Name: Karuna Bajirao Randive.

Prn: 2020BTECS00024

Batch: B2

Cryptography & Network Security Lab

Assignment No. 2

Theory:

1. Rail Fence Transposition Cipher Technique

The Rail Fence Transposition Cipher, also known as the Zigzag Cipher, is a simple columnar transposition cipher technique.

It involves arranging the plaintext characters in a zigzag pattern across multiple rows, known as "rails," and then reading them off row by row to create the encrypted message.

While this cipher is easy to understand and implement, it lacks strong security and is mainly used for educational purposes or simple puzzles.

Encryption:

- Choose the number of rails (rows) for the zigzag pattern.
- Write the message diagonally across the rails, moving up and down.
- Read the characters row by row to form the encrypted message.

Decryption:

- Create the zigzag pattern with the chosen number of rails.
- Leave blank spaces in the pattern for characters to be placed.
- Fill in the blanks with the encrypted characters, row by row.
- Read the characters diagonally to retrieve the original message.

Advantages:

- Easy to understand and implement.
- Provides basic encryption and breaks up character repetition.

Disadvantages:

- Not secure against modern cryptanalysis.
- Security depends on the number of rails, making it less practical for strong encryption.

Transposition Cipher:

1. Railfence Transposition

```
#include <bits/stdc++.h>
```

```
using namespace std;

string encryptRailFence(string text, int key){
    char rail[key][(text.length())];

    for(int i=0; i < key; i++)
        for (int j = 0; j < text.length(); j++)
            rail[i][j] = '\n';

    bool dir_down = false;
    int row = 0, col = 0;

    for (int i=0; i < text.length(); i++){
        if (row == 0 || row == key-1)
            dir_down = !dir_down;

        rail[row][col++] = text[i];
        dir_down?row++ : row--;
    }

    string result;
    for (int i=0; i < key; i++)
        for (int j=0; j < text.length(); j++)
            if (rail[i][j]!='\n')
                result.push_back(rail[i][j]);

    return result;
}

string decryptRailFence(string cipher, int key){
    char rail[key][cipher.length()];

    for (int i=0; i < key; i++)
        for (int j=0; j < cipher.length(); j++)
            rail[i][j] = '\n';

    bool dir_down;
    int row = 0, col = 0;


    for (int i=0; i < cipher.length(); i++){
        if (row == 0)
```

```
            dir_down = true;
        if (row == key-1)
            dir_down = false;

        rail[row][col++] = '*';
        dir_down?row++ : row--;
    }

    int index = 0;
    for (int i=0; i<key; i++)
        for (int j=0; j<cipher.length(); j++)
            if (rail[i][j] == '*' && index<cipher.length())
                rail[i][j] = cipher[index++];

    string result;
    row = 0, col = 0;
    for (int i=0; i< cipher.length(); i++){
        if (row == 0)
            dir_down = true;

        if (row == key-1)
            dir_down = false;

        if (rail[row][col] != '*')
            result.push_back(rail[row][col++]);

        dir_down?row++: row--;
    }
    return result;
}

int main(){
    string s;
    cout<<"Enter 1 for encryption and 2 for decryption: ";
    int x;
    int key;
    cin>>x;

    switch (x){
    case 1:
        cout<<"Plain text : ";
        cin>>s;
```

```
        cout<<"Key : ";
        cin>>key;
        cout << encryptRailFence(s, key) << endl;
        break;
    case 2:
        cout<<"Cipher text : ";
        cin>>s;

        cout<<"Key : ";
        cin>>key;
        cout << decryptRailFence(s, key) << endl;
        break;
    default:
        break;
    }
    return 0;
}
```

Output:

Encryption:

```
PS C:\Users\Shree Ram Samarth\Documents\CNS\Assign02> cd
nce.cpp -o railFence -lbgi -lgdi32 -lcomdlg32 -luuid -lo
Enter 1 for encryption and 2 for decryption: 2
Cipher text : GMiodonnorg
Key : 3
GoodMorning
PS C:\Users\Shree Ram Samarth\Documents\CNS\Assign02>
```

Decryption:

2. Row/Columnar Transposition

Code:

```
#include <bits/stdc++.h>
using namespace std;
string key = "HACK";
int keynumlist[4];

void setkeynumlist()
```

```
{
    string a = "ABCDEFGHJKLMNOPQRSTUVWXYZ";
    int id = 0;
    for (int i = 0; i < a.length(); i++)
    {
        for (int j = 0; j < key.length(); j++)
        {
            if (a[i] == key[j])
            {
                keynumlist[j] = id;
                id++;
            }
        }
    }

}
void encrypt(string text)
{
    int col = key.length();
    int row = text.length() / key.length();
    if (text.length() % key.length())
        row += 1;

    // cout<<row<<" "<<col<<endl;
    char matrix[row][col];
    int ind = 0;
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if (ind < text.length())
            {
                matrix[i][j] = text[ind++];
            }
            else
            {
                matrix[i][j] = '_';
            }
        }
    }

    string res = "";
    for (int i = 0; i < key.length(); i++)
    {
        int ii = keynumlist[i];
        for (int j = 0; j < row; j++)
        {
            if (matrix[j][ii] != '_')
```

```
                res += matrix[j][ii];
        }
    }

    cout << "Encrypted text: " << res << endl;
}
void decrypt(string text)
{
    int row = text.length() / key.length();
    if (text.length() % key.length())
    {
        row++;
    }
    int col = key.length();
    int rem = key.length() - (text.length() % key.length());

    // cout<<row<<" "<<col<<endl;

    int ind = 0;
    char matrix[row][col];
    for (int c = col - 1; c > 0, rem != 0; c--)
    {
        matrix[row - 1][c] = '_';
        rem--;
    }

    for (int i = 0; i < col; i++)
    {
        int ii = keynumlist[i];
        for (int j = 0; j < row; j++)
        {
            if (ind < text.length() && matrix[j][ii] != '_')
            {
                matrix[j][ii] = text[ind++];
            }
        }
    }

    string res = "";
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if (matrix[i][j] != '_')
                res += matrix[i][j];
        }
    }
```

```
    cout << "Decrypt text: " << res << endl;
}
int main()
{
    int key, opt, p = 1;
    string text;
    setkeynumlist();

    while (p)
    {
        cout << "Enter 1.Encrypt 2.Decrypt 3.Exit" << endl;
        cin >> opt;
        if (opt == 3)
            break;
        cin.ignore();
        cout << "Enter text" << endl;
        getline(cin, text);
        cin.ignore();

        switch (opt)
        {
        case 1:
            encrypt(text);
            break;
        case 2:
            decrypt(text);
            break;
        default:
            p = 0;
            break;
        }
    }
}
```

Output:

```
PS C:\Users\Shree Ram Samarth\Documents\CNS\Assign02> g++ rowColumnar.cpp
PS C:\Users\Shree Ram Samarth\Documents\CNS\Assign02> ./a.exe
Enter 1.Encrypt 2.Decrypt 3.Exit
1
Enter text
HiKaruna
pass
Encrypted text: KnHriuaa
```