**4CS372 : Cryptography and Network Security**

**B.Tech. (CSE) – I [ 2022-23 ]**

**Assignment No - 3**

# **Implementation of Play Fair Cipher**

**Aim:** Implementation of Play Fair Cipher in cpp

**Theory:**Playfair is type of substitution encryption technique. Where we create matrix of size 5\*5.

And 1 string key will be created with plain text.after that fill the matrix with key without duplication.after that fill remaining empty cells with remaining alphabets.

After this make pair from plain text.if any letter is appear more than once in pair then append x or z or any letter so no two same letter in 1 pair.

Search that pair in matrix

1. If both present in same row -> take letter from left to right.
2. If both present in same column-> take letter from top to bottom.
3. Else make rectangle and take swap letter from rectangle

In this way playfair performed!!

**Code:**

#include<bits/stdc++.h>

#include<string>

using namespace std;

class playfair {

    public:

        string msg;

        char n[5][5];

        char getChar( int a, int b )

        {

            return n[ (b + 5) % 5 ][ (a + 5) % 5 ];

        }

        bool getPos( char l, int &c, int &d )

        {

            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 )

        {

            msg.clear();

            for( string::iterator it = t.begin(); it != t.end(); it++ )

            {

                \*it = toupper( \*it );

                if( \*it < 65 || \*it > 90 )

                    continue;

                if( \*it == 'J')

                    \*it = 'I';

                msg += \*it;

            }

            if( e )

            {

                string nmsg = ""; size\_t len = msg.length();

                for( size\_t x = 0; x < len; x ++ )

                {

                    nmsg += msg[x];

                    if( x + 1 < len )

                    {

                        if( msg[x] == msg[x + 1] )

                            nmsg += 'X';

                        else

                        {

                            nmsg += msg[x + 1];

                            x++;

                        }

                    }

                }

                msg = nmsg;

            }

            if( msg.length() & 1 )

                msg += 'X';

        }

        void createEncoder( string key)

        {

            string s= "";

            vector <char> v;

            for(int i=0;i<key.size();i++)

            {

                if(key[i] == 'J')

                {

                    continue;

                }

                else

                {

                    if(find (v.begin(), v.end(), toupper(key[i])) != v.end())

                        continue;

                    v.push\_back(toupper(key[i]));

                }

            }

            for(int i=0;i<26;i++)

            {

                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))

                    {

                        if( j == p )

                        {

                            nmsg += getChar( j, (k + dir+5)%5 );

                            nmsg += getChar( p, (q + dir+5)%5);

                        }

                        else if( k == q )

                        {

                            nmsg += getChar( (j + dir+5)%5, k );

                            nmsg += getChar( (p + dir+5)%5, q );

                        }

                        else

                        {

                            nmsg += getChar( p, k );

                            nmsg += getChar( j, q );

                        }

                    }

                }

            }

            msg = nmsg;

        }

        string play( string k, string t, bool e )

        {

            createEncoder(k);

            getText( t, e );

            if( e )

                play( 1 );

            else

                play( -1 );

            return msg;

        }

};

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 << "Data is from\n 1. Manual Entering \n 2. File \nEnter Choice: ";

                cin>>datachoice;

                if(datachoice == 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;

                }

                else

                {

                    cout<<"Enter File Name:\n";

                    cin.ignore();

                    getline(cin,sample);

                    cout<<"Enter the key: ";

                    getline(cin,key);

                    fstream myfile;

                    myfile.open(sample.c\_str());

                    string str,s;

                    if(!myfile.is\_open())

                        cout << "Error while Opening File";

                    while(getline(myfile,str))

                        s+=str;

                    myfile.close();

                    s=pf.play(key,s,true);

                    myfile.open("CipherText.txt",ios\_base::out);

                    if(myfile.is\_open())

                        myfile.write(s.data(),s.size());

                    cout<<"File Encrypted\n";

                    myfile.close();

                }

                break;

            case 2:

                cout << "Data is from\n 1. Manual Entering \n 2. File \nEnter Choice: ";

                cin>>datachoice;

                if(datachoice == 1)

                {

                    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;;

                }

                else

                {

                    cout<<"Enter File Name:\n";

                    cin.ignore();

                    getline(cin,sample);

                    cout<<"Enter the key: ";

                    getline(cin,key);

                    fstream myfile;

                    myfile.open(sample.c\_str());

                    string str,s;

                    if(!myfile.is\_open())

                        cout << "Error while Opening File";

                    while(getline(myfile,str))

                        s+=str;

                    myfile.close();

                    s=pf.play(key,s,false);

                    myfile.open("PlainText.txt",ios\_base::out);

                    if(myfile.is\_open())

                        myfile.write(s.data(),s.size());

                    cout<<"File Decrypted\n";

                    myfile.close();

                }

                break;

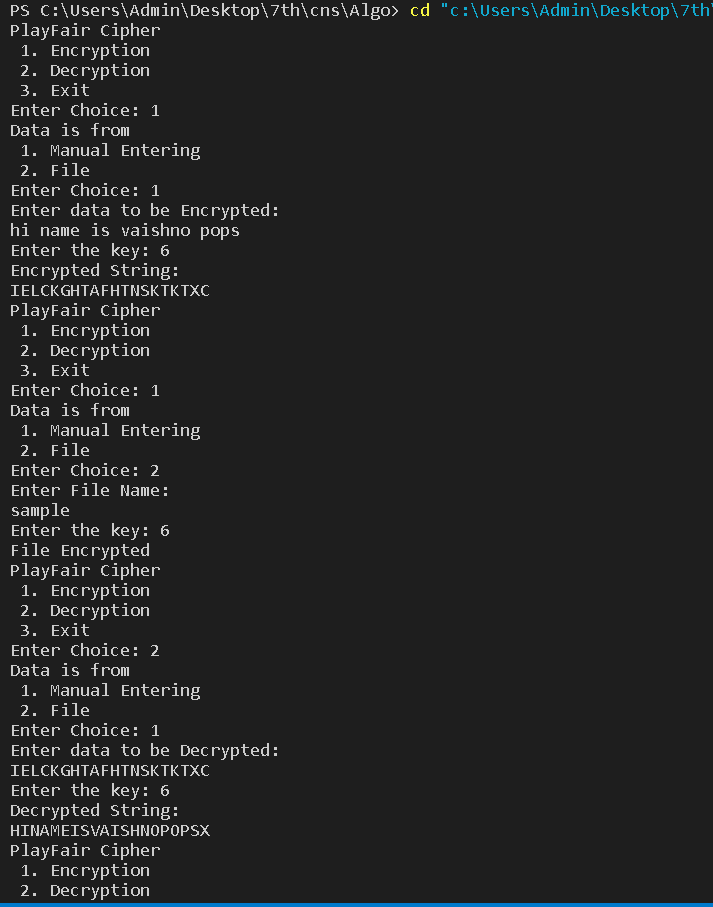
        }

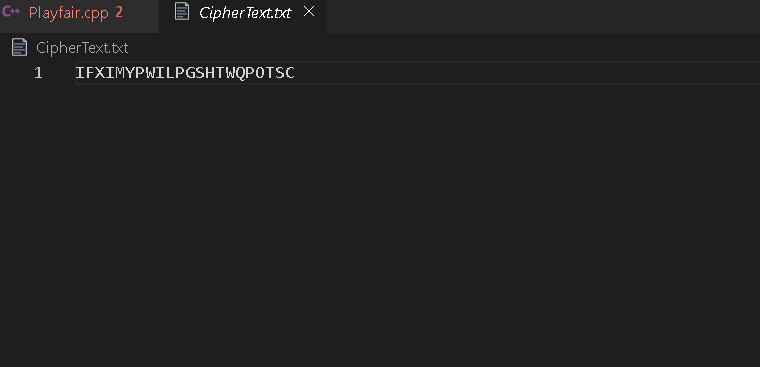
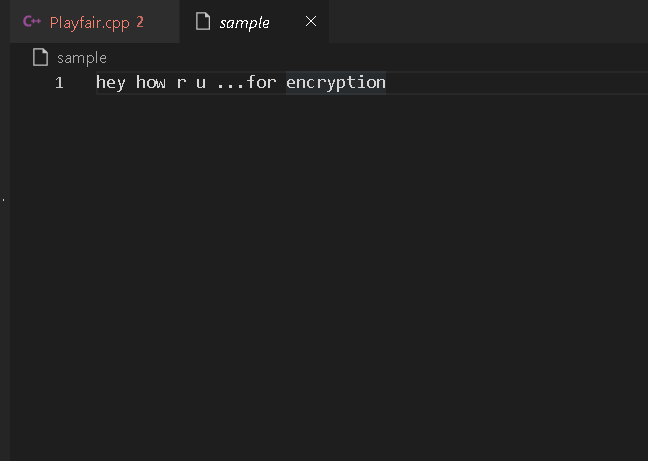
    }

    return 0;

}

**Output:**





**Conclusion:**

Playfair is quite complicated than ceasar cipher ..but Hard to crack.