Hacettepe University Department of Computer Engineering
BBM465 Information Security Laboratory
Experiment 1

Subject	:	Implementing a Feistel Cipher
Language	:	Java
Due Date	:	04/11/2020 - 23:59

1 Experiment

The aim of this project is to develop a Feistel Cipher and understand the internals of Feistel ciphers e.g. DES, IDEA, RC4/5. You will learn multi round ciphers that enable to encrypt/decrypt data with the right key. You will also learn to implement ECB, CBC and OFB encryption modes for your Feistel Cipher.

A Feistel Cipher has the below format as discussed in the class:

- **Encryption**

  $L_1=R_0 \qquad R_1=L_0 \oplus f_0(R_0,K_0)$

  $L_2=R_1 \qquad R_2=L_1 \oplus f_1(R_1,K_1)$

  $\ldots$

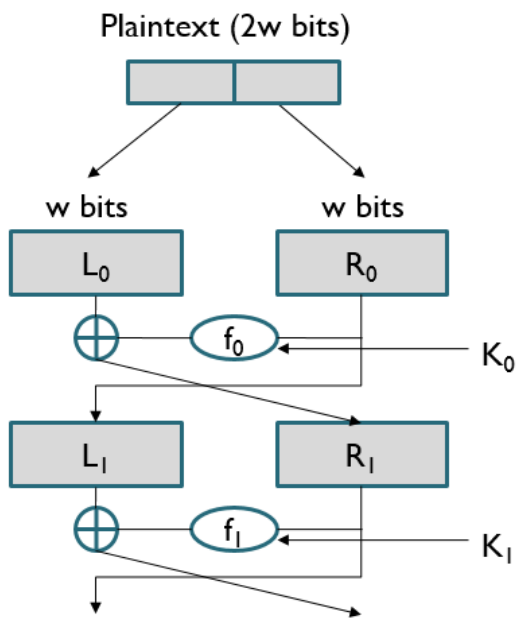
  $L_d=R_{d-1} \qquad R_d=L_{d-1} \oplus f_{d-1}(R_{d-1},K_{d-1})$

- **Decryption**

  $R_{d-1}=L_d \qquad L_{d-1}=R_d \oplus f_{d-1}(L_d,K_{d-1})$

  $\ldots$

  $R_0=L_1 \qquad L_0=R_1 \oplus f_0(L_1,K_0)$



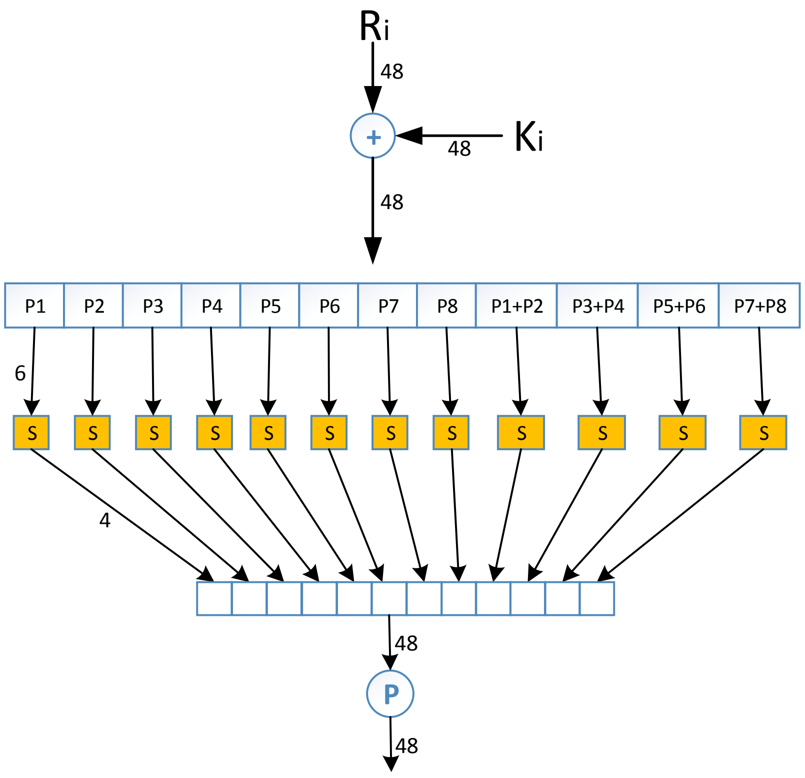
A Feistel Cipher has d rounds of encryption, where in i^{th} round, a one-way scramble function f is applied to right half of the data R_i with subkey K_i.

In this project, the main parameters of your Feistel Cipher will be as follows:

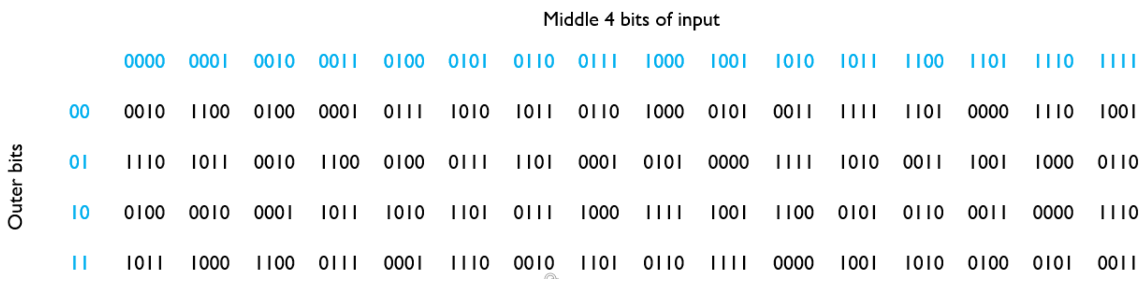
- 10 rounds of encryption/decryption
- 96 bit block size
- 96 bit key size

Scramble Function:

In our project, we will assume that scramble functions $f_0, f_1, f_2 \ldots f_{d-1}$ applied on each round are <u>all same</u>. In other words, you will have only one scramble function, which is defined as follows



In the above scramble function, R_i is XORed with K_i first and the result is divided into 8 pieces of 6-bit parts (*P1, P2, P3, ... P8*). Then, P1 and P2 are XORed and saved as the 9th part, P3 and P4 are XORed and saved as the 10th part and so on. As a result of these operations, you will have 12 pieces of 6-bit parts. Then, each 6-bit part is sent to a substitution box like in the DES algorithm. You are expected to use below substitution box:



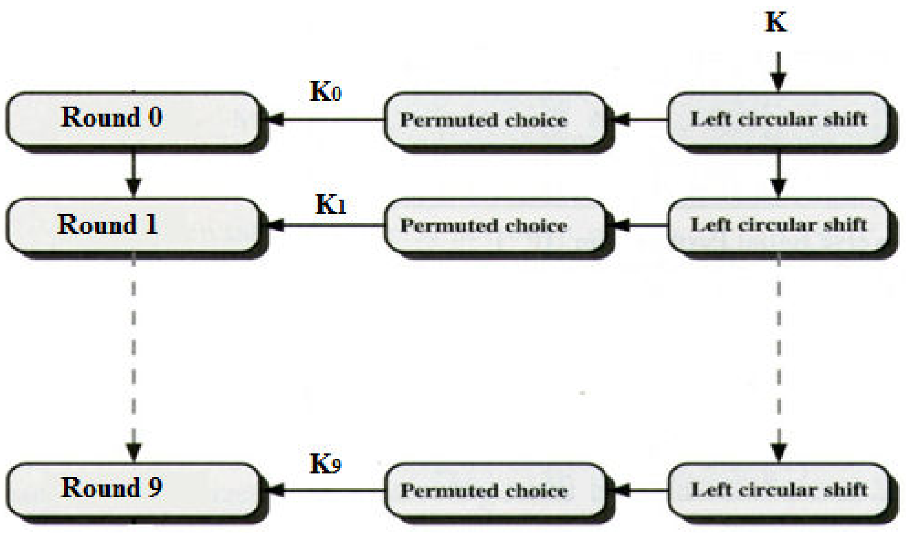
Outer bits	Middle 4 bits of input															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

The substitution box works in similar way to DES boxes. Outer bits of the input are used as the row number, inner bits are used as the column number. The results of 12 substitution box operations will be combined as a 48 bit output and a permutation function will be applied on this output. The permutation function swaps each even digit with the previous odd digit. Assuming B_i represents ith bit of the output, the permutation function will do following operation for all bits of the result:

$$\begin{aligned} &\text{Temp} = B_{2i} \\ &B_{2i} = B_{2i-1} \\ &B_{2i-1} = \text{Temp} \end{aligned}$$

Subkey Generation:

In each round, you will do apply a left circular shift on the key and a permuted choice on the key as shown in below figure:



In the even numbered rounds, the permuted choice will output even digits of the key to the related round. In other words, in rounds 0, 2, 4, 6, and 8, the permuted choice function will output 0, 2, 4, 6, 8, …. 94th bits, which is 48 bits in total. In the other (1,3,5,7,9) rounds, the permuted choice function will output odd digits, which are 1, 3, 5, 7, 9, …. 95th bits.

Encryption modes:

You will implement ECB, CBC, and OFB modes for your encryption algorithm. In case, if CBC or OFB modes are selected, your program should use a vector filled with 1s as the Initialization Vector.

2 Usage and testing

<u>**Your program must be named as "BBMcrypt"**</u> and should run from the command line with the following parameter format:

```
BBMcrypt enc|dec -K key -I input -O output -M mode
```

- *enc|dec* specifies the action, which can be either encryption or decryption
- *-K key* specifies the file name that contains encryption/decryption key, which is encoded in base64 encoding
- *-I input* specifies the input file name, which can be either plaintext or ciphertext
- *-O output* specifies the output file name, which can be either plaintext or ciphertext
- *-M mode* specifies the encryption mode, which can be ECB, CBC or OFB.

Your program will be tested on different sized files using the above command line format. <u>**Thus, it is important to obey this command line format or otherwise, you will lose points.**</u>

We will not test wrong parameters of users, such as missing input file name, key file name. However, -K, -I, -O, and -M parameters might be given in different orders on the command line (The first parameter must be always *enc|dec*).

3 Notes

1. You can ask questions about the experiment via Piazza group: piazza.com/hacettepe.edu.tr/fall2020/bbm465
2. Late submission will not be accepted!
3. You must compile and run your program on Ubuntu operating system before submission.
4. You are going to submit your experiment to online submission system:

   www.submit.cs.hacettepe.edu.tr

   The submission format is given below:

```
<Group id>.zip
  -java/
  --*.java
```