Assignment 3 : PlayFair Encryption and Decryption

PRN No: 2020BTECS00057

Name: Mrunal Anil Khade

Batch: B4

Aim:

To develop and implement the Playfair Cipher and to encryption and decryption on the input plaintext

Theory:

- The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher.
- It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II.
- The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.
- Decrypting the Playfair cipher is as simple as doing the same process in reverse. The receiver has the same key and can create the same key table, and then decrypt any messages made using that key.

Code :

```
utp
```

3.cpp

```cpp
//playfair

#include<bits/stdc++.h>
#include<string>
using namespace std;

class playfair {
   public:
        string msg;
        char n[5][5];

        char getChar( int a, int b )
        {
            //get the characters
            return n[ (b + 5) % 5 ][ (a + 5) % 5 ];
        }

        bool getPos( char l, int &c, int &d )
        {
            //get the position
            for( int y = 0; y < 5; y++ )
            {
                for( int x = 0; x < 5; x++ )
                {
                    if( n[y][x] == l )
                    {
                        c = x;
                        d= y;
                        return true;
                    }
                }
            }
            return false;
        }

        void getText( string t, bool e )
        {
            //get the original message
            msg.clear();
```

3.cpp

```
38          //get the original message
39          msg.clear();
40          for( string::iterator it = t.begin(); it != t.end(); it++ )
41          {
42              //to choose J = I.
43              *it = toupper( *it );
44              if( *it < 65 || *it > 90 )
45                  continue;
46              if( *it == 'J')
47                  *it = 'I';
48              msg += *it;
49          }
50          if( e )
51          {
52              string nmsg = ""; size_t len = msg.length();
53              for( size_t x = 0; x < len; x ++ )
54              {
55                  nmsg += msg[x];
56                  if( x + 1 < len )
57                  {
58                      if( msg[x] == msg[x + 1] )
59                          nmsg += 'X';
60                      else
61                      {
62                          nmsg += msg[x + 1];
63                          x++;
64                      }
65                  }
66              }
67              msg = nmsg;
68          }
69          if( msg.length() & 1 )
70              msg += 'X';
71      }
72
73      void createEncoder( string key)
74      {
75          //creation of the key table
76          string s= "";
```

3.cpp

```
        {
            if('A'+i == 'J')
            {
                if(find (v.begin(), v.end(), 'I') != v.end())
                    continue;
                v.push_back('I');
            }
            else
            {
                if(find (v.begin(), v.end(), 'A'+i) != v.end())
                    continue;
                v.push_back('A'+i);
            }
        }
        for(int i=0;i<v.size();i++)
        {
            s+=v[i];
        }
        copy( s.begin(), s.end(), &n[0][0] );
    }

    void play( int dir )
    {
        int j,k,p,q;
        string nmsg;
        for( string::const_iterator it = msg.begin(); it != msg.end(); it++ )
        {
            if( getPos( *it++, j, k ) )
            {
                if( getPos( *it, p, q))
                {
                    //for same row
                    if( j == p )
                    {
                        nmsg += getChar( j, (k + dir+5)%5 );
                        nmsg += getChar( p, (q + dir+5)%5);
                    }
                    //for same column
                    else if( k == q )
```

3.cpp

```cpp
int main()
{
    playfair pf;

    int choice;
    int datachoice;
    string sample,key;
    int shift;
    while(1)
    {
        cout << "PlayFair Cipher\n 1. Encryption \n 2. Decryption\n 3. Exit\nEnter Choice: ";
        cin>>choice;
        if(choice>2)
            break;
        switch(choice)
        {
            case 1:
                cout<<"Enter data to be Encrypted:\n";
                cin.ignore();
                getline(cin,sample);
                cout<<"Enter the key: ";
                getline(cin,key);
                cout<<"Encrypted String:\n";
                cout<<pf.play(key,sample,true)<<endl;

                break;
            case 2:

                cout<<"Enter data to be Decrypted:\n";
                cin.ignore();
                getline(cin,sample);
                cout<<"Enter the key: ";
                getline(cin,key);
                cout<<"Decrypted String:\n";
                cout<<pf.play(key,sample,false)<<endl;
                break;
        }
    }
    return 0;
```

```
mrunal@mrunal:~/Desktop/CNS$ cd "/home/mrunal/Desktop/CNS/" && g++ 3.cpp -o 3 && "/home/mrunal/Desktop/CNS/"3
PlayFair Cipher
 1. Encryption
 2. Decryption
 3. Exit
Enter Choice: 1
Enter data to be Encrypted:
MrunalKhade
Enter the key: Miss
Encrypted String:
SPTOSNLKIFLS
PlayFair Cipher
 1. Encryption
 2. Decryption
 3. Exit
Enter Choice: 2
Enter data to be Decrypted:
SPTOSNLKIFLS
Enter the key: Miss
Decrypted String:
MRUNALKHADEX
PlayFair Cipher
 1. Encryption
 2. Decryption
 3. Exit
Enter Choice: 3
mrunal@mrunal:~/Desktop/CNS$
```

You have Docker installed on your system. Do
install the recommended extensions from Mic

Conclusion:

Performed the experiment successfully. Encrypted the data with the provided key. Output of this encryption is decrypted to match the plaintext that was inputted by the user as shown in the above diagram.