Introduction to Cryptography with Coding Theory
Trappe & Washington

4.2 A Simplified DES-Type Algorithm

The DES algorithm is rather unwieldy to use for examples, so in the present section we present an algorithm that has many of the same features, but is much smaller. Like DES, the present algorithm is a block cipher. Since the blocks are encrypted separately, we assume throughout the present discussion that the full message consists of only one block.

The message has 12 bits and is written in the form L_0R_0, where L_0 consists of the first 6 bits and R_0 consists of the last 6 bits. The key K has 9 bits. The ith round of the algorithm transforms an input $L_{i-1}R_{i-1}$ to the output L_iR_i using an 8-bit key K_i derived from K.

The main part of the encryption process is a function $f(R_{i-1}, K_i)$ that

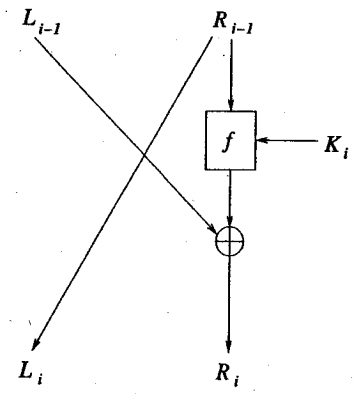

Figure 4.1: One Round of a Feistel System.

takes a 6-bit input R_{i-1} and an 8-bit input K_i and produces a 6-bit output. This will be described later.

The output for the ith round is defined as follows:

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

where $\oplus$ denotes XOR, namely bit-by-bit addition mod 2. This is depicted in Figure 4.1.

This operation is performed for a certain number of rounds, say n, and produces the ciphertext L_nR_n.

How do we decrypt? Start with L_nR_n and switch left and right to obtain R_nL_n. (*Note: This switch is built into the DES encryption algorithm, so it is not needed when decrypting DES.*) Now use the same procedure as before, but with the keys K_i used in reverse order $K_n, \ldots, K_1$. Let's see how this works. The first step takes R_nL_n and gives the output

$$[L_n] \quad [R_n \oplus f(L_n, K_n)].$$

We know from the encryption procedure that $L_n = R_{n-1}$ and $R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$. Therefore,

$$\begin{aligned}[L_n] \quad [R_n \oplus f(L_n, K_n)] &= [R_{n-1}] \quad [L_{n-1} \oplus f(R_{n-1}, K_n) \oplus f(L_n, K_n)] \\ &= [R_{n-1}] \quad [L_{n-1}].\end{aligned}$$

The last equality again uses $L_n = R_{n-1}$, so that $f(R_{n-1}, K_n) \oplus f(L_n, K_n)$ is 0. Similarly, the second step of decryption sends $R_{n-1}L_{n-1}$ to $R_{n-2}L_{n-2}$.

Continuing, we see that the decryption process leads us back to R_0L_0. Switching the left and right halves, we obtain the original plaintext L_0R_0, as desired.

Note that the decryption process is essentially the same as the encryption process. We simply need to switch left and right and use the keys K_i in reverse order. Therefore both the sender and receiver use a common key and they can use identical machines (though the receiver needs to reverse left and right inputs).

So far, we have said nothing about the function f. In fact, any f would work in the above procedures. But some choices of f yield much better security than others. The type of f used in DES is similar to that which we describe next. It is built up from a few components.

The first function is an expander. It takes an input of 6 bits and outputs 8 bits. The one we use is given in Figure 4.2.

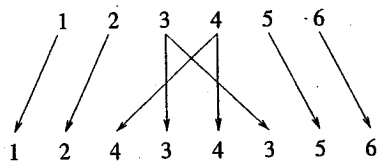

Figure 4.2: The Expander Function.

This means that the first input bit yields the first output bit, the third input bit yields both the fourth and the sixth output bits, etc. For example, 011001 is expanded to 01010101.

The main components are called S-boxes. We use two:

$$S_1 \quad \begin{bmatrix} 101 & 010 & 001 & 110 & 011 & 100 & 111 & 000 \\ 001 & 100 & 110 & 010 & 000 & 111 & 101 & 011 \end{bmatrix}$$

$$S_2 \quad \begin{bmatrix} 100 & 000 & 110 & 101 & 111 & 001 & 011 & 010 \\ 101 & 011 & 000 & 111 & 110 & 010 & 001 & 100 \end{bmatrix}.$$

The input for an S-box has 4 bits. The first bit specifies which row will be used: 0 for the first row, 1 for the second. The other 3 bits represent a binary number that specifies the column: 000 for the first column, 001 for the second, ..., 111 for the last column. The output for the S-box consists of the three bits in the specified location. For example, an input of 1010 for S_1 means we look at the second row, third column, which yields the output 110.

The key K consists of 9 bits. The key K_i for the ith round of encryption is obtained by using 8 bits of K, starting with the ith bit. For example, if

$K = 010011001$, then $K_4 = 01100101$ (after 5 bits, we reached the end of K, so the last 2 bits were obtained from the beginning of K).

We can now describe $f(R_{i-1}, K_i)$. The input R_{i-1} consists of 6 bits. The expander function is used to expand it to 8 bits. The result is XORed with K_i to produce another 8-bit number. The first 4 bits are sent to S_1, and the last 4 bits are sent to S_2. Each S-box outputs 3 bits, which are concatenated to form a 6-bit number. This is $f(R_{i-1}, K_i)$. We present this in Figure 4.3.

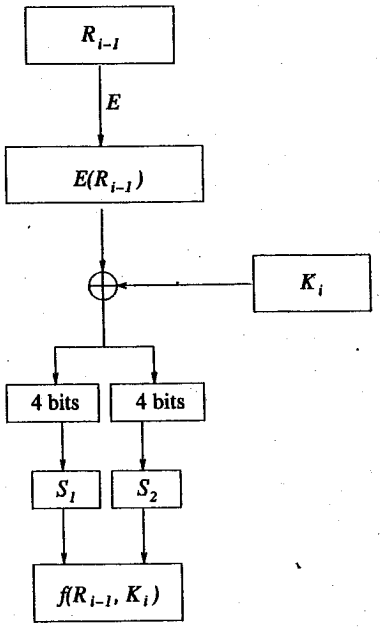

Figure 4.3: The Function $f(R_{i-1}, K_i)$.

For example, suppose $R_{i-1} = 100110$ and $K_i = 01100101$. We have

$$E(100110) \oplus K_i = 10101010 \oplus 01100101 = 11001111.$$

The first 4 bits are sent to S_1 and the last 4 bits are sent to S_2. The second row, fifth column of S_1 contains 000. The second row, last column of S_2 contains 100. Putting these outputs one after the other yields $f(R_{i-1}, K_i) = 000100$.

We can now describe what happens in one round. Suppose the input is

$$L_{i-1}R_{i-1} = 011100100110$$

and $K_i = 01100101$, as previously. This means that $R_{i-1} = 100110$, as in the example just discussed. Therefore $f(R_{i-1}, K_i) = 000100$. This is XORed with $L_{i-1} = 011100$ to yield $R_i = 011000$. Since $L_i = R_{i-1}$, we obtain

$$L_iR_i = 100110011000.$$

The output becomes the input for the next round.