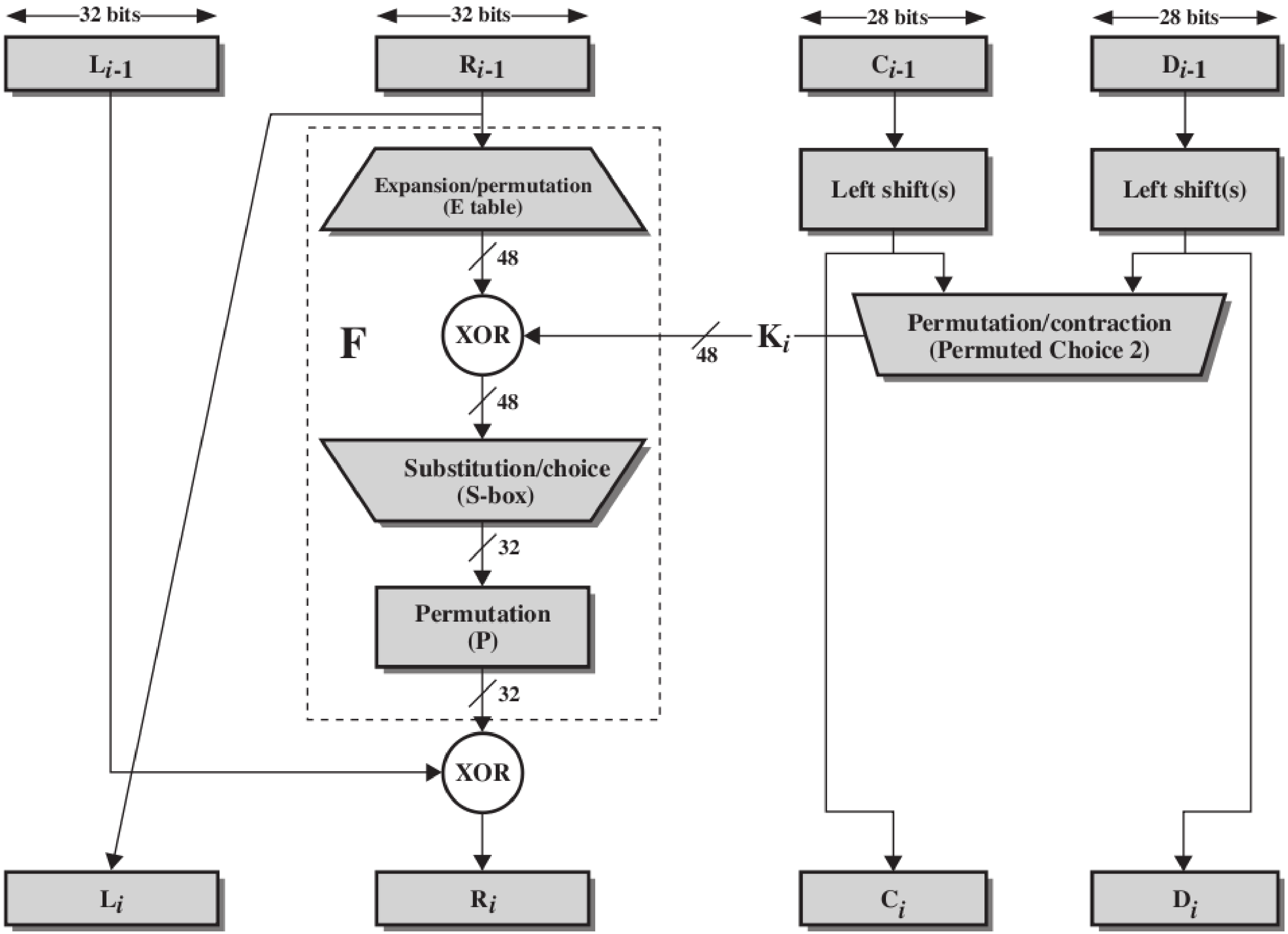
- As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key
- the processing of the plaintext proceeds in three phases.

1. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input.*
2. This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions.
3. The left and right halves of the output are swapped to produce the *preoutput.*
4. Finally, the preoutput is passed through a permutation [IP^{-1}] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.

Key generation

- Initially, the key is passed through a permutation function.
- Then, for each of the sixteen rounds, a *subkey* *(K_i) is produced by the* combination *of a left circular* shift and a permutation.

Single Round of DES Algorithm



A DES Decryption

- 1. As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

- 2. Additionally, the initial and final permutations are reversed.

Permutation Tables for DES

Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Final Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Input bit 58 goes to output bit 1

Input bit 50 goes to output bit 2, …

Even bits to LH half, odd bits to RH half

Quite regular in structure (easy in h/w)

Permutation Tables for DES

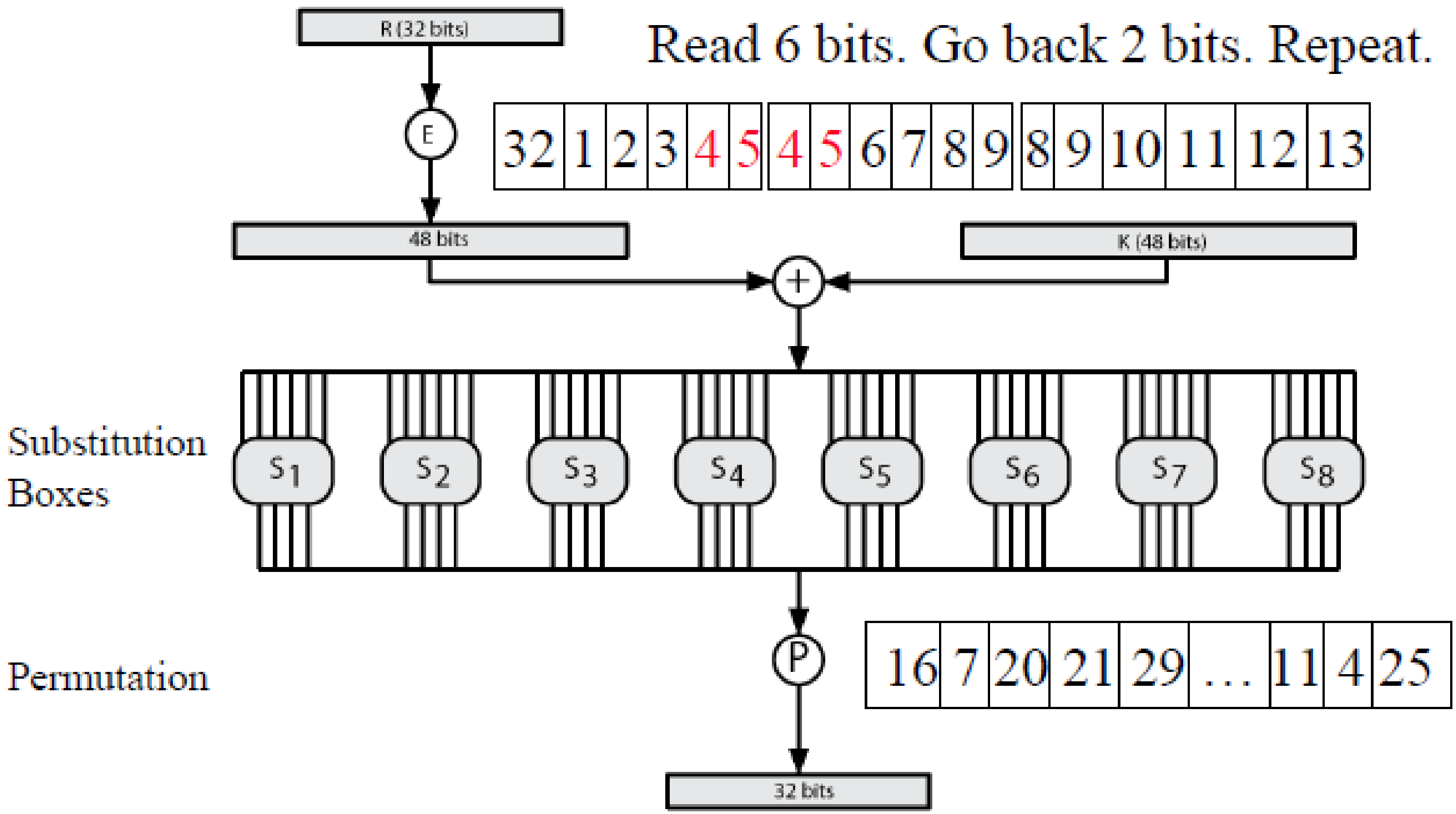
(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Calculation of F(R,K)



Substitution boxes

- Map 6 to 4 bits
- Outer bits 1 & 6 (**row** bits) select one row of 4
- Inner bits 2-5 (**column** bits) are substituted
- Example:

Input bits 1 and 6 / Input bits 2 thru 5

↓	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Definition of DES S-Boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Definition of DES S-Boxes

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES Key Schedule Calculation

- Permutation PC1 divides 56-bits in two 28-bit halves
- Rotate **each half** separately either 1 or 2 places depending on the **key rotation schedule** K
- Select 24-bits from each half & permute them by PC2

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

The Avalanche Effect

- Aim: small change in key (or plaintext) produces large change in ciphertext
- Avalanche effect is present in DES (good for security)
- Following examples show the number of bits that change in output when two dierent inputs are used, differing by 1 bit
  - Plaintext 1: 02468aceeca86420
  - Plaintext 2: 12468aceeca86420
  - Ciphertext difference: 32 bits
    - Key 1: 0f1571c947d9e859
    - Key 2: 1f1571c947d9e859
    - Ciphertext difference: 307

shows the result when the fourth bit of the plaintext is changed, so that the plaintext is **12468aceeca86420.**

DES example

- For this example, the plaintext is a hexadecimal palindrome. The plaintext, key, and resulting ciphertext are as follows:

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Results

Table 3.2 DES Example

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Note: DES subkeys are shown as eight 6-bit values in hex format

shows the progression of the algorithm.

Avalanche Effect in DES: Change in Plaintext

The second column of the table shows the intermediate 64-bit values at the end of each round for the two plaintexts.

The third column shows the number of bits that differ between the two intermediate values.

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b	32

Avalanche Eect in DES: Change in Key

shows a similar test using the original plaintext of with two keys that differ in only the fourth bit position:

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Concerns of DES

Key size and the nature of the algorithm

- Although 64 bit initial key, only 56 bits used in encryption (other 8 for parity check)
- $2^{56} = 7.2 * 10^{16}$
  - 1977: estimated cost $US20m to build machine to break in 10 hours
  - 1998: EFF built machine for $US250k to break in 3 days
  - Today: 56 bits considered too short to withstand brute force attack
- Recent offerings confirm this. Both Intel and AMD now offer hardware-based instructions to accelerate the use of AES. Test run on a contemporary multicore Intel machine resulted in an encryption rate of about half a billion encryptions per second.
- 3DES uses 128-bit keys

Table 3.5 Average Time Required for Exhaustive Key Search

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

DES Design Controversy (Concerns)

- although DES standard is public, considerable controversy over design (two concerns)
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
  - especially in financial applications
  - still standardised for legacy application use

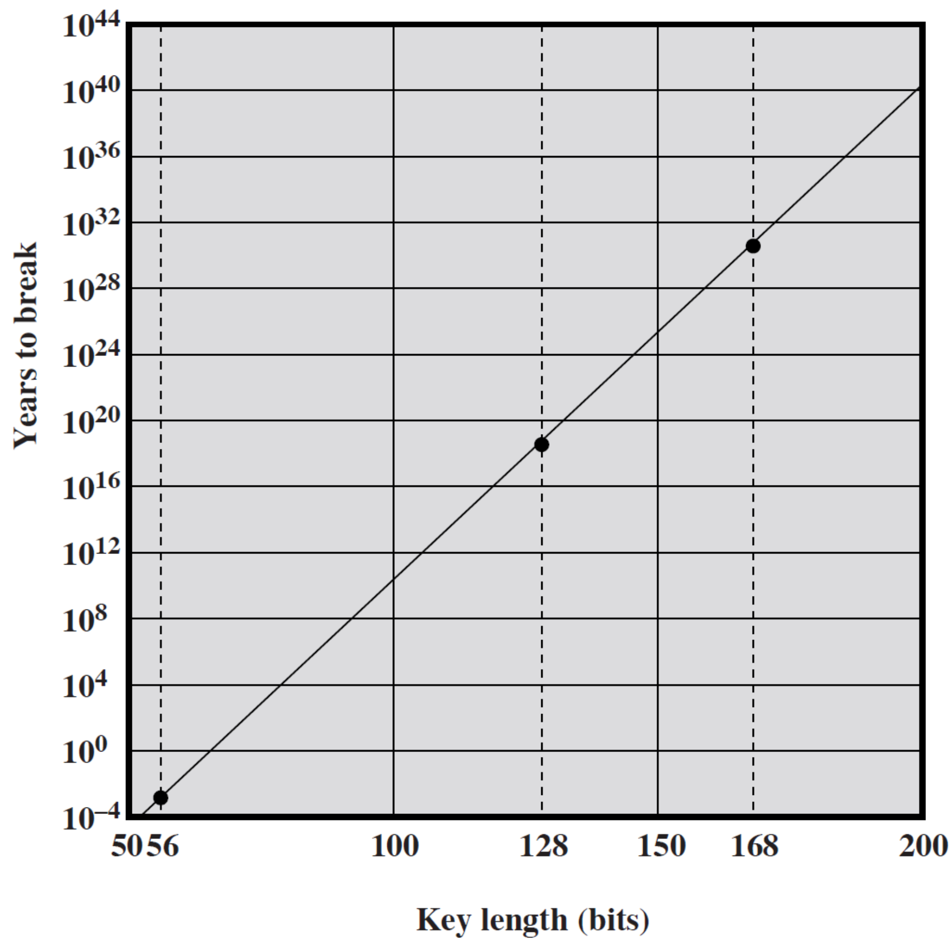
Concern of DES

- **The Nature of the DES Algorithm**
- Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm
- Because the design criteria for these S-boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes.

Time to Break a DES Code (assuming 10^6 decryptions/µs)

Using Electronic Frontier Foundation (EFF) DES cracker

Appx 10 hrs. for DES



Attacks on DES

Timing Attacks

- Information gained about key/plaintext by observing how long implementation takes to decrypt
- No known useful attacks on DES

Differential Cryptanalysis

- Observe how pairs of plaintext blocks evolve
- Break DES in 247 encryptions (compared to 255); but require 247 chosen plaintexts

Linear Cryptanalysis

- Find linear approximations of the transformations
- Break DES using 243 known plaintexts

Differential Cryptanalysis

- Chosen Plaintext attack: Get ciphertext for a given plaintext
- Get the (ΔX, ΔY) pairs, where ΔX is the difference in plaintext and ΔY is the difference in ciphertext
- Some (ΔX, ΔY) pairs are more likely than others, if those pairs are found, some key values are more likely so you can reduce the amount of brute force search
- Straightforward brute force attack on DES requires 2^{55} plaintexts
- Using differential cryptanalysis, DES can be broken with 2^{47} plaintexts.
  But finding appropriate plaintexts takes some trials and so the total amount of effort is $2^{55.1}$ which is more than straight forward brute force attack
  $\Rightarrow$ DES is resistant to differential cryptanalysis

Linear Cryptanalysis

- Bits in plaintext, ciphertext, and keys may have a linear relationship. For example:

$$P1 \oplus P2 \oplus C3 = K2 \oplus K5$$

- In a good cipher, the relationship should hold w probability ½. If any relationship has probability 1, the cipher is easy to break. If any relationship has probability 0, the cipher is easy to break.
- Bias = |Probability of linear relationship – 0.5|
- Find the linear approximation with the highest bias
  $\Rightarrow$ Helps reduce the brute force search effort.
- This method can be used to find the DES key given 2^{43} plaintexts.

Choosing F

- Non-linerity in rough terms, the more difficult it is to approximate F by a set of linear equations, the more nonlinear F is.

- A more stringent version of this is the **strict avalanche criterion (SAC),** which states that any output bit *j of an S-box* should change with probability 1/2 when any single input bit *i is inverted* for all *i, j*.

- Another criterion proposed is the **bit independence criterion** (BIC), **which states that output bits *j and k should change independently when any*** single input bit *i is inverted for all i, j, and k*.

DES Algorithm Design

DES was designed in private; questions about the motivation

of the design

- S-Boxes provide non-linearity: important part of DES, generally considered to be secure
- S-Boxes provide increased confusion
- Permutation P chosen to increase diffusion

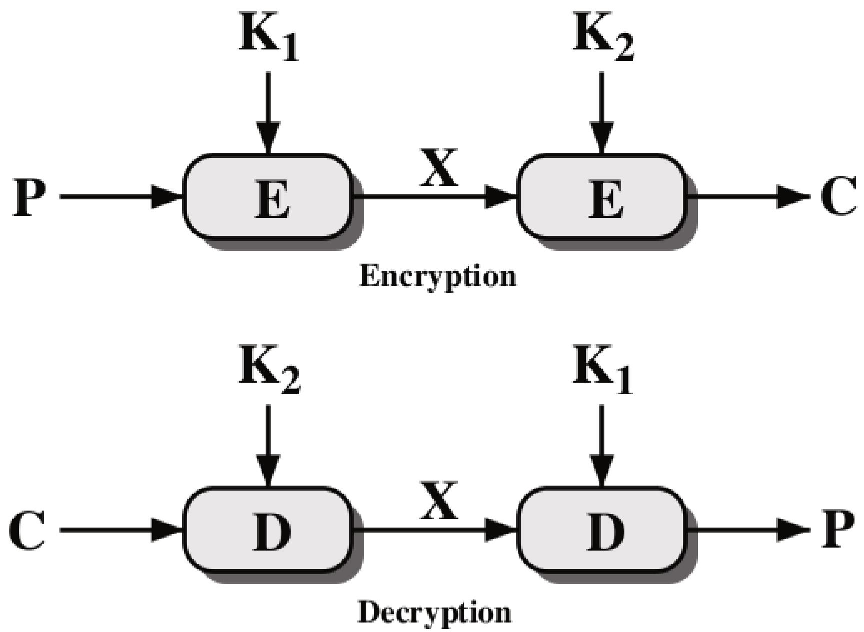
Multiple Encryption with DES

- DES is vulnerable to brute force attack
- Alternative block cipher that makes use of DES software/equipment/knowledge: encrypt multiple times with different keys

Options:

- 1. Double DES: not much better than single DES
- 2. Triple DES (3DES) with 2 keys: brute force 2^{112}
- 3. Triple DES with 3 keys: brute force 2^{168}

Double Encryption

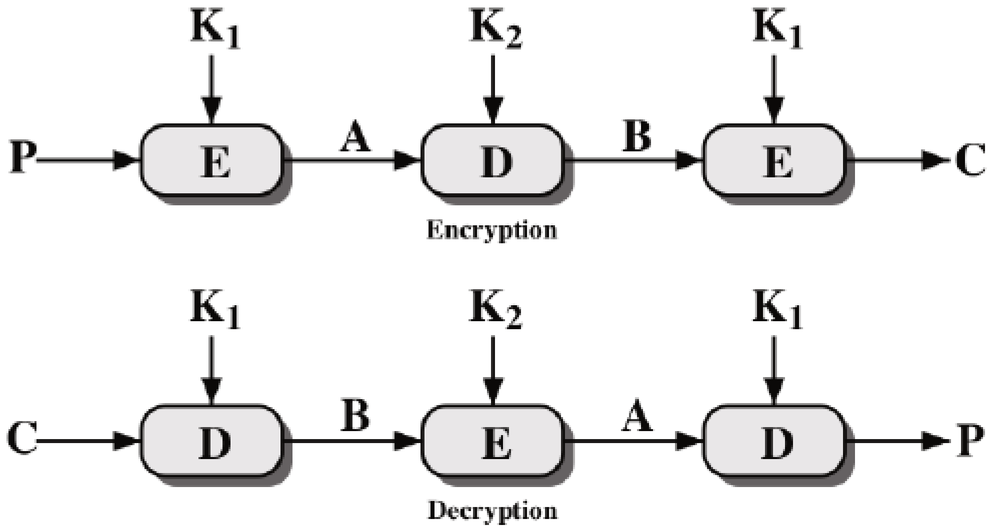


- For DES, 2 56-bit keys, meaning 112-bit key length
- Requires 2111 operations for brute force?
- Meet-in-the-middle attack makes it easier

Meet-in-the-Middle Attack

- Double DES Encryption: $C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$
- Say $X = \mathrm{E}(K_1, P) = \mathrm{D}(K_2, C)$
- Attacker knows two plaintext, ciphertext pairs (P_a, C_a) and (P_b, C_b)
  1. Encrypt P_a using all 2^{56} values of K_1 to get multiple values of X
  2. Store results in table and sort by X
  3. Decrypt C_a using all 2^{56} values of K_2
  4. As each decryption result produced, check against table
  5. If match, check current K_1, K_2 on C_b. If P_b obtained, then accept the keys
- With two known plaintext, ciphertext pairs, probability of successful attack is almost 1
- Encrypt/decrypt operations required: 2^{56} (twice as many as single DES)

Triple Encryption



- 2 keys, 112 bits
- 3 keys, 168 bits
- Why E-D-E? To be compatible with single DES:

$$C = \mathrm{E}(K_1, \mathrm{D}(K_1, \mathrm{E}(K_1, P))) = \mathrm{E}(K_1, P)$$

Other Symmetric Encryption Algorithms

- Blowfish (Schneier, 1993): 64 bit blocks/32–448 bit keys; Feistel structure
- Twofish (Schneier et al, 1998): 128/128, 192, 256; Feistel structure
- Serpent (Anderson et al, 1998): 128/128, 192, 256; Substitution-permutation network
- Camellia (Mitsubishi/NTT, 2000): 128/128, 192, 256; Feistel structure
- IDEA (Lai and Massey, 1991): 64/128
- CAST-128 (Adams and Tavares, 1996): 64/40–128; Feistel structure
- CAST-256 (Adams and Tavares, 1998): 128/up to 256; Feistel structure
- RC5 (Rivest, 1994): 32, 64 or 128/up to 2040; Feistel-like structure
- RC6 (Rivest et al, 1998): 128/128, 192, 256; Feistel structure

Cryptanalysis on Block Ciphers

Cipher	Method	Key space	Required resources: Time	Memory	Known data
DES	Brute force	2^{56}	2^{56}	-	-
3DES	MITM	2^{168}	2^{111}	2^{56}	2^{2}
3DES	Lucks	2^{168}	2^{113}	2^{88}	2^{32}
AES 128	Biclique	2^{128}	$2^{126.1}$	2^{8}	2^{88}
AES 256	Biclique	2^{256}	$2^{254.4}$	2^{8}	2^{40}

- Known data: chosen pairs of (plaintext, ciphertext)
- MITM: Meet-in-the-middle
- Lucks: S. Lucks, Attacking Triple Encryption, in *Fast Software Encryption*, Springer, 1998
- Biclique: Bogdanov, Khovratovich and Rechberger, Biclique Cryptanalysis of the Full AES, in *ASIACRYPT2011*, Springer, 2011

Multiple Encryption & DES

- clear a replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
  - prior to this alternative was to use multiple encryption with DES implementations
  - Triple-DES is the chosen form

Double-DES?

- could use 2 DES encrypts on each block
  - $C = E_{K2}(E_{K1}(P))$
- issue of reduction to single stage
- and have "meet-in-the-middle" attack
  - works whenever use a cipher twice
  - since $X = E_{K1}(P) = D_{K2}(C)$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - takes $O(2^{56})$ steps

Triple-DES with Two-Keys

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - nb encrypt & decrypt equivalent in security
  - if K1=K2 then can work with single DES
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - several proposed impractical attacks might become basis of future attacks

Triple-DES with Three-Keys

- although no practical attacks on two-key Triple-DES have some concerns
  - Two-key: key length = 56*2 = 112 bits
  - Three-key: key length = 56*3 = 168 bits
- can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- has been adopted by some Internet applications, eg PGP, S/MIME