Name: Nikita Shivchandra Khot

PRN No: 2020BTECS00041

Assignment No 2

Study and implementation of Transposition Cipher Technique

1. Row Transposition technique

Aim: To Study and implementation of Row Transposition Cipher Technique

Theory: A simple example for a transposition cipher is row transposition cipher where each character in the plain text is written horizontally with specified alphabet width. The cipher is written vertically, which creates an entirely different cipher text.

Code:

```
// Assignment 2 b)
#include<bits/stdc++.h>
using namespace std;

string encryptMessage(string msg,string const key,map<int,int> keyMap)
{
    int row,col,j;
    string cipher = "";

    col = key.length();
    row = msg.length()/col;

    if (msg.length() % col)
        row += 1;

    char matrix[row][col];
    for (int i=0,k=0; i < row; i++)
    {
        for (int j=0; j<col; )
        {
            if(msg[k] == '\0')
            {
                matrix[i][j] = '_';
                j++;
            }

            if( isalpha(msg[k]) || msg[k]==' ')
            {
                /* Adding only space and alphabet into matrix*/
```

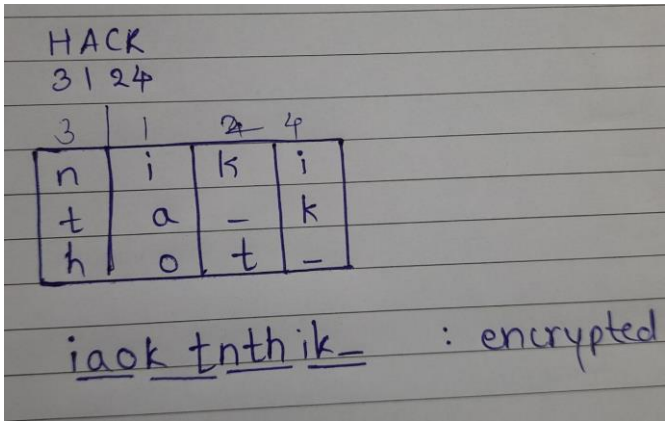
```
                matrix[i][j] = msg[k];
                j++;
            }
            k++;
        }
    }

    for (map<int,int>::iterator ii = keyMap.begin(); ii!=keyMap.end();
++ii)
    {
        j=ii->second;

        // getting cipher text from matrix column wise using permuted
key
        for (int i=0; i<row; i++)
        {
            if( isalpha(matrix[i][j]) || matrix[i][j]==' ' ||
matrix[i][j]=='_')
                cipher += matrix[i][j];
        }
    }

    return cipher;
}

// Decryption
string decryptMessage(string cipher,string const key,map<int,int>
keyMap)
{
    int col = key.length();

    int row = cipher.length()/col;
    char cipherMat[row][col];

    for (int j=0,k=0; j<col; j++)
        for (int i=0; i<row; i++)
            cipherMat[i][j] = cipher[k++];

    int index = 0;
    for( map<int,int>::iterator ii=keyMap.begin(); ii!=keyMap.end();
++ii)
        ii->second = index++;

    char decCipher[row][col];
    map<int,int>::iterator ii=keyMap.begin();
    int k = 0;
```

```
    for (int l=0,j; key[l]!='\0'; k++)
    {
        j = keyMap[key[l++]];
        for (int i=0; i<row; i++)
        {
            decCipher[i][k]=cipherMat[i][j];
        }
    }

    string msg = "";
    for (int i=0; i<row; i++)
    {
        for(int j=0; j<col; j++)
        {
            if(decCipher[i][j] != '_')
                msg += decCipher[i][j];
        }
    }
    return msg;
}

int main()
{
    string msg = "nikita khot";
    string key = "hack";
    map<int,int> keyMap;

    for(int i=0; i < key.length(); i++)
        keyMap[key[i]] = i;

    string cipher = encryptMessage(msg, key, keyMap);
    cout << "Encrypted Message: " << cipher << endl;
    cout << "Decrypted Message: " << decryptMessage(cipher, key,
keyMap) << endl;

    return 0;
}
```

Output:

```
Encrypted Message: iaok tnthik_
Decrypted Message: nikita khot
```

Example:



Limitations: It is not very good at hiding the most common letters in a language, making it easier for someone to figure out the message. If someone knows some parts of the original message, they can use that to break the code more easily. Also, dealing with the secret way you shuffle the message can be tricky. For long messages, it's not too hard for someone to try lots of different shuffles until they find the right one. Plus, it doesn't have all the advanced features needed to protect important information in today's digital world, like making sure the message is real and hasn't been changed. Overall, it's not a strong choice for keeping things safe in the modern age.

2. Rail Fence Cipher technique

Aim: To study and implement Rail Fence Cipher Technique

Theory: It is a simple encryption technique that works by writing a message in a zigzag pattern on a set number of rails (horizontal lines), and then reading the characters in a specific order to create the ciphertext. It's named after the shape of a fence made by drawing the zigzag lines.

Code:

```
#include <bits/stdc++.h>
using namespace std;
string encryptRailFence(string text, int key)
{
    char rail[key][(text.length())];
    int i, j;
    for (i=0; i < key; i++)
    {
        for (j = 0; j < text.length(); j++)
            rail[i][j] = '\n';
    }
    bool dir_down = false;
```

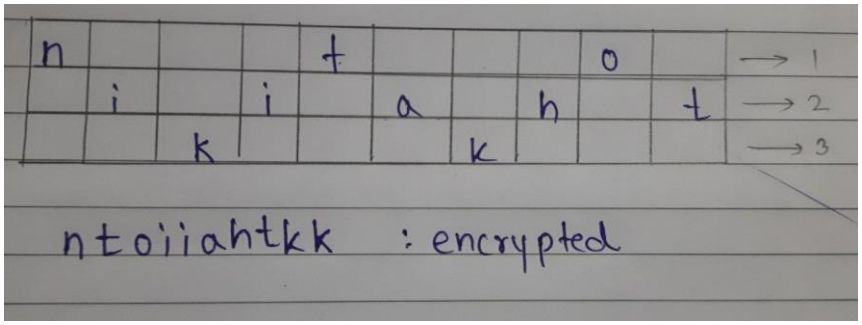
```
    int row = 0, col = 0;
    for (i=0; i < text.length(); i++)
    {
        if (row == 0 || row == key-1)
            dir_down = !dir_down;
        rail[row][col++] = text[i];
        dir_down ? row++ : row--;
    }
    string result;
    for (i=0; i < key; i++)
        for (j=0; j < text.length(); 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()];
    int i, j;
    for (i=0; i < key; i++)
        for (j=0; j < cipher.length(); j++)
            rail[i][j] = '\n';
    bool dir_down;
    int row = 0, col = 0;
    for (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 (i=0; i<key; i++)
        for (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 (i=0; i< cipher.length(); i++)
    {
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;
```

```
        result.push_back(rail[row][col++]);
        dir_down?row++: row--;
    }
    return result;
}
int main()
{
    string plain_text = "nikitakhot";
    int key = 3;
    string cipher_text, decrypted_text;
    cipher_text = encryptRailFence(plain_text, key);
    cout << "(Encrypted) Cipher Text: " << cipher_text << endl;
    decrypted_text = decryptRailFence(cipher_text, key);
    cout << "(Decrypted) Plain Test: " << decrypted_text << endl;
    return 0;
}
```

Output:

```
(Encrypted) Cipher Text: ntoiiahtkk
(Decrypted) Plain Test: nikitakhot
```

Example:



Limitations: It is easy to understand and use, but it's not very secure because it doesn't change the frequency distribution of letters in the message. It can be easily deciphered, especially for longer messages, through methods like trial and error. As a result, it's mainly used for educational purposes or in situations where only minimal security is required.