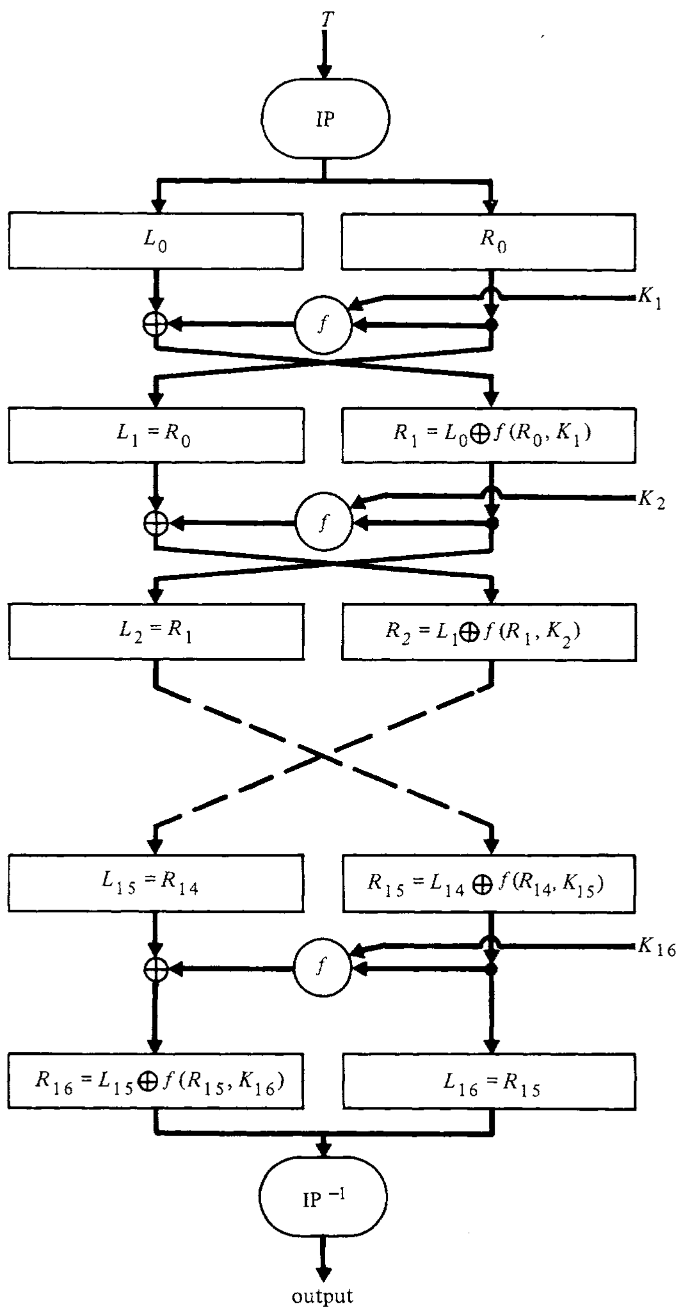
FIGURE 2.13 DES enciphering algorithm.


T
IP
L_0
R_0
K_1
f
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
K_2
f
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
f
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$L_{16} = R_{15}$
IP^{-1}
output

12-bits, each bit of plaintext can conceivably affect each bit of ciphertext. (See [Feis70,Feis75,Kam78] for details on the design of substitution-permutation ciphers.)

2.6.2 The Data Encryption Standard (DES)

In 1977 the National Bureau of Standards announced a Data Encryption Standard to be used in unclassified U.S. Government applications [NBS77]. The encryption algorithm was developed at IBM, and was the outgrowth of LUCIFER.

DES enciphers 64-bit blocks of data with a 56-bit key. There is disagreement over whether a 56-bit key is sufficiently strong; we shall discuss this controversy after describing the algorithm.

The algorithm—which is used both to encipher and to decipher—is summarized in Figure 2.13. An input block T is first transposed under an initial permutation IP, giving $T_0 = \mathrm{IP}(T)$. After it has passed through 16 iterations of a function f, it is transposed under the inverse permutation IP^{-1} to give the final result. The permutations IP and IP^{-1} are given in Tables 2.3(a) and 2.3(b), respectively. These tables (as well as the other permutation tables described later) should be read left-to-right, top-to-bottom. For example, IP transposes $T = t_1 t_2 \dots t_{64}$ into $T_0 = t_{58} t_{50} \dots t_7$. All tables are fixed.

Between the initial and final transpositions, the algorithm performs 16 iterations of a function f that combines substitution and transposition. Let T_i denote the result of the ith iteration, and let L_i and R_i denote the left and right halves of T_i, respectively; that is, $T_i = L_i R_i$, where

$$L_i = t_1 \dots t_{32}$$
$$R_i = t_{33} \dots t_{64} \ .$$

Then

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

where "$\oplus$" is the exclusive-or operation and K_i is a 48-bit key described later. Note that after the last iteration, the left and right halves are not exchanged; instead the

TABLE 2.3(a) Initial permutation IP.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

TABLE 2.3(b) Final permutation IP^{-1}.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

shown in Table 2.4. This table is used in the same way as the permutation tables, except that some bits of R_{i-1} are selected more than once; thus, given $R_{i-1} = r_1 r_2 \ldots r_{32}$, $\mathrm{E}(R_{i-1}) = r_{32} r_1 r_2 \ldots r_{32} r_1$.

Next, the exclusive-or of $\mathrm{E}(R_{i-1})$ and K_i is calculated and the result broken into eight 6-bit blocks $B_1, \ldots, B_8$, where

$$\mathrm{E}(R_{i-1}) \oplus K_i = B_1 B_2 \ldots B_8 \ .$$

TABLE 2.6 Selection functions (S-boxes).

								Column									
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Each 6-bit block B_j is then used as input to a selection (substitution) function (**S-box**) S_j, which returns a 4-bit block $S_j(B_j)$. These blocks are concatenated together, and the resulting 32-bit block is transposed by the permutation P shown in Table 2.5. Thus, the block returned by $f(R_{i-1}, K_i)$ is

$$P(S_1(B_1) \dots S_8(B_8)) \ .$$

Each S-box S_j maps a 6-bit block $B_j = b_1b_2b_3b_4b_5b_6$ into a 4-bit block as defined in

FIGURE 2.15 Key schedule calculation.

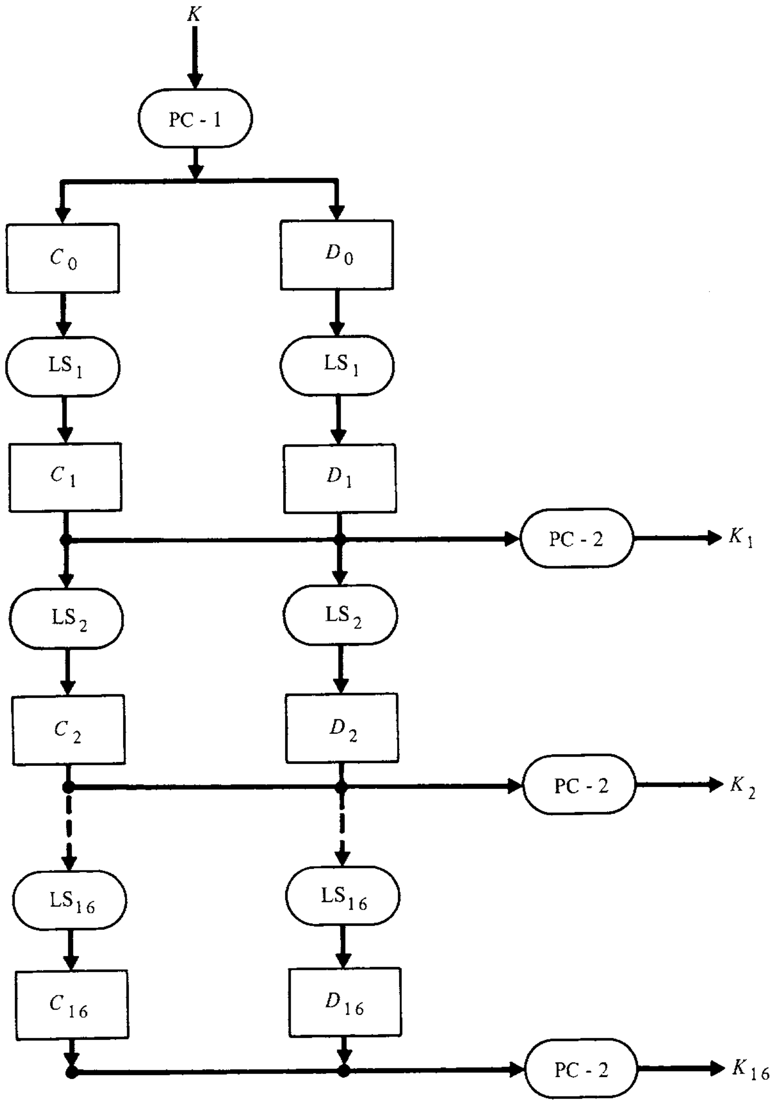


Table 2.6. This is done as follows: The integer corresponding to b_1b_6 selects a row in the table, while the integer corresponding to $b_2b_3b_4b_5$ selects a column. The value of $S_j(B_j)$ is then the 4-bit representation of the integer in that row and column.

> ***Example:***
> If $B_1 = 010011$, then S_1 returns the value in row 1, column 9; this is 6, which is represented as 0110. ■

Key calculation. Each iteration i uses a different 48-bit key K_i derived from the initial key K. Figure 2.15 illustrates how this is done. K is input as a 64-bit block, with 8 parity bits in positions 8, 16, . . . , 64. The permutation PC-1 (permuted choice 1) discards the parity bits and transposes the remaining 56 bits as shown in Table 2.7. The result PC-1(K) is then split into two halves C and D of 28 bits each. The blocks C and D are then successively shifted left to derive each key K_i. Letting C_i and D_i denote the values of C and D used to derive K_i, we have

$$C_i = LS_i(C_{i-1}),\ D_i = LS_i(D_{i-1}),$$

where LS_i is a left circular shift by the number of positions shown in Table 2.8, and C_0 and D_0 are the initial values of C and D. Key K_i is then given by

$$K_i = \text{PC-2}(C_i\, D_i), \text{ where}$$

PC-2 is the permutation shown in Table 2.9.

Deciphering. Deciphering is performed using the same algorithm, except that K_{16} is used in the first iteration, K_{15} in the second, and so on, with K_1 used in the

TABLE 2.7 Key permutation PC-1.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

TABLE 2.8 Key schedule of left shifts LS.

Iteration i	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

TABLE 2.9 Key permutation PC-2.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

16th iteration. This is because the final permutation IP^{-1} is the inverse of the initial permutation IP, and

$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus f(L_i, K_i) .$$

Note that whereas the order of the keys is reversed, the algorithm itself is not.

Implementation. DES has been implemented both in software and in hardware. Hardware implementations achieve encryption rates of several million bps (bits per second).

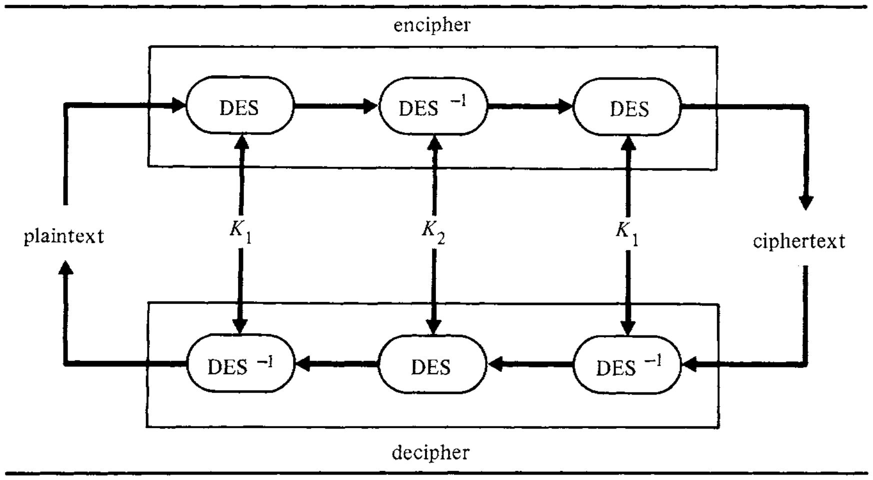
In 1976 (before DES was adopted), the National Bureau of Standards held two workshops to evaluate the proposed standard. At that time, Diffie and Hellman, and others, were concerned about possible weaknesses (see [Hell76, Diff77]. After the second workshop, Morris, Sloane, and Wyner [Morr77] reported that they believed the DES had two major weaknesses:

1. *Key size:* 56 bits may not provide adequate security.
2. *S-boxes:* The S-boxes may have hidden trapdoors.

Diffie and Hellman argue that with 56-bit keys, DES may be broken under a known-plaintext attack by exhaustive search. In [Diff77] they show that a special-purpose machine consisting of a million LSI chips could try all $2^{56} \cong 7 \times 10^{16}$ keys in 1 day. Each chip would check one key per μsec or 8.64×10^{10} keys per day. Whereas it would take almost 10^6 days for one chip to check all keys, 10^6 chips can check the entire key space in 1 day. The cost of such a machine would be about \$20 million. Amortized over 5 years, the cost per day would be about \$10,000. Because on the average only half the key space would have to be searched, the average search time would be half a day, making the cost per solution only \$5,000. More recently, Diffie [Diff81] has increased this estimate to a 2-day average search time on a \$50M machine (using 1980 technology). But he and Hellman [Hell79] both predict the cost of building a special-purpose DES search machine will drop substantially by 1990.

Hellman [Hell80] has also shown that it is possible to speed up the searching process by trading time for memory in a chosen-plaintext attack (see following section). The cost per solution would be \$10 on a \$5M machine. Because the

FIGURE 2.16 Multiple encipherment with DES.



attack can be thwarted with techniques such as cipher block chaining (see Chapter 3), the search strategy itself does not pose a real threat.

Hellman and others argue that the key size should be doubled (112 bits). Tuchman [Tuch79] claims that the same level of security can be obtained with 56-bit keys, using a multiple encryption scheme invented by Matyas and Meyer. Let DES_K denote the encipering transformation for key K, and DES_K^{-1} the corresponding deciphering transformation. A plaintext message M is enciphered as

$$C = \text{DES}_{K_1}\left(\text{DES}_{K_2}^{-1}(\text{DES}_{K_1}(M))\right);$$

that is, it is successively enciphered, deciphered, and then enciphered again, using one key (K_1) for the encipherments and another (K_2) for the decipherment (see Figure 2.16). The message M is restored by reversing the process and applying the inverse transformations:

$$\text{DES}_{K_1}^{-1}\left(\text{DES}_{K_2}(\text{DES}_{K_1}^{-1}(C))\right) = M.$$

Merkle and Hellman [Merk81] believe this approach may be less secure than using a single 112-bit key (or even using 3 separate keys in a triple-encryption scheme), showing that it can be theoretically broken under a chosen-plaintext attack using about 2^{56} operations and 2^{56} keys stored on 4 billion tapes.

Hellman and others have also questioned whether the S-boxes are secure (the analysis behind their design is presently classified). They believe that the design should be unclassified so that it may be publicly evaluated. (See [Suga79, Hell79,Davi79,Bran79,Tuch79] for a debate on these issues.)

2.6.3 Time-Memory Tradeoff

There are two naive approaches to breaking a cipher with n keys: exhaustive search and table lookup. **Exhaustive search** uses a known-plaintext attack: Given