
CHAPTER 6

Data Encryption Standard (DES)

(Solution to Practice Set)

Review Questions

1. The block size in DES is 64 bits. The cipher key size is 56 bits. The round key size is 48 bits.
2. DES uses 16 rounds.
3. In the first approach, DES uses 16 mixers and 15 swappers in encryption or decryption algorithm; in the second (alternative approach), DES use 16 mixers and 16 swappers in encryption or decryption algorithm.
4. In DES, encryption or decryption uses $16 \times 2 + 2 = 34$ permutations, because each mixer uses two permutations and there are two permutations before and after the rounds. The round-key generator uses 17 permutation operations: one parity drop and 16 compression permutation operations for each round.
5. The total number of exclusive-or operations is $16 \times 2 = 32$, because each round uses two exclusive-or operations (one inside the function and one outside of the function).
6. The input to the function is a 32-bit word, but the round-key is a 48-bit word. The expansion permutation is needed to increase the number of bits in the input word to 48.
7. The cipher key that is used for DES include the parity bits. To remove the parity bits and create a 56-bit cipher key, a parity drop permutation is needed. Not only does the parity-drop permutation drop the parity bits, it also permutes the rest of the bits.

8. A weak key is the one that, after parity drop operation, consists either of all 0s, all 1s, or half 0s and half 1s. Each weak key is the inverse of itself: $E_k(E_k(P)) = P$. A semi-weak key creates only two different round keys and each of them is repeated eight times. The semi-weak round keys come in pairs, where a key in the pair is the inverse of the other key in the pair: $E_{k_1}(E_{k_2}(P)) = P$. A possible weak key is a key that creates only four distinct round keys; in other words, the sixteen round keys are divided into four groups and each group is made of four equal round keys.
-
9. Double DES uses two instances of DES ciphers for encryption and two instances of reverse ciphers for decryption. Each instance uses a different key, which means that the size of the key is 112 bits. However, double DES is vulnerable to meet-in-the-middle attack.
10. Triple DES uses three stages of DES for encryption and decryption. Two versions of triple DES are in use today: triple DES with two keys and triple DES with three keys. In triple DES with two keys, there are only two keys: K_1 and K_2 . The first and the third stages use K_1 ; the second stage uses K_2 . In triple DES with three keys, there are three keys: K_1 , K_2 , and K_3 .

Exercises

11.

a.

Input: 1 1011 1 → **3, 11** → **Output: 03 (0011)**

b.

Input: 0 0110 0 → **0, 6** → **Output: 09 (1001)**

c.

Input: 0 0000 0 → **0, 0** → **Output: 04 (0100)**

d.

Input: 1 1111 1 → **3, 15** → **Output: 09 (1001)**

12. The following table shows the output from all boxes. No pattern can be found:

<i>Input</i>	<i>Box 1</i>	<i>Box 2</i>	<i>Box 3</i>	<i>Box 4</i>	<i>Box 5</i>	<i>Box 6</i>	<i>Box 7</i>	<i>Box 8</i>
000000	1110	1111	1010	0111	0010	1100	0100	1101

13. The following table shows the output from all boxes. No pattern can be found:

<i>Input</i>	<i>Box 1</i>	<i>Box 2</i>	<i>Box 3</i>	<i>Box 4</i>	<i>Box 5</i>	<i>Box 6</i>	<i>Box 7</i>	<i>Box 8</i>
111111	1101	1001	1100	1110	0011	0011	1101	1011

14.

a. The following shows that 3 bits will be changed in the output.

Input: 0 0000 0 → **00, 00** → **Output: 10 (1010)**
Input: 0 0000 1 → **01, 00** → **Output: 13 (1101)**

b. The following shows that 2 bits will be changed in the output.

Input: 1 1111 1 → **03, 15** → **Output: 09 (1001)**
Input: 1 1101 1 → **03, 14** → **Output: 14 (1100)**

15.

a. The following shows that 2 bits will be changed in the output.

Input: 0 0110 0 → **00, 06** → **Output: 03 (0011)**
Input: 0 0000 0 → **00, 00** → **Output: 15 (1111)**

b. The following shows that 2 bits will be changed in the output.

Input: 1 1001 1	→	03, 09	→	Output: 06 (0110)
Input: 1 1111 1	→	03, 15	→	Output: 09 (1001)

16.

a. The following shows that two outputs are different.

Input: 0 0110 0	→	00, 06	→	Output: 09 (1001)
Input: 1 1000 0	→	02, 08	→	Output: 15 (1111)

b. The following shows that two outputs are different.

Input: 1 1001 1	→	03, 09	→	Output: 04 (0100)
Input: 0 0111 1	→	01, 07	→	Output: 03 (0011)

17. The following table shows 32 input pairs and 32 output pairs. The last column is the difference between the outputs.

<i>Input pairs</i>		<i>Output pairs</i>		<i>d</i>
000000	000001	0010	1110	1100
000010	000011	1100	1011	0111
000100	000101	0100	0010	0110
000110	000111	0001	1100	1111
001000	001001	0111	0100	0011
001010	001011	1010	0111	1001
001100	001101	1011	1101	0110
001110	001111	0110	0001	0111
010000	010001	1000	0101	1101
010010	010011	0101	0000	0101
010100	010101	0011	1111	1100
010110	010111	1111	1010	0101
011000	011001	1101	0011	1110
011010	011011	0000	1001	1001
011100	011101	1110	1000	0110
011110	011111	1001	0110	1111
100000	100001	0110	1011	1101
100010	100011	0010	1000	1010
100100	100101	0001	1100	1101
100110	100111	1011	0111	1100

011010	011011	0000	1001	1001
011100	011101	1110	1000	0110
011110	011111	1001	0110	1111
100000	100001	0110	1011	1101
100010	100011	0010	1000	1010
100100	100101	0001	1100	1101
100110	100111	1011	0111	1100
101000	101001	1010	0001	1011
101010	101011	1101	1110	0011
101100	101101	0111	0010	0101
101110	101111	1000	1101	0101
110000	110001	1111	0110	1001
110010	110011	1001	1111	0110
110100	110101	1100	0000	1100
110110	110111	0101	1001	1100
111000	111001	0110	1010	1100
111010	111011	0011	0100	0111
111100	111101	0000	0101	0101
111110	111111	1110	0011	1101

If we sort the table on last column (output differences), we get the following: two (0011)'s, five (0101)'s, four (0110)'s, three (0111)'s, three (0111)'s, one (1010), one (1011), one (1100), five (1100)'s, four (1101)'s, one (1110), and two (1111)'s.

None of the group has a size larger than eight.

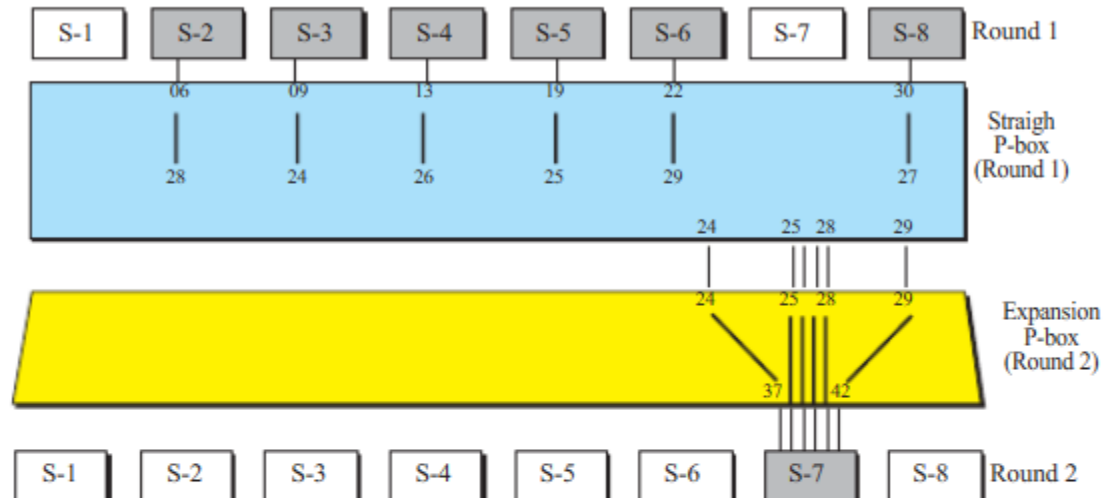
- 18.** For S-Box 7, we set the first (leftmost) input bit to 0. We change the other five input bits. We then observe one of the output bits (we have chosen the third bit from the left) and count how many 0's and how many 1's we obtain for this bit. These two counts must be close to each other. The following table shows inputs and outputs.

<i>Input</i>	<i>Output</i>	<i>Third bit</i>
0 0000 0	0 1 0 0	0
0 0000 1	1 1 0 1	0
0 0001 0	1 0 1 1	1
0 0001 1	0 0 0 0	0
0 0010 0	0 0 1 0	1
0 0010 1	1 0 1 1	1
0 0011 0	1 1 1 0	1
0 0011 1	0 1 1 1	1
0 0100 0	1 1 1 1	1
0 0100 1	0 1 0 0	0
0 0101 0	0 0 0 0	0
0 0101 1	1 0 0 0	0
0 0110 0	1 0 0 0	0
0 0110 1	0 0 0 1	0
0 0111 0	1 1 0 1	0
0 0111 1	1 0 1 0	1
0 1000 0	0 0 1 1	1
0 1000 1	1 1 1 0	1
0 1001 0	1 1 0 0	0
0 1001 1	0 0 1 1	1
0 1010 0	1 0 0 1	0
0 1010 1	0 1 0 1	0
0 1011 0	0 1 1 1	1
0 1011 1	1 1 0 0	0
0 1100 0	0 1 0 1	0
0 1100 1	0 0 1 0	1
0 1101 0	1 0 1 0	1
0 1101 1	1 1 1 1	1
0 1110 0	0 0 0 0	0
0 1110 1	1 0 0 0	0
0 1111 0	0 0 0 1	0
0 1111 1	0 1 1 0	1

As the table shows we have **17 0's** and **15 1's** close enough.

- 19.** Figure S6.19 shows the situation. The inputs to S-box 7 in round 2 comes from six different S-boxes in round 1.

Figure S6.19 *Solution to Exercise 19*

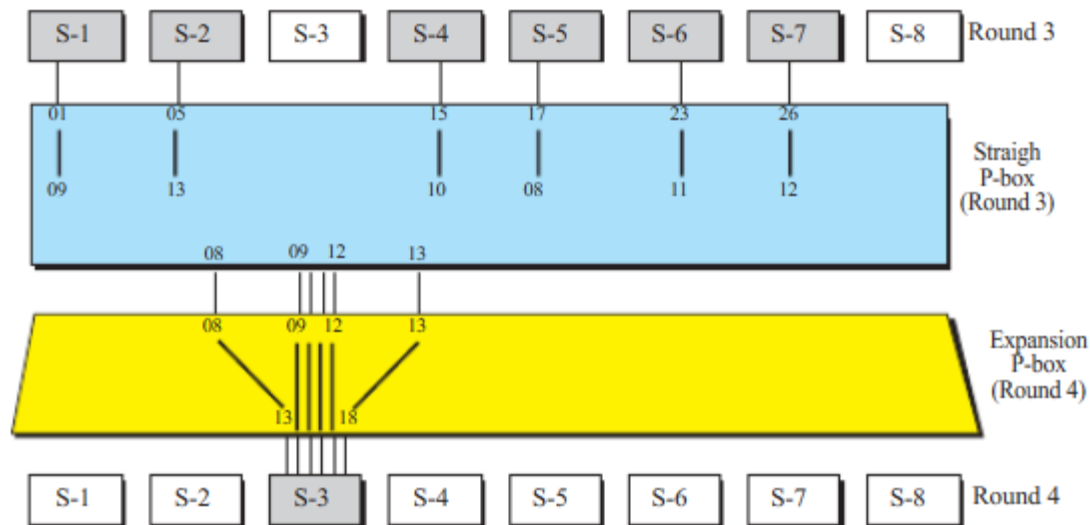


- a.** The six inputs to S-box 7 in round 2 come from six outputs (37, 38, 39, 40, 41, 42) of expansion permutation box.
- b.** The above six outputs correspond to the six inputs (24, 25, 26, 27, 28, 29) in the expansion permutation box (See Table 6.2 in the textbook).
- c.** The above six inputs correspond to the six inputs (09, 19, 13, 30, 06, 22) inputs in the straight permutation box (See Table 6.11 in the textbook).
- d.** The above six inputs correspond to the outputs of six different S-Boxes (S-3, S-5, S-4, S-8, S-2, and S-6) in round 1.

- d. The above six inputs correspond to the outputs of six different S-Boxes (S-3, S-5, S-4, S-8, S-2, and S-6) in round 1.

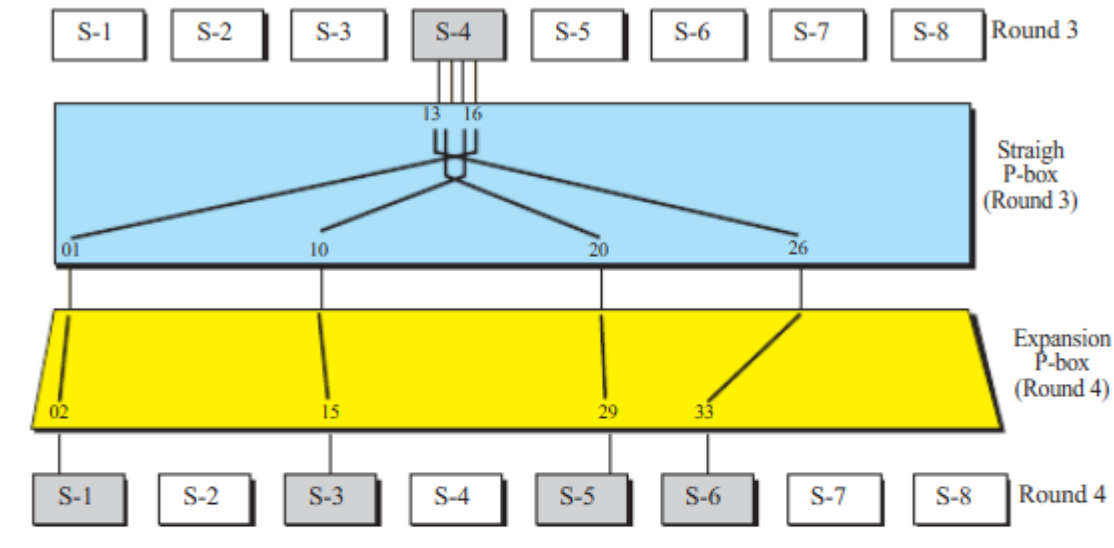
20. Figure S6.20 shows the situation. The inputs to S-box 7 in round 2 comes from six S-boxes in round 1.

Figure S6.20 *Solution to Exercise 20*



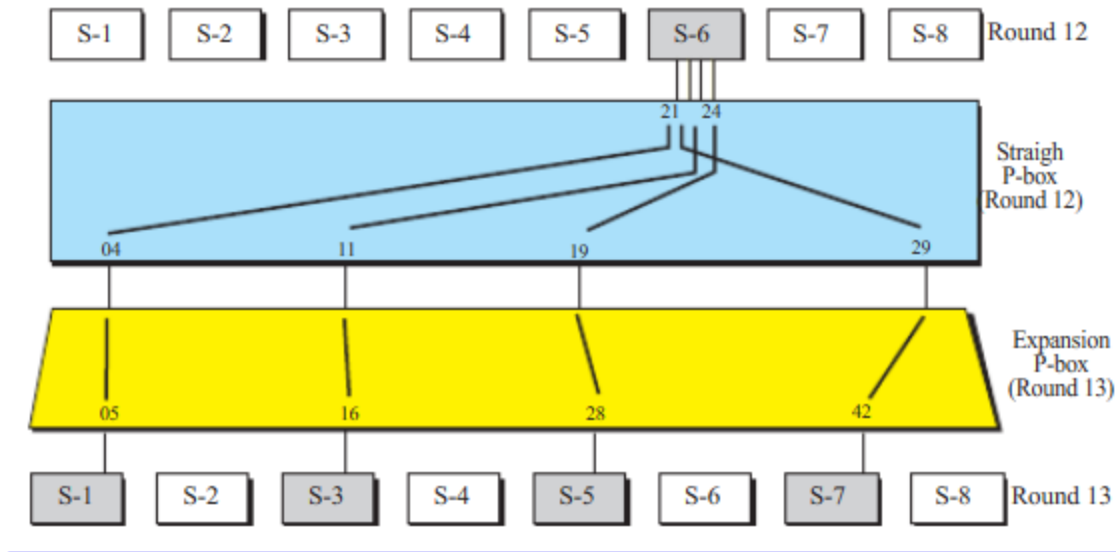
- a. The six inputs to S-box 3 in round 4 come from six outputs (13, 14, 15, 16, 17, 18) of expansion permutation box in round 4.
- b. The above six outputs corresponds to the six inputs (08, 09, 10, 11, 12, 13) in the expansion permutation box in round 4 (See Table 6.2 in the textbook).
- c. The above six inputs correspond to the six inputs (17, 01, 15, 23, 26, 05) inputs in the straight permutation box (See Table 6.11 in the textbook).
- d. The above six inputs correspond to the outputs of six different S-Boxes (S-5, S-1, S-4, S-6, S-7, and S-2) in round 3. None of them come from S-3.
21. Figure S6.21 shows the situation. The outputs from S-box 4 in round 3 go to four S-boxes in round 4.

Figure S6.21 *Solution to Exercise 21*



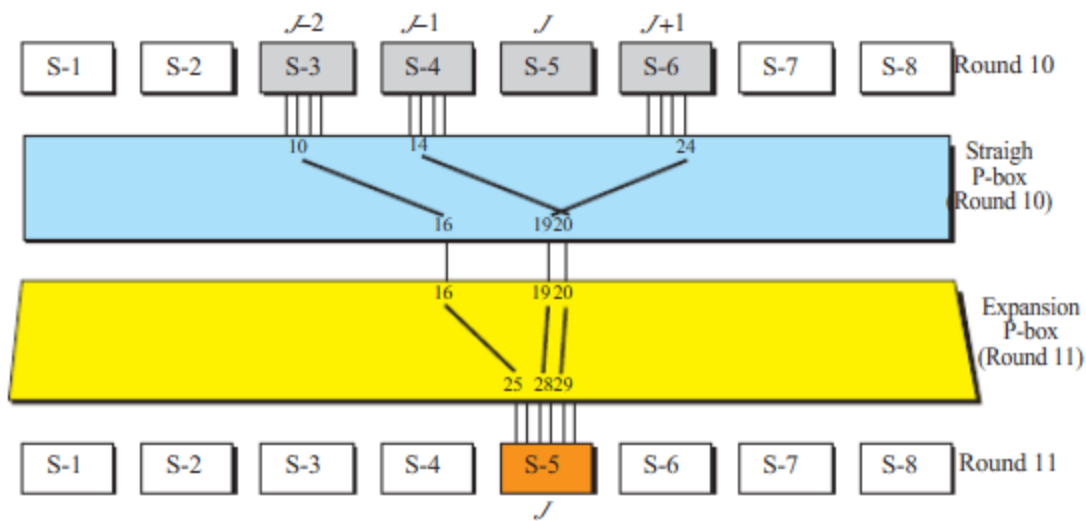
- a. The four outputs from S-box 3 in round 4 go to six outputs (26, 20, 10, 01) of straight permutation box (See Table 6.1).
 - b. The above four outputs correspond to four outputs (33, 29, 15, 02) in the expansion permutation box (See Table 6.11).
 - c. The above four inputs corresponds to the inputs of four different S-Boxes (S-6, S-5, S-3, and S-1) in round 4.
22. Figure S6.22 shows the situation. The outputs from S-box 6 in round 12 goes to four different S-boxes in round 13. None of them go to S-box 6. So the criterion cannot actually be tested by these exercise, but it does not violate the criterion either.

Figure S6.22 *Solution to Exercise 22*



23. Figure S6.23 shows the situation. We assume $j=5$.

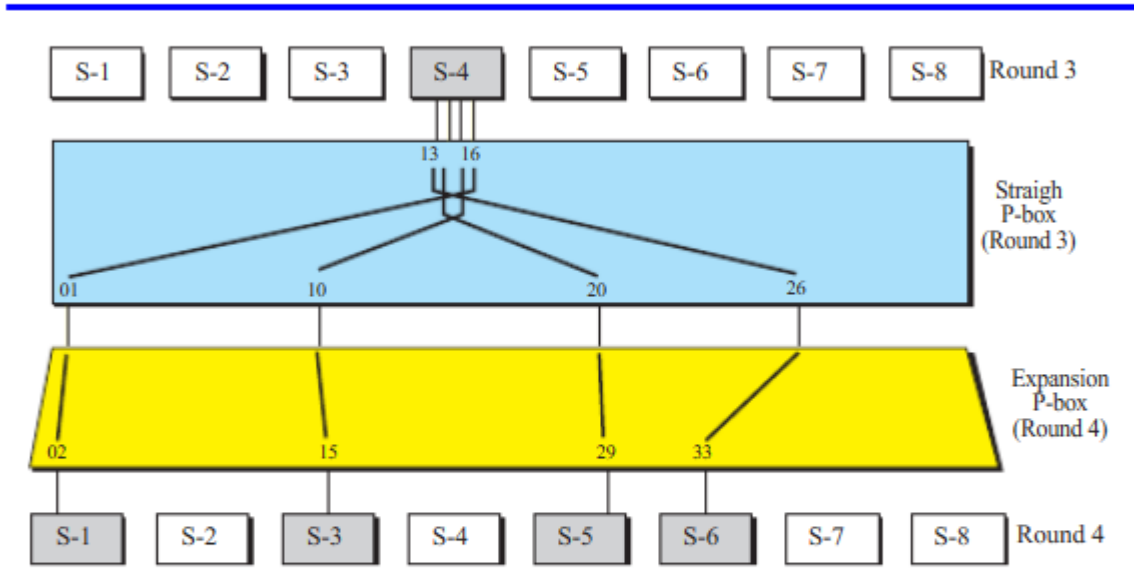
Figure S6.23 *Solution to Exercise 23*



- a.** One output from S-box 3 (output 10) goes to the first input of S-box 5 (input 25).
- b.** An output from S-box 4 (output 14) goes one the last two inputs of S-box 6 (input 29).
- c.** An output from S-box 6 (output 24) goes to one of the middle input of S-box 5 (input 19).

- 24.** The solution can be found in Figure S6.24. The outputs of S-4 is distributed between S-1, S-3, S-5, and S-7 in the next round.

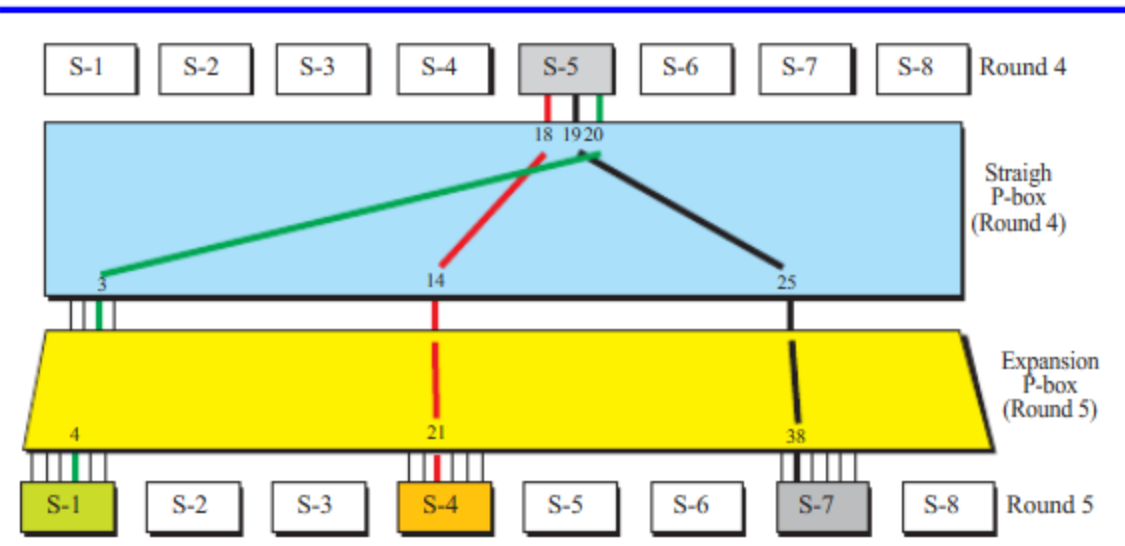
Figure S6.24 *Solution to Exercise 24*



- The output 16 goes to bit 2 (one of the first two bits of S-box 1).
- The output 14 goes to bit 29 (the last bit of S-box 5).
- The output 15 goes to bit 15 (one of middle bit of S-box 3).
- The output 13 goes to bit 33 (one of the middle bit of S-6).

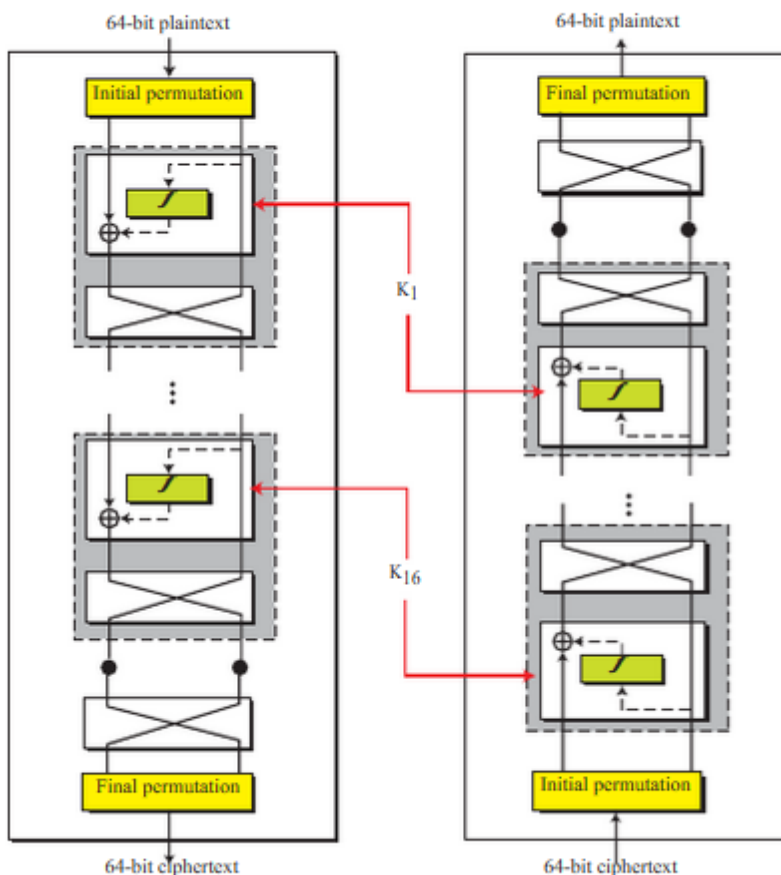
- 25.** Figure S6.25 will help us in this problem.

Figure S6.25 *Solution to Exercise 25*



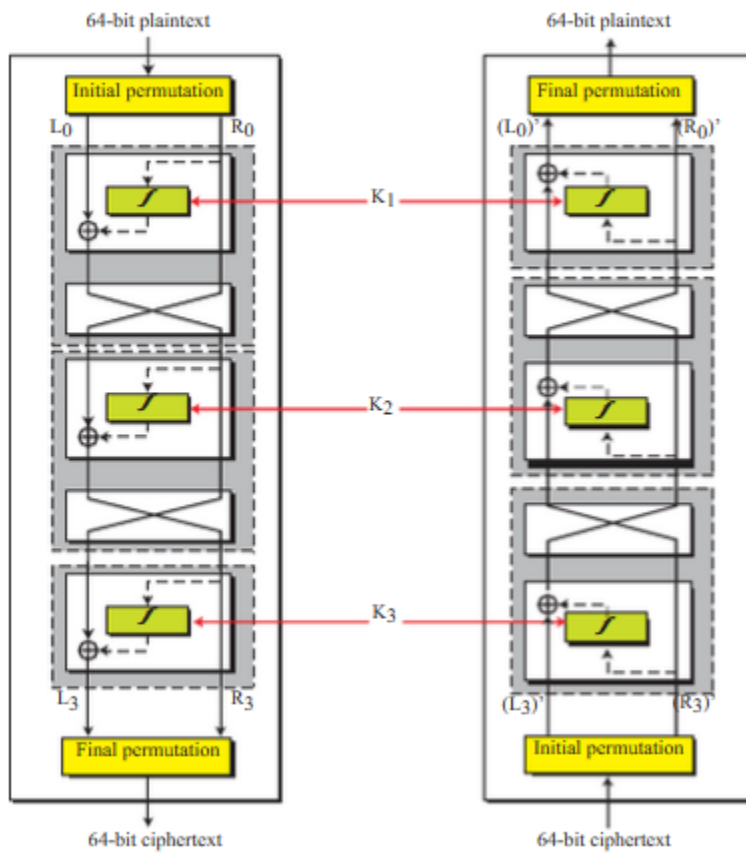
- a. Only output 19 from S-box 5 (in round 4) goes to S-box 7 (in round 5). However, the input 38 is not a middle input, so the criterion does not apply. This answers the question about this exercise, but we do some more investigations.
- b. Output 18 from S-box 5 goes a middle input (21) in S-box 4 in the next round. To check the criteria, we need to see if any input from S-box 4 goes to a middle input in next round. Looking at Figure S6.24, we can see that this is not the case. The criterion applies here.
- c. We also observe that output 19 from S-box 5 goes a middle input in S-box 1 (input 4). However, none of the inputs from S-box 1 in round 4 goes to a middle input of S-box 5 in the next round. Only one output from S-box 1 (output 2 goes to S-box 5, input 26, but it is not a middle input). The criterion applies here.

Figure S6.26 Solution to Exercise 26



- 27.** Figure S6.27 shows a three-round cipher. We prove the equalities between the L's and R's from bottom to top. We have labeled each left section in the encryption L_i and in the decryption $(L_i)'$; we have labeled each right sections R_i in the encrypting and $(R_i)'$ in decryption.

Figure S6.27 *Solution to Exercise 27*



- a. Since final permutation and initial permutations are inverse of each other (if there is no corruption during transmission), we have

$$(\mathbf{L}_3)' = (\mathbf{L}_3)$$

$$(\mathbf{R}_3)' = (\mathbf{R}_3)$$

- b. Using the previous equalities and the relations in the mixers, we have

$$(\mathbf{L}_2)' = (\mathbf{R}_3)' = \mathbf{R}_3 = \mathbf{R}_2$$

$$(\mathbf{L}_2)' = \mathbf{R}_2$$

$$(\mathbf{R}_2)' = (\mathbf{L}_3)' \oplus \mathcal{F}[(\mathbf{R}_3)', K_3]$$

$$(\mathbf{R}_2)' = \mathbf{L}_3 \oplus \mathcal{F}[\mathbf{R}_3, K_3]$$

$$(\mathbf{R}_2)' = \mathbf{L}_2 \oplus \mathcal{F}[\mathbf{R}_2, K_3] \oplus \mathcal{F}[\mathbf{R}_2, K_3]$$

$$(\mathbf{R}_2)' = \mathbf{L}_2$$

- c. Using the previous equalities and the relations in the mixers, we have

$$(\mathbf{L}_1)' = (\mathbf{R}_2)' = \mathbf{L}_2 = \mathbf{R}_1$$

$$(\mathbf{L}_1)' = \mathbf{R}_1$$

$$(\mathbf{R}_1)' = \mathbf{R}_2 \oplus \mathcal{F}[\mathbf{L}_2, K_2]$$

$$(\mathbf{R}_1)' = \mathbf{R}_2 \oplus \mathcal{F}[\mathbf{R}_1, K_2]$$

$$(\mathbf{R}_1)' = \mathbf{L}_1 \oplus \mathcal{F}[\mathbf{R}_1, K_1] \oplus \mathcal{F}[\mathbf{R}_1, K_1]$$

$$(\mathbf{R}_1)' = \mathbf{L}_1$$

- d. Using the previous equalities and the relations in the mixers, we have

$$(\mathbf{L}_0)' = (\mathbf{L}_1)' \oplus \mathcal{F}[(\mathbf{R}_1)', K_1]$$

$$(\mathbf{L}_0)' = \mathbf{R}_1 \oplus \mathcal{F}[\mathbf{L}_1, K_1]$$

$$(\mathbf{L}_0)' = \mathbf{L}_0 \oplus \mathcal{F}[\mathbf{L}_0, K_1] \oplus \mathcal{F}[\mathbf{L}_0, K_1]$$

$$(\mathbf{L}_1)' = \mathbf{L}_0$$

$$(\mathbf{R}_0)' = (\mathbf{R}_1)' = \mathbf{L}_1 = \mathbf{R}_0$$

$$(\mathbf{R}_0)' = \mathbf{R}_0$$

We have proved that $(\mathbf{L}_1)' = \mathbf{L}_0$ and $(\mathbf{R}_0)' = \mathbf{R}_0$. Since the final permutation and initial permutation are inverse of each other, the plaintext created at the destination is the same as the plaintext started at the source.

28. If we create a table of input and output, we can answers the three questions.

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

In	Out	In	Out
41	25	44	37
52	26	49	38
31	27	39	39
37	28	56	40
47	29	34	41
55	30	53	42
30	31	46	43
40	32	42	44
51	33	50	45
45	34	36	46
33	35	29	47
48	36	32	48

- The missing inputs are 09, 18, 22, 25, 35, 38, 43 and 54.
- From above table, we can see that the left 24 bits come from the left 28 bits (except bits 09, 18, 22, and 25, which are blocked).
- From the above table, we can see the right 24 bits come from the right 28 bits (except bits 35, 38, 43, and 54, which are blocked).

29. The following shows the result:

(1066 0099 0088 0088)₁₆

30. The following shows the result:

(0F55 AAF5 0F55 AAF5)₁₆

31. The first round key is

(1437 4013 3784)₁₆

32. The following shows the effect:

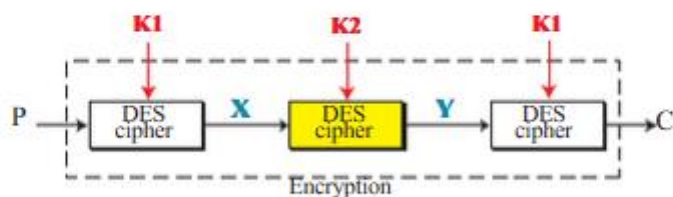
	Originals				Complements			
Plaintext:	0000	0000	0000	0000	FFFF	FFFF	FFFF	FFFF
Key:	0000	0000	0000	0000	FFFF	FFFF	FFFF	FFFF
Ciphertext:	0808	02AA	AA02	A8AA	F7F7	FD55	55FD	5755

When we complement the plaintext and the key, the ciphertext is complemented.

33. Figure S6.33 shows the encryption using 3DES with two keys, in which X or Y are the intermediate texts.

33. Figure S6.33 shows the encryption using 3DES with two keys, in which X or Y are the intermediate texts.

Figure S6.33 Solution to Exercise 33



Van Oorschot and Wiener have devised a meet-in-the-middle attack on the above configuration. The attack is basically a known-plaintext attack. It follows the steps shown below:

- a. Eve intercepts n plaintext/ciphertext pairs and stores them in a table which is sorted on values of P as shown below:

Plaintext	Ciphertext
P_1	C_1
P_2	C_2
...	...
P_n	C_n

Table 1: n P/C pairs

- b. Eve now chooses a value for X (see Figure S6.33) and uses the decryption algorithm, and all 2^{56} possible K1's values to create 2^{56} different P values as shown below:

$$P_1 = D(K1_1, X) \quad P_2 = D(K1_2, X) \quad \dots \quad P_m = D(K1_m, X) \quad \text{where } m = 2^{56}$$

- c. If a value of P created in step b matches one of the value of P in Table 1, Eve uses the corresponding value for K1 (from the list in step b) and the value of C from Table 1 and calculates a value for second intermediate text $Y = D(K1, C)$. Now Eve creates a second table, Table 2, which is sorted on the value of Y. Eve has now r possible candidate for K1 keys.

Y	K1
Y_1	$K1_1$
Y_2	$K1_2$
...	...
Y_r	$K1_r$

Table 2: r Y/K1 pairs

- d. Eve now searches for K2. For each 2^{56} possible values of K2, Eve uses the decryption algorithm and the value of X chosen in step b to create 2^{56} different values for Y's.

$$Y_1 = D(K2_1, X) \quad Y_2 = D(K2_2, X) \quad \dots \quad Y_m = D(K2_m, X) \quad \text{where } m = 2^{56}$$

If a value of Y_i created in this step matches one of the value in Table 2, Eve have found a pair of keys: K1 is extracted from Table 2 and K2 is extracted from the decryption algorithm that matches the value of Y in Table 2.

35.

```
split(n, m, inBlock[1 ... n], leftBlock[1 ... m], rightBlock[1 ... m])
{
    i ← 1
    while (i ≤ m)
    {
        leftBlock[i] ← inBlock[i]
        rightBlock[i] ← inBlock[i + m]
        i ← i + 1
    }
    return
}
```

36.

```
combine(n, m, leftBlock[1 ... n], rightBlock[1 ... n], outBlock[1 ... m])
{
    i ← 1
    while (i ≤ m)
    {
        outBlock[i] ← leftBlock[i]
        outBlock[i + m] ← rightBlock[i]
        i ← i + 1
    }
    return
}
```

37.

```
exclusiveOr(n, firstBlock[1 ... n], secondBlock[1 ... n], outBlock[1 ... n])
{
    i ← 1
    while (i ≤ n)
    {
        outBlock[i] ← firstBlock[i] ⊕ secondBlock[i]
        i ← i + 1
    }
    return
}
```

38.

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

39. We have added one extra parameter ED (encrypt/decrypt). If ED = E, we do encryption; if ED = D, we do decryption.

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