Experiment 03

Aim: Write a program in Java or Python to perform Cryptanalysis or decoding Playfair Cipher.

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D15-B
Subject	Internet Security Lab
LO Mapped	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers.

AIM: Write a program in Java or Python to perform Cryptanalysis or decoding of Playfair Cipher.

THEORY:

The Playfair cipher or Playfair square or Wheatstone–Playfair cipher is a manual symmetric encryption technique and was the first literal diagram substitution cipher.

The technique encrypts pairs of letters (*bigrams* or *digrams*), instead of single letters as in the simple substitution cipher and rather more complex Vigenère cipher systems then in use. The Playfair is thus significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. The frequency analysis of bigrams is possible, but considerably more difficult. With 600[1] possible bigrams rather than the 26 possible monograms (single symbols, usually letters in this context), a considerably larger cipher text is required in order to be useful

The Playfair Cipher Encryption Algorithm: Generate the key Square(5×5): 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.
Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

CODE:

```
import java.awt.Point;
import java.util.Scanner;

public class Main
 {

private int length = 0;

private String [][] table;
public static void main(String args[])
 {
Main pf = new Main();
 }

private Main()
 {

System.out.print("Enter the key for playfair cipher: ");
Scanner sc = new Scanner(System.in);
String key = parseString(sc);
while(key.equals(""))
key = parseString(sc);
table = this.cipherTable(key);

System.out.print("Enter the plaintext to be encipher: ");
String input = parseString(sc);
while(input.equals(""))
input = parseString(sc);

String output = cipher(input);
String decodedOutput = decode(output);

this.keyTable(table);
this.printResults(output,decodedOutput);
 }

private String parseString(Scanner sc)
 {
String parse = sc.nextLine();

parse = parse.toUpperCase();

parse = parse.replaceAll("[^A-Z]", "");
```

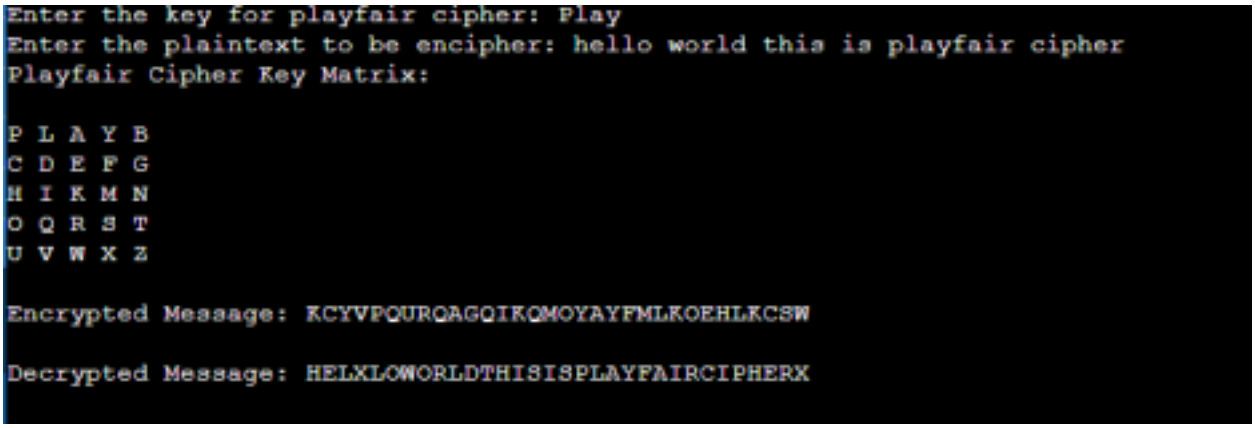
```
parse = parse.replace("J", "I");
return parse;
}

private String[][] cipherTable(String key)
 {

String[][] playfairTable = new String[5][5];
String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";

for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
playfairTable[i][j] = "";
for(int k = 0; k < keyString.length(); k++)
 {
boolean repeat = false;
boolean used = false;
for(int i = 0; i < 5; i++)
 {
for(int j = 0; j < 5; j++)
 {
if(playfairTable[i][j].equals("" + keyString.charAt(k))) {
repeat = true;
}
else if(playfairTable[i][j].equals("") && !repeat && !used) {
playfairTable[i][j] = "" + keyString.charAt(k); used
= true;
}
}
}
}
return playfairTable;
}

private String cipher(String in)
 {
length = (int) in.length() / 2 + in.length() % 2;


for(int i = 0; i < (length - 1); i++)
 {
if(in.charAt(2 * i) == in.charAt(2 * i + 1))
 {
in = new StringBuffer(in).insert(2 * i + 1, 'X').toString();
length = (int) in.length() / 2 + in.length() % 2; }
}
```

```
String[] digraph = new String[length];

for(int j = 0; j < length ; j++)
 {

if(j == (length - 1) && in.length() / 2 == (length - 1))

in = in + "X";
digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1); }
 String out = "";
 String[] encDigraphs = new String[length];
encDigraphs = encodeDigraph(digraph); for(int k
= 0; k < length; k++)
out = out + encDigraphs[k];
return out;
}

private String[] encodeDigraph(String di[]) {
 String[] encipher = new String[length];
for(int i = 0; i < length; i++)
 {
char a = di[i].charAt(0);
char b = di[i].charAt(1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();


if(r1 == r2)
 {
c1 = (c1 + 1) % 5;
c2 = (c2 + 1) % 5;
}

else if(c1 == c2)
 {
r1 = (r1 + 1) % 5;
r2 = (r2 + 1) % 5;
}

else
 {
int temp = c1;
c1 = c2;
c2 = temp;
}
```

```
encipher[i] = table[r1][c1] + "" + table[r2][c2]; }
return encipher;
}

private String decode(String out)
 {
String decoded = "";
for(int i = 0; i < out.length() / 2; i++)
 {
char a = out.charAt(2*i);
char b = out.charAt(2*i+1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
 {
c1 = (c1 + 4) % 5;
c2 = (c2 + 4) % 5;
}
else if(c1 == c2)
 {
r1 = (r1 + 4) % 5;
r2 = (r2 + 4) % 5;
}
else
 {

int temp = c1;
c1 = c2;
c2 = temp;
}
decoded = decoded + table[r1][c1] + table[r2][c2]; }

return decoded;
}

private Point getPoint(char c)
 {
Point pt = new Point(0,0);
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
if(c == table[i][j].charAt(0))
pt = new Point(i,j);
return pt;
}
```

```
private void keyTable(String[][] printTable) {
System.out.println("Playfair Cipher Key Matrix: ");
System.out.println();

for(int i = 0; i < 5; i++)
{

for(int j = 0; j < 5; j++)
{

System.out.print(printTable[i][j]+" ");
}
System.out.println();
}
System.out.println();
}

private void printResults(String encipher, String dec) {
System.out.print("Encrypted Message: ");

System.out.println(encipher);
System.out.println();
System.out.print("Decrypted Message: ");

System.out.println(dec);
}

}
```

OUTPUT:



CONCLUSION: We successfully implemented playfair cipher in java