**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;

    }

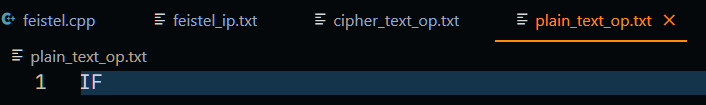
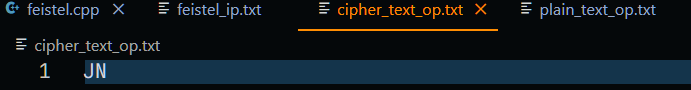
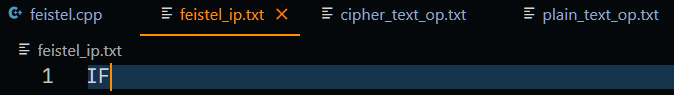
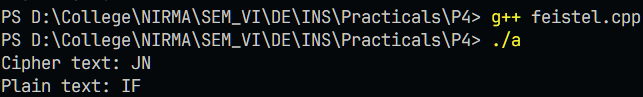
    cout << *"Cipher text: "* << CT << endl;

    cout << *"Plain text: "* << PT << endl;

    return 0;

}

**O/P:  
Text file output**

**Terminal output**