

# ETC005 Bluetooth

## Laboratory exercise 3

### Simplified DES

Name	Date	Approved



# Simplified DES

Overview .....	3
Task 1 .....	3
Task 2 .....	3
The encryption algorithm involves five functions .....	4
S-DES Key Generation .....	4
function_key.....	5
Initial and Final Permutations .....	5
The Function F and $f_k$ .....	6
function_F .....	6
function_fk .....	6
The Switch Function .....	10
function_switch .....	10
Figure 1: Simplified DES Scheme .....	11
Figure 2: Key Generation for Simplified DES.....	12
Figure 3: Simplified DES Scheme Encryption Detail.....	13

## Overview

Figure 1 illustrates the overall structure of the simplified DES, which we will refer to as S-DES. The S-DES encryption algorithm takes an 8-bit block of plaintext (examples: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output. The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.

## Task 1

Make a Matlab code according to this document. Create the following main functions:

- function\_key
- function\_f
- function\_fk
- function\_switch

Make a program and use the function in the list.

Test the main program according to this example:

```
key = [1010000010]
plaintext = [01000001] = A (char)
ciphertext = [00010101] = 15 (hex)
```

## Task 2

Make a Brute Force program. This program must use every possible keys ( $0 - 2^{10} - 1$ ) to force the encryption.

Brute Force this encrypted messages:

```
AF224F62772FE86A9D7762D4F88E8E
```

and give the appropriate key as well.



## The encryption algorithm involves five functions

- An initial permutation (IP).
- A complex function labeled  $f_k$ , which involves both permutation and substitution operations and depends on a key input.
- A simple permutation function that switches (SW) the two halves of the data.
- The function  $f_k$  again.
- A permutation function that is the inverse of the initial permutation ( $IP^{-1}$ ).

## S-DES Key Generation

S-DES depends on the use of a 10-bit key shared between sender and receiver. From this key, two 8-bit subkeys are produced for use in particular stages of the encryption and decryption algorithm, see figure 2.

First, permute the key in the following fashion. Let the 10-bit key be designated as  $(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10})$ . Then the permutation P10 is defined as:

$P10(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6)$ .

P10									
3	5	2	7	4	10	1	9	8	6

**Step 1:** For example, the key (1010000010) is permuted to (1000001100).

**Step 2:** Divide (1000001100) into a left part 5-bit value (10000) and a right part 5-bit value (01100).

**Step 3:** Perform a circular left shift (LS-1), or rotation, separately. The left value (10000) becomes (00001). The right value (01100) becomes (11000). Concatenate the left part (00001) and the right part (11000) into a 10-bit value (0000111000).

**Step 4:** Pick out and permutes 8, (don't use 1 and 2), of the 10 bits according to the following rules:

P8							
6	3	7	4	8	5	10	9

The result is subkey  $k_1$ . The value (00 00111000) becomes:  
 $k_1 = (10100100)$ .

**Step 5:** Go back to the pair of 5-bit strings produced by the two LS-1 functions, the results of step 3, and perform a circular left shift of 2 bit positions on each string. The left value (00001) becomes (00100) and the right value (11000) becomes (00011). Concatenate the left part (00100) and the right part (00011) into a 10-bit value (0010000011).

**Step 6:** Finally, P8 is applied again to produce  $k_2$ , the value (00 10000011) becomes:  
 $k_2 = (01000011)$ .

## ***function\_key***

function [k1,k2] = function\_key(key)

## **Initial and Final Permutations**

The input to the algorithm is an 8-bit block of plaintext (10111101), which we first permute using the IP function.

**Step first:** Permute the plaintext (10111101) with IP and the value becomes (01111110). Divide the value into two parts. The left part becomes (0111) and the right part becomes (1110).

IP							
2	6	3	1	4	8	5	7

This retains all 8 bits of the plaintext but mixes them up. At the end of the algorithm, the inverse permutation is used.

**Step last:** Permute the value (11101100) with  $IP^{-1}$  and the ciphertext becomes (01110101).

$IP^{-1}$							
4	1	3	5	7	2	8	6

It is easy to show by example that the second permutation is indeed the reverse of the first; that is,  $IP^{-1}(IP(X)) = X$ .



## The Function F and $f_k$

The most complex component of S-DES is the function  $f_k$ , which consists of a combination of permutation and substitution functions, see figure 3.

$$f_k(L,R) = (L \oplus F(R,SK),R)$$

### *function\_F*

function F = function\_ff(key,R)

### *function\_fk*

function [L,R] = function\_fk(key,L,R)

SK is a subkey,  $k_1$  or  $k_2$ , and  $\oplus$  is the bit-by-bit exclusive-OR function.

#### **Step 1:**

First loop: Suppose the output of the IP stage is (01111110), the left 4-bit value is (0111) and the right 4-bit value is (1110).

Second loop: The output of the SW stage is (11101100), the left 4-bit value is (1110) and the right 4-bit value is (1100).

#### **Step 2:**

First loop: Expand the 4-bit value (1110) and concatenate it twice into an 8-bit value (11101110).

Second loop: Expand the 4-bit value (1100) and concatenate it twice into an 8-bit value (11001100).

#### **Step 3:**

First loop: Permute (11101110) and we have (01111101).

Second loop: Permute (11001100) and we have (01101001).

E/P							
4	1	2	3	2	3	4	1

**Step 4:**

First loop: Create a matrix based on the result of step 3. Row 1 is (0111). Row 2 is (1101).

$n_4$	$n_1$	$n_2$	$n_3$
$n_2$	$n_3$	$n_4$	$n_1$

0	1	1	1
1	1	0	1

Second loop: Create a matrix based on the result of step 3. Row 1 is (0110). Row 2 is (1001).

$n_4$	$n_1$	$n_2$	$n_3$
$n_2$	$n_3$	$n_4$	$n_1$

0	1	1	0
1	0	0	1

**Step 5:**

First loop: Create a matrix based on the 8-bit subkey  $K_1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18})$ . We use the key (10100100).

$k_{11}$	$k_{12}$	$k_{13}$	$k_{14}$
$k_{15}$	$k_{16}$	$k_{17}$	$k_{18}$

1	0	1	0
0	1	0	0

Second loop: Create a matrix based on the 8-bit subkey  $K_2 = (k_{21}, k_{22}, k_{23}, k_{24}, k_{25}, k_{26}, k_{27}, k_{28})$ . We use the key (01000011).

$k_{21}$	$k_{22}$	$k_{23}$	$k_{24}$
$k_{25}$	$k_{26}$	$k_{27}$	$k_{28}$

0	1	0	0
0	0	1	1

**Step 6:**

First loop: Perform an exclusive-OR function on the matrix in step 4 and step 5.

$n_4 \oplus k_1$	$n_1 \oplus k_2$	$n_2 \oplus k_3$	$n_3 \oplus k_4$
$n_2 \oplus k_5$	$n_3 \oplus k_6$	$n_4 \oplus k_7$	$n_3 \oplus k_8$

1	1	0	1
1	0	0	1

Second loop: Perform an exclusive-OR function on the matrix in step 4 and step 5.

$n_4 \oplus k_1$	$n_1 \oplus k_2$	$n_2 \oplus k_3$	$n_3 \oplus k_4$
$n_2 \oplus k_5$	$n_3 \oplus k_6$	$n_4 \oplus k_7$	$n_3 \oplus k_8$

0	0	1	0
1	0	1	0

**Step 7:**

First loop: Rename the result of matrix in step 6. Create groups according to these roles:  $(P_{0,0} P_{0,3})$ ,  $(P_{0,1} P_{0,2})$ ,  $(P_{1,0} P_{1,3})$  and  $(P_{1,1} P_{1,2})$ . Convert the binary value to decimal.

$P_{0,0}$	$P_{0,1}$	$P_{0,2}$	$P_{0,3}$
$P_{1,0}$	$P_{1,1}$	$P_{1,2}$	$P_{1,3}$

1	1	0	1
1	0	0	1

$$(P_{0,0} P_{0,3}) = (11) = 3$$

$$(P_{0,1} P_{0,2}) = (10) = 2$$

$$(P_{1,0} P_{1,3}) = (11) = 3$$

$$(P_{1,1} P_{1,2}) = (00) = 0$$

Second loop: Rename the result of matrix in step 6. Create groups according to these roles:  $(P_{0,0} P_{0,3})$ ,  $(P_{0,1} P_{0,2})$ ,  $(P_{1,0} P_{1,3})$  and  $(P_{1,1} P_{1,2})$ . Convert the binary value to decimal.

$P_{0,0}$	$P_{0,1}$	$P_{0,2}$	$P_{0,3}$
$P_{1,0}$	$P_{1,1}$	$P_{1,2}$	$P_{1,3}$

0	0	1	0
1	0	1	0

$$(P_{0,0} P_{0,3}) = (00) = 0$$

$$(P_{0,1} P_{0,2}) = (01) = 1$$

$$(P_{1,0} P_{1,3}) = (10) = 2$$

$$(P_{1,1} P_{1,2}) = (01) = 1$$

**Step 8:** The S-boxes operate as follows. The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the S-box. The entry in that row and column, in base 2, is the 2-bit row 0, column 2 of  $S_0$ , which is 3, or (11) in binary. Add the two 2-bit results into a one 4-bit value.

First loop:

$$S0_y = (P_{0,0} P_{0,3}) = (11) = 3$$

$$S0_x = (P_{0,1} P_{0,2}) = (10) = 2$$

$$S0(y,x) = 3 \text{ (dec)} = 11 \text{ (bin)}$$

$$S1_y = (P_{1,0} P_{1,3}) = (11) = 3$$

$$S1_x = (P_{1,1} P_{1,2}) = (00) = 0$$

$$S1(y,x) = 2 \text{ (dec)} = 10 \text{ (bin)}$$

Concatenate  $S_0$  (3=11) and  $S_1$  (2=10) into a 4-bit value (1110).

		0	1	2	3			0	1	2	3	
S0	=	0	1	2	3		S1	=	0	1	2	3
		1	0	3	2				0	1	2	3
		3	2	1	0				2	0	1	3
		2	0	2	1				3	0	1	0
		3	3	1	3				3	1	0	3



Second loop:

$$S0_y = (P_{0,0} P_{0,3}) = (00) = 0$$

$$S0_x = (P_{0,1} P_{0,2}) = (01) = 1$$

$$S0(y,x) = 0 \text{ (dec)} = 00 \text{ (bin)}$$

$$S1_y = (P_{1,0} P_{1,3}) = (10) = 2$$

$$S1_x = (P_{1,1} P_{1,2}) = (01) = 1$$

$$S1(y,x) = 0 \text{ (dec)} = 00 \text{ (bin)}$$

Concatenate S0 (0=00) and S1 (0=00) into a 4-bit value (0000).

$$S0 = \begin{array}{c|c|c|c|c} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & \mathbf{0} & 3 & 2 \\ 1 & 3 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 & 3 \\ 3 & 3 & 1 & 3 & 2 \end{array} \quad S1 = \begin{array}{c|c|c|c|c} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 1 & 3 \\ 2 & 3 & \mathbf{0} & 1 & 0 \\ 3 & 2 & 1 & 0 & 3 \end{array}$$

### Step 9:

First loop:

Permute the value (1110) which becomes (1011) using P4.

P4			
2	4	3	1

Second loop:

Permute the value (0000) which becomes (0000) using P4.

P4			
2	4	3	1

### Step 10:

First loop:

Take the left value from step 1 (0111) and exclusive-OR the value from step 9 (1011).

$$(0111) \oplus (1011) = (1100).$$

Second loop:

Take the left value from step 1 (1110) and exclusive-OR the value from step 9 (0000).

$$(1110) \oplus (0000) = (1110).$$

### Step 11:

First loop:

Take the value from step 10 (1100) and the right value from step 1 (1110).

Second loop:

Take the value from step 10 (1110) and the value from step 10 first loop (1100).



---

## The Switch Function

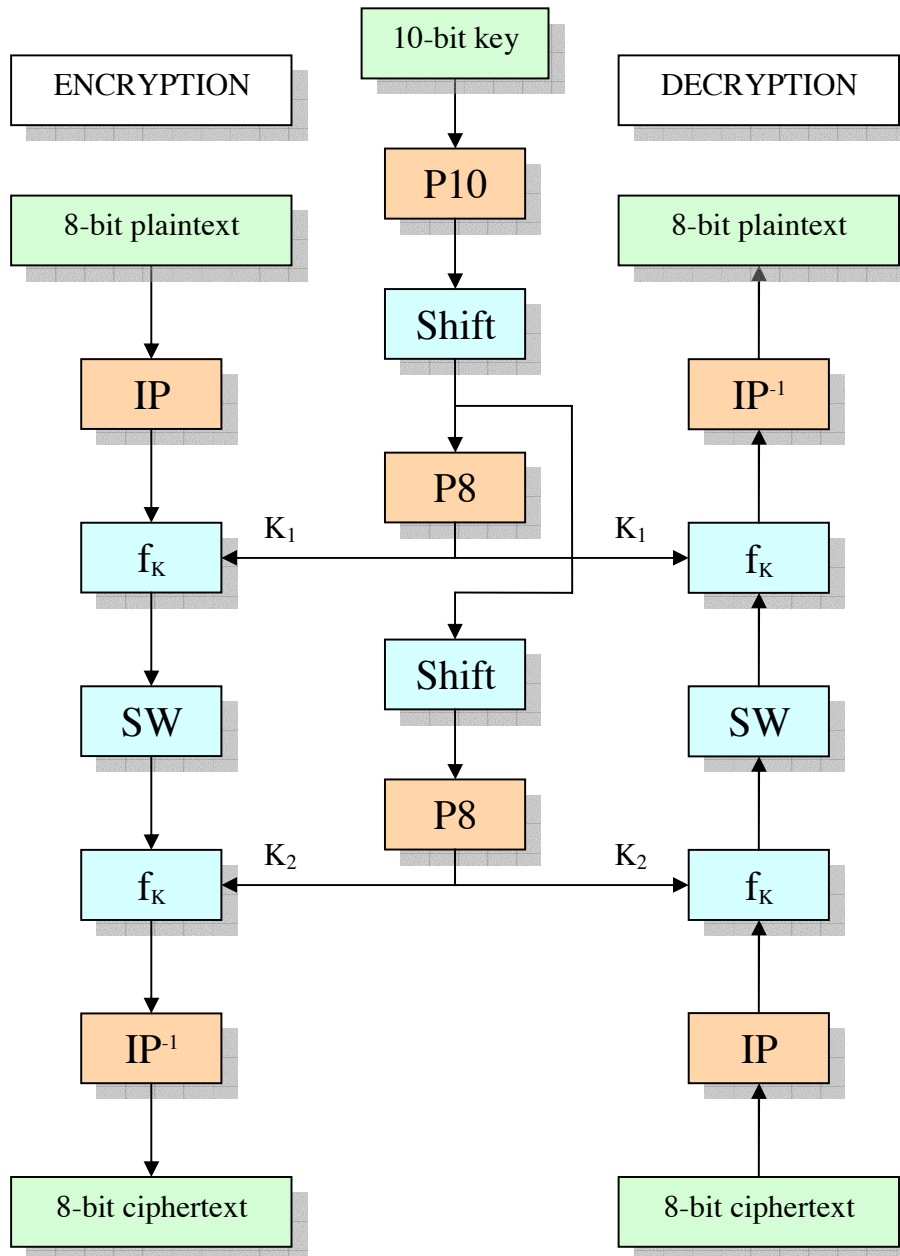
The function  $f_k$  only alters the leftmost 4-bits of the input. The switch function (SW) interchanges the left and right 4-bits so that the second instance of  $f_k$  operates on a different 4-bit.

Take the value from step 10 first loop (1100) and the right 4-bit value from IP (1110). Swap places with each other. Use (1110) as input to the left side in the  $f_k$  in the second loop. Use (1100) as input to the right side in the  $f_k$  in the second loop.

### ***function\_switch***

function [L,R] = function\_switch(L,R)

**Figure 1: Simplified DES Scheme**



**Figure 1:** Simplified DES Scheme.

## Figure 2: Key Generation for Simplified DES

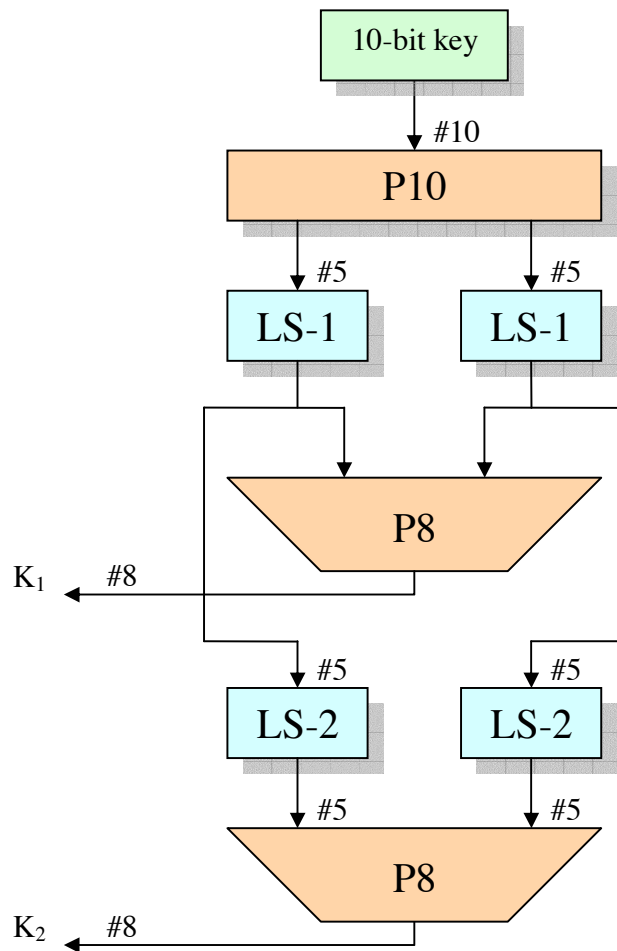
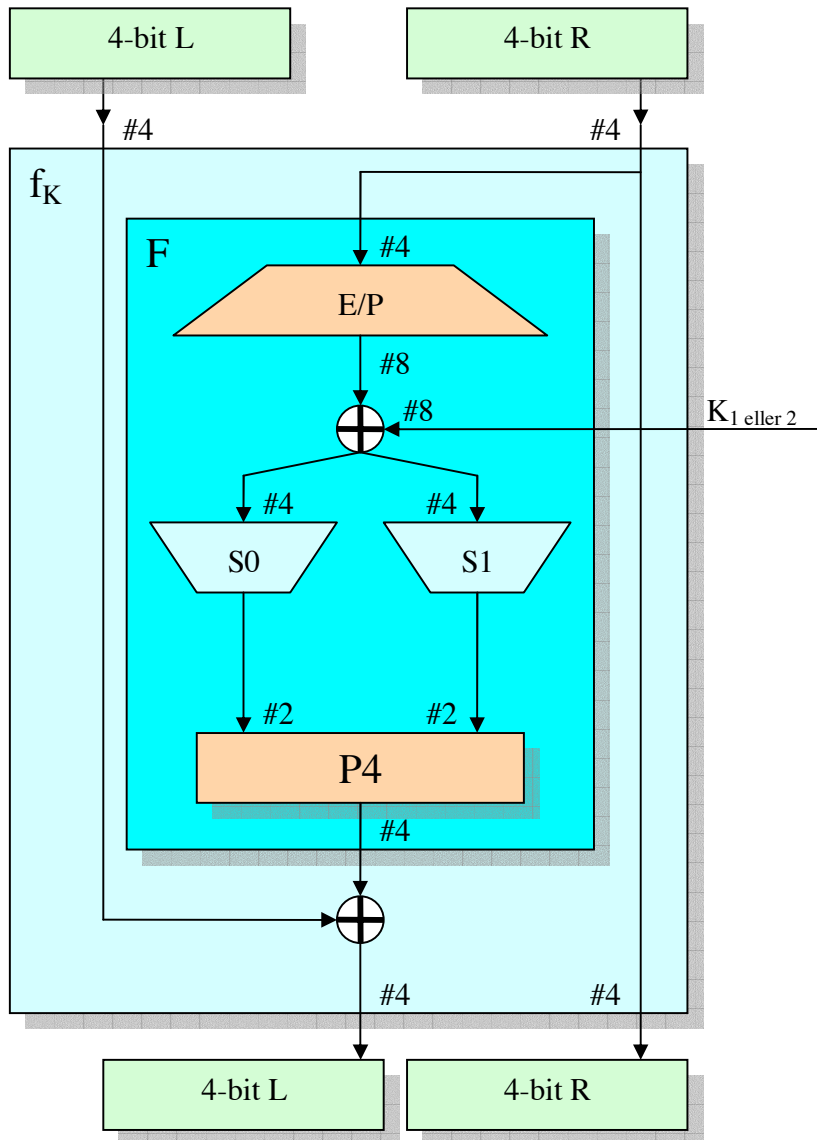


Figure 2: Key Generation for Simplified DES.

**Figure 3: Simplified DES Scheme Encryption Detail**



**Figure 3:** Simplified DES Scheme Encryption Detail.