SIMPLIFIED DES

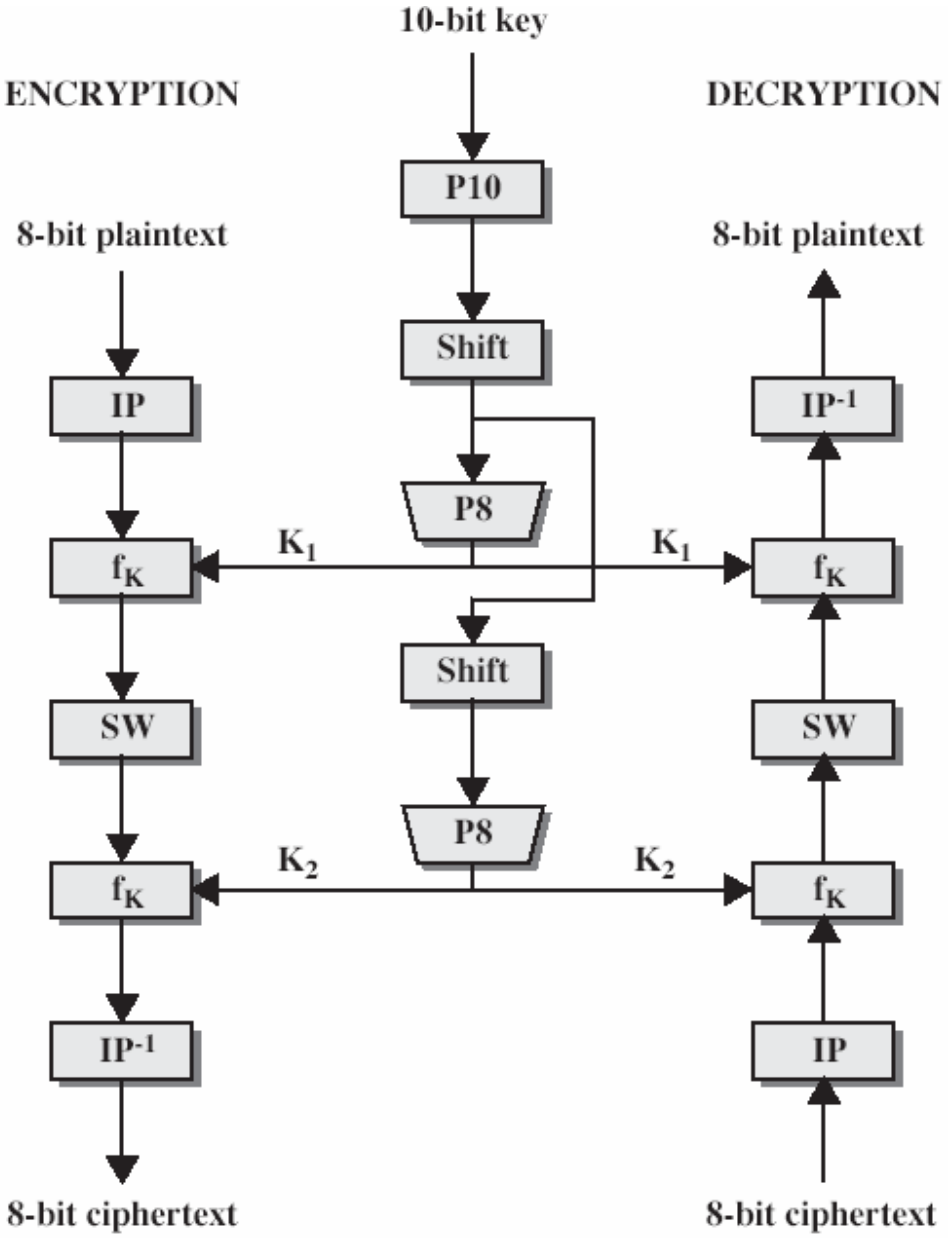

Figure 3.1 Simplified DES Scheme

S-DES encryption (decryption) algorithm takes 8-bit block of plaintext (ciphertext) and a 10-bit key, and produces 8-bit ciphertext (plaintext) block. Encryption algorithm involves 5 functions: an initial permutation (IP); a complex function f_K, which involves both permutation and substitution and depends on a key input; a simple permutation function that switches (SW) the 2 halves of the data; the function f_K again; and finally, a permutation

function that is the inverse of the initial permutation (IP^{-1}). Decryption process is similar.

The function f_K takes 8-bit key which is obtained from the 10-bit initial one two times. 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 (K1). The output of the shift operation also feeds into another shift and another instance of P8 to produce the 2nd subkey K2.

We can express encryption algorithm as superposition:

$$IP^{-1} \circ f_{K_2} \circ SW \circ f_{K_1} \circ IP$$

or

$$\text{Ciphertext= } IP^{-1}\left(f_{K_2}(SW(f_{K_1}(IP(pla\,\text{int}\,ext))))\right)$$

Where

$$K_1 = P8(Shift(P10(key)))$$

$$K_2 = P8(Shift(Shift(P10(key))))$$

Decryption is the reverse of encryption:

$$\text{Plaintext= } IP^{-1}\left(f_{K_1}(SW(f_{K_2}(IP(ciphertext))))\right)$$

We now examine S-DES in more details

S-DES KEY GENERATION

Scheme of key generation:

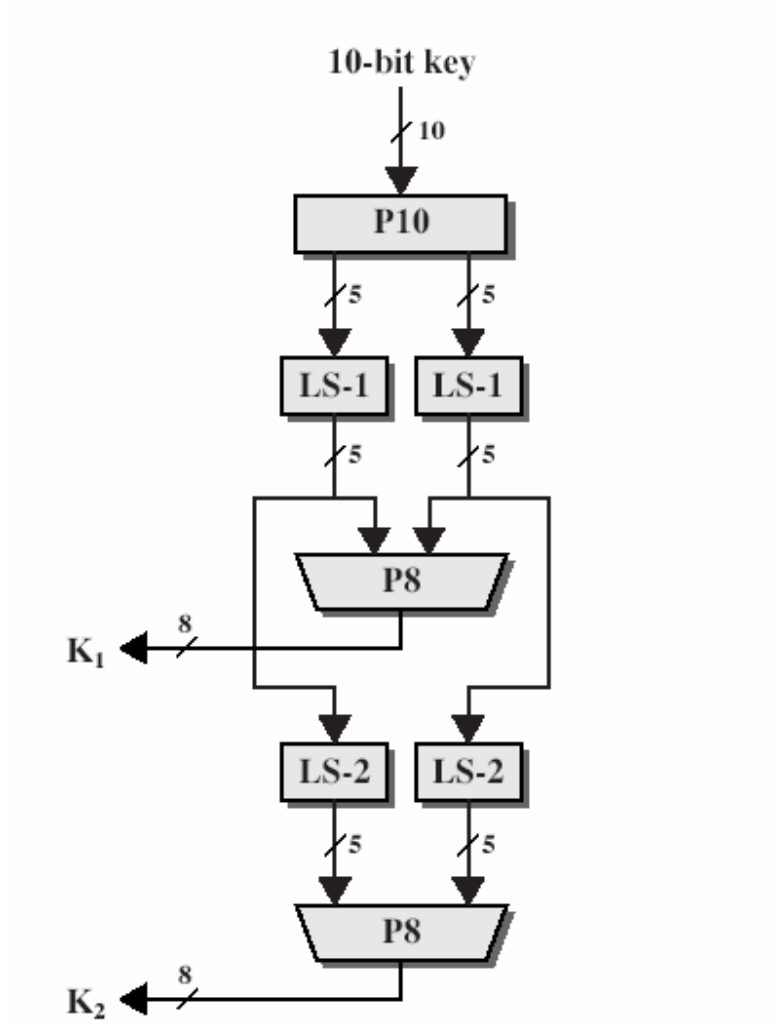

Figure 3.2 Key Generation for Simplified DES

First, permute the 10-bit key k1,k2,..,k10:
P10(k1,k2,k3,k4,k5,k6,k7,k8,k9,k10)=(k3,k5,k2,k7,k4,k10,k1,k9,k8,k6)
Or it may be represented in such a form

P10
3 5 2 7 4 10 1 9 8 6

Each position in this table gives the identity of the input bit that produces the output bit in this position. So, the 1st output bit is bit 3 (k3), the 2nd is k5 and so on. For example, the key (1010000010) is permuted to (1000001100). Next, perform a circular shift (LS-1), or rotation, separately on the 1st 5 bits and the 2nd 5 bits. In our example, the result is (00001 11000)
Next, we apply P8, which picks out and permutes 8 out of 10 bits according to the following rule:

P8
6 3 7 4 8 5 10 9

The result is subkey K1. In our example, this yields (10100100)
We then go back to the pair of 5-bit strings produced by the 2 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 K2. In our example, the result is (01000011)

S-DES ENCRYPTION

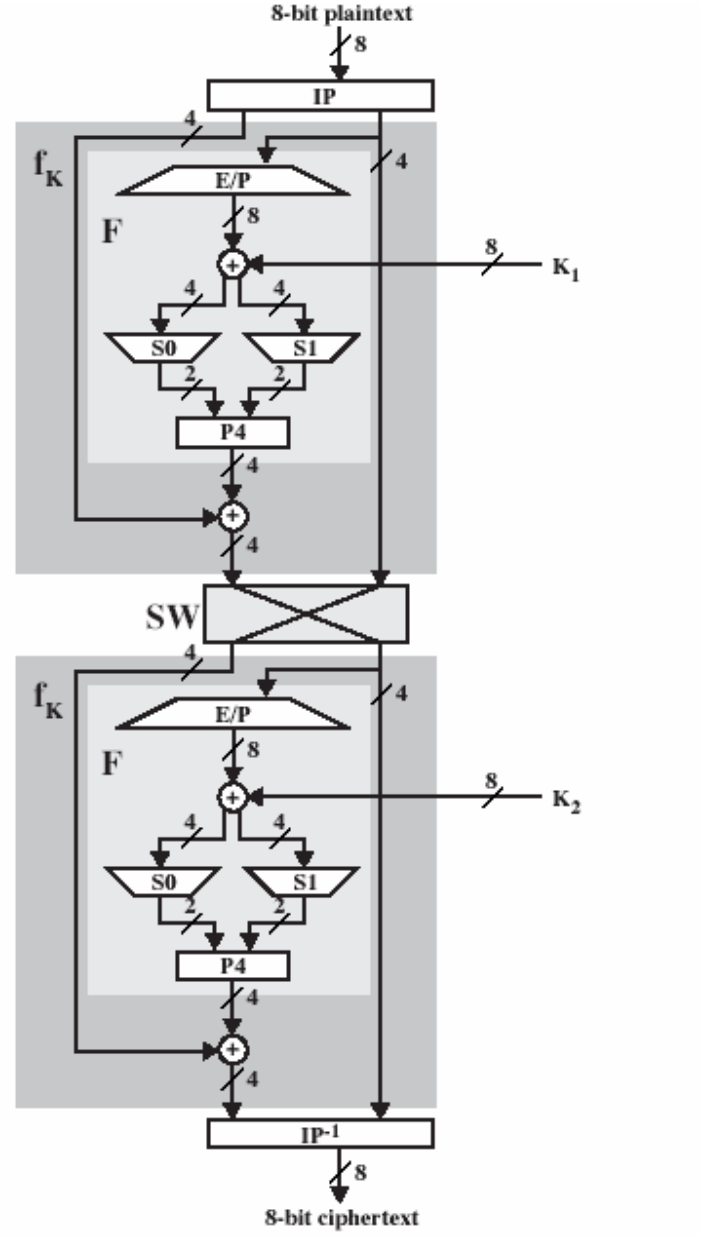

Figure 3.3 Simplified DES Encryption Detail

The input to the algorithm is an 8-bit block of plaintext, which is permuted by IP function:

IP
2 6 3 1 4 8 5 7

At the end of the algorithm, the inverse permutation is used:

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

It may be verified, that $IP^{-1}(IP(X)) = X$.

The most complex component of S-DES is the function f_K, which consists of a combination of permutation and substitution functions. The function 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 XOR operation. For example, suppose the output of the IP stage in Fig.3.3 is (1011 1101) and F(1101,SK) = (1110) for some key SK. Then f_K(1011 1101) = (0101 1101) because (1011) $\oplus$ (1110) = (0101).

We now describe the mapping F. The input is a 4-bit number (n1 n2 n3 n4). The 1st operation is an expansion/permutation:

E/P
4 1 2 3 2 3 4 1

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

n4|n1 n2|n3
n2|n3 n4|n1

The 8-bit subkey K1 = (k11, k12, k13, k14, k15, k16, k17, k18) is added to this value using XOR:

n4+k11|n1+k12 n2+k13|n3+k14
n2+k15|n3+k16 n4+k17|n1+k18

Let us rename these bits:

p00|p01 p02|p03
p10|p11 p12|p13

The 1st 4 bits (1st row of the preceding matrix) are fed into the S-box S0 to produce a 2-bit output, and the remaining 4 bits (2nd row) are fed into S1 to produce another 2-bit output. These 2 boxes are defined as follows:

$$S0 = \begin{matrix} 0\,1\,2\,3 \\ \end{matrix} \quad \begin{matrix} 0\,1\,2\,3 \\ \end{matrix}$$

$$\begin{pmatrix} 1\,0\,3\,2 \\ 3\,2\,1\,0 \\ 0\,2\,1\,3 \\ 3\,1\,3\,2 \end{pmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} \qquad S1 = \begin{pmatrix} 0\,1\,2\,3 \\ 2\,0\,1\,3 \\ 3\,0\,1\,0 \\ 2\,1\,0\,3 \end{pmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

The S-boxes operate as follows. The 1st and 4th input bits are treated as a 2-bit number that specify a row of the S-box, and the 2nd and 3rd 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 (p00, p03) = (00) and (p01, p02) = (10), then the output is from row 0, column 2 of S0, which is 3, or (11) in binary.

Similarly, (p10, p13) and (p11, p12) 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 function F.
The function f_K only alters the leftmost 4 bits of input.
The switch function SW interchanges the left and right bits so that the 2^{nd} instance of f_K operates on a different 4 bits. In the 2^{nd} instance, the E/P, S0, S1, and P4 functions are the same. The key input is K2.