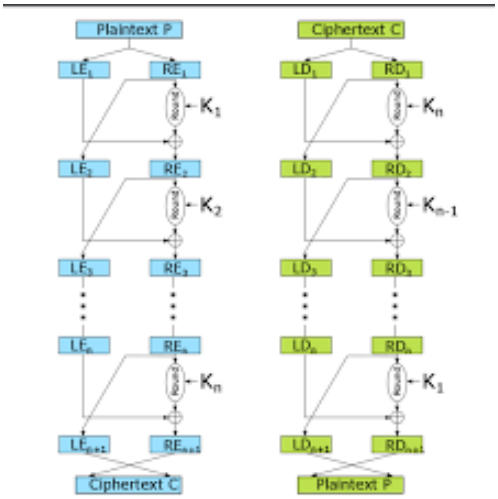
Name: Anuj Singh
Roll no: PG18 PRN: 1032201083
Batch : G2

Lab 2: Implement simple DES (fiestel block) symmetric key algorithm using python or java or C++

Feistel Cipher model is a structure or a design used to develop many block ciphers such as DES. Feistel cipher may have invertible, non-invertible and self invertible components in its design. Same encryption as well as decryption algorithm is used. A separate key is used for each round. However same round keys are used for encryption as well as decryption.

Feistel cipher algorithm

- Create a list of all the Plain Text characters.
- Convert the Plain Text to Ascii and then 8-bit binary format.
- Divide the binary Plain Text string into two halves: left half (L1)and right half (R1)
- Generate a random binary keys (K1 and K2) of length equal to the half the length of the Plain Text for the two rounds.



The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext, each round consisting of a “substitution” step followed by a permutation step

- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.

- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function 'f' that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a subkey) is derived from the encryption key. This means that each round uses a different key, although all these subkeys are related to the original key.
- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a 'round'. The number of rounds are specified by the algorithm design.
- Once the last round is completed then the two sub blocks, 'R' and 'L' are concatenated in this order to form the ciphertext block.

Decryption Process

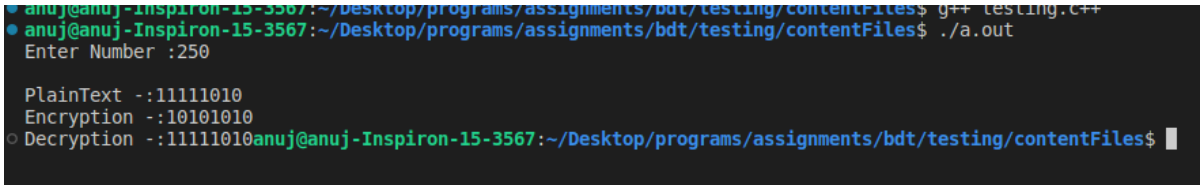
The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.
The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.
The final swapping of 'L' and 'R' in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.

Code-:

```
#include <iostream>
#include <vector>
#include <bits/stdc++.h>
using namespace std;

vector<int> XOR(vector<int> num, vector<int> num2)
{
   vector<int> result;
   for (int i = 0; i < num.size()/2; i++)
   {
      result.push_back(num.at(i)^(num2.at(i) ^ num.at((num.size()/2)+i)));
   }
   return result;
}
```

```
vector<int> binary(int num)
{
   vector<int> result;
   for (int i = 0; num > 0; i++)
   {
      result.push_back(num % 2);
      num = num / 2;
   }
   std::reverse(result.begin(), result.end());
   return result;
}
int main()
{
   //Encryption

   int num;
   cout << "Enter Number :";
   cin >> num;
   vector<int> plainText = binary(num), result, key{1, 1, 1, 1},temp ,decrypt;
   cout << "\nPlainText -:";
   for (int i = 0; i < plainText.size(); i++)
   {
      cout << plainText.at(i);
   }
   for (int i = plainText.size() / 2; i < plainText.size(); i++)
   {
      result.push_back(plainText.at(i));
   }
   temp = XOR(plainText, key);
   for(int i=0;i<temp.size();i++){
      result.push_back(temp.at(i));
   }

   cout << "\nEncryption -:";
   for (int i = 0; i < result.size(); i++)
   {
      cout << result.at(i);
   }


   //Decryption
```

```
    decrypt=XOR(result,key);
    for(int i=0;i<result.size()/2;i++){
        decrypt.push_back(result.at(i));
    }

    cout << "\nDecryption -:";
    for (int i = 0; i < decrypt.size(); i++)
    {
        cout << decrypt.at(i);
    }
    return 0;
}
```



FAQs

LAB- 2

FAQ's

Q1 State different steps in DES

Ans 1. Initial Permutation
The IP step only performed once at the start. The step shuffles the bits of plaintext according to permutation matrix.

2. ~~Example~~ Expansion Permutation
The 32-bit plaintext is expanded into a 48-bits. for operation with 48 bit key.

3. Substitution Box
The next converts 6 bits to a position on a box matrix & replaced by its value to return a 32 bit value.

4. Straight permutation.
The Sbox output is permuted with a set matrix

Q2 State features of DES
1. Data Encryption Standard is a symmetric key block cipher
2. It uses feistal Cipher
3. It is a 16 rounds feistal structure
4. Block size is 64 bits.
5. key length : 64 bit : Effective key length : 56 bits

Q3 Explain different operation modes

Ans The different operational modes are:

1.) Electronic Codeblock → Each 64 bit blocks is encrypted independently

2.) Cipher Block Chaining → Each 64 bit blocks depended on the previous one & uses initialization vector.

3.) Cipher Feedback: A preceding cipher text is used for encryption with plaintext to make next ciphertext.

4.) Output Feedback: Like CFB but input to encryption algorithm is from preceding DES

5.) Counter: Plaintext is Xor'd with a encrypted counter which is encrypted for subsequent block.