Information and Network Security
2CSDE54
Practical 4

21BCE020

Perform encryption and decryption using Feistel Cipher

Code:

```
#include <iostream>
#include <bitset>
#include <string>
#include <fstream>
#include <sstream>
#include <utility>
#include <functional>
using namespace std;

string ascii_to_bin(string ip) {
    string binstr;
    for (char c : ip) {
        binstr += bitset<8>(c).to_string();
        binstr = binstr.substr(0, 16); //remove any trailing
zeros
    }
    return binstr;
}

string bin_to_str(string resultant_bin){
    string str = "";
    stringstream sstream(resultant_bin);
    while (sstream.good()) { //more data to read
        bitset<8> bits;
        sstream >> bits;
        str += static_cast<char>(bits.to_ulong()); //explicit
type conversion to unsigned long
    }
    return str;
}

pair<string, string> splitting(string binstr) {
```

```
    string l = "";
    string r = "";
    pair<string, string> p;
    int len = binstr.length();
    l = binstr.substr(0, len / 2);
    r = binstr.substr(len / 2);
    p.first = l;
    p.second = r;
    return p;
}

string XOR(string& str1, string& str2) {
    string result;
    for (size_t i = 0; i < str1.length(); ++i) { //size_t to
avoid negative size
        if (str1[i] == str2[i]) {
            result += '0';
        } else {
            result += '1';
        }
    }
    return result;
}


string encrypt(string ip, string key1, string key2) {
    string binstr = ascii_to_bin(ip);
    string k1_bin = ascii_to_bin(key1);
    string k2_bin = ascii_to_bin(key2);
    pair<string, string> p = splitting(binstr);
    string l = p.first;
    string r = p.second;
    // First round of encryption
    /* f1 = XOR(r1, k1)
    r2 = XOR(f1, l1)
    l2 = r1*/
    string f1 = XOR(r, k1_bin);
    string l2 = r;
    string r2 = XOR(f1, l);

    // Second round of encryption
    /* f2 = XOR(r2, k2)
```

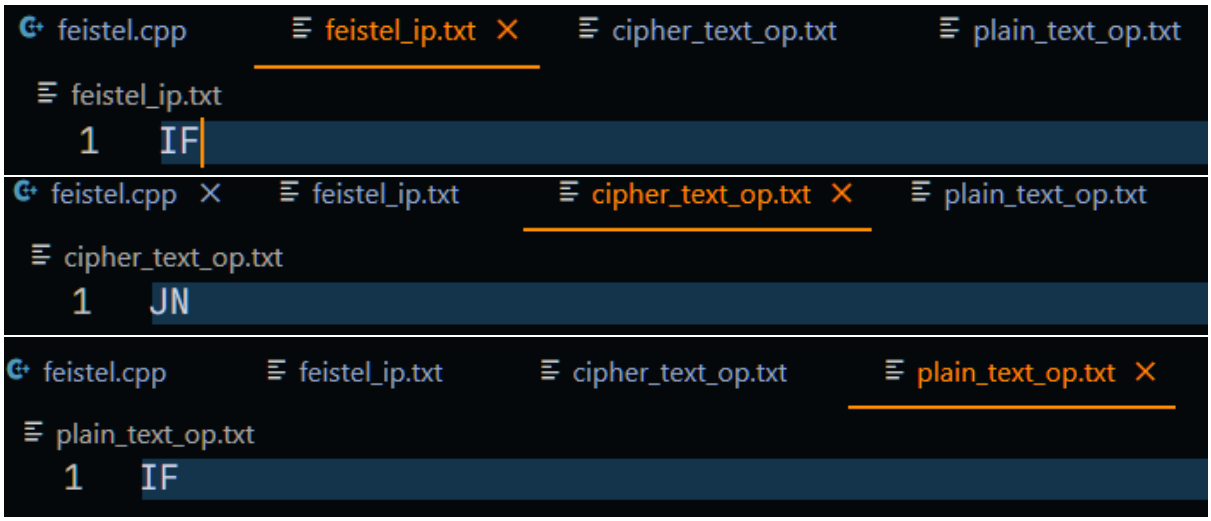
```
    r3 = XOR(f2, l2)
    l3 = r2*/
    string f2 = XOR(r2, k2_bin);
    string l3 = r2;
    string r3 = XOR(f2, l2);


    string result = r3 + l3;

    // Resultant binary string after encryption
    // string result = bitset<8>(r3).to_string() +
bitset<8>(l3).to_string();
    // Cipher text after binary -> string
    string cipher_text = bin_to_str(result);
    return cipher_text;
}

string decrypt(string CT, string key1, string key2) {
    string binstr = ascii_to_bin(CT);
    // binstr = binstr.substr(0, 16);
    string k1_bin = ascii_to_bin(key1);
    string k2_bin = ascii_to_bin(key2);
    pair<string, string> p = splitting(binstr);
    string l = p.first;
    string r = p.second;
    // First round of encryption
    /* f1 = XOR(r1, k1)
    r2 = XOR(f1, l1)
    l2 = r1*/
    // int f1 = stoi(r, nullptr, 2) ^ stoi(k2_bin, nullptr,
2);
    // int r2 = f1 ^ stoi(l, nullptr, 2);
    // int l2 = stoi(r, nullptr, 2);
    string f1 = XOR(r, k2_bin);
    string l2 = r;
    string r2 = XOR(f1, l);

    // Second round of encryption
    /* f2 = XOR(r2, k2)
    r3 = XOR(f2, l2)
    l3 = r2*/
    // int f2 = r2 ^ stoi(k1_bin, nullptr, 2);
```

```
    // int r3 = f2 ^ l2;
    // int l3 = r2;
    string f2 = XOR(r2, k1_bin);
    string l3 = r2;
    string r3 = XOR(f2, l2);
    string result = r3 + l3;
    // Intermediate binary string after decryption
    // string inter_result = bitset<8>(r3).to_string() +
bitset<8>(l3).to_string();
    // Split resultant binary string
    // pair<string, string> final_p = splitting(inter_result);
    // string new_l = final_p.first;
    // string new_r = final_p.second;
    // string result = bitset<8>(new_l).to_string() +
bitset<8>(new_r).to_string();
    string plain_text = bin_to_str(result);
    return plain_text;

}


int main() {
    // string ip = "IF";
    string key1 = "A";
    string key2 = "B";
    // cout<<"Cipher Text: "<<CT;
    // cout<<"\nPlain Text: "<<decrypt("JN", key1, key2);
    ifstream ipf("feistel_ip.txt");
    string ip;
    if (ipf.is_open()) {
        ipf >> ip;
        ipf.close();
    }
    else {
        cout<<"Error"<<endl;
    }

    string CT;
    CT = encrypt(ip, key1, key2);
    ofstream ctf("cipher_text_op.txt");
    if (ctf.is_open()) {
        ctf << CT;
```

```
        ctf.close();
    }
    else {
        cout<<"Error"<<endl;
    }

    string PT = decrypt(CT, key1, key2);

    ofstream decf("plain_text_op.txt");
    if (decf.is_open()) {
        decf << PT;
        decf.close();
    }
    else {
        cout<<"Error"<<endl;
        return 1;
    }

    cout << "Cipher text: " << CT << endl;
    cout << "Plain text: " << PT << endl;
    return 0;
}
```

O/P:

Text file output



Terminal output

```
PS D:\College\NIRMA\SEM_VI\DE\INS\Practicals\P4> g++ feistel.cpp
PS D:\College\NIRMA\SEM_VI\DE\INS\Practicals\P4> ./a
Cipher text: JN
Plain text: IF
```