Assignment: 4-

Date of Assignment: 4 Sept 2017

Date of Submission: 25 Sept 2017 Time: 5:30 pm

Figure 1 illustrates the overall structure of the simplified DES, referred to as S-DES. The S-DES encryption algorithm takes an 8-bit block of plaintext 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 plaintext.

1. Create a program for encryption and decryption using the S-DES algorithm, described in this document. For testing the algorithm, the following key-plaintext pair generates the following ciphertext.

key = 1010000010

plaintext = 01000001 (Binary ASCII-value for character 'A')

ciphertext = 00010101 (Binary value 21)

2. Make a brute force program that uses every possible key to force the encryption. Decrypt the following message and give the encryption key aswell. The plaintext can be assumed to be written text containing characters and white spaces only. The ciphertext, split over two lines, is presented as a hexadecimal string.

15629177698EF862D42F77E8F862B7F8E87706CB8EC72F5A
62C75A6215CBCB0DC7F8B762447706CB8E2FA9779DCB4C06

3 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 the use in particular stages of the encryption and decryption algorithm. Figure 2 shows the block diagram for subkey generation.

Step 1 Permute the key using P10. 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

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

The permutation table for P10 is presented in table 1.

Example key: `1010000010` *to* `1000001100`.

Step 2 Divide the key into a left part 5-bit value and a right part 5-bit value.

Example key: `1000001100` *to* `10000` *and* `01100`.

Step 3 Perform a circular left shift one step on each 5-bit value. Concatenate the two parts into a 10-bit value.

Example key: `10000` *and* `01100` *to* `00001` *and* `11000` *after circular shift, and* `0000111000` *after concatenation.*

Step 4 Pick out 8 of the 10 bits according to the permutation P8 presented in table 2. The result is subkey K_1.

Example key: `0000111000` *to* $K_1 =$ `10100100`.

Step 5 Divide the resulting value from step 3 after concatenation into two left and right 5-bit values. Perform a circular left shift two steps on each 5-bit value, and concatenate the two values.

Example key: `0000111000` *to* `0010000011`.

Step 6 Pick out 8 of the 10 bits according to the permutation P8. The result is the subkey K_2.

Example key: `0010000011` *to* $K_2 =$ `01000011`.

4 Encryption

The input to the algorithm is an 8-bit block of plaintext. The example throughout the description will be `10111101`. The diagram for encryption is shown in figure 1, left part.

4.1 Initial permutation

Permute the plaintext using IP. The perutation table for IP is presented in table 3.
Example encryption: `10111101` *to* `01111110`.

4.2 Function f_K

The most complex part of S-DES is the function f_K which consists of a combination of permutations and substitutions. The block diagram for the f_K function is shown in figure 3.

Step 1 Divide the output value from the initial permutation into two 4-bit values, referred to as left and right value, respectively.

Example encryption: `01111110` *to left value* `0111` *and right value* `1110`.

Step 2 Expand the right 4-bit by concatenating it with itself into an 8-bit value.

Example encryption: `1110` *to* `11101110`.

Step 3 Permute the expanded 8-bit value by E/P. The perutation table for E/P is presented in table 4.

Example encryption: `11101110` *to* `01111101`.

Step 4 Create a 2×4 matrix based on the result of step 3. First 4 bits is the first row, last 4 bits is the second row.

Example encryption: `01111101` *to* $n = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$.

Step 5 Create a 2×4 matrix based on the key K_1.

Example encryption: $K_1 =$ `10100100` *to* $k = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$.

Step 6 Perform an elementwise exclusive OR-operation on the matrices n and k from step 4 and 5.

Example encryption: n *and* k *to* $P = n \oplus k = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$.

Step 7 Extract bits from P and convert binary value to decimal for indices into the S-boxes.

$$P = \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} & P_{0,3} \\ P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \end{bmatrix}$$

$$\begin{aligned} S_{0,x} &= (P_{0,1}, P_{0,2}) \\ S_{0,y} &= (P_{0,0}, P_{0,3}) \\ S_{1,x} &= (P_{1,1}, P_{1,2}) \\ S_{1,y} &= (P_{1,0}, P_{1,3}) \end{aligned}$$

Example encryption: P *to* $S_{0,x} = (10) = 2$, $S_{0,y} = (11) = 3$, $S_{1,x} = (00) = 0$ *and* $S_{1,y} = (11) = 3$.

Step 8 Look up the value in the first S-box using row-index $S_{0,y}$ and column-index $S_{0,x}$. Lookup the value in the second S-box using row-index $S_{1,y}$ and column-index $S_{1,x}$. Convert the values from the S-boxes to 2-bit values and concatenate into a 4-bit value. The first and second S-box, S0 and S1, respectively, is presented in table 5.

Example encryption: $S0_{S_{0,y},S_{0,x}} = S0_{3,2} = 3 =$ `11` *and* $S1_{S_{1,y},S_{1,x}} = S1_{3,0} = 2 =$ `10` *to* `1110`.

Step 9 Permute the output from step 8 using P4. The permutation table for P4 is presented in table 6

Example encryption: `1110` *to* `1011`.

Step 10 Take the left value from step 1 and exclusive OR it with the value from step 9.

Example encryption: `0111` *and* `1011` *to* `1100`.

Step 11 Concatenate the 4-bit value from step 10 with the right 4-bit value from step 1 into an 8-bit value.

Example encryption: `1100` *and* `1110` *to* `11001110`.

4.3 Switch function

The f_K function only alters the leftmost 4 bits of the input. The switch function interchanges the left and right 4-bit values so the second instance of f_K operates on a different 4 bits. Interchange the left and right 4-bit values of the output value from step 11 in the f_K function.
Example encryption: `11001110` *to* `11101100`.

4.4 Function f_K

Perform step 1-11 of the f_K function a second time on the new value from the switch function, but use K_2 instead of K_1 as key in step 5.
Example encryption: `11101100` *to* `11101100`.

4.5 Final permutation

Permute the output from the second f_K function using IP^{-1}. The perutation table for IP^{-1} is presented in table 7. The output from the final permutation is the ciphertext
Example encryption: `11101100` *to* `01110101`.

5 Decryption

Decryption is similar to encryption, see right part of figure 1. Decryption is performed using the same functions as when encrypting, only the subkeys are used in reverse order (first K_2, then K_1).

A Permutation tables, S-boxes and diagrams

Table 1: Permutation table for P10.

3	5	2	7	4	10	1	9	8	6

Table 2: Permutation table for P8.

6	3	7	4	8	5	10	9

Table 3: Permutation table for IP.

2	6	3	1	4	8	5	7

Table 4: Permutation table for E/P.

4	1	2	3	2	3	4	1

Table 5: S-boxes.

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

Table 6: Permutation table for P4.

2	4	3	1

Table 7: Permutation table for IP^{-1}.

4	1	3	5	7	2	8	6

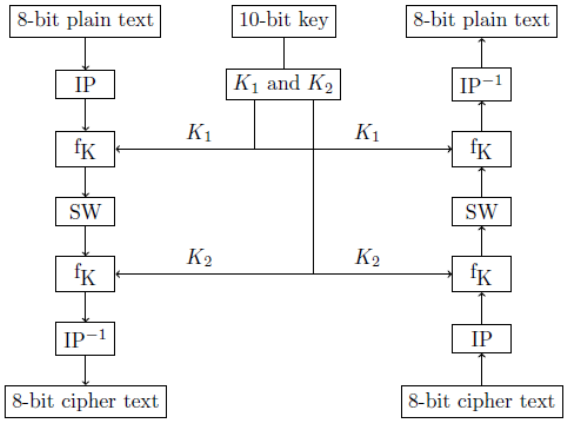

Figure 1: S-DES encryption and decryption structure.

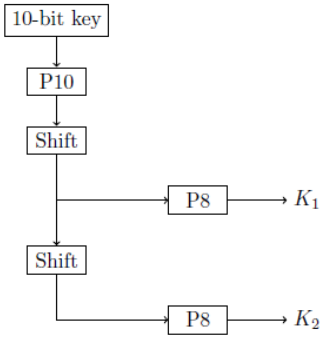

Figure 2: Subkey generation structure.

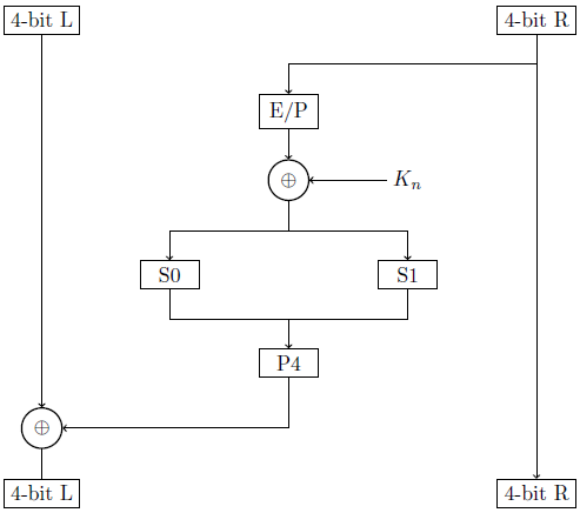

Figure 3: f_K structure.