Overview

Figure 3.1 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 (example: 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.

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; and finally a permutation function that is the inverse of the initial permutation (IP^{-1}). As was mentioned in Chapter 2, the use of multiple stages of permutation and substitution results in a more complex algorithm, which increases the difficulty of cryptanalysis.

The function f_K takes as input not only the data passing through the encryption algorithm, but also an 8-bit key. The algorithm could have been designed to

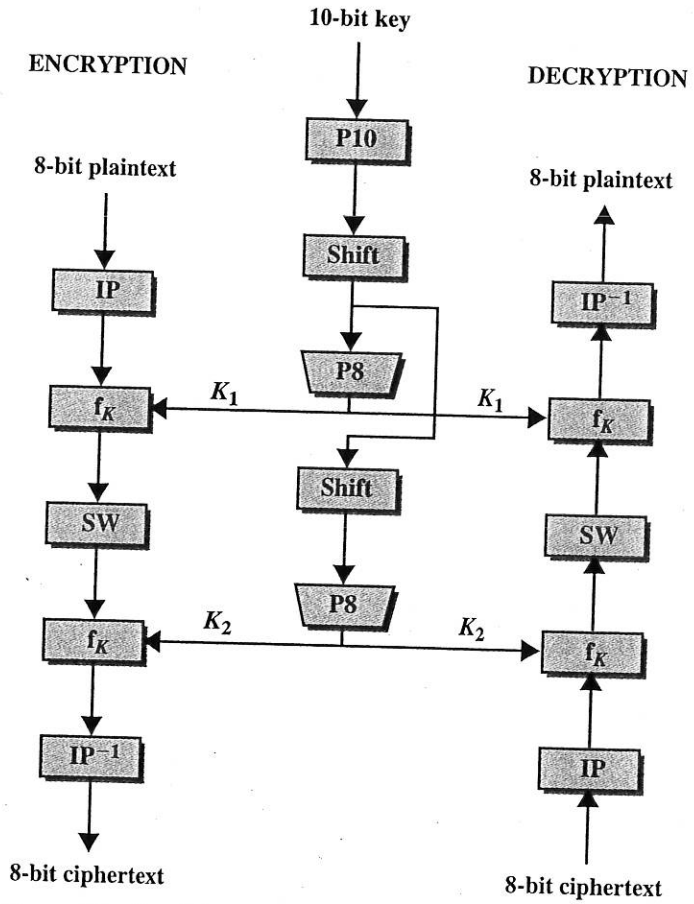

Figure 3.1 Simplified DES Scheme

work with a 16-bit key, consisting of two 8-bit subkeys, one used for each occurrence of f_K. Alternatively, a single 8-bit key could have been used, with the same key used twice in the algorithm. A compromise is to use a 10-bit key from which two 8-bit subkeys are generated, as depicted in Figure 3.1. In this case, the key is first subjected to a permutation (P10). Then a shift operation is performed. The output of the shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first subkey (K_1). The output of the shift operation also feeds into another shift and another instance of P8 to produce the second subkey (K_2).

We can concisely express the encryption algorithm as a composition[3] of functions:

$$\text{IP}^{-1} \circ \text{f}_{K_2} \circ \text{SW} \circ \text{f}_{K_1} \circ \text{IP}$$

which can also be written as

$$\text{ciphertext} = \text{IP}^{-1}(\text{f}_{K_2}(\text{SW}(\text{f}_{K_1}(\text{IP}(\text{plaintext})))))$$

where

$$K_1 = \text{P8}(\text{Shift}(\text{P10}(\text{key})))$$
$$K_2 = \text{P8}(\text{Shift}(\text{Shift}(\text{P10}(\text{key}))))$$

Decryption is also shown in Figure 3.1 and is essentially the reverse of encryption:

$$\text{plaintext} = \text{IP}^{-1}(\text{f}_{K_1}(\text{SW}(\text{f}_{K_2}(\text{IP}(\text{ciphertext})))))$$

We now examine the elements of S-DES in more detail.

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. Figure 3.2 depicts the stages followed to produce the subkeys.

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

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

P10 can be concisely defined by the following display:

P10									
3	5	2	7	4	10	1	9	8	6

This table is read from left to right; each position in the table gives the identity of the input bit that produces the output bit in that position. So the first output

[3] **Definition:** If f and g are two functions, then the function F with the equation $y = F(x) = g[f(x)]$ is called the **composite** of f and g and is denoted as $F = g \circ f$.

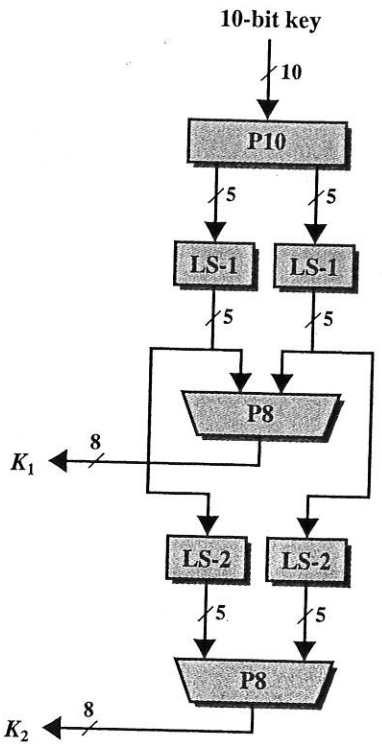

Figure 3.2 Key Generation for Simplified DES

bit is bit 3 of the input; the second output bit is bit 5 of the input, and so on. For example, the key (1010000010) is permuted to (1000001100). Next, perform a circular left shift (LS–1), or rotation, separately on the first five bits and the second five bits. In our example, the result is (00001 11000).

Next we apply P8, which picks out and permutes 8 of the 10 bits according to the following rule:

P8							
6	3	7	4	8	5	10	9

The result is subkey 1 (K_1). In our example, this yields (10100100).

We then go back to the pair of 5-bit strings produced by the two LS–1 functions and perform a circular left shift of 2 bit positions on each string. In our example, the value (00001 11000) becomes (00100 00011). Finally, P8 is applied again to produce K_2. In our example, the result is (01000011).

S-DES Encryption

Figure 3.3 shows the S-DES encryption algorithm in greater detail. As was mentioned, encryption involves the sequential application of five functions. We examine each of these.

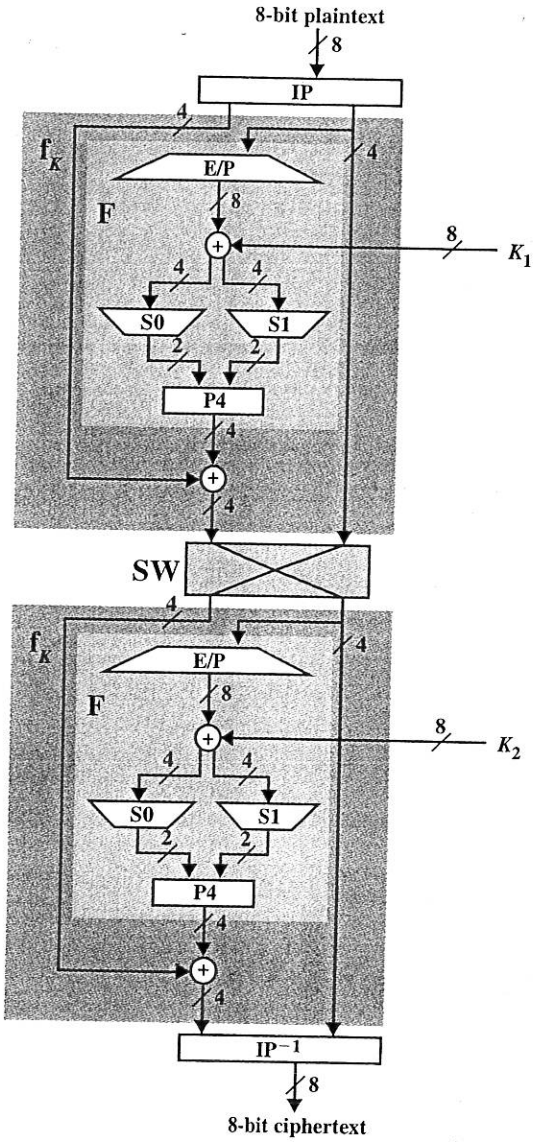

Figure 3.3 Simplified DES Scheme Encryption Detail

Initial and Final Permutations

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

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:

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_K

The most complex component of S-DES is the function f_K, which consists of a combination of permutation and substitution functions. The functions can be expressed as follows. Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to f_K, and let F be a mapping (not necessarily one to one) from 4-bit strings to 4-bit strings. Then we let

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

where SK is a subkey and $\oplus$ is the bit-by-bit exclusive-OR function. For example, suppose the output of the IP stage in Figure 3.3 is (10111101) and F(1101, SK) = (1110) for some key SK. Then f_K(10111101) = (01011101) because (1011) $\oplus$ (1110) = (0101).

We now describe the mapping F. The input is a 4-bit number $(n_1n_2n_3n_4)$. The first operation is an expansion/permutation operation:

E/P							
4	1	2	3	2	3	4	1

For what follows, it is clearer to depict the result in this fashion:

$$\begin{array}{c|c|c|c} n_4 & n_1 & n_2 & n_3 \\ n_2 & n_3 & n_4 & n_1 \end{array}$$

The 8-bit subkey $K_1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18})$ is added to this value using exclusive-OR:

$$\begin{array}{c|c|c|c} n_4 + k_{11} & n_1 + k_{12} & n_2 + k_{13} & n_3 + k_{14} \\ n_2 + k_{15} & n_3 + k_{16} & n_4 + k_{17} & n_1 + k_{18} \end{array}$$

Let us rename these 8 bits:

$$\begin{array}{c|c|c|c} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \end{array}$$

The first 4 bits (first row of the preceding matrix) are fed into the S-box S0 to produce a 2-bit output, and the remaining 4 bits (second row) are fed into S1 to produce another 2-bit output. These two boxes are defined as follows:

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

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 output. For example, if $(p_{0,0}p_{0,3}) = (00)$ and $(p_{0,1}p_{0,2}) = (10)$, then the output is from row 0, column 2 of S0, which is 3, or (11) in binary. Similarly, $(p_{1,0}p_{1,3})$ and $(p_{1,1}p_{1,2})$ are used to index into a row and column of S1 to produce an additional 2 bits.

Next, the 4 bits produced by S0 and S1 undergo a further permutation as follows:

P4			
2	4	3	1

The output of P4 is the output of the function F.

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 bits. In this second instance, the E/P, S0, S1, and P4 functions are the same. The key input is K_2.

Analysis of Simplified DES

A brute-force attack on simplified DES is certainly feasible. With a 10-bit key, there are only $2^{10} = 1024$ possibilities. Given a ciphertext, an attacker can try each possibility and analyze the result to determine if it is reasonable plaintext.

What about cryptanalysis? Let us consider a known plaintext attack in which a single plaintext $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$ and its ciphertext output $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$ are known and the key $(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10})$ is unknown. Then each c_i is a polynomial function g_i of the p_j's and k_j's. We can therefore express the encryption algorithm as 8 nonlinear equations in 10 unknowns. There are a number of possible solutions, but each of these could be calculated and then analyzed. Each of the permutations and additions in the algorithm is a linear mapping. The nonlinearity comes from the S-boxes. It is useful to write down the equations for these boxes. For clarity, rename $(p_{0,0}, p_{0,1}, p_{0,2}, p_{0,3}) = (a, b, c, d)$ and $(p_{1,0}, p_{1,1}, p_{1,2}, p_{1,3}) = (w, x, y, z)$, and let the 4-bit output be (q, r, s, t) Then the operation of the S0 is defined by the following equations:

$$\begin{aligned} q &= abcd + ab + ac + b + d \\ r &= abcd + abd + ab + ac + ad + a + c + 1 \end{aligned}$$