04 (05) September, 2024

Block Cipher Structure and Design Principles
Feistel Cipher Structure

(mostly taken from William Stallings, *Cryptography and Network Security: Principles and Practices*)

Traditional Block Cipher Structure

Several important symmetric block encryption algorithms in current use are based on a structure referred to as a *Feistel block cipher*. For that reason, it is important to examine the design principles of the *Feistel cipher*. We begin with a comparison of stream ciphers and block ciphers. Then we discuss the motivation for the Feistel block cipher structure. Finally, we discuss some of its implications.

Stream Ciphers and Block Ciphers

A *stream cipher* is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the auto-keyed Vigenère cipher and the Vernam cipher. In the ideal case, a one-time pad version of the Vernam cipher would be used (Figure 3.7), in which the keystream (k_i) is as long as the plaintext bit stream (p_i).

If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream. However, the keystream must be provided to both users in advance via some independent and secure channel. This introduces insurmountable logistical problems if the intended data traffic is very large.

Accordingly, for practical reasons, the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users. In this approach (Figure 4.1a), the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong. That is, it must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream. The two users need only share the generating key, and each can produce the keystream.

A *block cipher* is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key (Figure 4.1b). Using some of the modes of operation explained in Chapter 7, a block cipher can be used to achieve the same effect as a stream cipher.

Far more effort has gone into analyzing block ciphers. In general, they seem applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers. Accordingly, the concern in this chapter, and in our discussions throughout the book of symmetric encryption, will primarily focus on block ciphers.

Motivation for the Feistel Cipher Structure

A block cipher operates on a plaintext block of *n* bits to produce a ciphertext block of *n* bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular. The following examples illustrate nonsingular and singular transformations for n = 2.

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is $2^n!$.

Figure 4.2 illustrates the logic of a general substitution cipher for n = 4. A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits. The encryption and decryption mappings can be defined by a tabulation, as shown in Table 4.1. This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. *Feistel* refers to this as the ***ideal block cipher***, because it allows for the maximum number of possible encryption mappings from the plaintext block.

But there is a practical problem with the ***ideal block cipher***. If a small block size, such as n = 4, is used, then the system is equivalent to a classical substitution cipher. Such systems, as we have seen, are vulnerable to a statistical analysis of the plaintext. This weakness is not inherent in the use of a substitution cipher but rather results from the use of a small block size. If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.

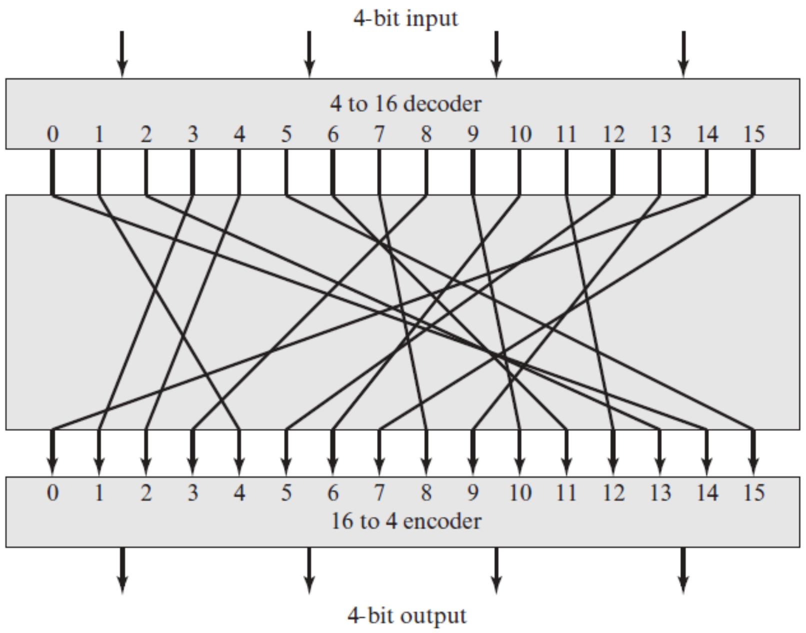


Figure 4.2 General n-bit-n-bit Block Substitution (shown with $n = 4$)

An arbitrary reversible substitution cipher (the ***ideal block cipher***) for a large block size is not practical, however, from an implementation and performance point of view. For such a transformation, the *mapping itself constitutes the key*. Consider again Table 4.1, which defines one particular reversible mapping from plaintext to ciphertext for n = 4. *The mapping can be defined by the entries in the second column, which show the value of the ciphertext for each plaintext block.* This, in essence, is the key that determines the specific mapping from among all possible mappings. In this case, using this straightforward method of defining the key, the required key length is (4 bits) * (16 rows) = 64 bits. In general, for an n-bit ideal block cipher, the length of the key defined in this fashion is $n * 2^n$ bits. For a 64-bit block, which is a desirable length to thwart statistical attacks, the required key length is $64 * 2^{64} = 2^{70} \approx 10^{21}$ bits.

Table 4.1 Encryption and Decryption Tables for Substitution Cipher of Figure 4.2

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

In considering these difficulties, Feistel points out that what is needed is an *approximation to the ideal block cipher system for large n, built up out of components* that are easily realizable [FEIS75]. But before turning to Feistel's approach, let us make one other observation. We could use the general block substitution cipher but, to make its implementation tractable, confine ourselves to a subset of the $2^n!$ possible reversible mappings. *For example, suppose we define the mapping in terms of a set of linear equations. In the case of n = 4, we have*

$$\begin{aligned}
y_1 &= k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4 \\
y_2 &= k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4 \\
y_3 &= k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4 \\
y_4 &= k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4
\end{aligned}$$

where the x_i are the four binary digits of the plaintext block, the y_i are the four binary digits of the ciphertext block, the k_{ij} are the binary coefficients, and arithmetic is mod 2. The key size is just n^2, in this case 16 bits. The danger with this kind of formulation is that it may be vulnerable to cryptanalysis by an attacker that is aware of the structure of the algorithm. In this example, what we have is essentially the Hill cipher discussed in Chapter 3 (we omitted that topic), applied to binary data rather than characters. As we saw in Chapter 3, a simple linear system such as this is quite vulnerable.

The Feistel Cipher

Feistel proposed [FEIS73] that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible transformations, rather than the $2^n!$ transformations available with the ideal block cipher.

In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:

- ***Substitution***: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- ***Permutation***: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

In fact, Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions [SHAN49]. We look next at these concepts of *diffusion* and *confusion* and then present the Feistel cipher. But first, it is worth commenting on this remarkable fact: The *Feistel cipher structure*, which dates back over a quarter century and which, in turn, is based on Shannon's proposal of 1945, is the structure used by a number of significant symmetric block ciphers currently in use. In particular, the Feistel structure is used for Triple Data Encryption Algorithm (TDEA), which is one of the two encryption algorithms (along with AES), approved for general use by the National Institute of Standards and Technology (NIST). The Feistel structure is also used for several schemes for format-preserving encryption, which have recently come into prominence. In addition, the Camellia block cipher is a Feistel structure; it is one of the possible symmetric ciphers in TLS and a number of other Internet security protocols. Both TDEA and format-preserving encryption are covered in Chapter 7.

DIFFUSION AND CONFUSION The terms *diffusion* and *confusion* were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system [SHAN49]. Shannon's concern was to thwart cryptanalysis based on statistical analysis. The reasoning is as follows. Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message (probable words). If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, part of the key, or at least a set of keys likely to contain the exact key. In what Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used. The arbitrary substitution cipher that we discussed previously (Figure 4.2) is such a cipher, but as we have seen, it is impractical.

Other than recourse to ideal systems, Shannon suggests two methods for frustrating statistical cryptanalysis: diffusion and confusion. In *diffusion*, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally, this is equivalent to having each ciphertext digit be affected by many plaintext digits. An example of diffusion is to encrypt a message M = $m_1, m_2, m_3, \ldots$ of characters with an averaging operation:

$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

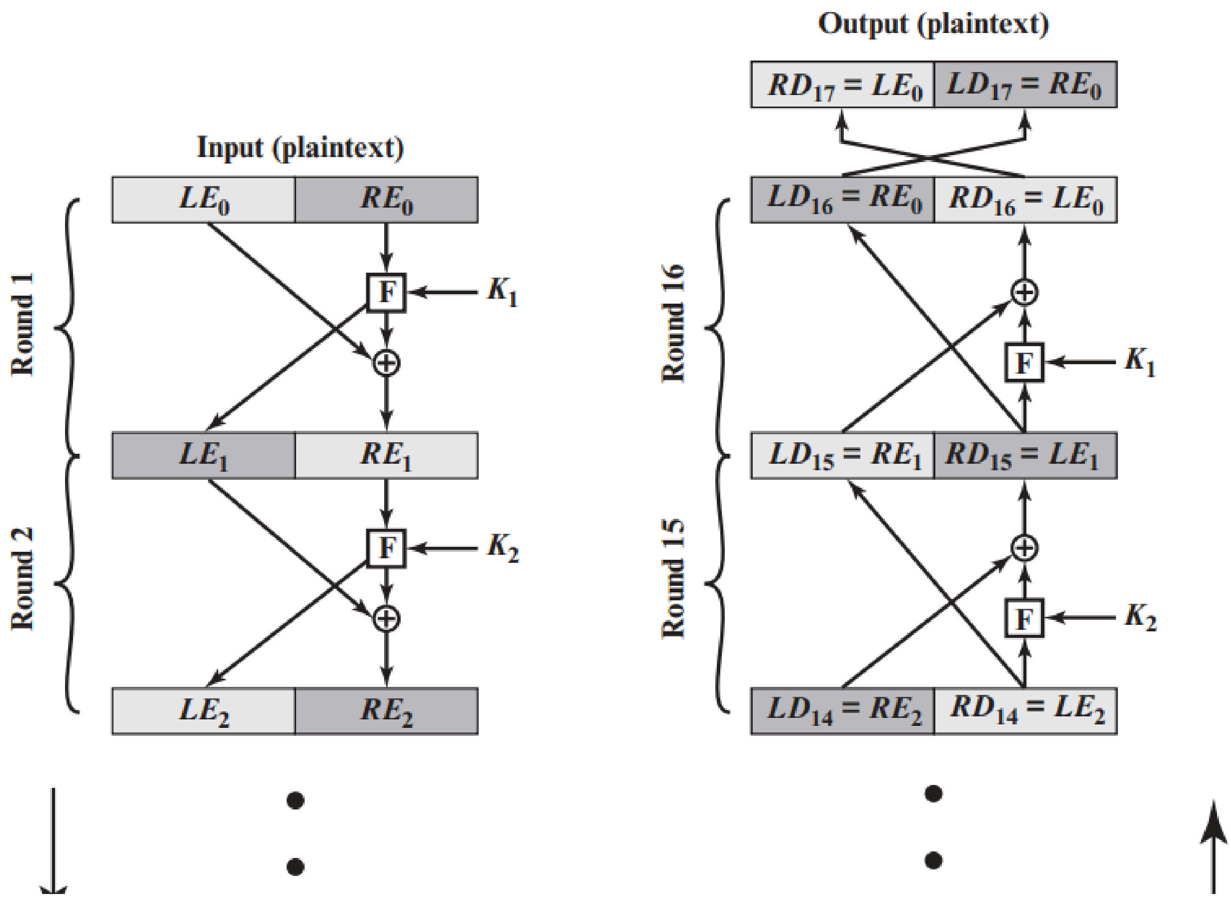
adding k successive letters to get a ciphertext letter y_n. One can show that the statistical structure of the plaintext has been dissipated. Thus, the letter frequencies in the ciphertext will be more nearly equal than in the plaintext; the digram frequencies will also be more nearly equal, and so on. In a binary block cipher, diffusion can be achieved by repeatedly performing some permutation on the data followed by applying a function to that permutation; the effect is that bits from different positions in the original plaintext contribute to a single bit of ciphertext.

Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key. The mechanism of diffusion seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce the key. On the other hand, *confusion* seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by the use of a complex substitution algorithm. In contrast, a simple linear substitution function would add little confusion.

As [ROBS95b] points out, so successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern block cipher design.

FEISTEL CIPHER STRUCTURE The left-hand side of Figure 4.3 depicts the encryption structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. *The plaintext block is divided into two halves, LE_0 and RE_0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block*. Each round i has as inputs LE_{i-1} and RE_{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. In Figure 4.3, 16 rounds are used, although any number of rounds could be implemented.

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 the *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.* Another way to express this is to say that F is a function of the right-half block of w bits and a subkey of y bits, which produces an output value of length w bits: F(RE_i, K_{i+1}). 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 example I discussed in class was taken from here:
https://www.youtube.com/watch?v=fenzYD2J9vs

We used four steps:

Step 1: Split the plaintext into two halves, L_0 and R_0
Step 2: E = fs(R_0), where E is a temporary variable, and fs is a function that is applied on a 3-bit string (essentially it does a right circular shift)
Step 3: $L_1 = R_0$, and, R_1 = (L_0 XOR E)
Step 4: Concatenate L_1 and R_1, thus we finally have our ciphertext

Plaintext : 011110 100001

Step 1: Split the plaintext into two halves

L_0 = 011110; R_0 = 100001

Step 2: E = fs(R_0) – note that fs applies on 3-bit strings, here R_0 is 6-bits.

What we simply do is break R_0 into 3-bit strings, and apply the function on each of the 3-bit strings, and then concatenate the results.

E = fs(100) concat fs(001) = 010 concat 100 = 010100

Step 3: L_1 = R_0, so L_1 = 100001

Secondly, R_1 = (L_0 XOR E) = 011110 XOR 010100 = 001010

Step 4: Concatenate L_1 and R_1

Ciphertext = 100001 concat 001010

Ciphertext = 100001001010