The Feistel Cipher

Feistel proposed , can approximate the ideal block cipher by utilizing the concept of a product cipher(use of a cipher that alternates substitutions and permutations) , 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.

In fact, this is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions.

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 possible transformations, rather than the 2 ! transformations available with the ideal block cipher.

Diffusion and Confusion

The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system. Shannon's concern was to thwart cryptanalysis based on statistical analysis.

If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or 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.

In **diffusion**, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext.

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.

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.

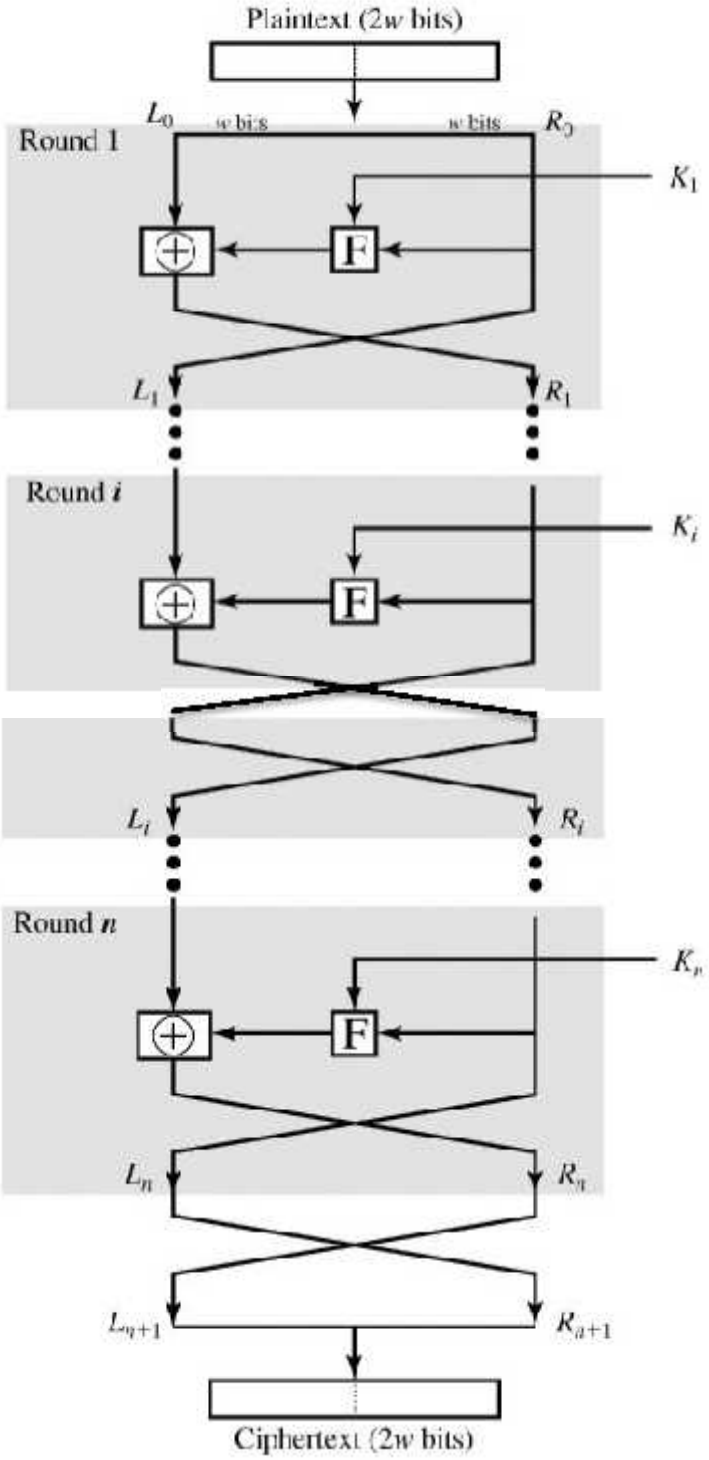
On the other hand, **confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.

This is achieved by the use of a complex substitution algorithm.

Feistel Cipher Structure(encryption):

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, L(left) and R (right) . 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 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.

Figure below depicts the structure proposed by Feistel.



Classical Feistel Network

All rounds have the same structure.

A- **Substitution** is performed on the left half of the data.

B- **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:

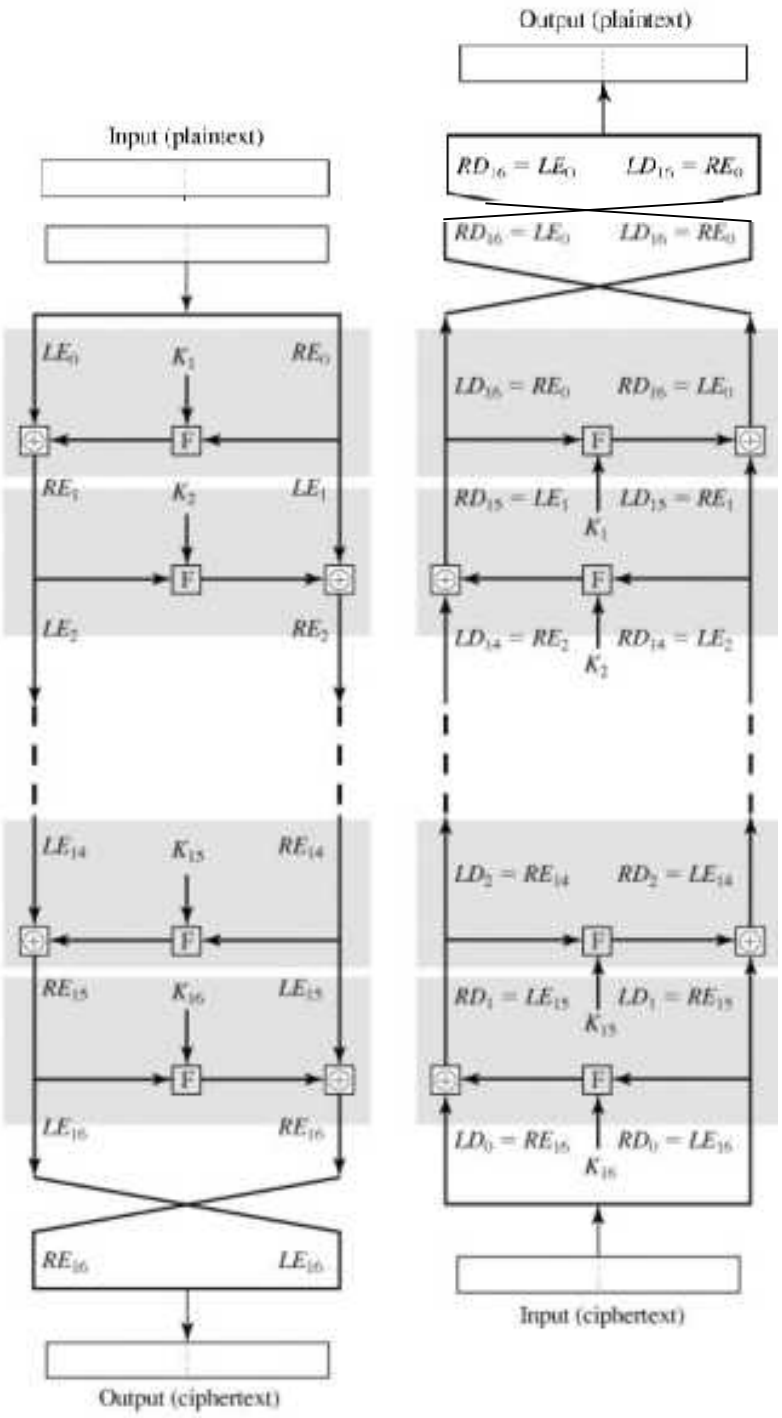
- **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm.
- **Key size:** Larger key size means greater security but may decrease encryption/decryption speed. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function:** Again, greater complexity generally means greater resistance to cryptanalysis.
- **Fast software encryption/decryption:** The speed of execution of the algorithm becomes a concern.
- **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze.

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 use the subkeys K in reverse order.

That is, use K_n in the first round, K_{n-1} in the second round, and so on until K is used in the last round.

This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption.



Encryption Decryption

Feistel Encryption and Decryption

First, consider the encryption process. We see that:

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \times F(RE_{15}, K_{16})$$

On the decryption side,

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \times F(RD_0, K_{16})$$

$$= RE_{16} \times F(RE_{15}, K_{16})$$

$$= [LE_{15} \times F(RE_{15}, K_{15})] \times F(RE_{15}, K_{16})$$

The XOR has the following properties:

$[A \times B] \times C = A \times [B \times C]$

$D \times D = 0$

$E \times 0 = E$

Thus, we have LD 1= RE15 and RD1 = LE15 .

Therefore, the output of the first round of the decryption process is LE15||RE15 , which is the 32-bit swap of the input to the sixteenth round of the encryption. in general terms. For the ith iteration of the encryption algorithm,

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \times F(RE_{i-1}, K_i)$$

Rearranging terms,

$$RE_{i-1} = LE_i$$

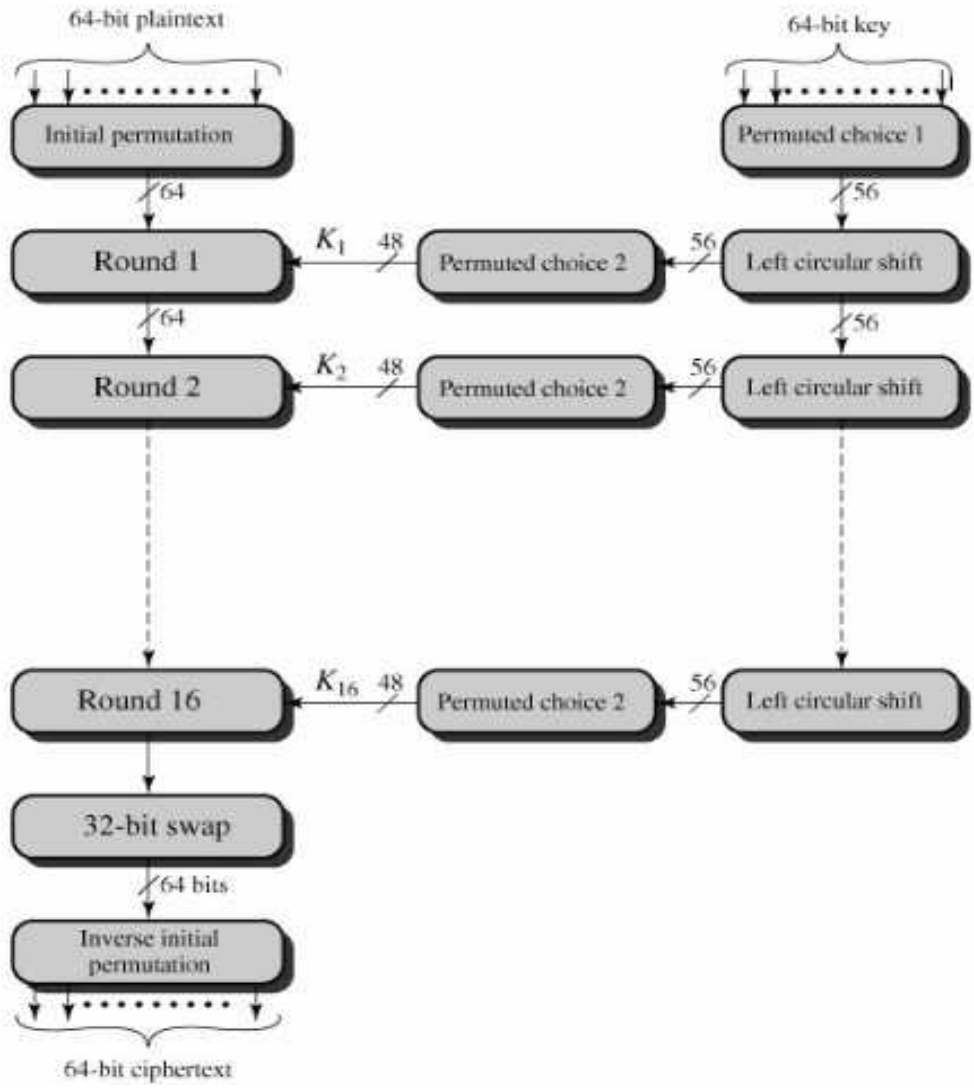
$$LE_{i-1} = RE_i \times F(RE_{i-1}, K_{i2} = RE_i \times F(LE_i, K_i)$$

The Data Encryption Standard:

For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

DES Encryption

The overall scheme for DES encryption is illustrated in Figure below. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.



General Depiction of DES Encryption Algorithm

Looking at the left-hand side of the figure up, we can see that the processing of the plaintext proceeds in three phases:

First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input. This is followed by **Second,** a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**.

Third: the preoutput is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.

The right-hand portion of Figure up shows the way in which :

First ,the 56-bit key is used. Initially, the key is passed through a permutation function.

Second, Then, for each of the 16 rounds, a *subkey* (*K*) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

Initial Permutation

The initial permutation and its inverse are defined by tables, as shown in Tables 1b and 2b, respectively. The tables are to be interpreted as follows.

- The input to a table consists of 64 bits numbered from 1 to 64.
- The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
- Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

Consider the following 64-bit input *M*:

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

Here *Mi* is a binary digit. Then the permutation $X = \mathrm{IP}(M)$ is as follows:

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

If we then take the inverse permutation $Y = IP^{-1}(X) = IP^{-1}(\mathrm{IP}(M))$, it can be seen that the original ordering of the bits is restored.