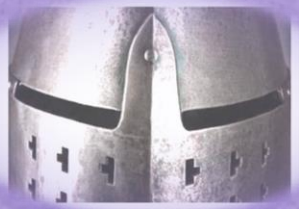


Chapter 4

Block Ciphers and the Data Encryption Standard

Stream Cipher

Encrypts a digital data stream one bit or one byte at a time

Examples:

- Autokeyed Vigenère cipher
- Vernam cipher

In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the keystream is as long as the plaintext bit stream

If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream

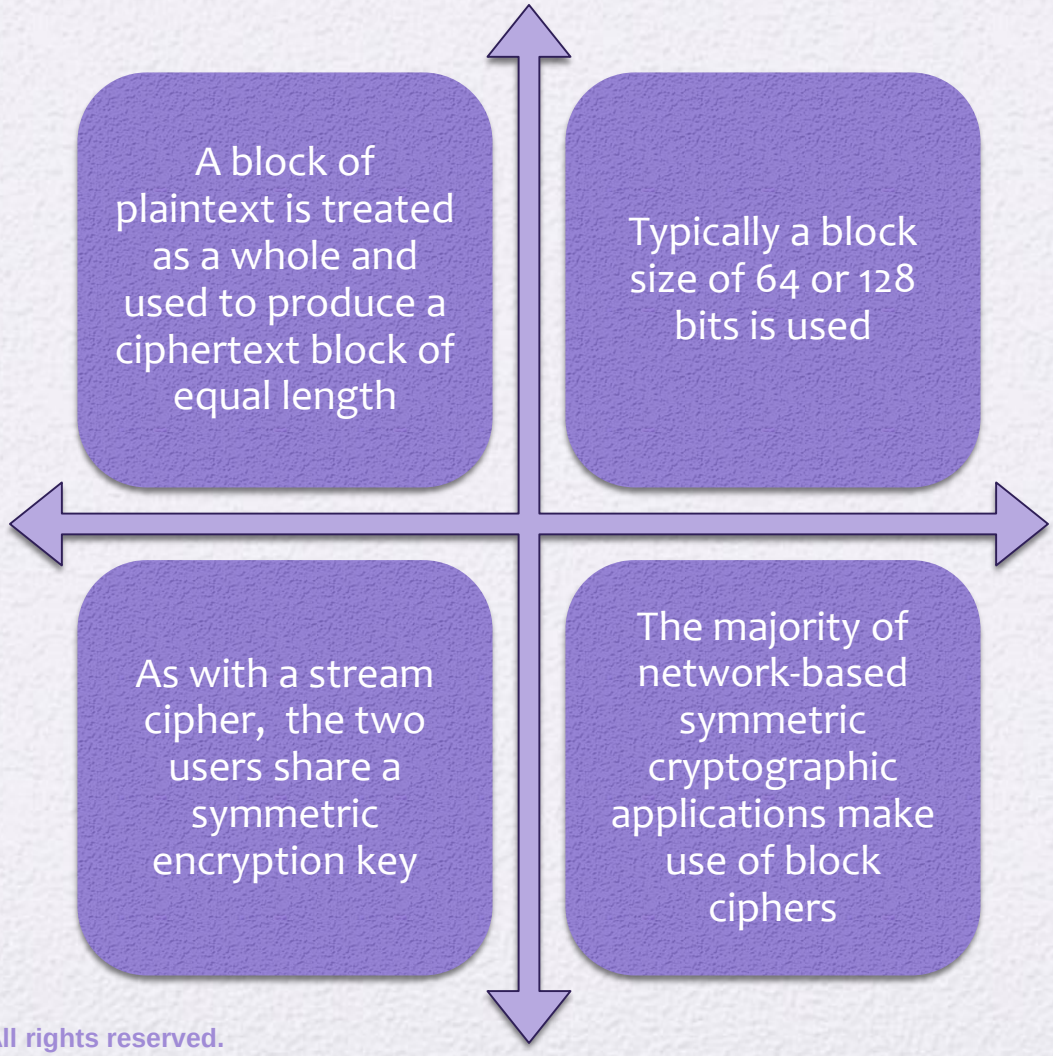
- **Keystream** must be provided to both users in advance via some independent and secure channel
- **This introduces insurmountable logistical problems if the intended data traffic is very large**

For **practical reasons the bit-stream generator** must be implemented as an algorithmic procedure so that the cryptographic bit stream can be produced by both users

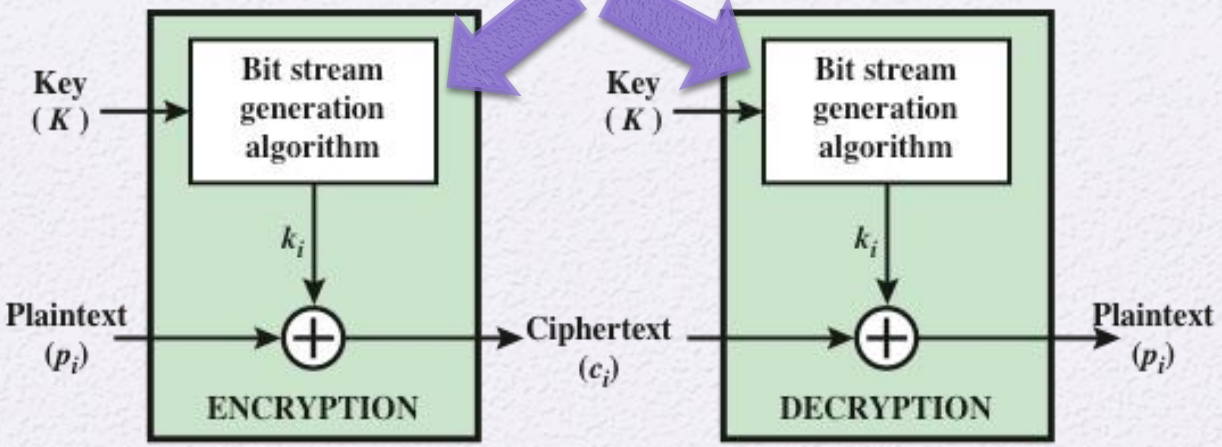
It must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream

The two users need only share the generating key and each can produce the keystream

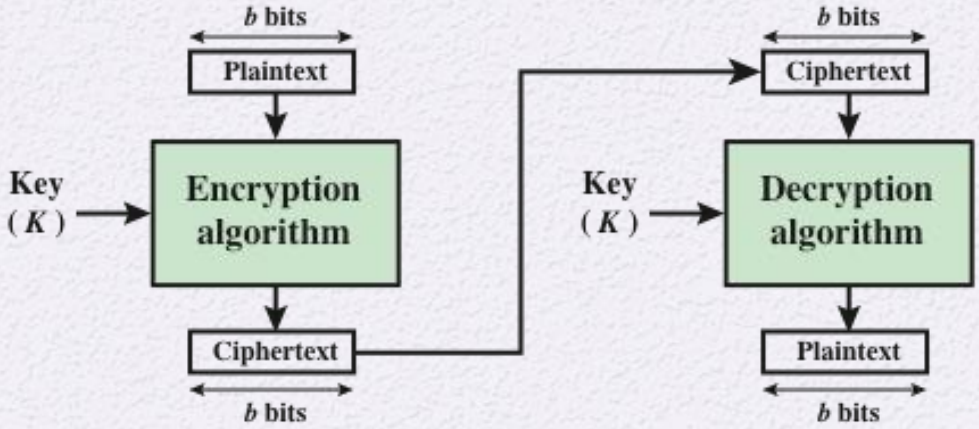
Block Cipher



pseudo-random generator



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Figure 4.1 Stream Cipher and Block Cipher

Diffusion and Confusion

- Terms introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system
  - Shannon's concern was to thwart cryptanalysis based on statistical analysis

Diffusion

- The statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext
- This is achieved by having each plaintext digit affect the value of many ciphertext digits

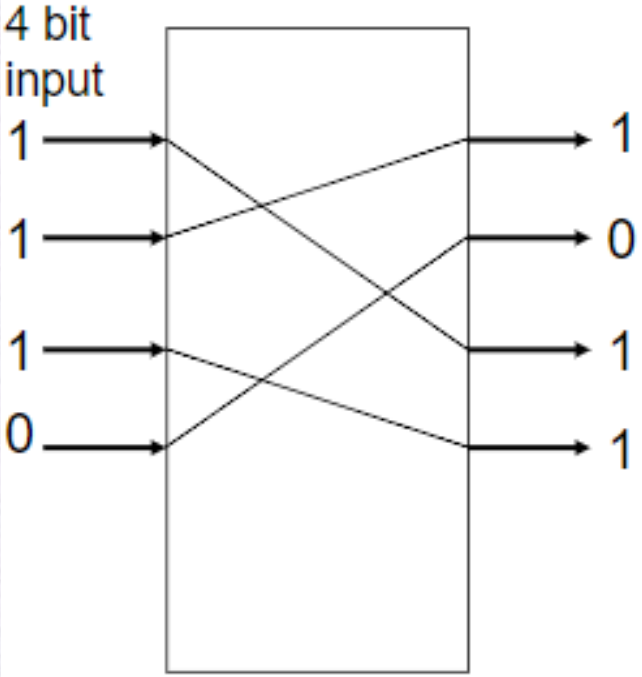
Confusion

- Seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible
- Even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key

Shannon's Building Blocks

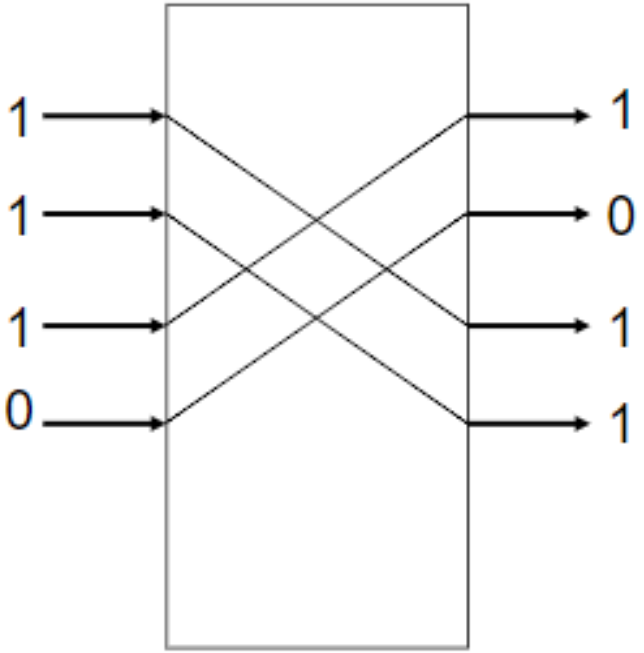
- Shannon proposed:
  - **S-box** (substitution): provides confusion of inputs
  - **P-Box** (permutation): provides diffusion across S-box inputs.
  - **N rounds** of S-boxes and P-boxes

P-box (permutation)



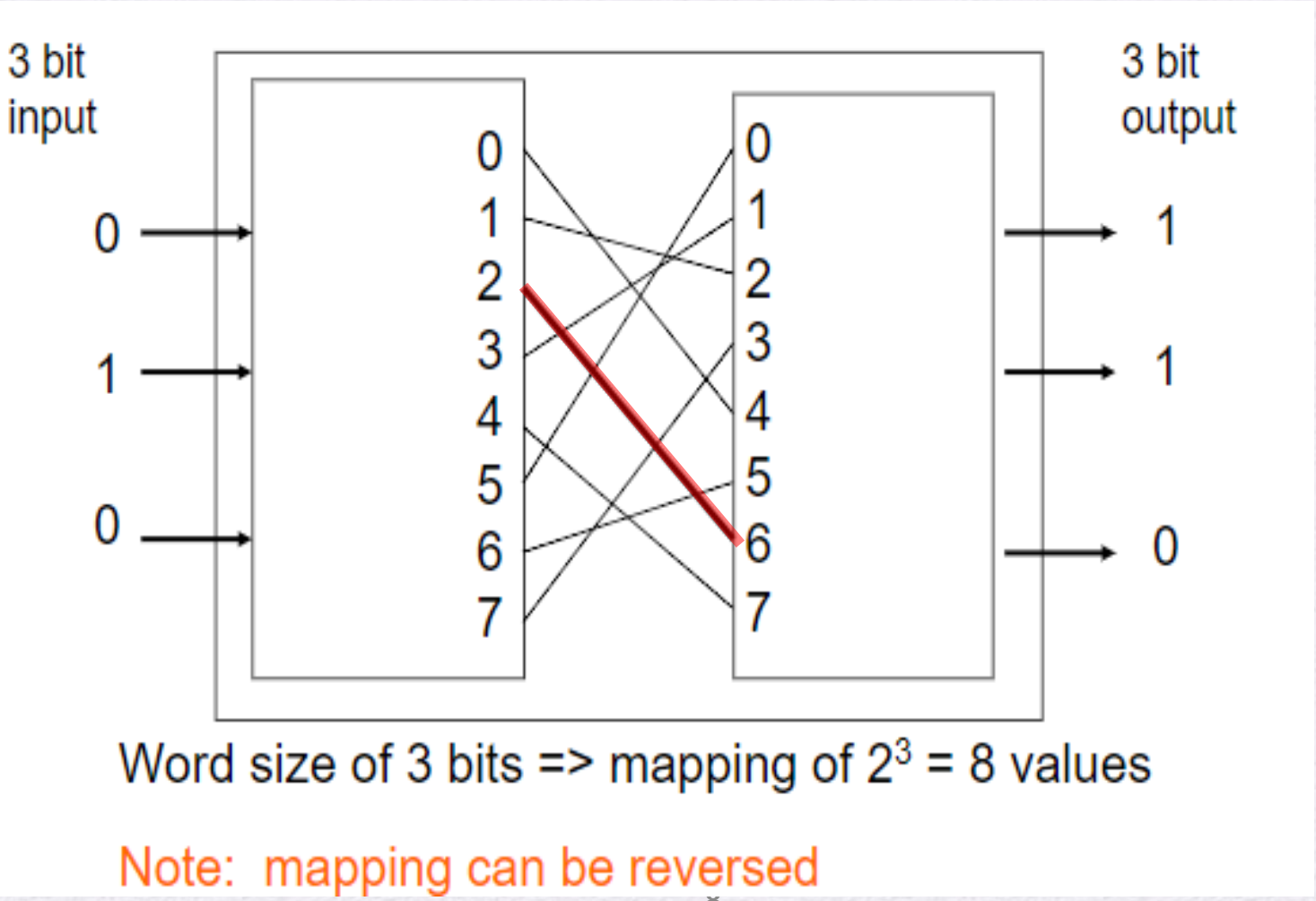
Example 1

Note: reversible

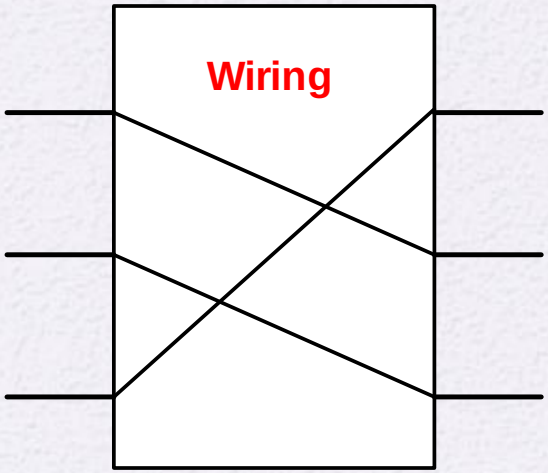


Example 2 - swap two halves of input

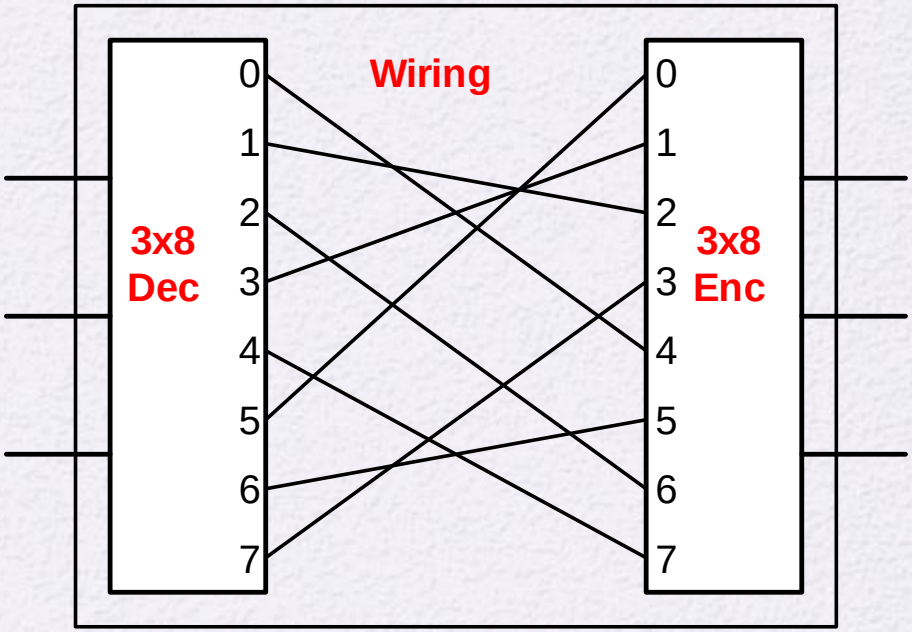
S-box (substitution): n=3



Hardwar Implementation of S-box and P-box



P-box



S-box

S-box implementation is more complex than P-Box.

(S-Box) = Decoder + (P-Box) + Encoder

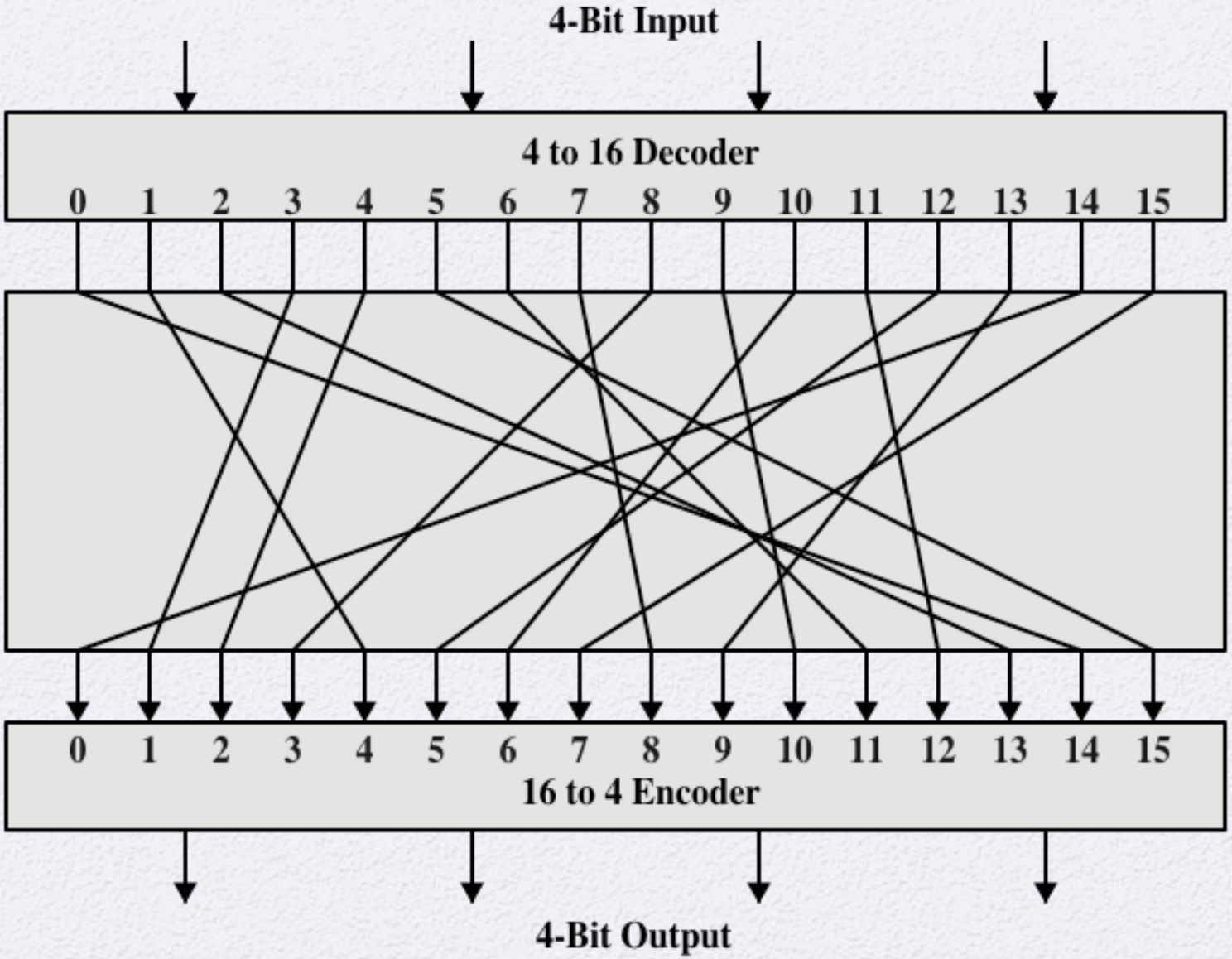


Figure 4.2 General *n*-bit-*n*-bit Block Substitution (shown with $n = 4$)

Table 4.1

Encryption and Decryption Tables for Substitution Cipher of Figure 4.2

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

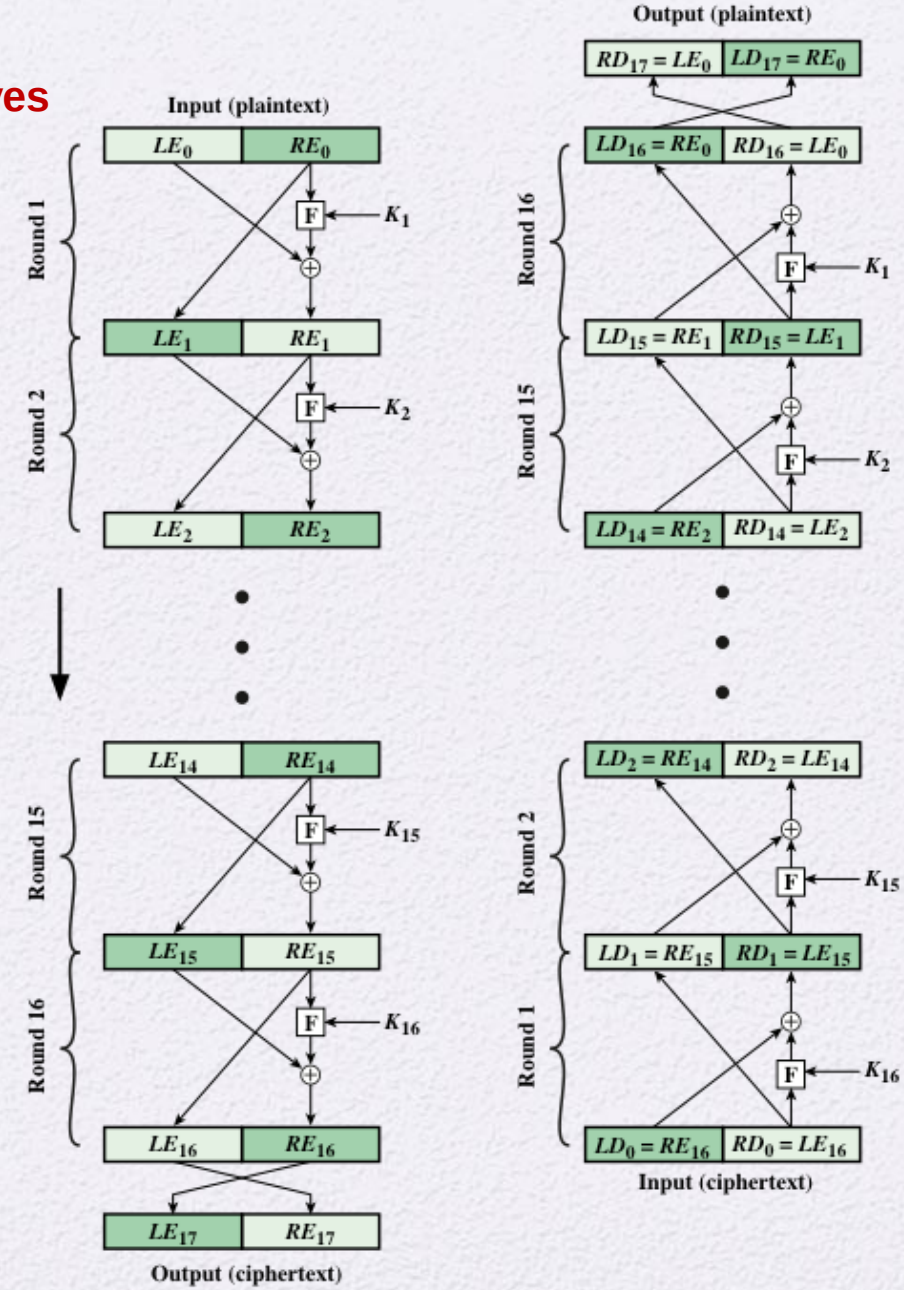
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations
  - Recall
    - Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements
    - Permutation: the order in which the elements appear in the sequence is changed
- Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions
- Is the structure used by many significant symmetric block ciphers currently in use

Algorithm:
- Input is split into 2 halves
- Number of rounds =16
- round function F
- K_i for $round_i$
-Decryption is opposite to encryption.

Encryption proposed by Feistel



Decryption proposed by Feistel

Figure 4.3 Feistel Encryption and Decryption (16 rounds)

Feistel Cipher Design Features

- **Block size**: larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm.
- **Key size**: larger key size means greater security but may decrease encryption/decryption speed.
- **Number of rounds**: the essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security
- **Subkey generation algorithm**: greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
- **Round function *F*** :greater complexity generally means greater resistance to cryptanalysis
- **Fast software encryption/decryption**: In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, 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

Feistel Example

Decryption is opposite to encryption.
We will demonstrate that round 1 (decryption) is reverse of round16 (encryption)

First, consider the encryption process. We see that

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

Encryption

On the decryption side,

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$
$$RD_1 = LD_0 \oplus \mathrm{F}(RD_0, K_{16})$$
$$= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16})$$
$$= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16})$$

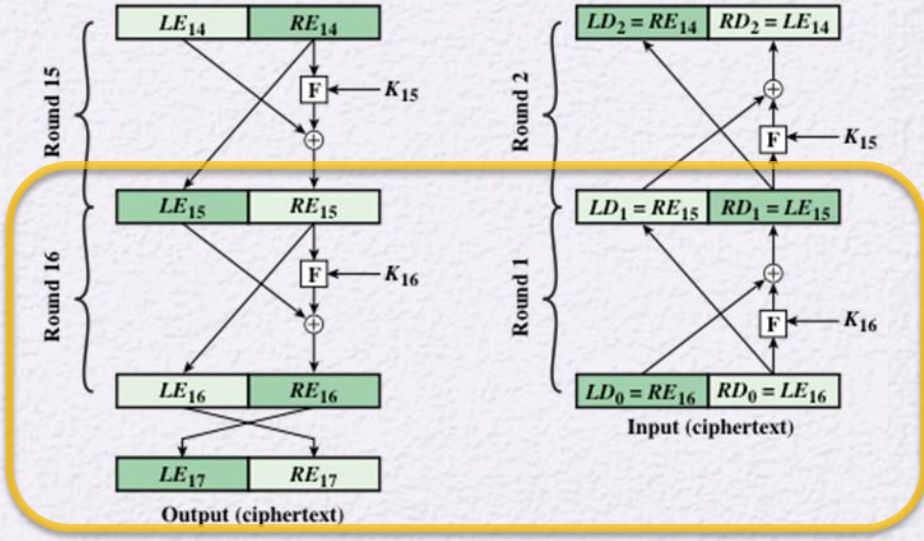
Note: $RE_{16} = LE_{15} \oplus K_{16}$

The XOR has the following properties:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$
$$D \oplus D = 0$$
$$E \oplus 0 = E$$

Thus, we have $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$.

Decryption

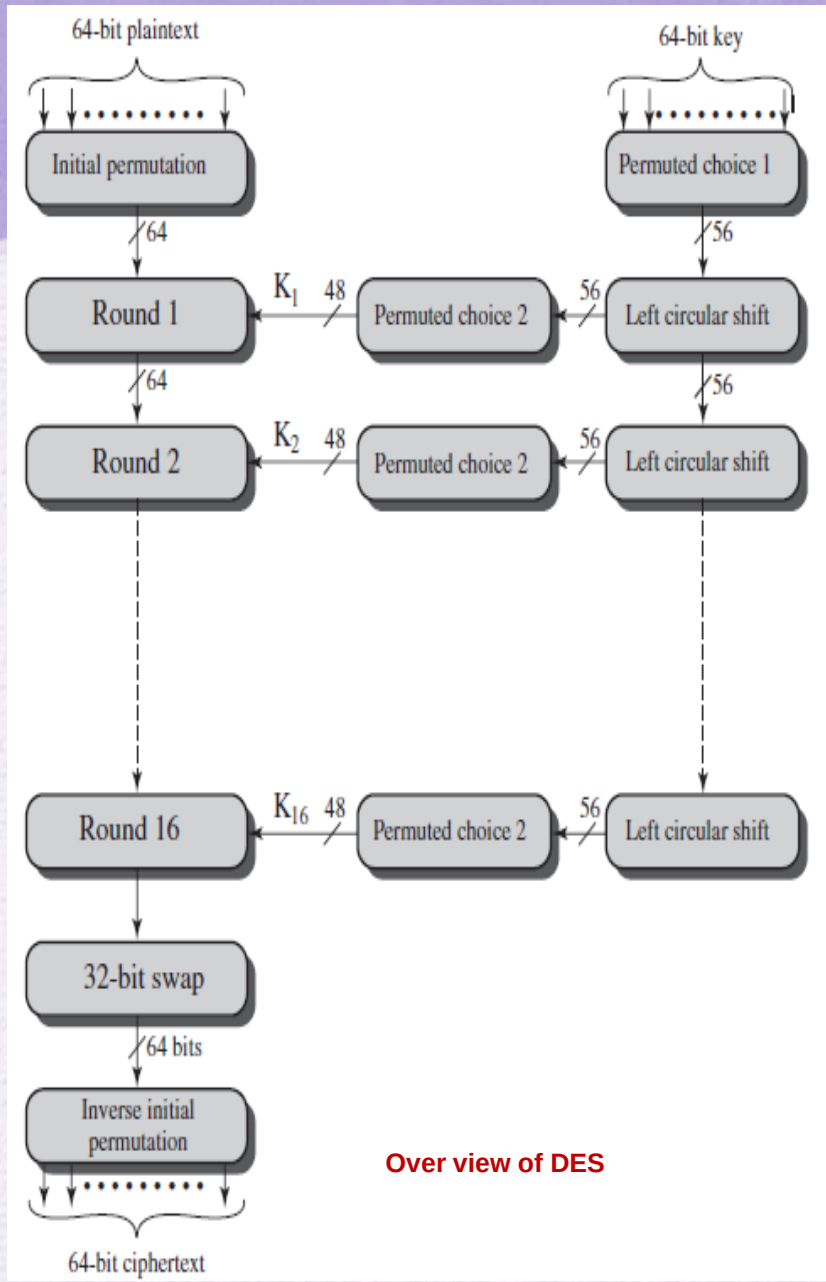


Data Encryption Standard (DES)

- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- Algorithm itself is referred to as the Data Encryption Algorithm (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

DES: Overview

- Key size = 56 bits (+ 8-bit parity)
- Block size = 64 bits
  - Input is split into two halves.
- Number of rounds/cycles =16
- Algorithm ensuring that the output bits have no obvious relationship to the input bits and spreading the effect of one plaintext bit to other bits in ciphertext.
- Substitution provides the confusion, and transposition provides the diffusion.
- DES algorithm operates on blocks of data. It splits a data block in half, scrambles each half independently, combines the key with one half, and swaps the two halves. This process is repeated 16 times.

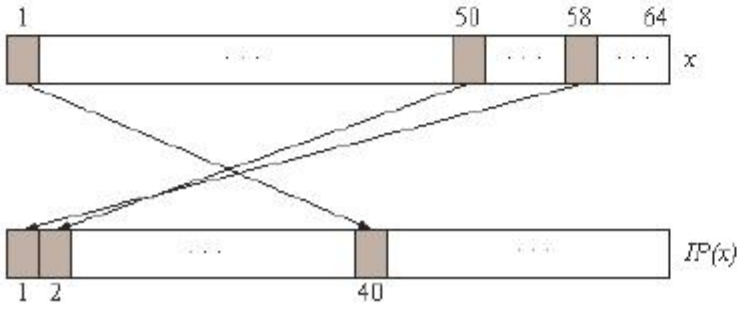


Over view of DES

Initial and Final Permutations

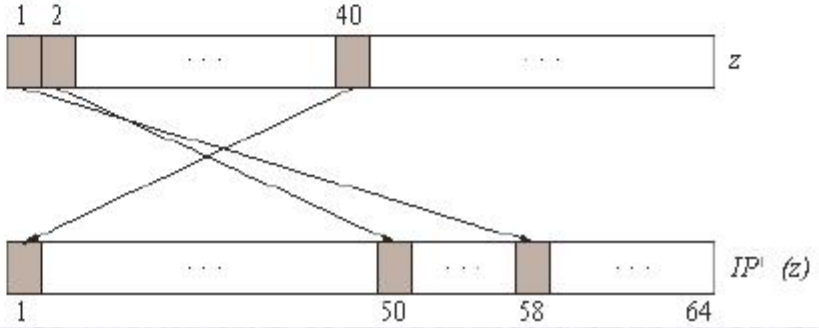
Initial Permutation

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



DES: Initial Permutation

Initial permutation permutes input bits according to the following table

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

Example:

02 46 8a ce ec a8 64 20 → 5a 00 5a 00 3c f0 3c 0f

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

Initial permutation

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

DES: Initial Permutation

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

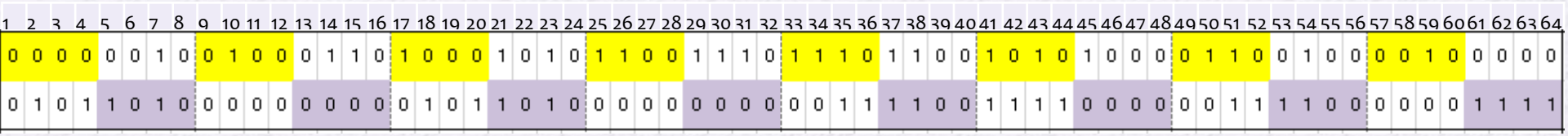
Initial permutation

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

Example:

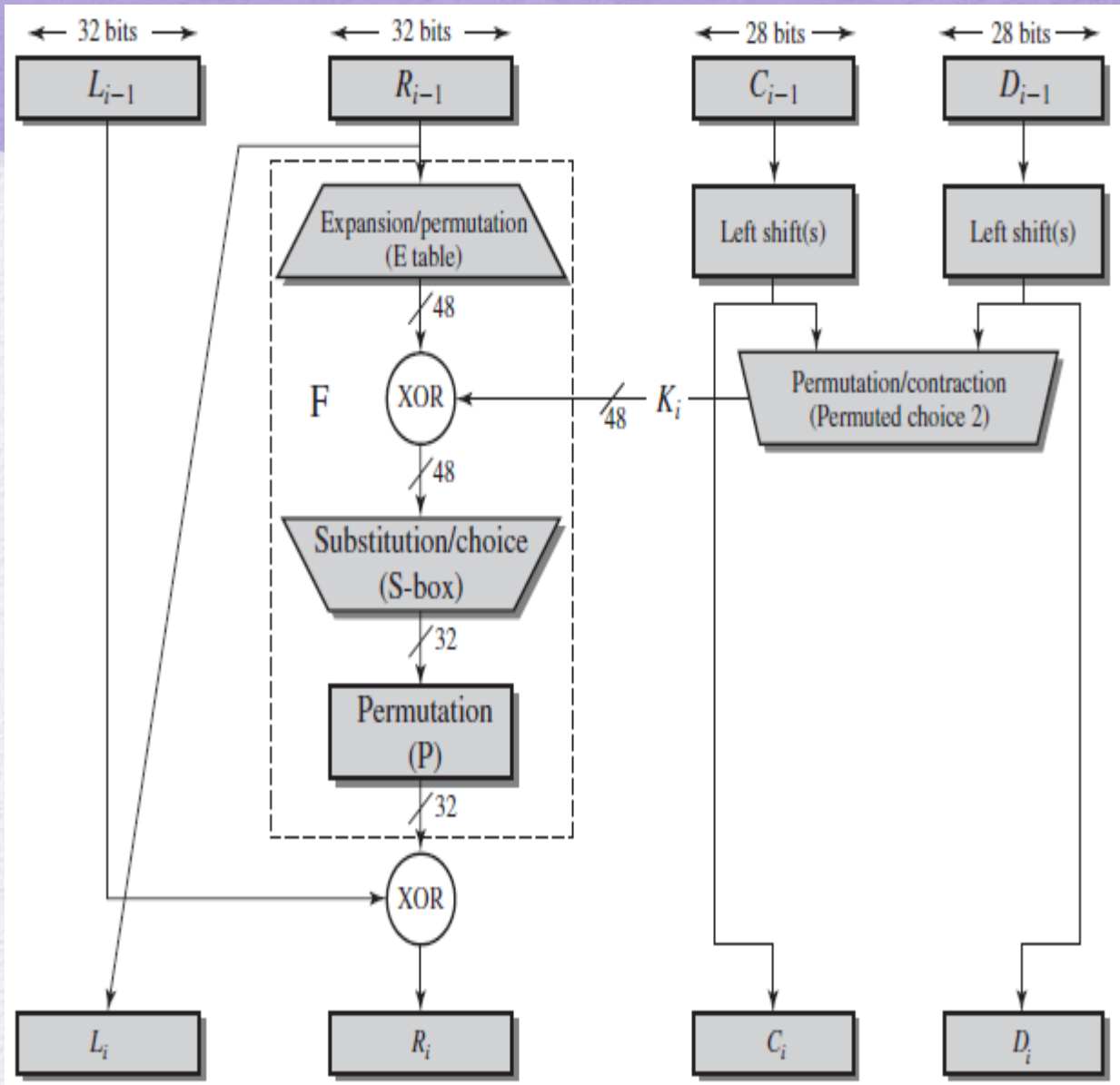
02 46 8a ce ec a8 64 20

5a 00 5a 00 3c f0 3c 0f



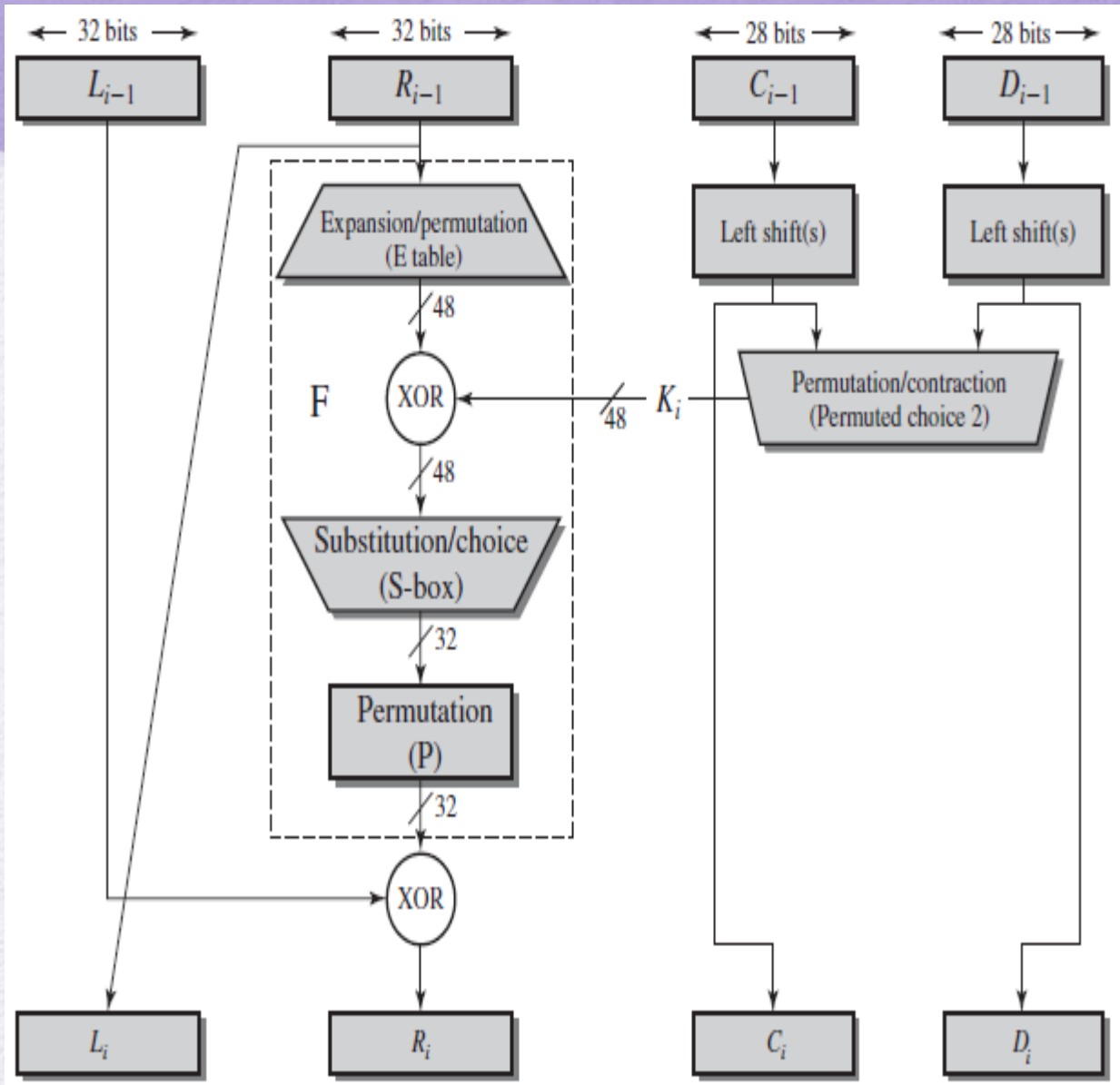
Single Round Algorithm

- Input is split into two 32-bit halves: L and R.
- Processing in each round$_i$:
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- ***F*** function includes
  - Expansion/permutation of R (E table)
  - XOR (R, Key)
  - Substitute (S-box)
  - Permutation (P)
- Key generation generates 48-bit key (K_i)for round$_i$.

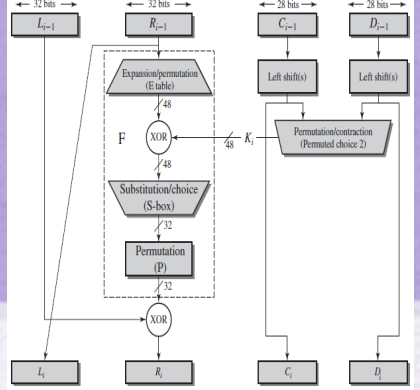


Single Round Algorithm

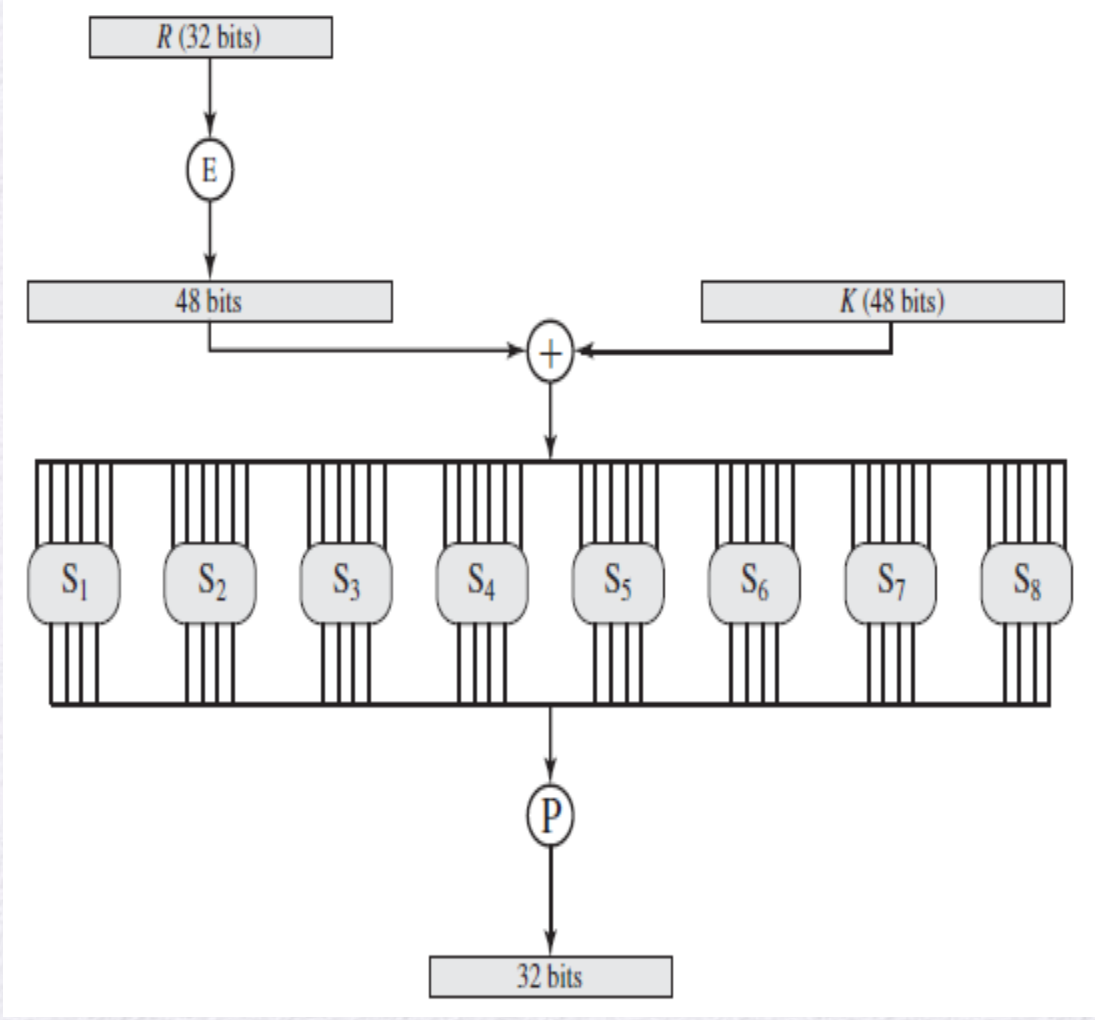
- Input is split into two 32-bit halves: L and R.
- Processing in each round$_i$:
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- ***F*** function includes
  - Expansion/permutation of R (E table)
  - XOR (R, Key)
  - Substitute (S-box)
  - Permutation (P)
- Key generation generates 48-bit key (K_i)for round$_i$.



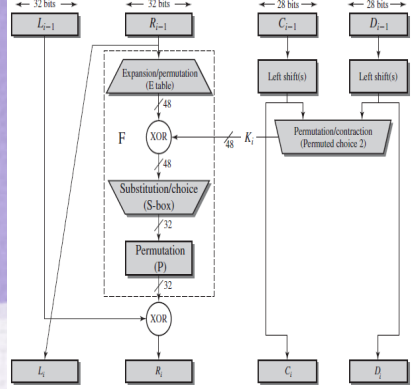
DES: F (R,K) function



- F function includes
  - Expansion/permutation of R (E table)
  - XOR (R, Key)
  - Substitute (S-box)
  - Permutation (P)
- S-box receives 6-bits and produces 4 bits.



DES: Expansion/Permutation of R



32-bit

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

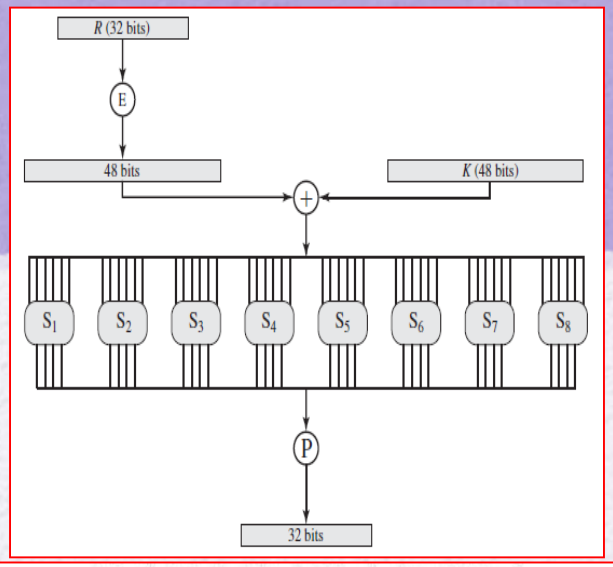
48-bit

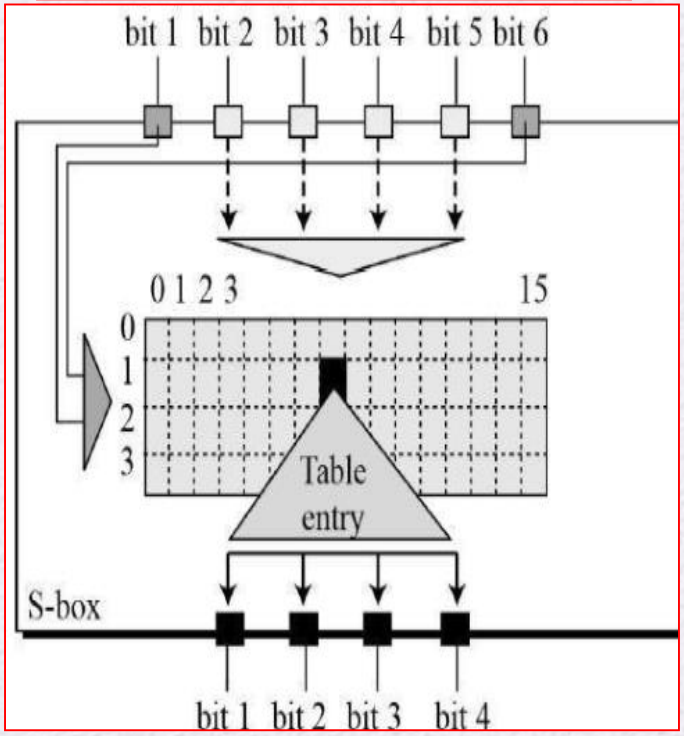
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Mapping and expanding of 32-bit to 48-bit by (E) step

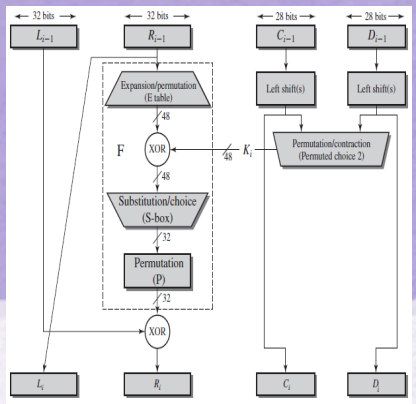
DES: Definition of S-boxes

- Used to implement confusion (i.e. obscure the relationship between the key and the ciphertext).
- For DES:
  - There are 8 S-boxes. They are described in detail in next slide.
  - S-box takes 6-bit input and produces 4-bit output.
  - S-box is a lookup-table with 4 rows and 16 columns
  - Outer 2 bits of input determine row.
  - Inner 4 bits of input determine column.
  - Once determined, row and column point to the location of 4-bit output.





DES: Definition of 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

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

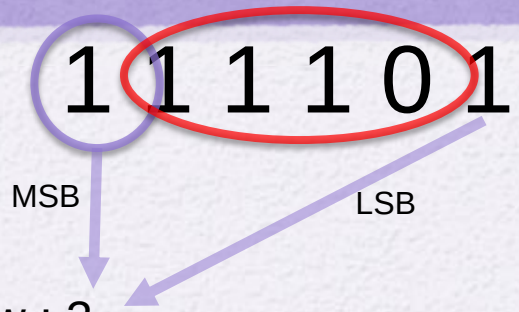
S_7

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

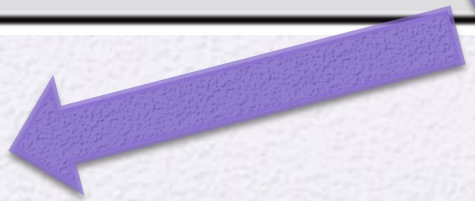
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S-box Example



Row : 3
Column: 14

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



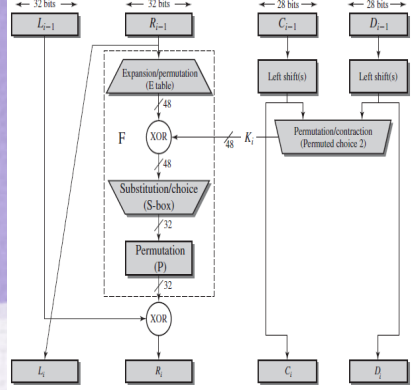
0 1 1 0

S-Box Nonlinearity

- The S-boxes are the most crucial elements of DES because they introduce a *nonlinearity* to DES Algorithm.
- Nonlinearity: $S(a) \oplus S(b) \neq S(a \oplus b)$
- Example:
  - a=100101 , b= 100111
  - $S_1(100101) \oplus S_1(100111) = 08_H \oplus 02_H = 0A_H$
  - $S_1(100101 \oplus 100111) = S_1(000010) = 04_H$

DES: Permutation function (P)

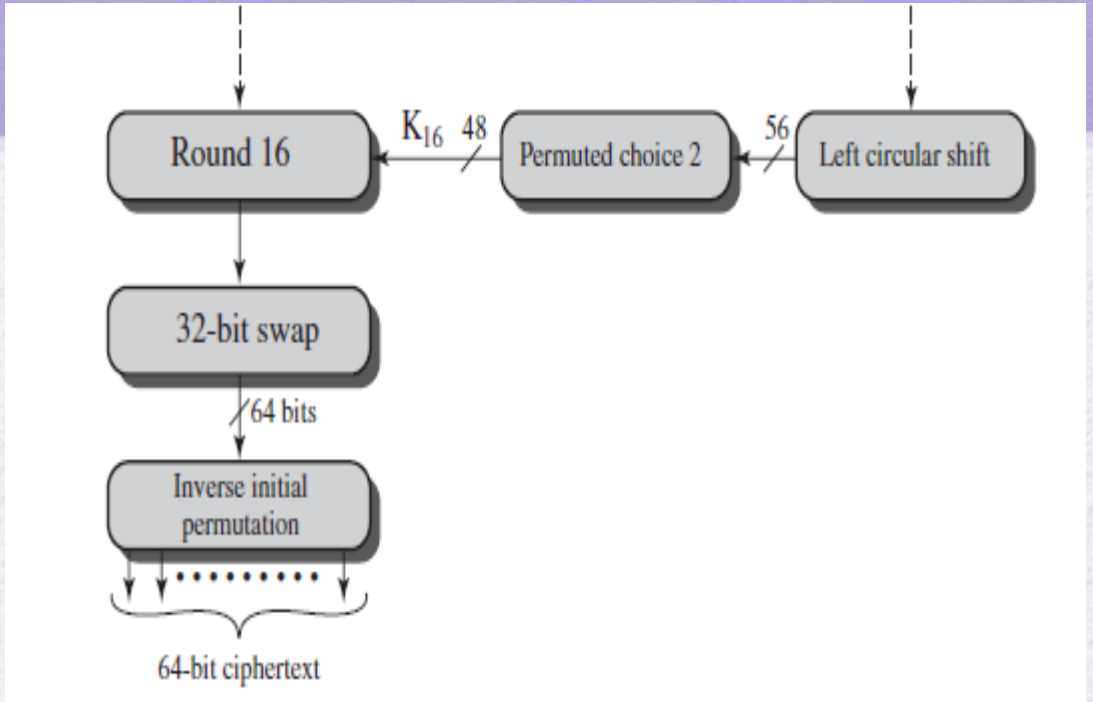
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32



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

DES: last steps

- After round 16:
  - 32-bit swap
    - Swap the two 32-bit halves: L and R.
  - Inverse initial permutation (shown in table)

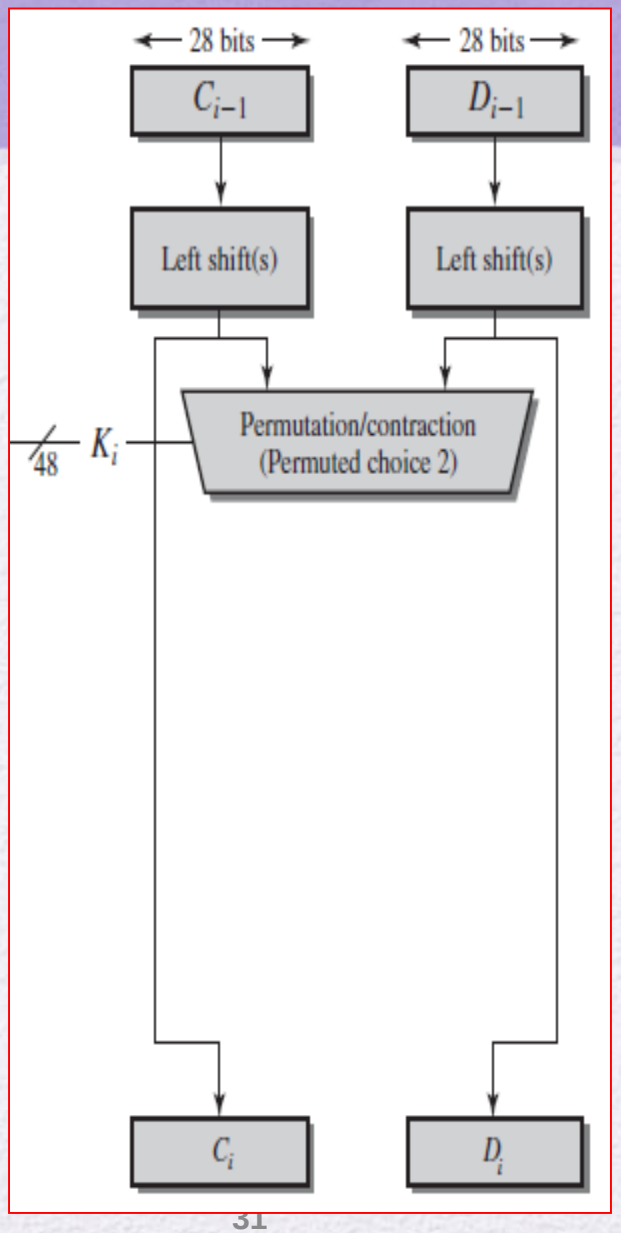


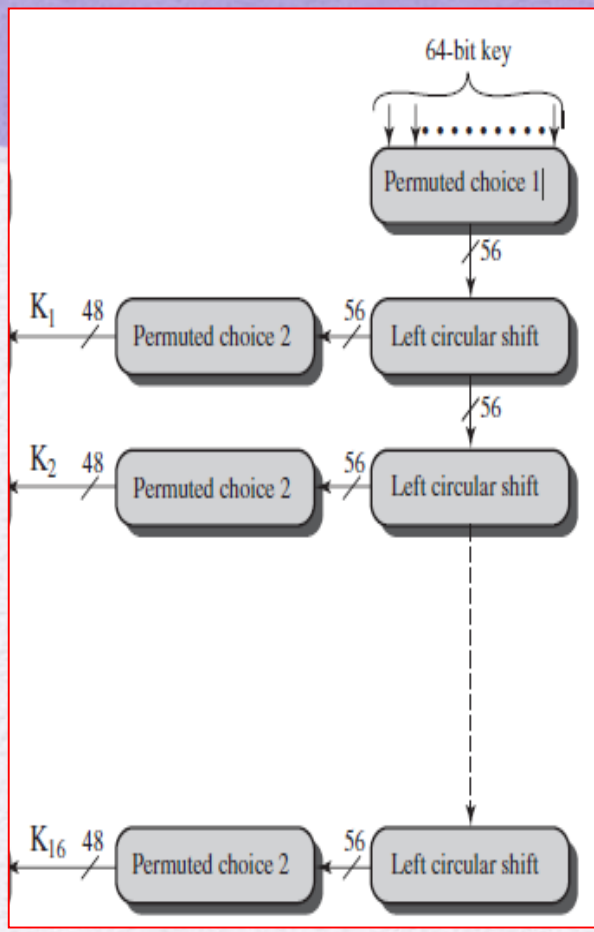
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Inverse initial permutation

DES: Key Generation

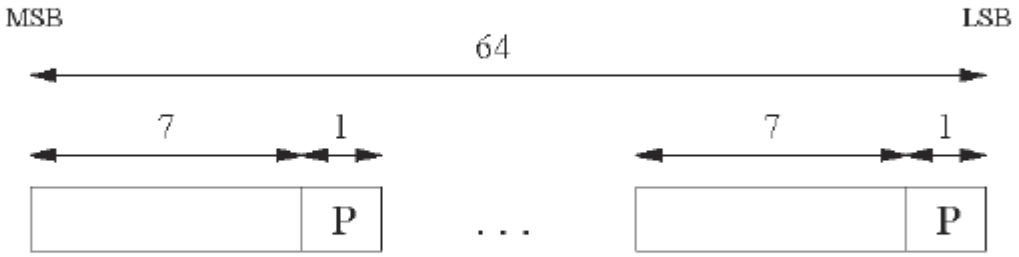
- Key is 64-bit.
  - The bits of the key are numbered from 1 through 64;
  - Every eighth bit is ignored, as indicated by the lack of shading in Input-Key Table (a) . (see next slides)
- The key is first goes through permutation governed by a table labeled Permuted Choice One, Table (b). (see next slides)
- The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_o and D_o.
- At each round C_{i-1} and D_{i-1} , and are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by Table (d).
- These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two Table(c), which produces a 48-bit output that serves as input to the function .





Key Size and Expansion

- Derives 16 round keys (or *subkeys*) k_i of 48 bits each from the original 56 bit key.
- The input key size of the DES is 64 bit: **56 bit key** and 8 bit parity:



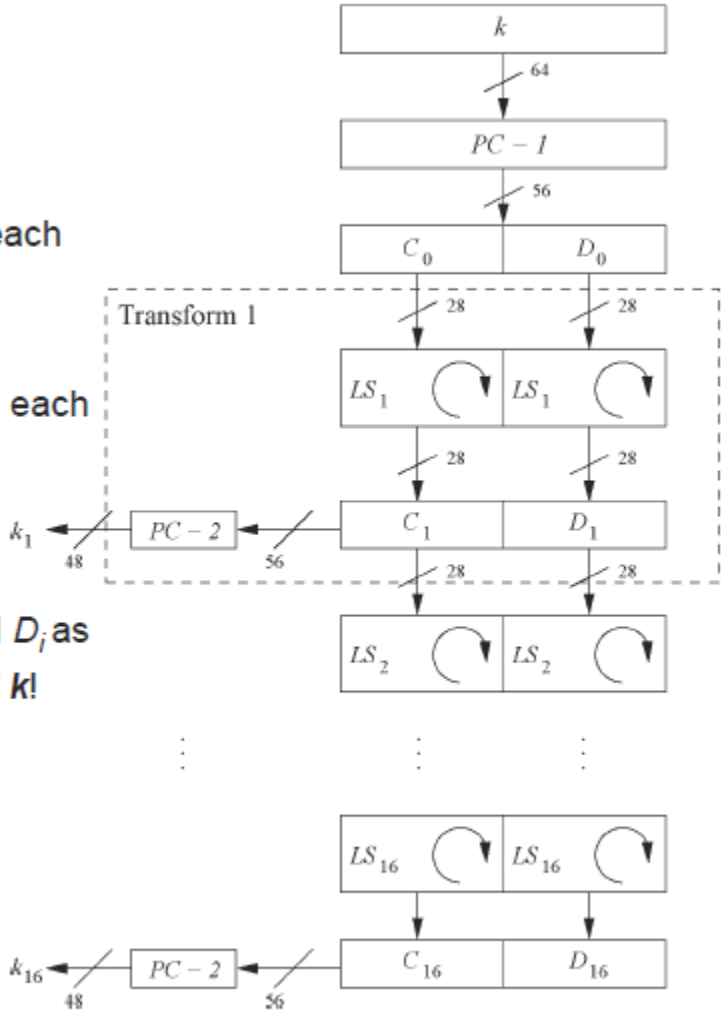
P = parity bit

- **Parity bits are removed** in a first **permuted choice** *PC-1*:
(note that the bits 8, 16, 24, 32, 40, 48, 56 and 64 are not used at all)

Key Scheduling (i.e. Expansion)

- Split key into 28-bit halves C_0 and D_0.
- In **rounds *i* = 1, 2, 9 ,16,** the two halves are each rotated left by **one bit**.
- In **all other rounds** where the two halves are each rotated left by **two bits**.
- *In each round i permuted choice* **PC-2** selects a permuted subset of 48 bits of C_i and D_i as round key k_i, i.e. **each k_i is a permutation of *k*!**

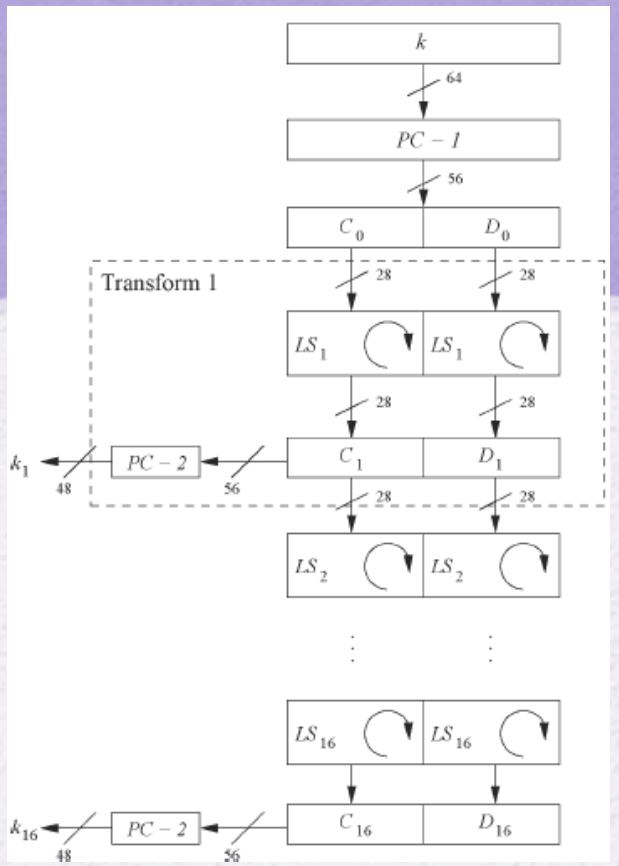
PC – 2							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32



- **Note:** The total number of rotations:
  4 x 1 + 12 x 2 = 28 $\Rightarrow D_0 = D_{16}$ and $C_0 = C_{16}$!

DES: Key Generation

Tables



(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

DES: Example

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

The first row shows the 32-bit values of the left and right halves of data after the initial permutation.
The next 16 rows show the results after each round.

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

Avalanche Effect

- Avalanche Effect
  - defined as: a change in one bit of the input should produce a change in many bits of the output.
  - measures diffusion in data and the change in key generation.
- Strict avalanche criterion (SAC)
  - Any single input change (e.g., flipping of a bit) should change every output bit with probability of ½.

Avalanche Effect in DES: Change in Plaintext

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 2b2cefbc99f91153	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 057cde97d7683f2a	32

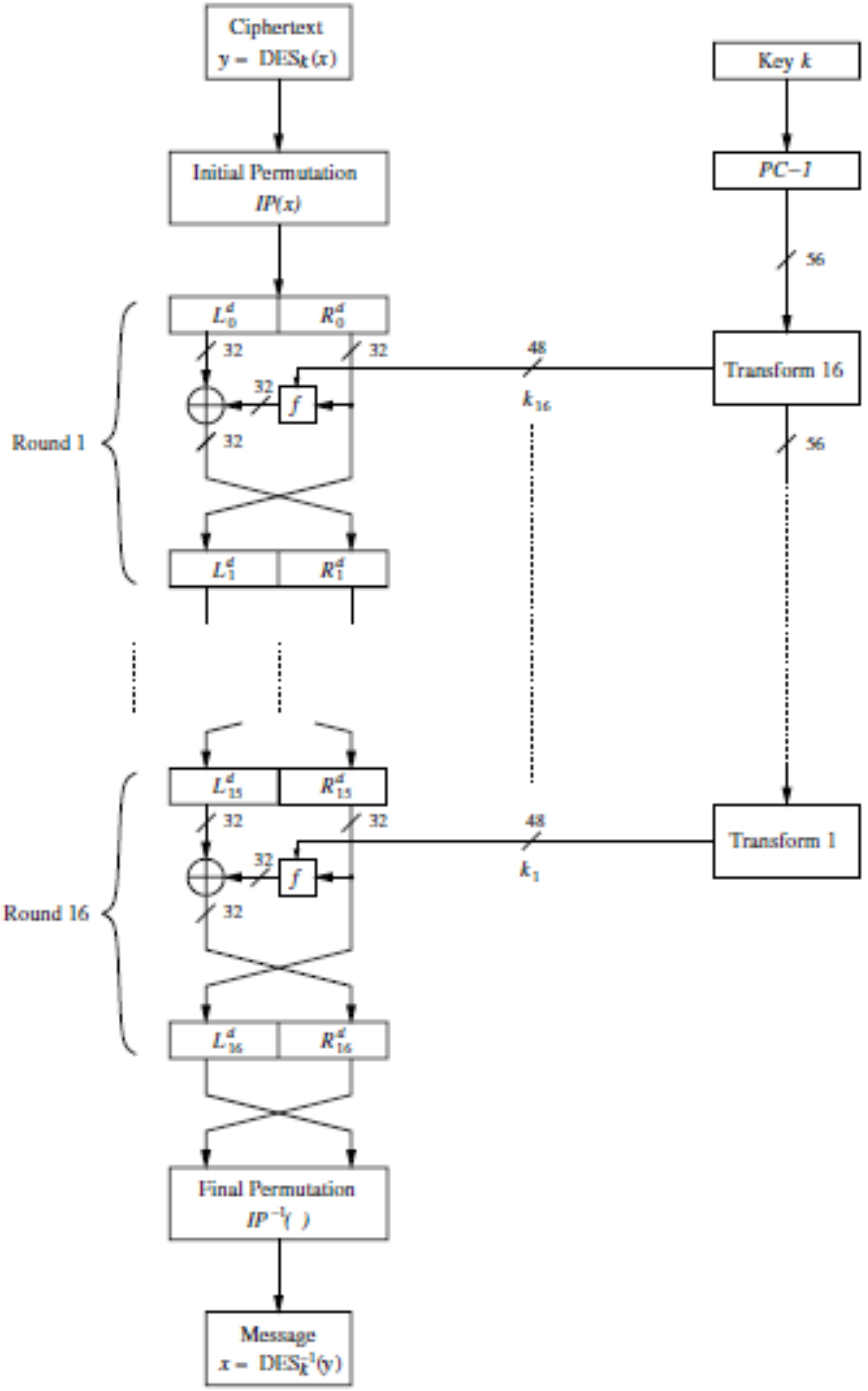
Avalanche Effect in DES: Change in Key

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

DES: Decryption

- Decryption is essentially the same function as encryption.
- This is because DES is based on a Feistel network.



DES: Decryption

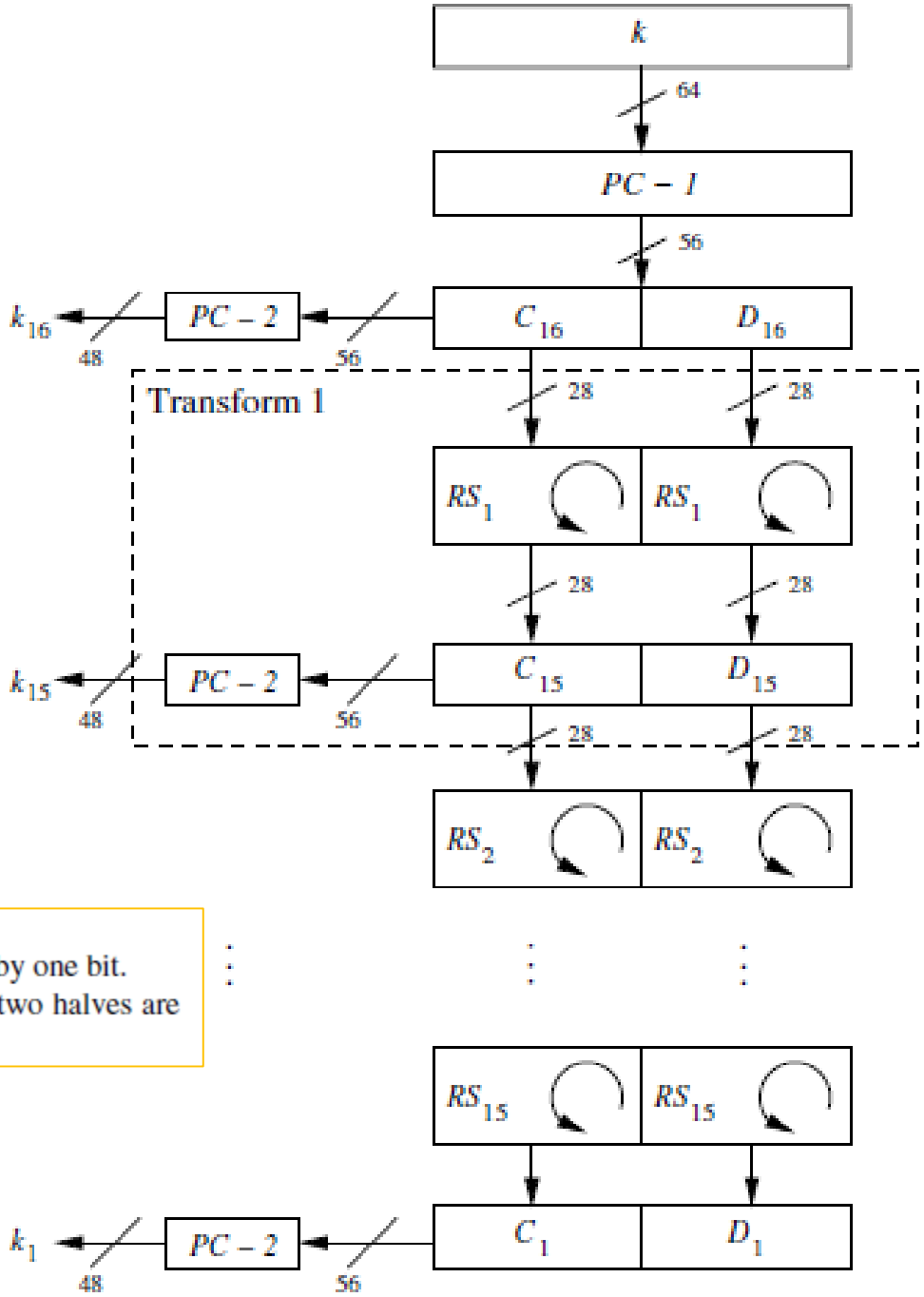
Reversed Key Schedule

- Note:
  - Co =C16 and Do = D16.
  - Hence $k16$ can be directly derived after PC−1.

$$
\begin{aligned}
k_{16} &= PC-2(C_{16}, D_{16}) \\
&= PC-2(C_0, D_0) \\
&= PC-2(PC-1(k))
\end{aligned}
$$

$$
\begin{aligned}
k_{15} &= PC-2(C_{15}, D_{15}) \\
&= PC-2(RS_2(C_{16}), RS_2(D_{16})) \\
&= PC-2(RS_2(C_0), RS_2(D_0))
\end{aligned}
$$

In decryption round 1, the key is not rotated.
In decryption rounds 2, 9, and 16 the two halves are rotated right by one bit.
In the other rounds 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14 and 15 the two halves are rotated right by two bits.



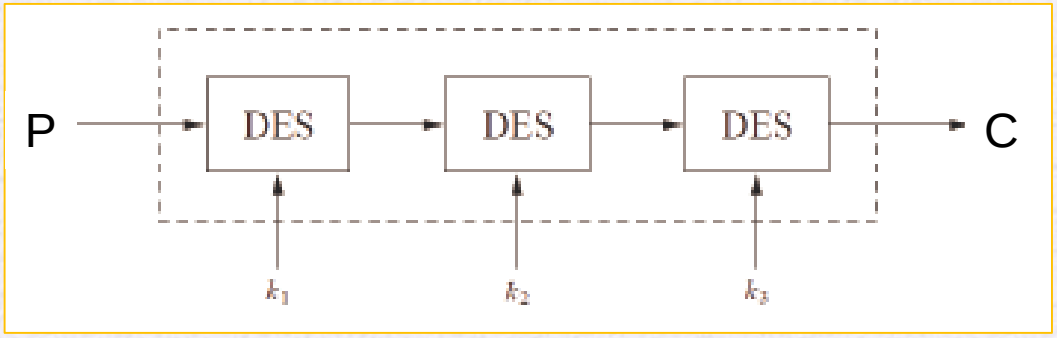
Security of DES

- **After proposal of DES two major criticisms arose:**
  1. Key space is too small (2^{56} keys)
  2. S-box design criteria have been kept secret: Are there any hidden analytical attacks (*backdoors*), only known to the NSA?

- **Analytical Attacks:** DES is highly resistent to both *differential* and *linear cryptanalysis*, which have been published years later than the DES. This means IBM and NSA had been aware of these attacks for 15 years!
  So far there is no known analytical attack which breaks DES in realistic scenarios.

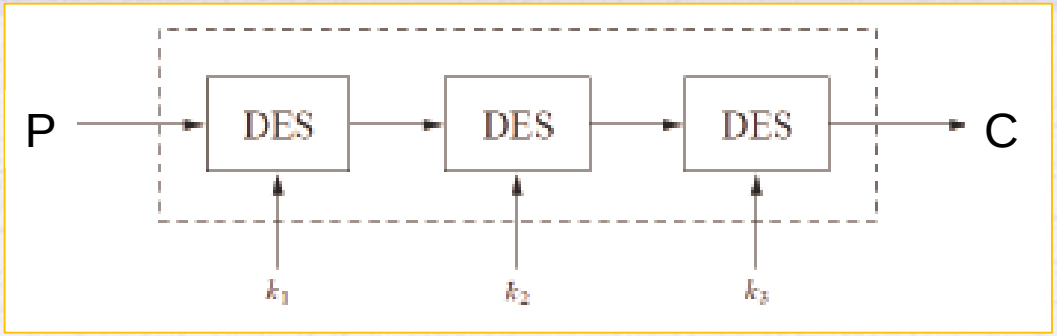
- **Exhaustive key search:** For a given pair of plaintext-ciphertext (x, y) test all 2^{56} keys until the condition $DES_k^{-1}(x)=y$ is fulfilled.
  $\Rightarrow$ Relatively easy given today's computer technology!

Triple DES (3DES)



- Triple DES increases
  - An extension for DES
  - Triple increases key size to protect against such attacks, without the need to design a completely new block cipher algorithm.
  - Used in many legacy applications, i.e., in banking systems.
- Encryption: $C=E_{k3}(D_{K2}(E_{k1}(P)))$
- Decryption: $P=D_{k1}(E_{K2}(D_{k3}(C)))$
- No practical attack known today.

Triple DES: Keys options



- All three keys are independent
  - total key size=3×56=168 bits
- K1=K3, K2 is independent
  - Total key size=2 ×56=112 bits
- K1=K2=K3
  - Triple DES becomes like original DES. Key size=56 bits

Alternatives to DES

Algorithm	I/O Bit	key lengths	remarks
AES / Rijndael	128	128/192/256	DES "replacement", worldwide used standard
Triple DES	64	112 (effective)	conservative choice
Mars	128	128/192/256	AES finalist
RC6	128	128/192/256	AES finalist
Serpent	128	128/192/256	AES finalist
Twofish	128	128/192/256	AES finalist
IDEA	64	128	patented

Table 4.5

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	$26! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

Block Cipher Design Principles: Number of Rounds

?

The greater the number of rounds, the more difficult it is to perform cryptanalysis

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack

If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search

Block Cipher Design Principles: Design of Function F

- The heart of a Feistel block cipher is the function F
- The more nonlinear F, the more difficult any type of cryptanalysis will be
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function

The algorithm should have good avalanche properties

Strict avalanche criterion (SAC)

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

Bit independence criterion (BIC)

States that output bits j and k should change independently when any single input bit i is inverted for all i , j , and k

Block Cipher Design Principles: Key Schedule Algorithm

- With any Feistel block cipher, the key is used to generate one subkey for each round
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key
- It is suggested that, at a minimum, the **key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion**

Summary

- Traditional Block Cipher Structure
  - Stream ciphers
  - Block ciphers
  - Motivation for the Feistel cipher structure
  - Feistel cipher
- The Data Encryption Standard (DES)
  - Encryption
  - Decryption
  - Avalanche effect

- The strength of DES
  - Use of 56-bit keys
  - Nature of the DES algorithm
  - Timing attacks
- Block cipher design principles
  - Number of rounds
  - Design of function F
  - Key schedule algorithm

Additional Slides

Additional Slides

Bit-wise compliment of Data, Key and Output

- DES has a somewhat surprising property related to bitwise complements of its inputs and outputs.

$$y = DES_k(x)$$

$$y' = DES_{k'}(x')$$

- *y`*, *k`*, and *x`* are bit-wise compliment of *y*, *k* and *x*.

Weak Key

- If ENC(P, K) = DEC(P,K) ➔ K is defined as weak Key
- **Formal Definition:**

A DES key K_w is called a weak key if encryption and decryption are identical operations:

$$\mathrm{DES}_{K_w}(x) = \mathrm{DES}^{-1}_{K_w}(x), \text{for all } x$$

For $\mathrm{DES}_{K_w}(x) = \mathrm{DES}^{-1}_{K_w}(x), \text{for all } x$ to hold true, the generated subkeys must have the following relation to one another:

$$k_{i+1} = k_{16-i} \text{ for } i = 0, 1, \ldots, 7$$

Rotating all 0s or all 1s is the only way to satisfy this criteria, since they always produce the same subkey in each round.

Weak Key

Rotating all 0s or all 1s is the only way to satisfy this criteria, since they always produce the same subkey in each round.

As such, the four weak keys are when:

$$C_0 = FFFFFFF_{16} \text{ or } 0000000_{16}$$

and

$$D_0 = FFFFFFF_{16} \text{ or } 0000000_{16}$$

Ignoring the effect of $PC-1$ (since it's trivial to reverse), the four possible weak keys can be defined as:

① $(C_0, D_0) = 00000000000000_{16}$

② $(C_0, D_0) = 0000000FFFFFFF_{16}$

③ $(C_0, D_0) = FFFFFFF0000000_{16}$

④ $(C_0, D_0) = FFFFFFFFFFFFFF_{16}$

The likelyhood of choosing one of these weak keys at random is as follows:

$$\frac{4}{2^{56}} = \frac{2^2}{2^{56}} = \frac{1}{2^{54}}$$

DES Implementation: Throughput and Frequency

- Suppose we have a DES implementation, in which:
  - Each round takes **c** cycles
  - The design frequency is ***F***
- Throughput is number of encrypted bits per second
- $Throughput = \frac{Encrypted\_Bits}{Second} = \frac{Encrypted\_Bits}{Cycle} \times \frac{Cycle}{Second}$
- $\frac{Encrypted\_Bits}{Cycle} = \frac{64-bits}{16\ Rounds \times \frac{\boldsymbol{C}\ cyles}{Round}} = \frac{4\ bits}{\boldsymbol{c}\ cyles}$
- $Throughput = \frac{Encrypted\_Bits}{Cycle} \times \frac{Cycle}{Second} = \frac{4\ bits}{\boldsymbol{c}\ cyles} \times \frac{Cycle}{Second}$
- $Throughput = \frac{4\ bits}{c\ cyles} \times F$

DES Implementation: Throughput and Frequency

- Example: What clock frequency is required for encrypting a fast network link running at a speed of 1 Gb/sec? Assume that one round can be performed in one clock cycle.
- Solution:
  - $Throughput = \frac{4\ bits}{c\ cyles} \times F$
  - 1×10^9 bits/sec = $\frac{4\ bits}{1\ cyles} \times F$
  - $F = 0.25\ GHz$