Chapter 2

Data Encryption Standard (DES)

Objectives:

- To review a short history of DES
- To define the basic structure of DES
- To describe the details of building elements of DES
- To describe the round keys generation process
- To analyze DES

Introduction

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

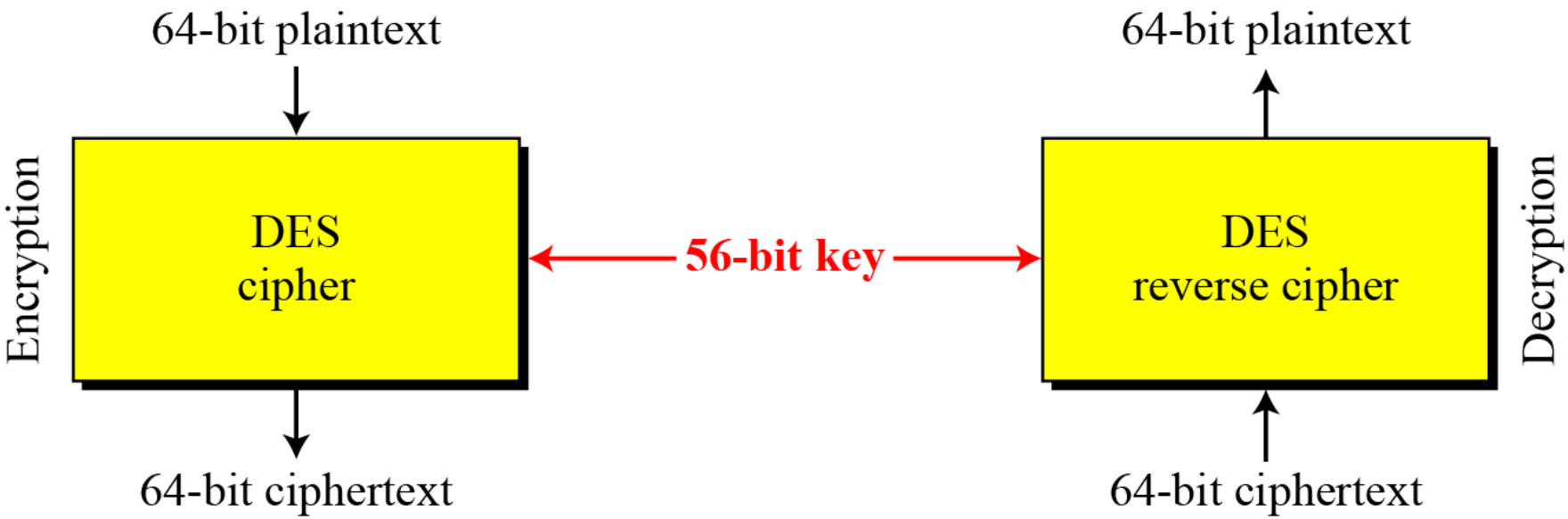
History

- In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem. A proposal from IBM, a modification of a project called Lucifer, was accepted as DES.
- DES was published in the Federal Register in March 1975 as a draft of the Federal Information Processing Standard (FIPS).

DES

DES is a block cipher, as shown in Figure

Figure ***Encryption and decryption with DES***



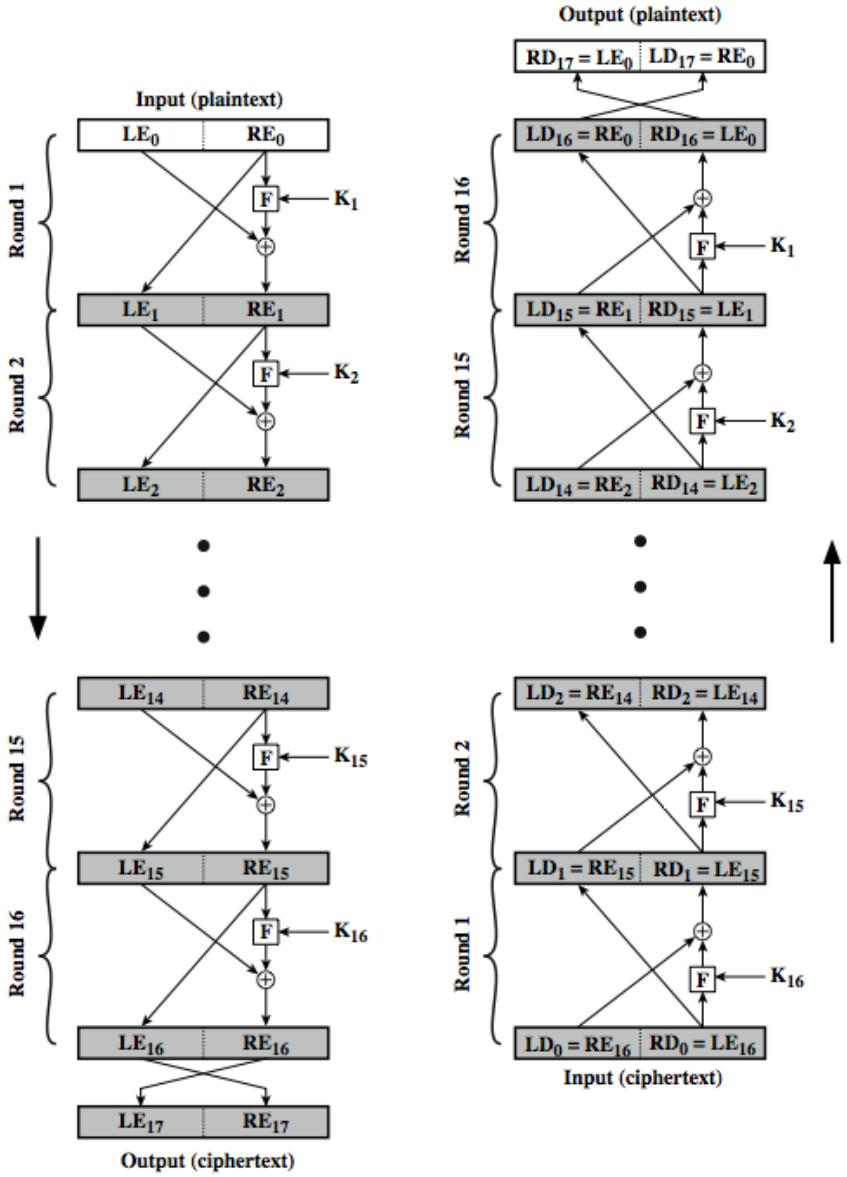
DES

- The Data Encryption Standard (DES) is a symmetric-key block cipher
- DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure.

Feistel Cipher Structure

- Horst Feistel devised the **Feistel cipher**
  - based on concept of invertible product cipher
- partitions input block into two halves
- Process halves through multiple rounds which perform a substitution on left half based on round function of right half & subkey
- Then have permutation swapping halves.

Feistel Cipher Structure



Feistel Cipher Design Elements

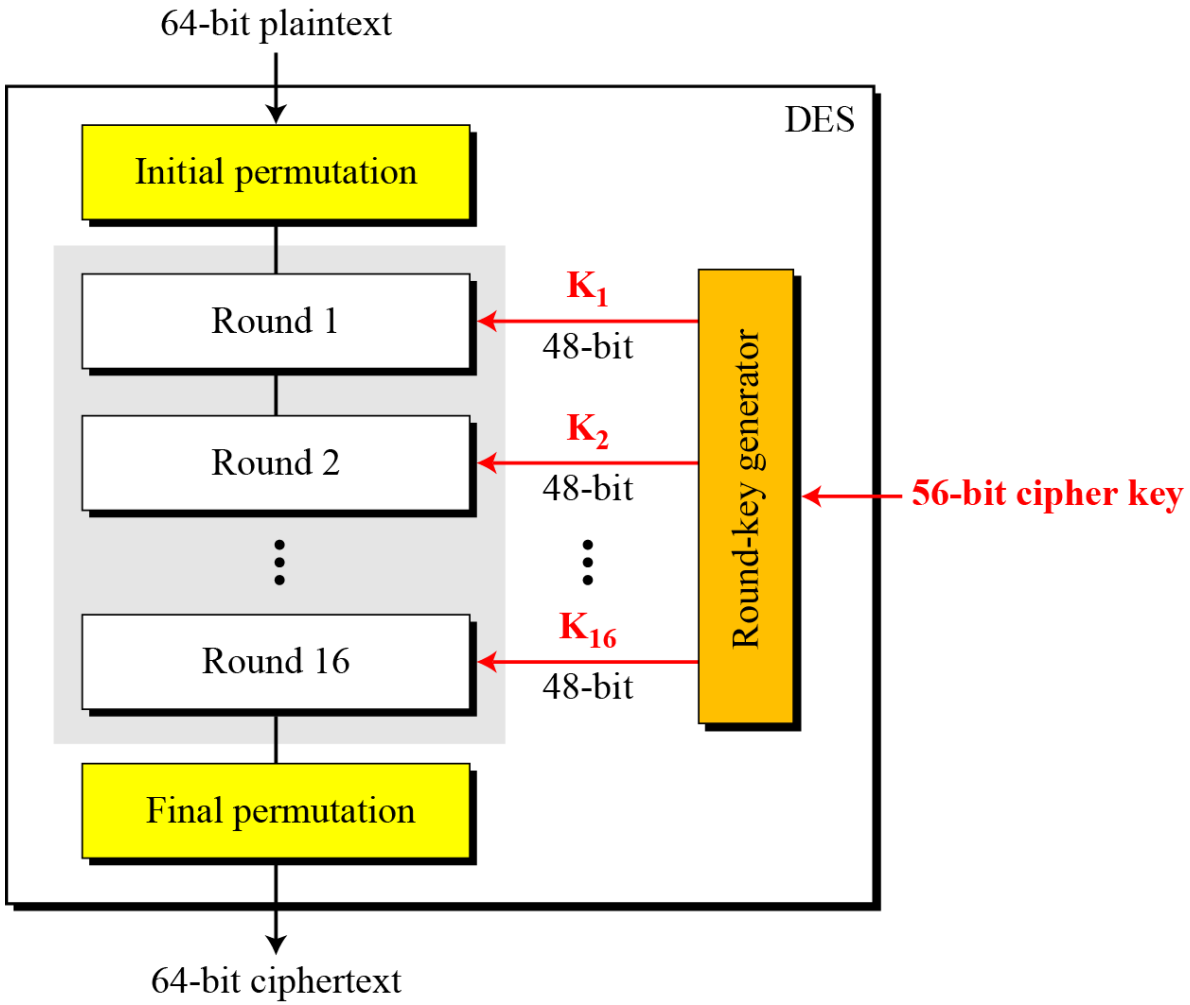
- block size
- key size
- number of rounds
- subkey generation algorithm
- round function
- fast software en/decryption
- ease of analysis

Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message
- more practically Shannon suggested combining S & P elements to obtain:
- diffusion – dissipates statistical structure of plaintext over bulk of ciphertext
- confusion – makes relationship between ciphertext and key as complex as possible

General structure of DES

Figure ***General structure of DES***



Initial and Final Permutation

- The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other.
- They have no cryptography significance in DES.

Initial and Final Permutations

Initial and final permutation steps in DES

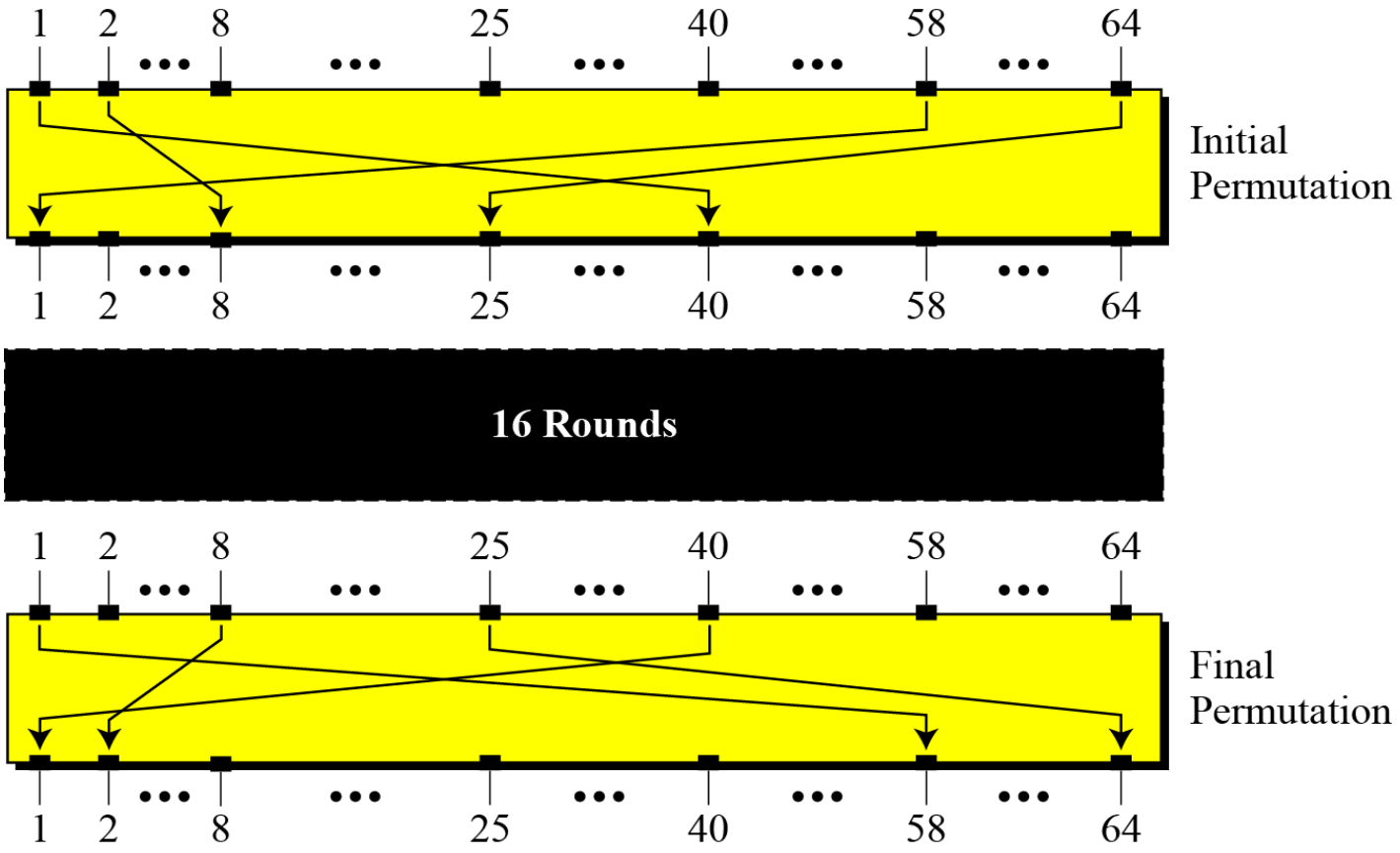


Table 6.1 *Initial and final permutation tables*

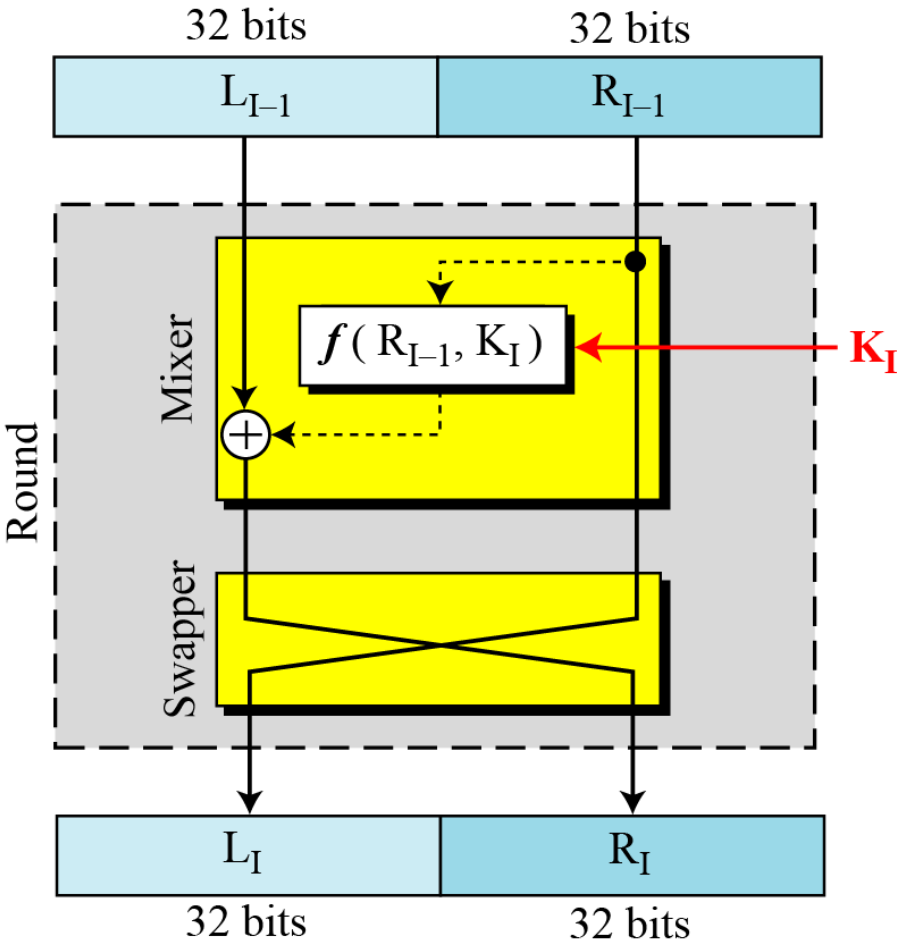
Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11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

Round Function

- The heart of this cipher is the DES function, *f*.
- The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

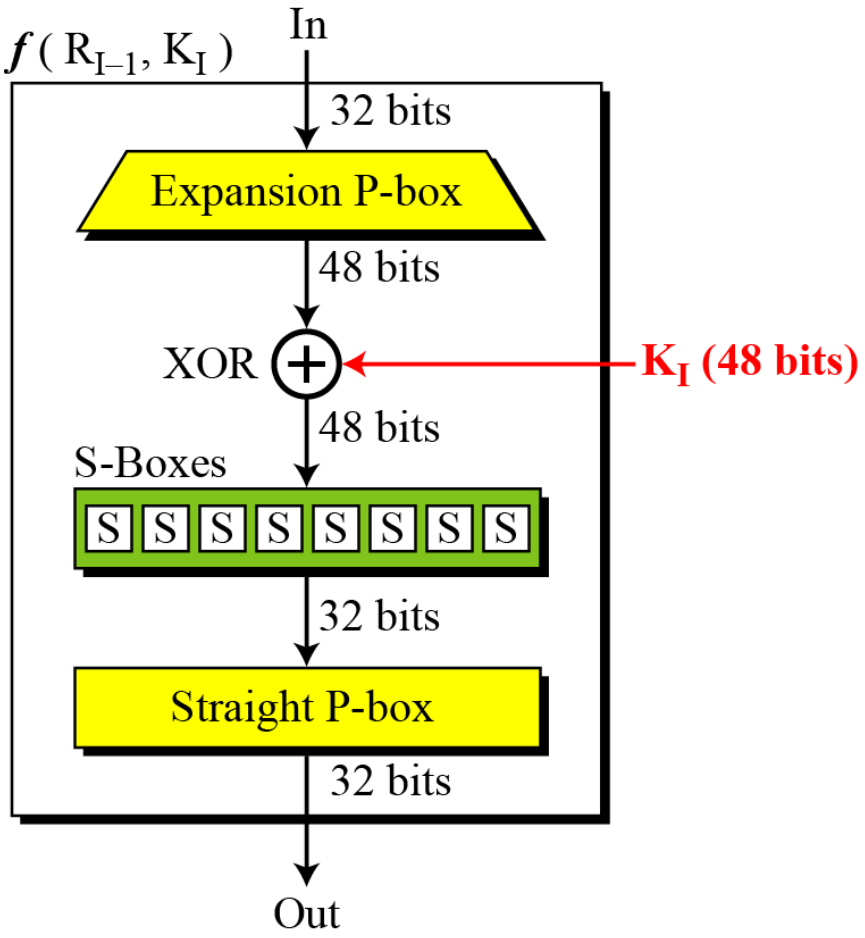
Rounds

DES uses 16 rounds. Each round of DES is a Feistel cipher.



A round in DES (encryption site)

DES Function

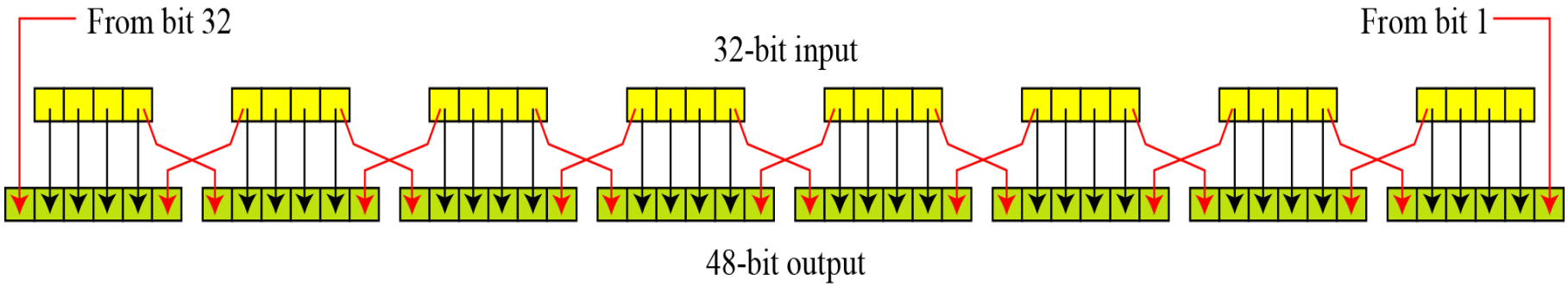


DES function

Expansion P Box

- Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits.

Expansion P-box



Expansion permutation

Continue

Although the relationship between the input and output can be defined mathematically, DES uses Table to define this P-box.

Table ***Expansion P-box table***

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

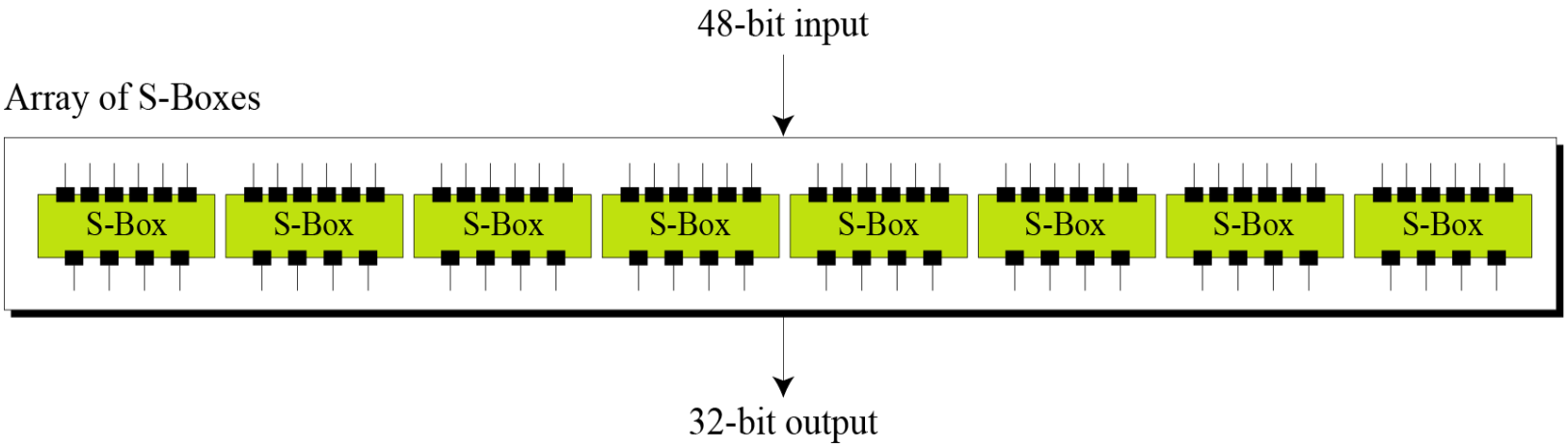
Continue

Whitener (XOR)

- After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key.
- Note that both the right section and the key are 48-bits in length.
- Also the round key is used only in this operation.

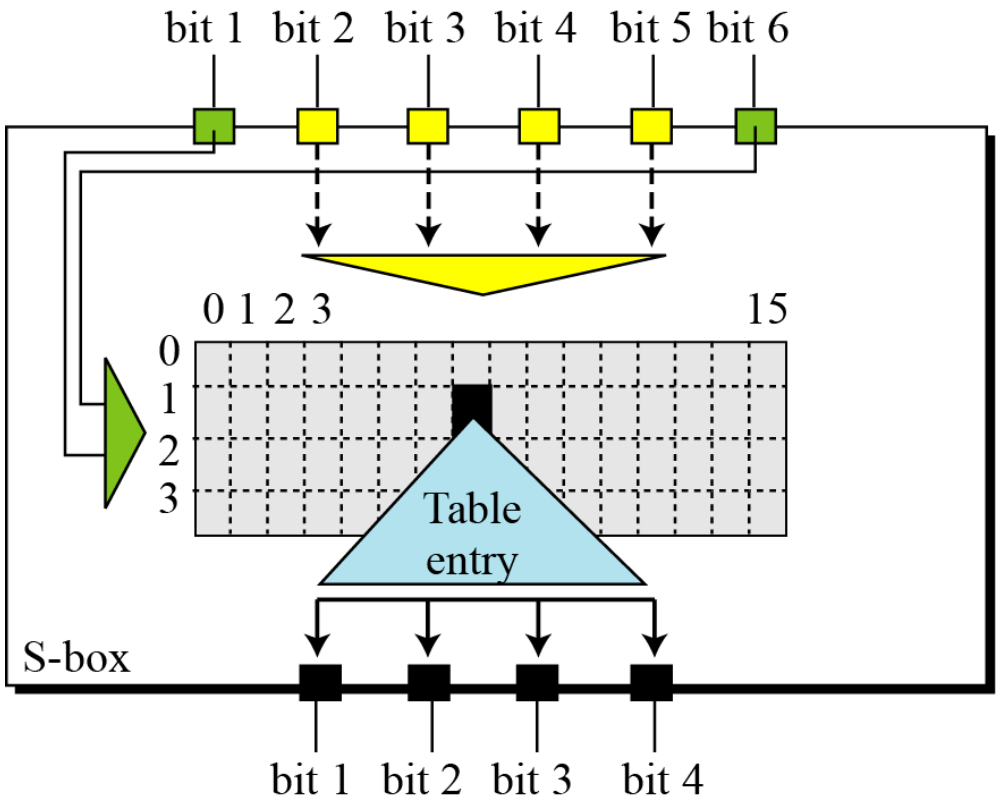
Continue

Substitution Boxes. − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.



S-boxes

Continue



S-box rule

Continue

Following Table shows the permutation for S-box 1.

S-box 1

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

Continued

Example : The input to S-box 1 is 100011. What is the output?

Solution

If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

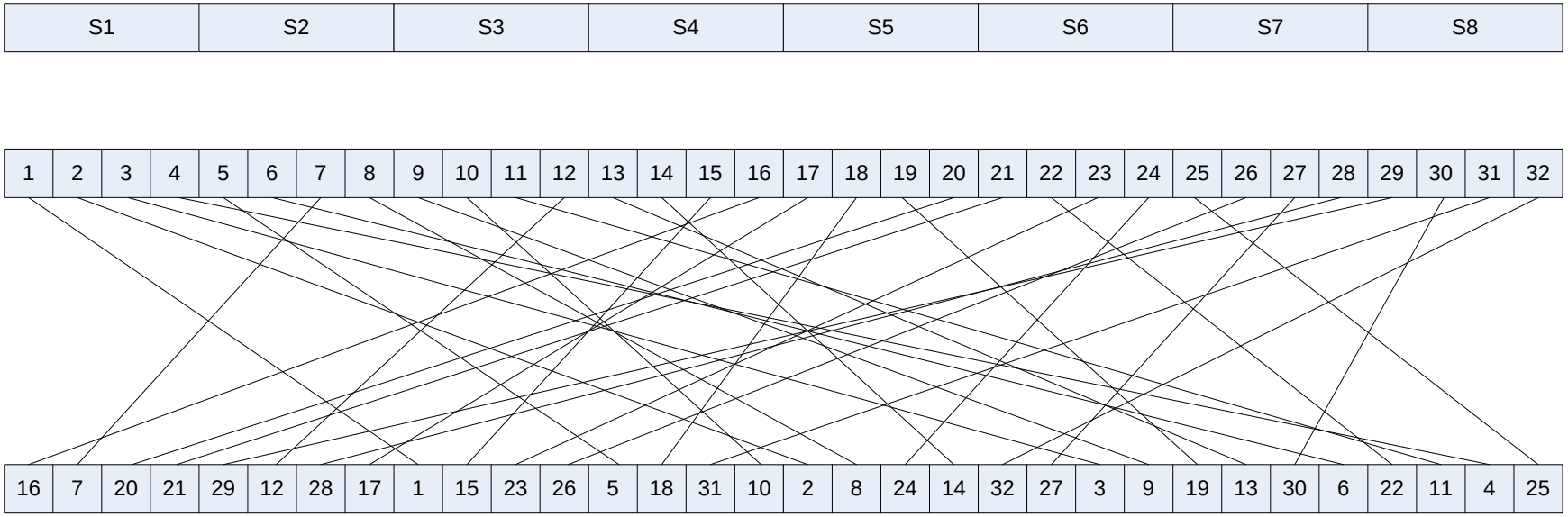
Continue

Straight Permutation − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

Straight permutation table

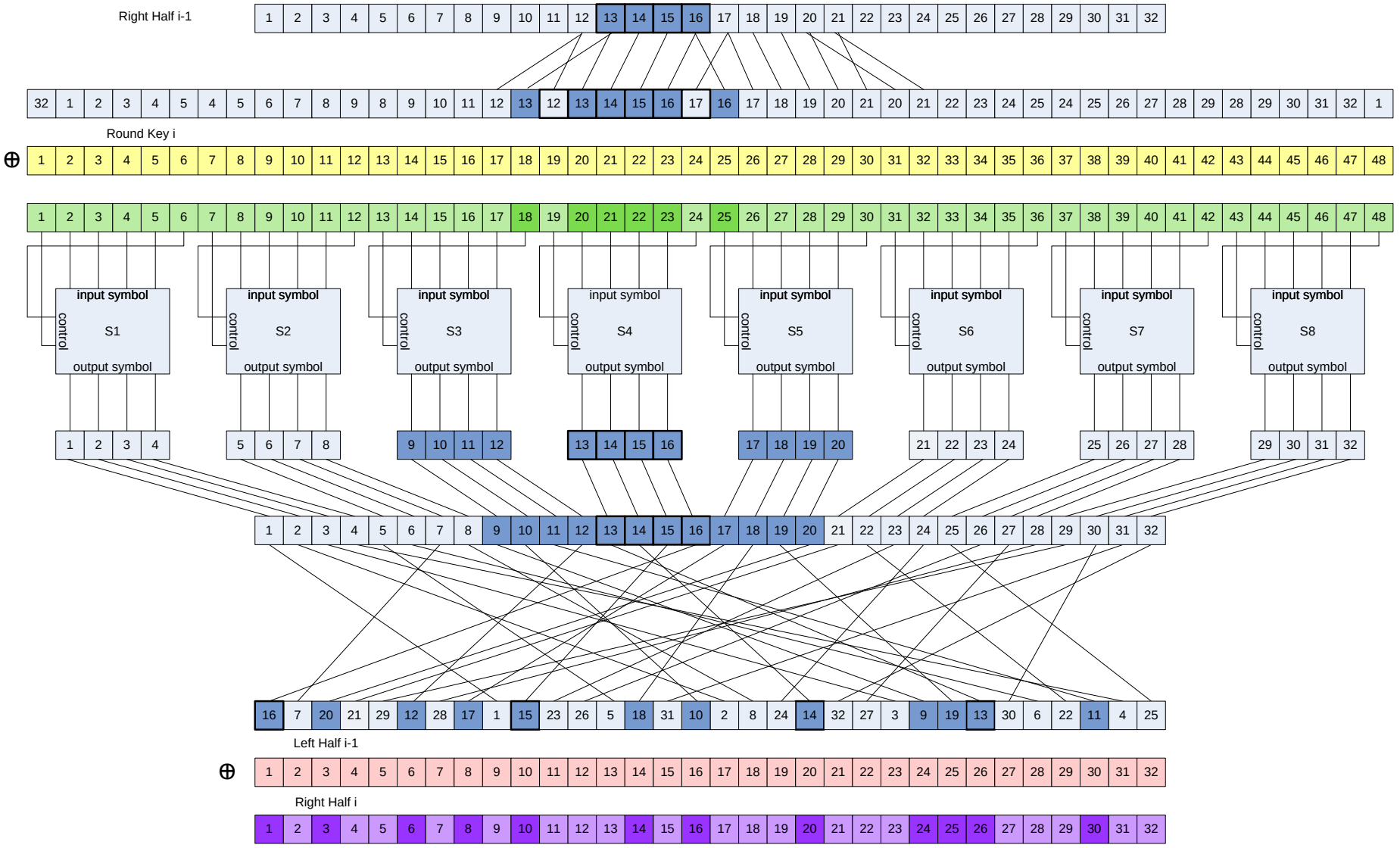
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Permutation Box P



- P-box applied at end of each round

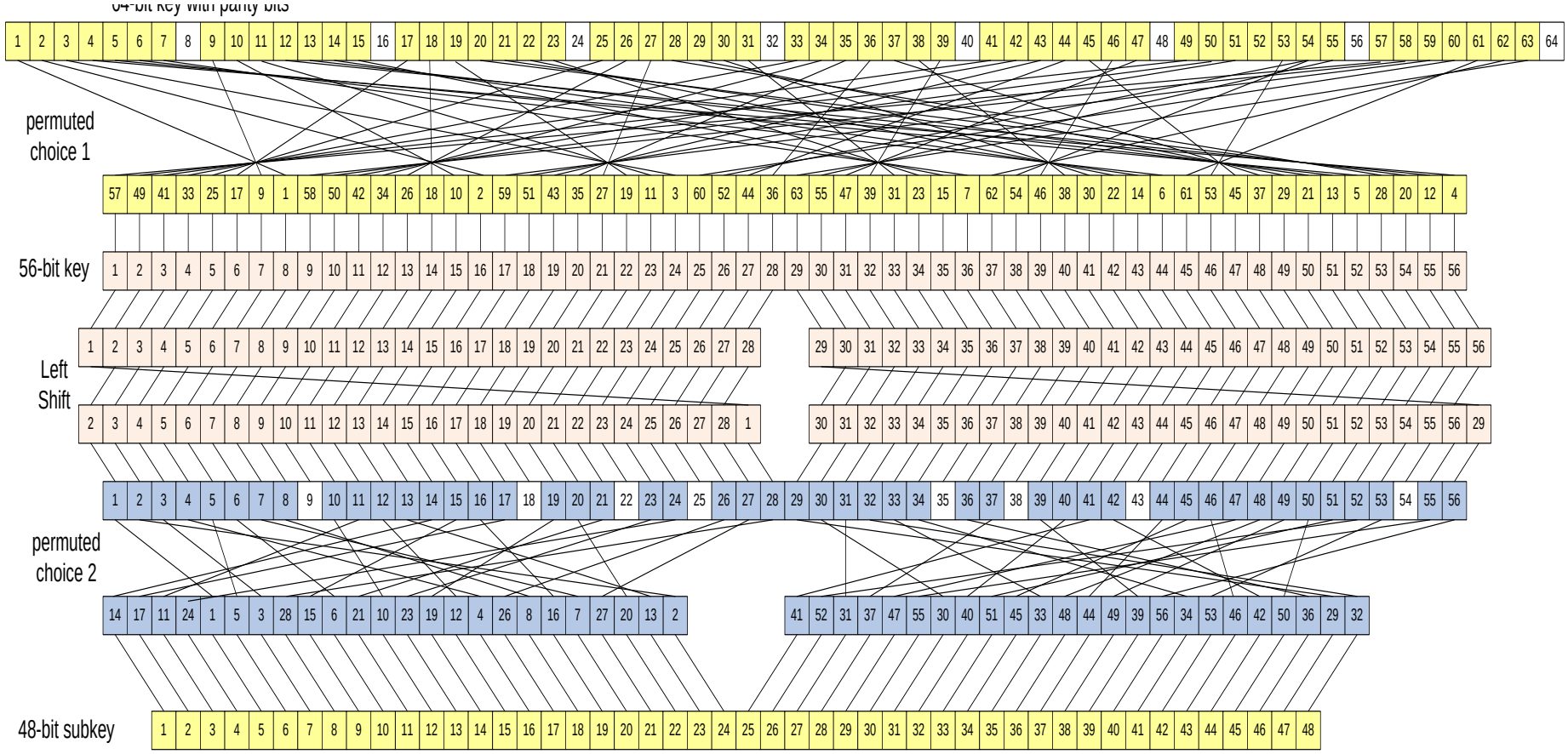
DES round in full



DES Key Schedule

57	49	41	33
58	50	42	34
59	51	43	35
60	52	44	36
31	23	15	07
30	22	14	06
29	21	13	05

14	17	11	24	01	05	03
15	06	21	10	23	19	12
26	08	16	07	27	20	13
41	52	31	37	47	55	30
51	45	33	48	44	49	39
34	53	46	42	50	36	29



Cipher and Reverse Cipher

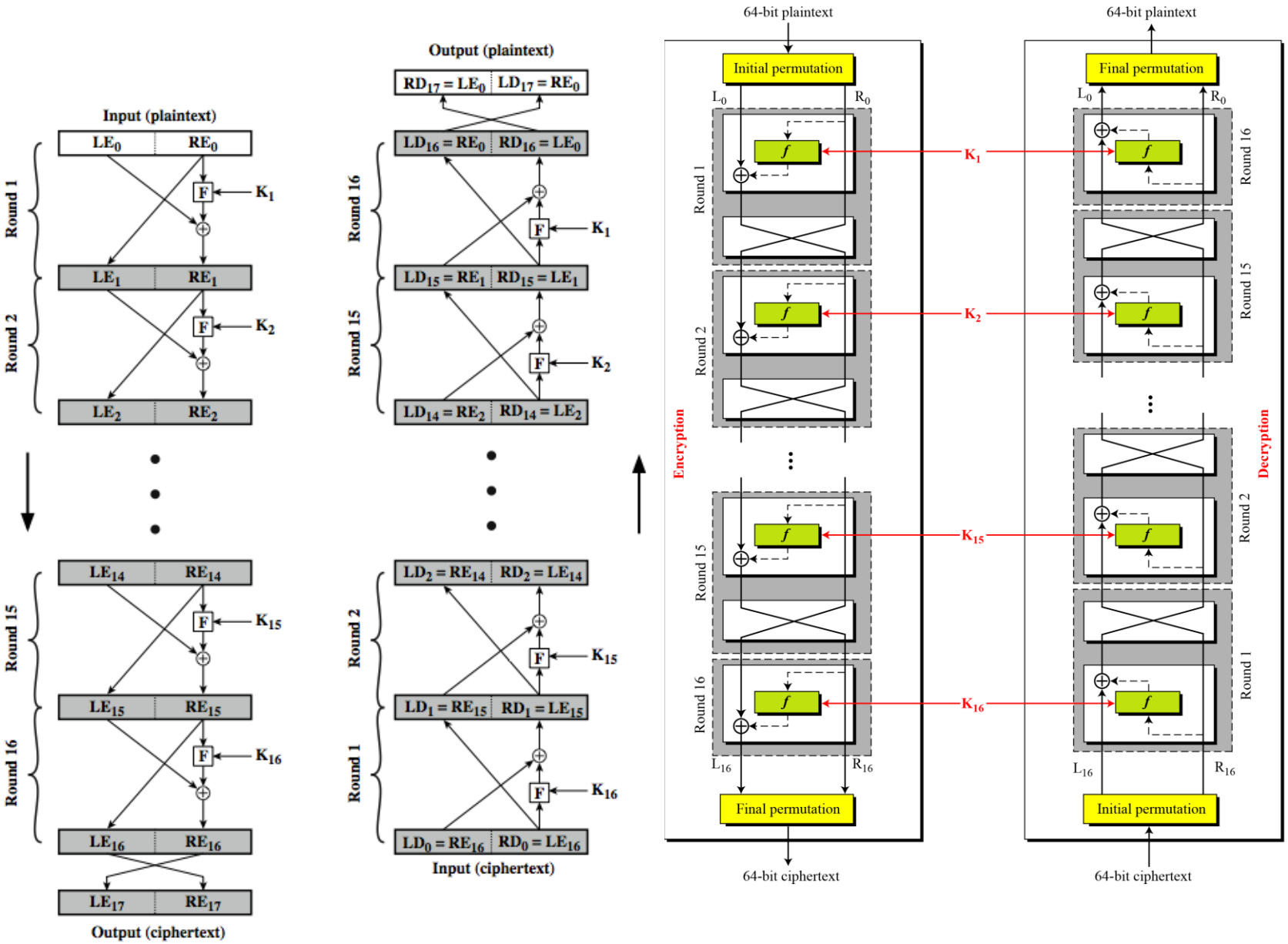
Using mixers and swappers, we can create the cipher and reverse cipher, each having 16 rounds.

First Approach

To achieve this goal, one approach is to make the last round (round 16) different from the others; it has only a mixer and no swapper.

Continued

DES cipher and reverse cipher for the first approach



Continued

Algorithm ***Pseudocode for DES cipher***

```
Cipher (plainBlock[64], RoundKeys[16, 48], cipherBlock[64])
{
    permute (64, 64, plainBlock, inBlock, InitialPermutationTable)
    split (64, 32, inBlock, leftBlock, rightBlock)
    for (round = 1 to 16)
    {
        mixer (leftBlock, rightBlock, RoundKeys[round])
        if (round!=16) swapper (leftBlock, rightBlock)
    }
    combine (32, 64, leftBlock, rightBlock, outBlock)
    permute (64, 64, outBlock, cipherBlock, FinalPermutationTable)
}
```

Continued

Algorithm *Pseudocode for DES cipher* (Continued)

```
mixer (leftBlock[48], rightBlock[48], RoundKey[48])
{
    copy (32, rightBlock, T1)
    function (T1, RoundKey, T2)
    exclusiveOr (32, leftBlock, T2, T3)
    copy (32, T3, rightBlock)
}
swapper (leftBlock[32], rigthBlock[32])
{
    copy (32, leftBlock, T)
    copy (32, rightBlock, leftBlock)
    copy (32, T, rightBlock)
}
```

Continued

Algorithm ***Pseudocode for DES cipher*** **(Continued)**

```
function (inBlock[32], RoundKey[48], outBlock[32])
{
    permute (32, 48, inBlock, T1, ExpansionPermutationTable)
    exclusiveOr (48, T1, RoundKey, T2)
    substitute (T2, T3, SubstituteTables)
    permute (32, 32, T3, outBlock, StraightPermutationTable)
}
```

Continued

Algorithm *Pseudocode for DES cipher* (Continued)

```
substitute (inBlock[32], outBlock[48], SubstitutionTables[8, 4, 16])
{
    for (i = 1 to 8)
    {
        row ← 2 × inBlock[i × 6 + 1] + inBlock [i × 6 + 6]
        col ← 8 × inBlock[i × 6 + 2] + 4 × inBlock[i × 6 + 3] +
              2 × inBlock[i × 6 + 4] + inBlock[i × 6 + 5]

        value = SubstitutionTables [i][row][col]

        outBlock[[i × 4 + 1] ← value / 8;        value ← value mod 8
        outBlock[[i × 4 + 2] ← value / 4;        value ← value mod 4
        outBlock[[i × 4 + 3] ← value / 2;        value ← value mod 2
        outBlock[[i × 4 + 4] ← value
    }
}
```

Continued

Alternative Approach

We can make all 16 rounds the same by including one swapper to the 16th round and add an extra swapper after that (two swappers cancel the effect of each other).

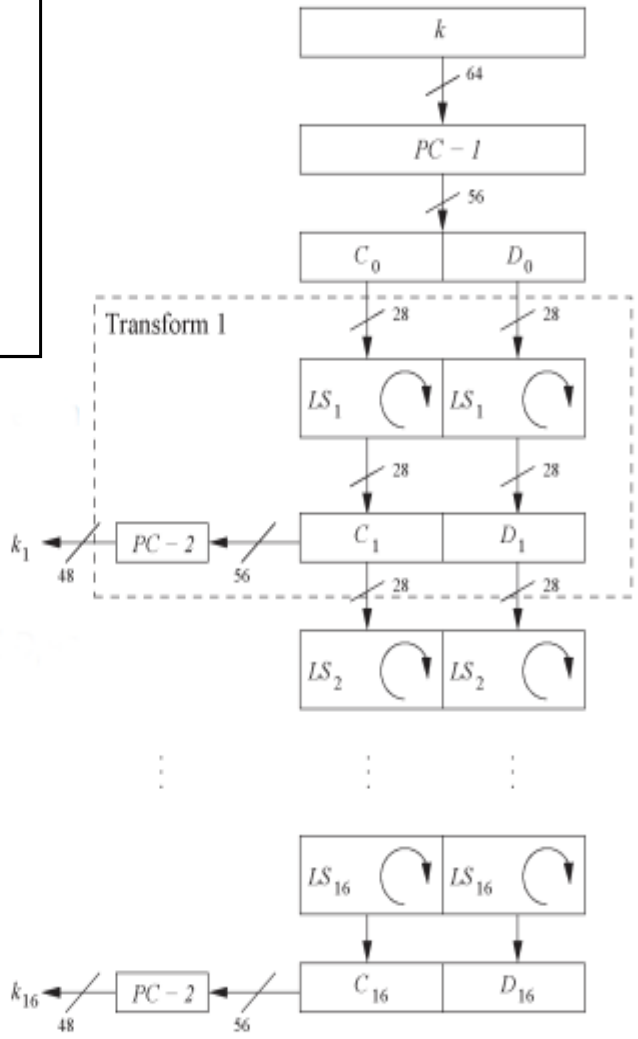
Key Generation

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

Key Generation Process

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

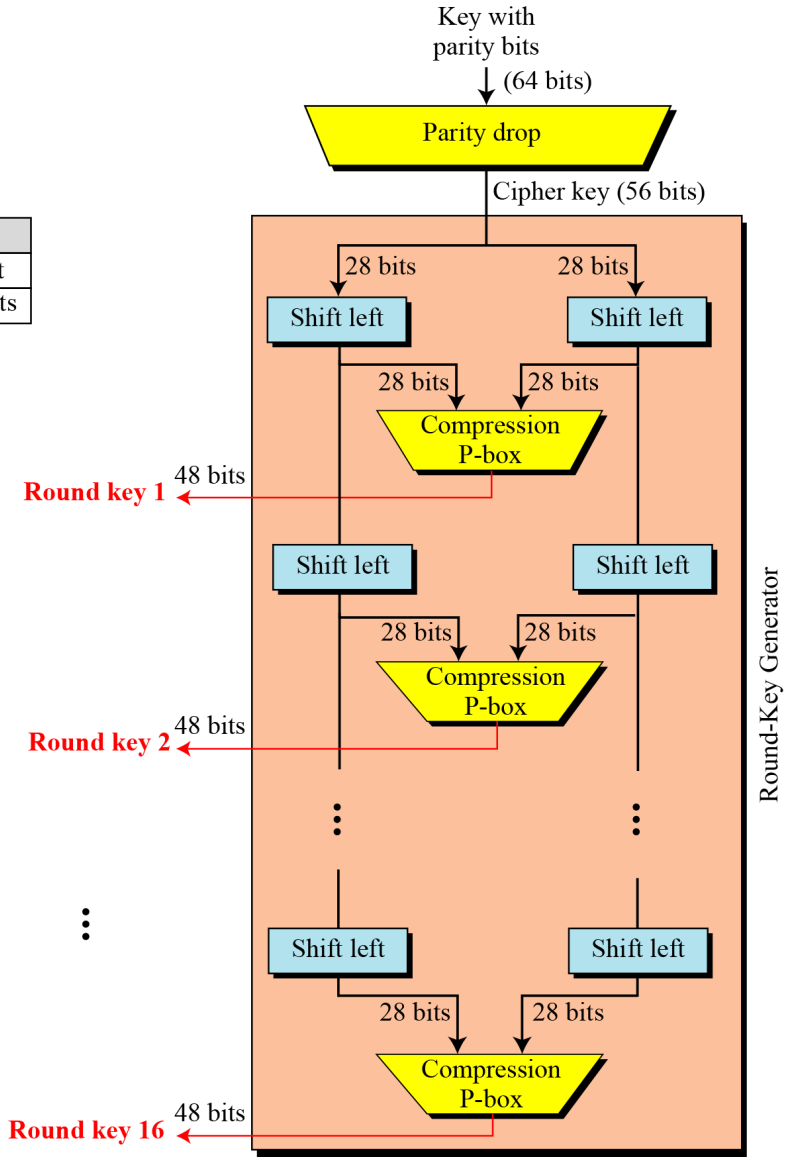
PC – 2							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32



Continued

Shifting

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



Key generation

Continued

- Parity bit drop: it drops parity bits (bits 8,16,24,…64) from 64 bit key and permutes the rest of the bits according to Parity-bit drop table.

Continued

Table ***Parity-bit drop table***

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

continued

- Shift left:

1. Key is divided into two 28 bit parts.
2. Each part is shifted left (circular) one or two bits.
3. The two parts are then combined to form a 56-bit part.

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

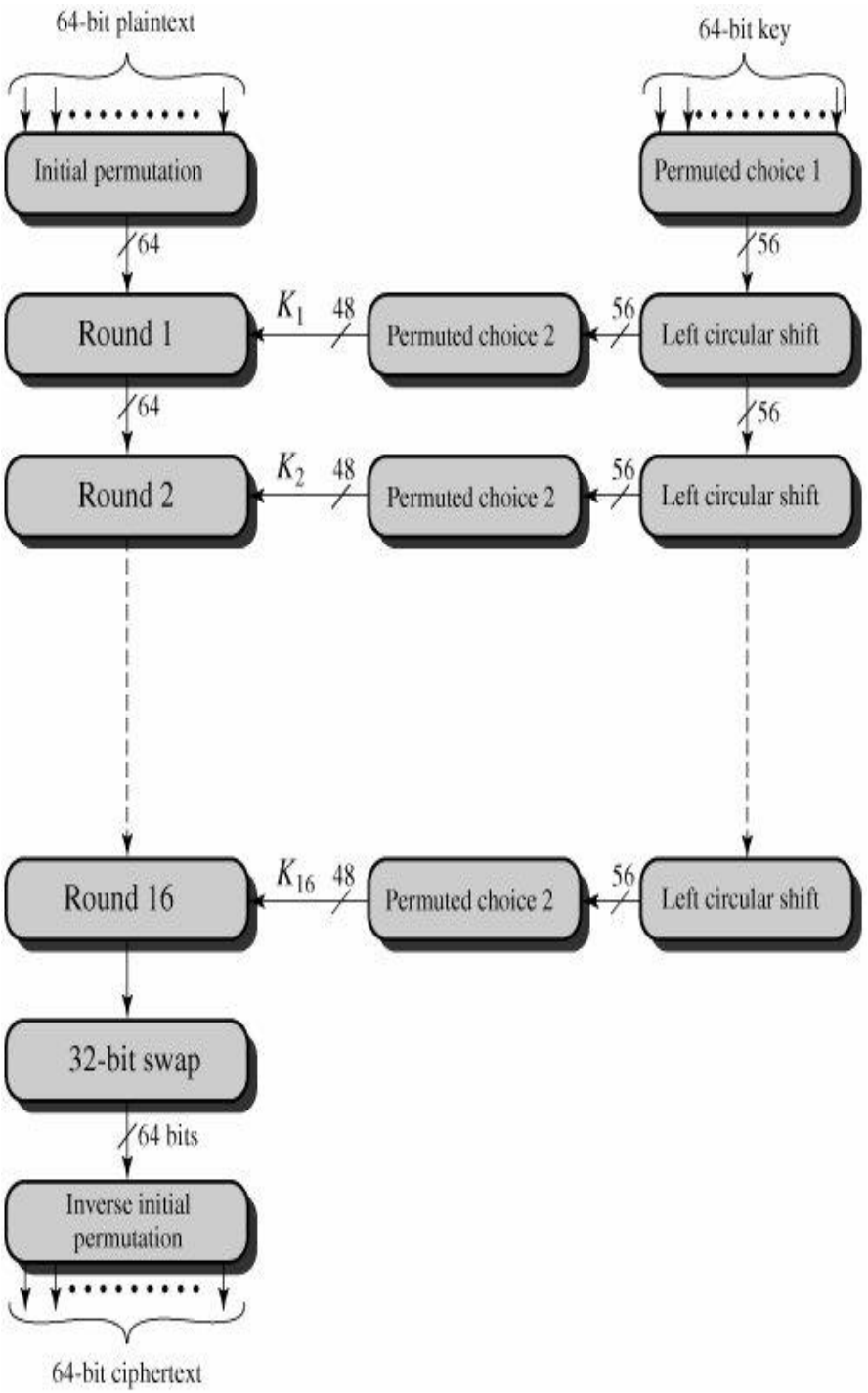
Continued

Compression D-Box:

It changes 56 bits to 48 bits which are used as a key for a round. (drops bits 9,18,22,25,35,38,43,54)

Table ***Key-compression table***

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32


64-bit plaintext
64-bit key
Initial permutation
Permuted choice 1
64
56
Round 1
K_1
48
Permuted choice 2
56
Left circular shift
64
56
Round 2
K_2
48
Permuted choice 2
56
Left circular shift
Round 16
K_{16}
48
Permuted choice 2
56
Left circular shift
32-bit swap
64 bits
Inverse initial permutation
64-bit ciphertext

Continued

Algorithm ***Algorithm for round-key generation***

```
Key_Generator (keyWithParities[64], RoundKeys[16, 48], ShiftTable[16])
{
    permute (64, 56, keyWithParities, cipherKey, ParityDropTable)
    split (56, 28, cipherKey, leftKey, rightKey)
     for (round = 1 to 16)
     {
          shiftLeft (leftKey, ShiftTable[round])
          shiftLeft (rightKey, ShiftTable[round])
          combine (28, 56, leftKey, rightKey, preRoundKey)
          permute (56, 48, preRoundKey, RoundKeys[round], KeyCompressionTable)
     }
}
```

Continued

Algorithm ***Algorithm for round-key generation (Continue)***

```
shiftLeft (block[28], numOfShifts)
{
    for (i = 1 to numOfShifts)
    {
        T ← block[1]
        for (j = 2 to 28)
        {
            block [j−1] ← block [j]
        }
        block[28] ← T
    }
}
```

Example

We choose a random plaintext block and a random key, and determine what the ciphertext block would be (all in hexadecimal):

Plaintext: 123456ABCD132536 Key: AABB09182736CCDD
CipherText: C0B7A8D05F3A829C

Trace of data for Example

Plaintext: 123456ABCD132536			
*After initial permutation:*14A7D67818CA18AD After splitting: L_0=14A7D678 R_0=18CA18AD			
Round	*Left*	*Right*	*Round Key*
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3

Continued

Trace of data for Example (Conintued)

Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			*(after final permutation)*

Continued

Example:

Let us see how Bob, at the destination, can decipher the ciphertext received from Alice using the same key. Table shows some interesting points.

Ciphertext: C0B7A8D05F3A829C			
After initial permutation: 19BA9212CF26B472 After splitting: L_0=19BA9212 R_0=CF26B472			
Round	*Left*	*Right*	*Round Key*
Round 1	CF26B472	BD2DD2AB	181C5D75C66D
Round 2	BD2DD2AB	387CCDAA	3330C5D9A36D
. . .	. . .	. . .	. . .
Round 15	5A78E394	18CA18AD	4568581ABCCE
Round 16	14A7D678	18CA18AD	194CD072DE8C
After combination: 14A7D67818CA18AD			
Plaintext:123456ABCD132536 (after final permutation)			

DES Design Controversy

- DES was subjected to intense and continuing criticism over :
  - choice of 56-bit key (vs Lucifer 128-bit)
  - the classified design criteria
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
  - especially in financial applications
  - still standardized for legacy application use
  - 3DES still strong (112 bit key)

DES ANALYSIS

Critics have used a strong magnifier to analyze DES. Tests have been done to measure the strength of some desired properties in a block cipher.

Properties

Two desired properties of a block cipher are the avalanche effect and the completeness.

- **Avalanche effect** – A small change in plaintext results in the very great change in the ciphertext.
- **Completeness** – Each bit of ciphertext depends on many bits of plaintext.

Properties (continued…)

avalanche effect

Example

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

Continued

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits. This means that changing approximately 1.5 percent of the plaintext creates a change of approximately 45 percent in the ciphertext.

Number of bit differences for Example

Rounds	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

Design Criteria

S-Boxes

The design provides confusion and diffusion of bits from each round to the next.

D-Boxes

They provide diffusion of bits.

Number of Rounds

DES uses sixteen rounds of Feistel ciphers. the ciphertext is thoroughly a random function of plaintext and ciphertext.

DES Weaknesses

During the last few years critics have found some weaknesses in DES.

Weaknesses in Cipher Design

1. ***Weaknesses in S-boxes***
2. ***Weaknesses in D-boxes***
3. ***Weaknesses in Key***

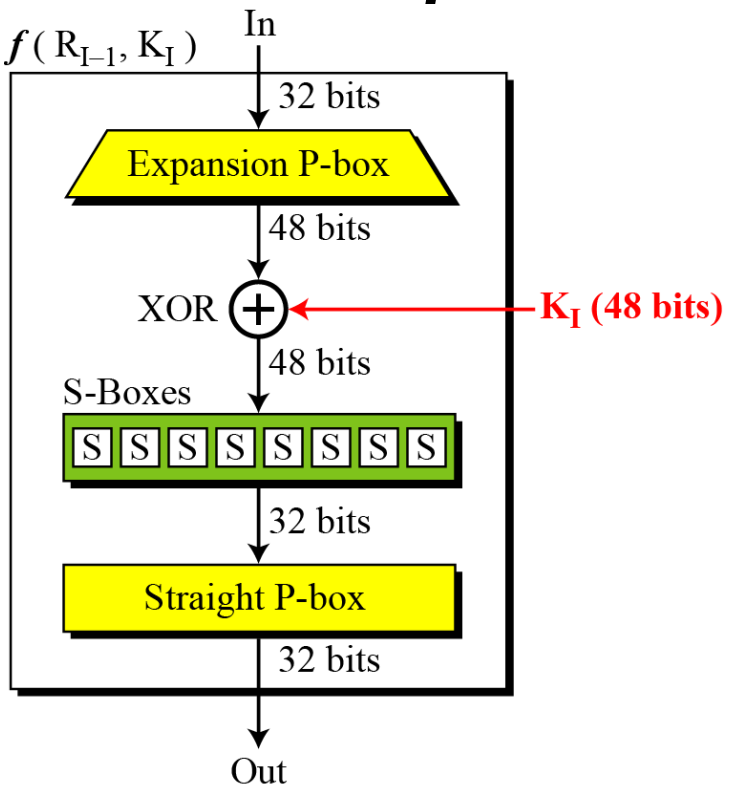
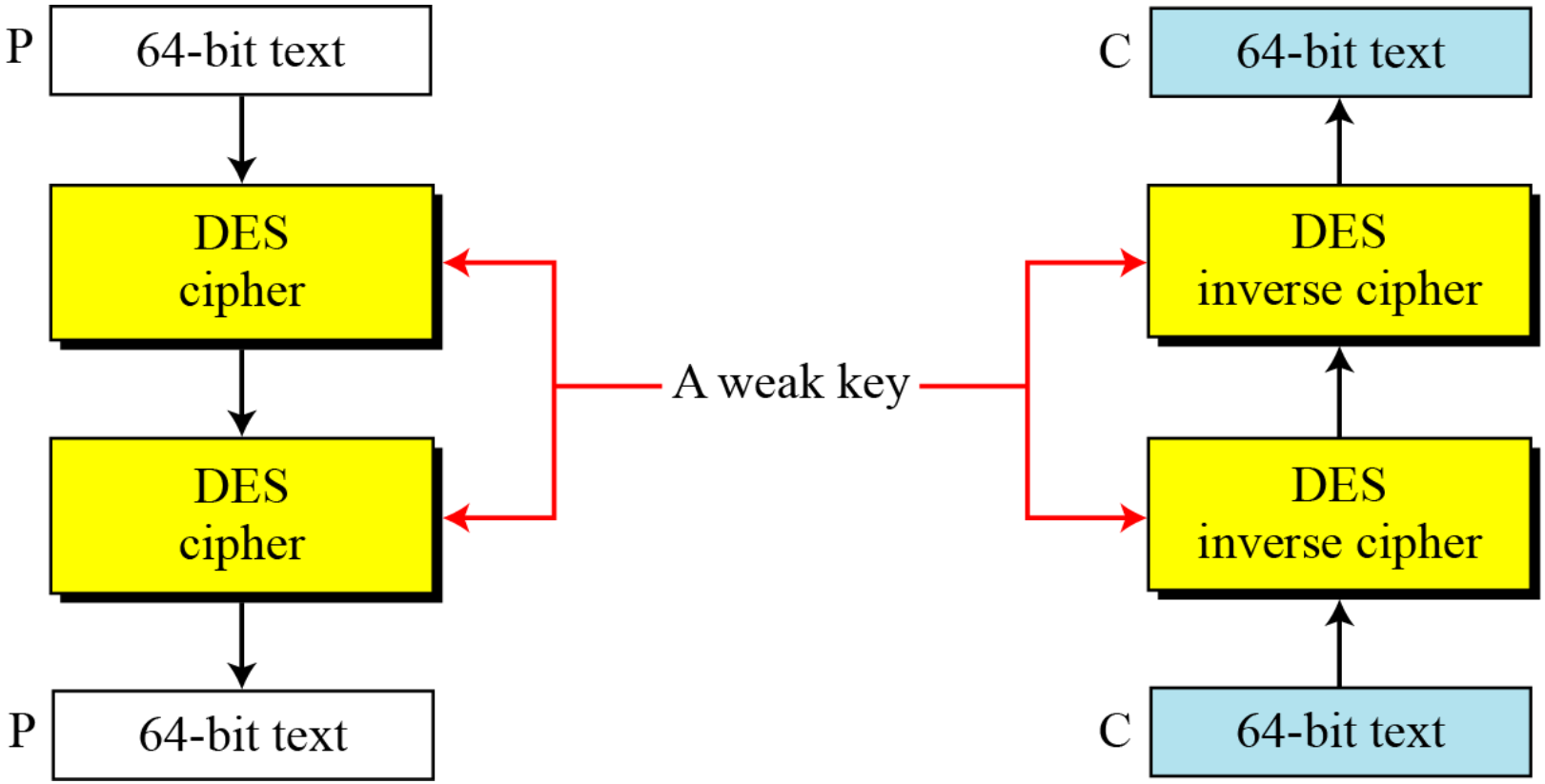


Table 6.18 *Weak keys*

Keys before parities drop (64 bits)	*Actual key (56 bits)*
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Continued

Double encryption and decryption with a weak key



Continued

Example

Let us try the first weak key in previous table to encrypt a block two times. After two encryptions with the same key the original plaintext block is created. Note that we have used the encryption algorithm two times, not one encryption followed by another decryption.

Key: 0x0101010101010101
Plaintext: *0x123456788765432 1* Ciphertext: 0x814FE938589154F7

Key: 0x0101010101010101
Plaintext: 0x814FE938589154F7 Ciphertext: *0x123456788765432 1*

Continued

Table 6.19 *Semi-weak keys*

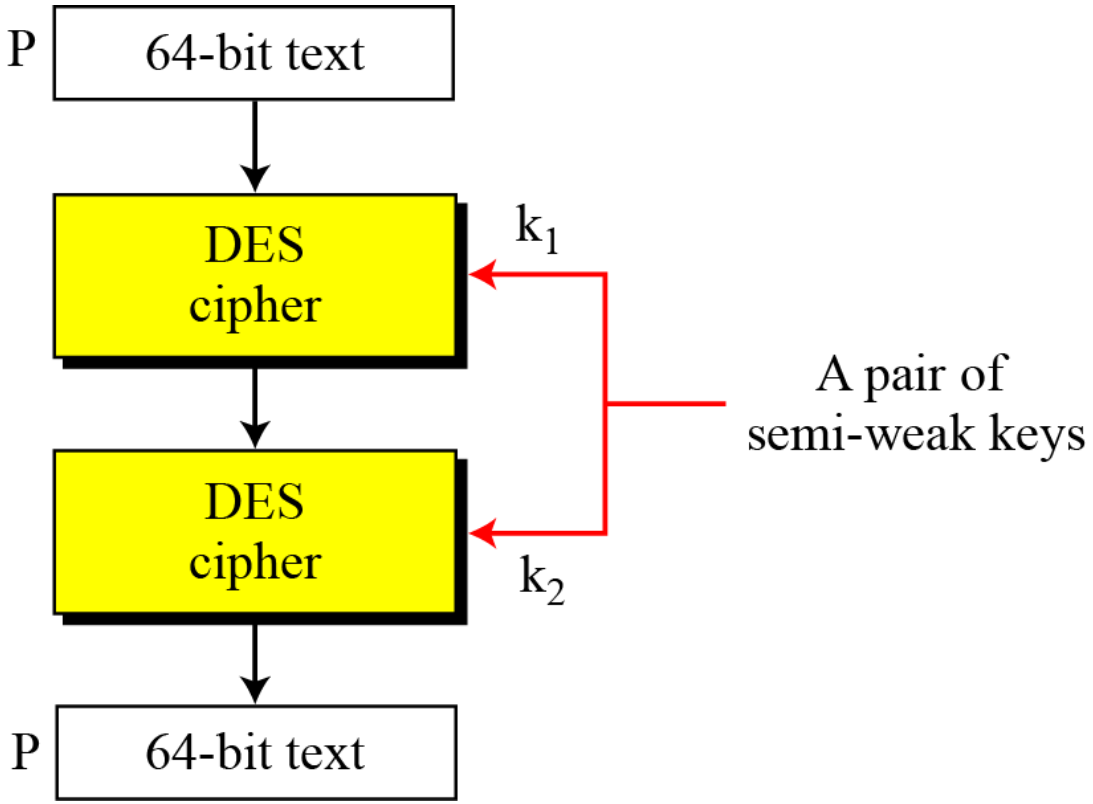
First key in the pair	*Second key in the pair*
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

Continued

Round key 1	9153E54319BD	6EAC1ABCE642
Round key 2	6EAC1ABCE642	9153E54319BD
Round key 3	6EAC1ABCE642	9153E54319BD
Round key 4	6EAC1ABCE642	9153E54319BD
Round key 5	6EAC1ABCE642	9153E54319BD
Round key 6	6EAC1ABCE642	9153E54319BD
Round key 7	6EAC1ABCE642	9153E54319BD
Round key 8	6EAC1ABCE642	9153E54319BD
Round key 9	9153E54319BD	6EAC1ABCE642
Round key 10	9153E54319BD	6EAC1ABCE642
Round key 11	9153E54319BD	6EAC1ABCE642
Round key 12	9153E54319BD	6EAC1ABCE642
Round key 13	9153E54319BD	6EAC1ABCE642
Round key 14	9153E54319BD	6EAC1ABCE642
Round key 15	9153E54319BD	6EAC1ABCE642
Round key 16	6EAC1ABCE642	9153E54319BD

Continued

A pair of semi-weak keys in encryption and decryption



Possible weak keys

There are 48 keys that are called possible weak keys. It is a key that creates only four distinct round keys.

Continued

Example

What is the probability of randomly selecting a weak, a semi-weak, or a possible weak key?

Solution

DES has a key domain of 2^{56}. The total number of the above keys are 64 (4 + 12 + 48). The probability of choosing one of these keys is 8.8×10^{-16}, almost impossible.

Key Compliment

- It is observed that half of the keys are compliment of the other half.
- It simplifies job of the cryptanalysis

$$C = E(K, P) \rightarrow \overline{C} = E(\overline{K}, \overline{P})$$

Multiple DES

- The major criticism of DES regards its key length.
- We can use double or triple DES to increase the key size.

Double DES

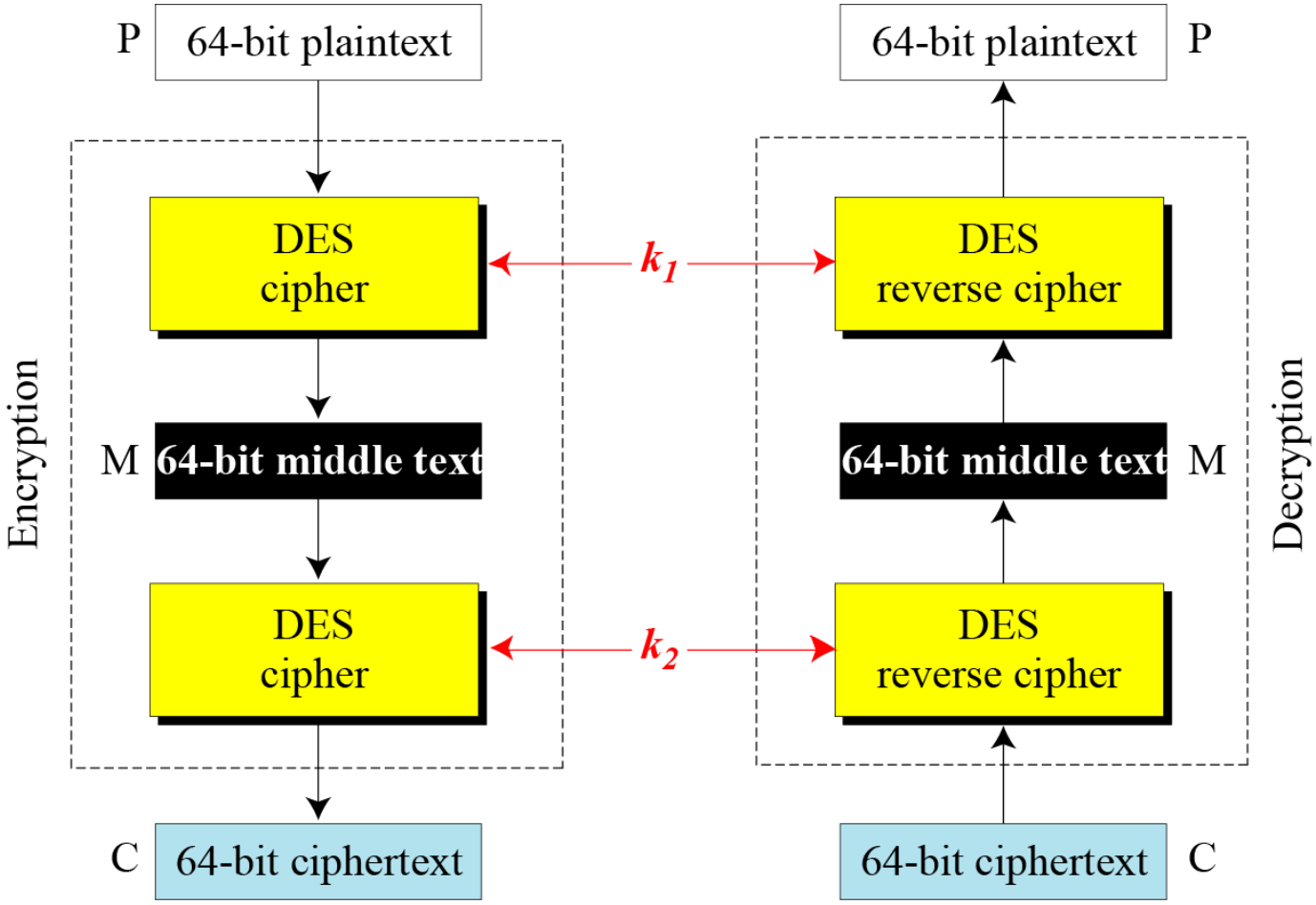
The first approach is to use double DES (2DES).

Meet-in-the-Middle Attack

However, using a known-plaintext attack called meet-in-the-middle attack proves that double DES improves this vulnerability slightly (to 2^{57} tests), but not tremendously (to 2^{112}).

Continued

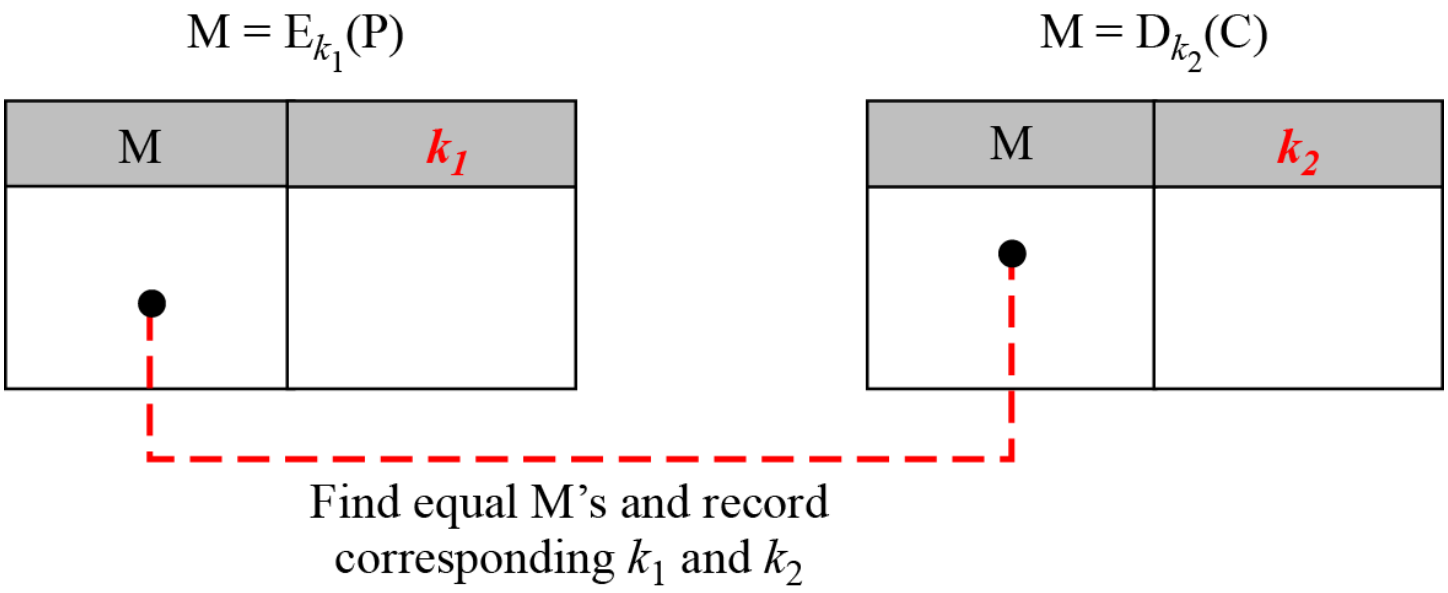
Figure ***Meet-in-the-middle attack for double DES***



$$M = E_{k_1}(P) \quad \text{and} \quad M = D_{k_2}(C)$$

Continued

Figure ***Tables for meet-in-the-middle attack***

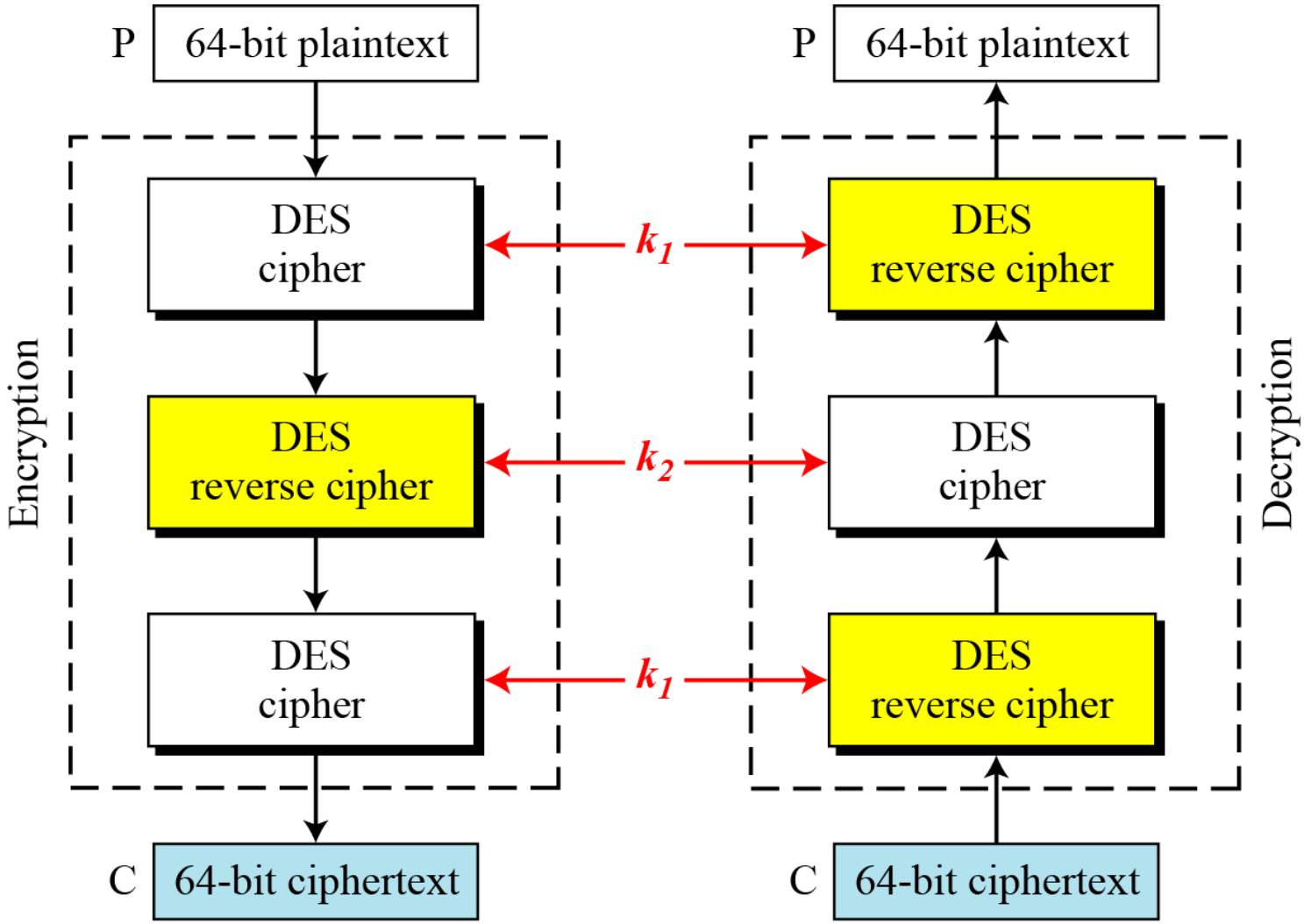


Triple DES

There are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2-key Triple DES (2TDES).

Triple DES

Figure *Triple DES with two keys*



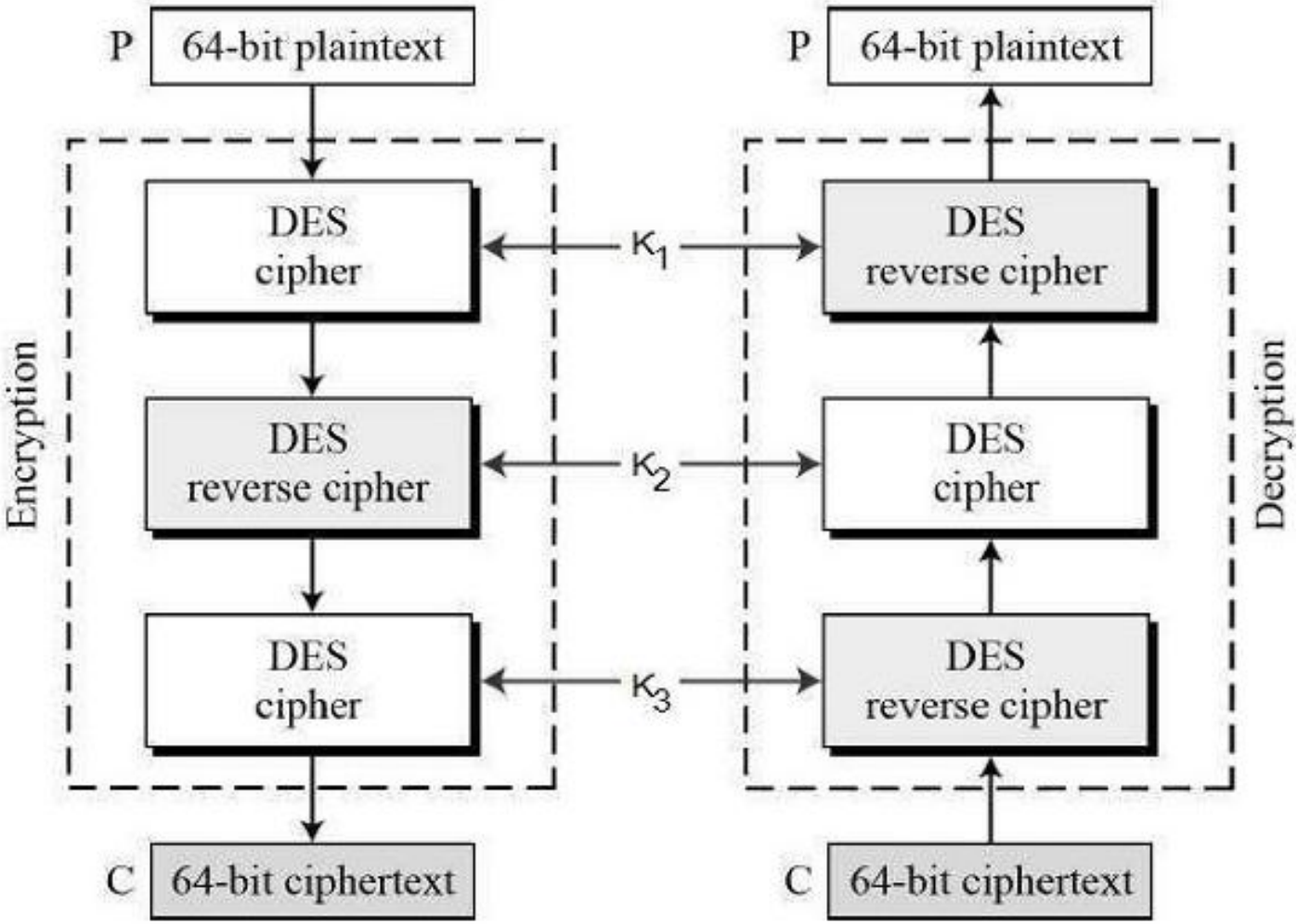
Continuous

Triple DES with Three Keys

- Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1, K_2 and K_3.
- This means that the actual 3TDES key has length 3×56 = 168 bits.

Continued…

Figure ***Triple DES with three keys***



Continued…

The encryption-decryption process

- Encrypt the plaintext blocks using single DES with key K_1.
- Now decrypt the output of step 1 using single DES with key K_2.
- Finally, encrypt the output of step 2 using single DES with key K_3.
- The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.

Security of DES

DES, as the first important block cipher, has gone through much scrutiny. Among the attempted attacks, three are of interest: brute-force, differential cryptanalysis, and linear cryptanalysis.

Brute-Force Attack

Combining key size weakness with the key complement weakness, it is clear that DES can be broken using 2^{55} encryptions.

Differential Cryptanalysis

- It has been revealed that the designers of DES already knew about this type of attack and designed S-boxes and chose 16 as the number of rounds to make DES specifically resistant to this type of attack.
- It has been shown that DES can be broken using 2^{47} pairs of chosen plaintexts or 2^{55} known plaintexts.
-

Linear Cryptanalysis

- Linear cryptanalysis is newer than differential cryptanalysis. DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis.
- S-boxes are not very resistant to linear cryptanalysis.
- It has been shown that DES can be broken using 2^{43} pairs of known plaintexts.
- However, from the practical point of view, finding so many pairs is very unlikely.

References:

1. Stallings
2. Forouzan