Block ciphers and the Data Encryption Standard

1. Simplified DES
2. Block cipher principles
3. DES algorithm
4. Strength of DES
5. Differential and linear cryptanalysis
6. Block cipher design principles
7. Block cipher modes of operation

SIMPLIFIED DES

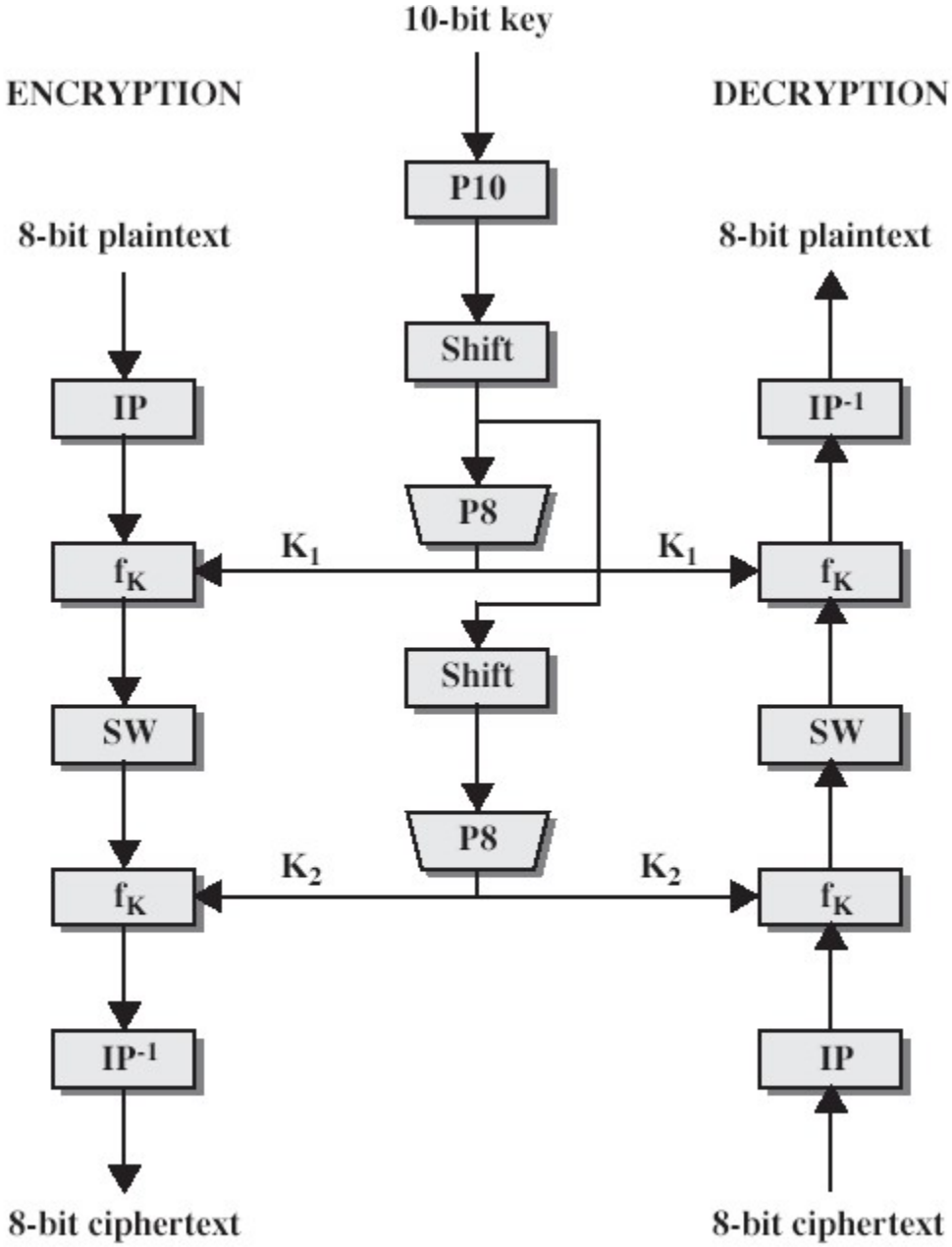

Figure 3.1 Simplified DES Scheme

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

SIMPLIFIED DES (CONT 1)

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

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

Where

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

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

Decryption is the reverse of encryption:

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

We now examine S-DES in more details

S-DES KEY GENERATION

Scheme of key generation:

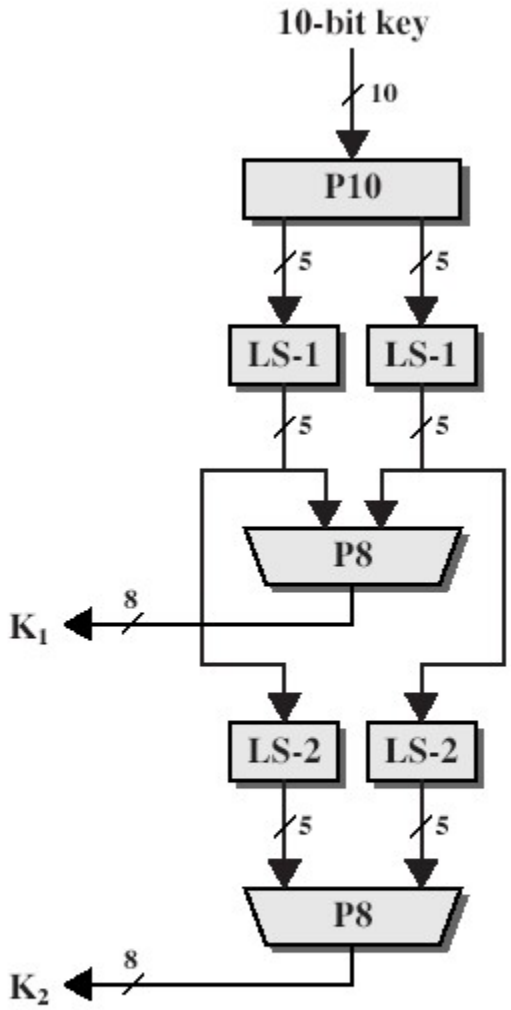

Figure 3.2 Key Generation for Simplified DES

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

S-DES KEY GENERATION (CONT 1)

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

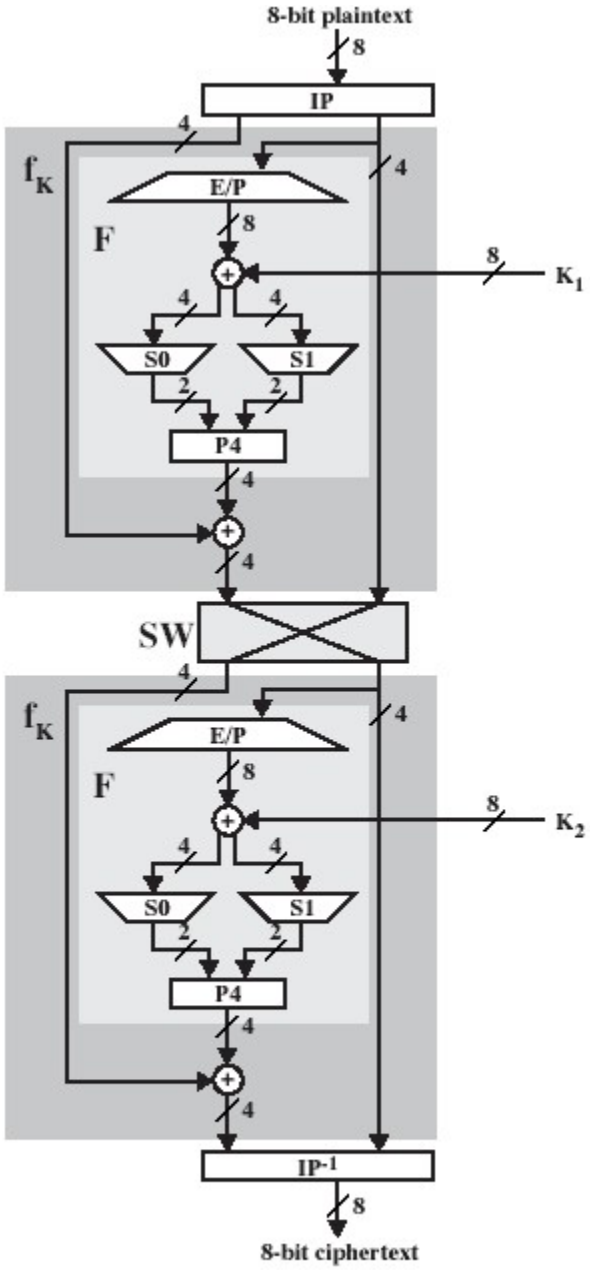

Figure 3.3 Simplified DES Encryption Detail

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:

S-DES ENCRYPTION (CONT 1)

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:

$$S0 = \begin{matrix} 0\,1\,2\,3 \\ \begin{pmatrix} 1\,0\,3\,2 \\ 3\,2\,1\,0 \\ 0\,2\,1\,3 \\ 3\,1\,3\,2 \end{pmatrix} \end{matrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} \qquad S1 = \begin{matrix} 0\,1\,2\,3 \\ \begin{pmatrix} 0\,1\,2\,3 \\ 2\,0\,1\,3 \\ 3\,0\,1\,0 \\ 2\,1\,0\,3 \end{pmatrix} \end{matrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

S-DES ENCRYPTION (CONT 2)

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 2nd instance of f_K operates on a different 4 bits. In the 2nd instance, the E/P, S0, S1, and P4 functions are the same. The key input is K2.

ANALYSIS OF SIMPLIFIED DES

A brute-force attack on S-DES is feasible since with a 10-bit key there are only 1024 possibilities.

What about cryptanalysis? If we know plaintext (p1p2p3p4p5p6p7p8) and respective ciphertext (c1c2c3c4c5c6c7c8), and key (k1k2k3k4k5k6k7k8k9k10) is unknown, then we can express this problem as a system of 8 nonlinear equations with 10 unknowns. The nonlinearity comes from the S-boxes. It is useful to write down equations for these boxes. For clarity, rename (p00,p01,p02,p03)=(a,b,c,d) and (p10,p11,p12,p13)=(w,x,y,z). Then the operation of S0 is defined in the following equations:

q=abcd+ab+ac+b+d

r=abcd+abd+ab+ac+ad+a+c+1

where all additions are made modulo 2. Similar equations define S1.

Let us show it.

ANALYSIS OF SIMPLIFIED DES (CONT 1)

Truth table for S0:

	q	r	a	d	b	c
0	0	1	0	0	0	0
1	0	0	0	0	0	1
2	1	1	0	0	1	0
3	1	0	0	0	1	1
4	1	1	0	1	0	0
5	1	0	0	1	0	1
6	0	1	0	1	1	0
7	0	0	0	1	1	1
8	0	0	1	0	0	0
9	1	0	1	0	0	1
10	0	1	1	0	1	0
11	1	1	1	0	1	1
12	1	1	1	1	0	0
13	0	1	1	1	0	1
14	1	1	1	1	1	0
15	1	0	1	1	1	1

$$q = (a \vee d \vee b \vee c)(a \vee d \vee b \vee \overline{c})(a \vee \overline{d} \vee \overline{b} \vee c)$$

$$(a \vee \overline{d} \vee \overline{b} \vee \overline{c})(\overline{a} \vee d \vee b \vee c)(\overline{a} \vee d \vee \overline{b} \vee c)(\overline{a} \vee \overline{d} \vee b \vee \overline{c})$$

$$= (a \vee d \vee b)(a \vee \overline{d} \vee \overline{b})(\overline{a} \vee d \vee c)(\overline{a} \vee \overline{d} \vee b \vee \overline{c}) =$$

$$(a \vee d\overline{b} \vee b\overline{d})(\overline{a} \vee bd \vee \overline{d}c \vee c\overline{d} \vee bc) =$$

$$abd \vee \overline{a}\overline{b}d \vee \overline{a}b\overline{d} \vee abc \vee a\overline{c}d \vee ac\overline{d} \vee bc\overline{d} \vee \overline{b}\overline{c}d =$$

$$(abd + (1+a)(1+b)d) \vee (1+a)b(1+d) \vee abc \vee (a(1+c)d + ac(1+d))$$

$$\vee (bc(1+d) + (1+b)(1+c)d) = (d + ad + bd) \vee (b + ab + bd + abd) \vee abc \vee$$

$$(ad + ac) \vee (bc + d + bd + cd) = (d + ad + b + ab + bd) \vee$$

$$(abd + acd + ac + bc + d + bd + cd + abcd + abc) = abcd + ac + ab + b + d$$

Alternating linear maps with these nonlinear maps results in very complex polynomial expressions for the ciphertext bits, making cryptanalysis difficult.

RELATIONSHIP TO DES

DES operates on 64-bit blocks of input. The encryption scheme can be defined as

$$IP^{-1} \circ f_{K_{16}} \circ SW \circ f_{K_{15}} \circ SW \circ ...SW \circ f_{K_1} \circ SW \circ IP$$

RELATIONSHIP TO DES (CONT 1)

A 56-bit key is used, from which 16 48-bit subkeys are calculated. There is an initial permutation of 56 bits followed by a sequence of shifts and permutations of 48 bits.

Within the encryption algorithm, instead of F acting on 4 bits (n1n2n3n4), it acts on 32 bits (n1n2..n32). After the initial expansion/permutation, the output of 48 bits can be diagrammed as

n32|n1 n2 n3 n4 |n5

n4 |n5 n6 n7 n8 |n9

…

…

…

n28|n29 n30 n31 n32|n1

This matrix is added (XOR) to a 48-bit subkey. There 8 rows, corresponding to 8 S-boxes. Each S-box has 4 rows and 16 columns. The 1st and last bit of a row of the preceding matrix picks out a row of an S-box, and the middle 4 bits pick out a column.

BLOCK CIPHER PRINCIPLES

Stream ciphers – Vigenere autokey, Vernam cipher – encrypts data element by element

Block ciphers treat a block of plaintext as a whole. Typically, a block size is 64 or 128 bits. They are more popular than stream ciphers and mostly based on Feistel cipher structure (Horst Feistel, IBM, 1973, http://en.wikipedia.org/wiki/Horst_Feistel).

MOTIVATION FOR THE FEISTEL CIPHER STRUCTURE

Encryption should be reversible. Fig. 3.4 shows the logic of a general substitution cipher for n=4 (block size).

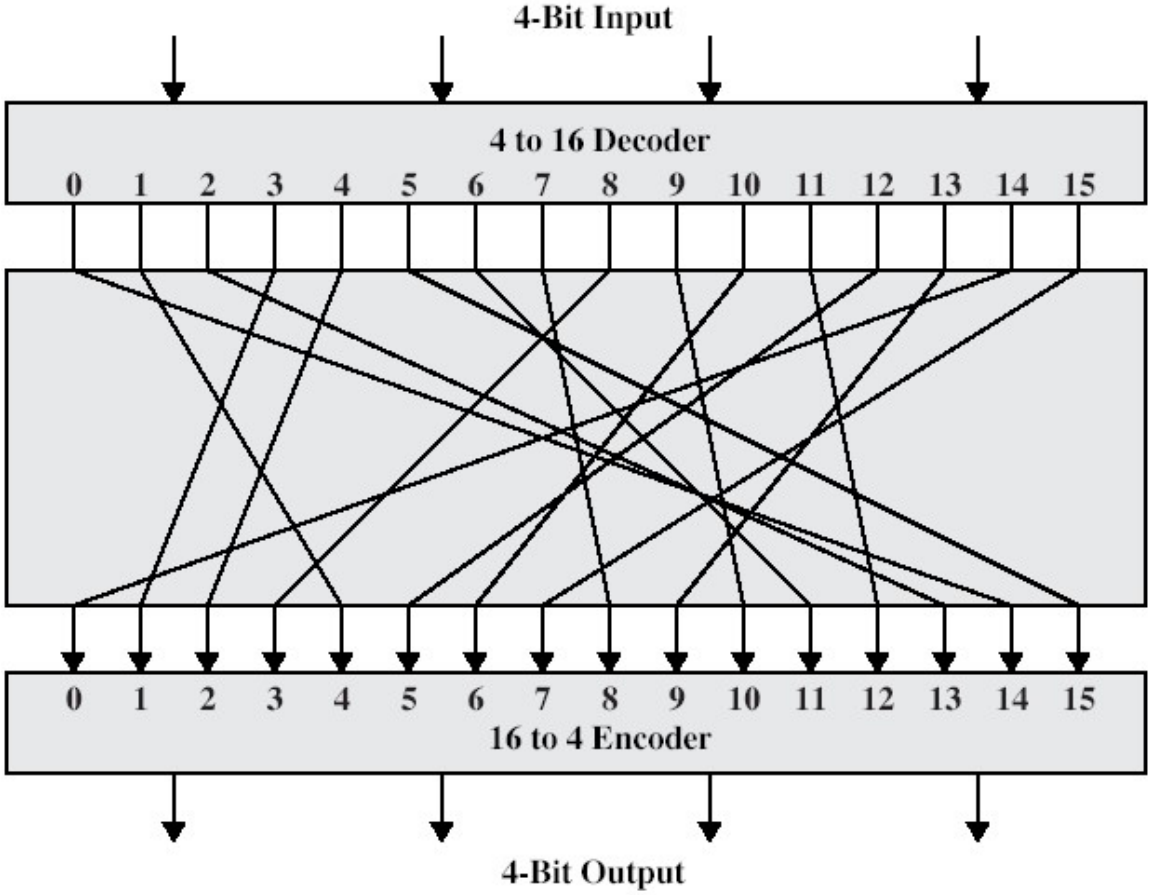

Figure 3.4 General *n*-bit-*n*-bit Block Substitution (shown with $n = 4$)

The encryption and decryption tables can be defined by tabulation, as shown in Table 3.1:

MOTIVATION FOR THE FEISTEL CIPHER STRUCTURE (CONT 1)

Table 3.1 Encryption and Decryption Tables for Substitution Cipher of Figure 3.4

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

If n is small, then statistical characteristics of plaintext survive in the ciphertext. If n is large, then number of possible mappings becomes large, each of them is a key of the cipher, the size of the key is $n2^n$. For 64 bits key size is $2^{70} \approx 10^{21}$ bits. Such enormous size of the key makes its use impossible. Feistel points out that what is needed is an approximation to this ideal block-cipher system for large n, built up out of components that are easily realizable.

THE FEISTEL CIPHER

Feistel proposed that we can approximate the simple substitution cipher by utilizing the concept of a product cipher, which is the performing of two or more basic ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations. In fact, this is a practical application of a

THE FEISTEL CIPHER (CONT 1)

proposal by Claude Shannon of 1945 (http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Shannon.html) to develop a product cipher that alternates confusion and diffusion functions

DIFFUSION AND CONFUSION

These are measures to thwart cryptanalysis based on statistical analysis. In diffusion, the statistical structure of the plaintext is dissipated into long range statistics of the ciphertext. This is achieved by having each plaintext letter affect the value of many ciphertext digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. An example of diffusion is to encrypt a message M=m1,m2,m3,.. of characters with an averaging operation:

$$y_n = \sum_{i=1}^{k} m_{n+i} \pmod{26}$$

adding k successive letters to get a ciphertext letter y_n. The letter frequencies in the ciphertext will be more nearly equal than in the plaintext (structure dissipated).

Confusion seeks to make the relationship between the statistics of the ciphertext and and the value of the encryption key as complex as possible. This is achieved by the use of a complex substitution algorithm. These operations became the cornerstone of modern block cipher design.

FEISTEL CIPHER STRUCTURE

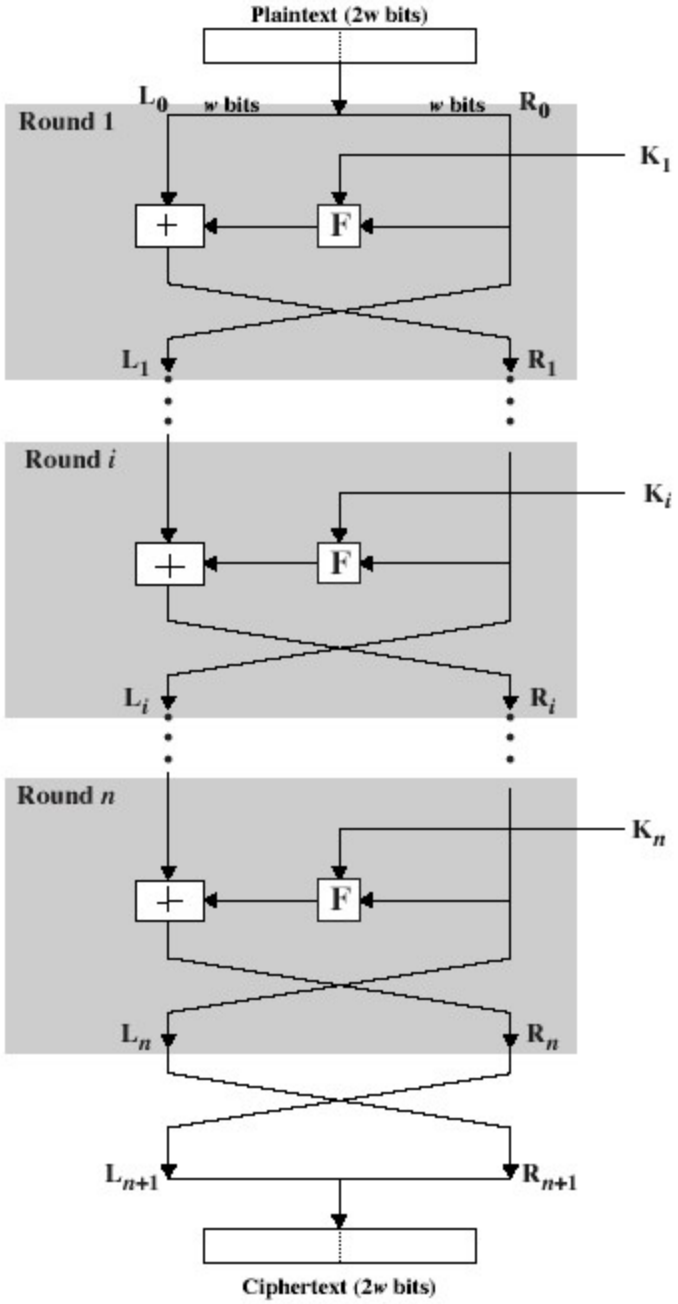

Figure 3.5 Classical Feistel Network

The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into 2 halves, L0 and R0. The 2 halves of the data pass through n rounds of processing and the combine to produce the ciphertext block. Each round i has as inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as a subkey K_i, derived from the overall K. In general, the subkeys K_i are different from K and from each other.

FEISTEL CIPHER STRUCTURE (CONT 1)

All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking exclusive –OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey K_i. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

Block size: large size means greater security but greater overhead (64, 128 bits)

Key size: large size means greater security but greater overhead (64, 128 bits)

Number of rounds: multiple rounds increase security (16 rounds)

Subkey generation algorithm: greater complexity – more secure

Round function: greater complexity – more secure

Additionally:

Fast software encryption/decryption: speed of execution becomes a concern

Ease of analysis: it should be difficult to cryptanalyze, but easy to analyze for cryptanalytic vulnerabilities.

We can see that SDES exhibits a Feistel structure with 2 rounds. The one difference from a "pure" Feistel structure is that the algorithm begins and ends with a permutation function. This difference also appears in full DES.

FEISTEL DECRYPTION ALGORITHM

The process of decryption with a Feistel cipher is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the algorithm, but the subkeys K_i in the reverse order. That is, use K_n in the 1^{st} round, and so on, K_1 in the last round. This is a nice feature, because we can use just one algorithm both for encryption and decryption.

FEISTEL DECRYPTION ALGORITHM (CONT 1)

Consider encryption/decryption processes:

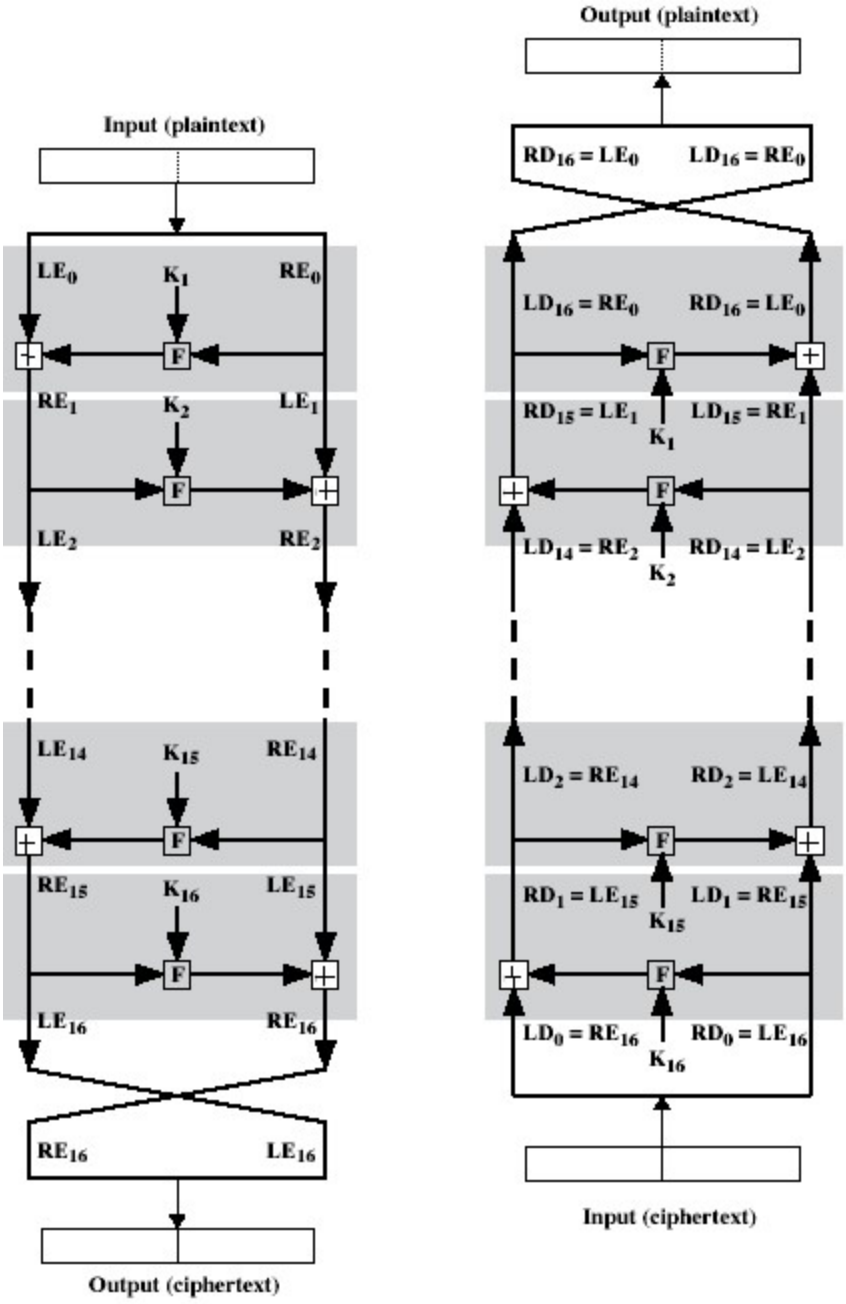

Figure 3.6 Feistel Encryption and Decryption

Let, REi – data travelling through encryption, LDi, RDi – data travelling through decryption. Output of ith encryption round is LEi||REi (concatenation). To simplify the diagram, it is untwisted, not showing the

swap that occurs at the end of each interaction. But intermediate result at the end of ith stage of the encryption process is the 2w-bit LEi||REi, and

FEISTEL DECRYPTION ALGORITHM (CONT 2)

the intermediate result at the end of the ith stage of decryption is LDi||RDi. Then the corresponding input to (16-i)th decryption round is LEi||REi, or, equivalently, RD_{16-i} ||LD_{16-i}. Let's prove that.

After the last iteration, the two halves are swapped, so that the ciphertext is RE16||LE16. Now take the ciphertext and use it as input to the same algorithm. The input to the 1st round is RE16||LE16, which is equal to the 32-bit swap of the output of the 16th round of the encryption process. Now we show that the output of the 1st round of the decryption process is equal to a 32-bit swap of the output of the 15th round of the encryption process. First, consider encryption process,

$$LE16=RE15$$
$$RE16=LE15+F(RE15,K16)$$

On the decryption side,

$$LD1=RD0=LE16=RE15$$
$$RD1=LD0+F(RD0,K16)=RE16+F(RE15,K16)=$$
$$[LE15+F(RE15,K16)]+F(RE15,K16)=LE15$$

Thus, we have

$$LD1=RE15$$
$$RD1=LE15,$$

So, we got that output of the 1st stage of decryption process is equal to 32-bit swap of the 15th round of the encryption process: LD1||RD1=RE15||LE15, and continuing these considerations, we come to

$$LDi||RDi=RE(16-i)||LE(16-i).$$

Also, we can write

$$LEi=RE(i-1)$$
$$REi=LE(i-1)+F(RE(i-1),Ki)$$

or

$$RE(i-1)=LEi$$
$$LE(i-1)=REi+F(RE(i-1),Ki)= REi+F(LEi,Ki)$$

and these equations confirm the assignments shown in the right-hand side of Figure 3.6.

Output of the last round of the decryption process is

$$LD16||RD16=RE0||LE0$$

A 32-bit swap recovers the original plaintext. Note that the derivation does not require that F be a reversible function (for example, it may be a constant value 1).