Experiment 03

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

Roll No.	19
Name	Manav Jawrani
Class	D15-A
Subject	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.

Introduction:

1. What is Cipher?

Ciphers, also called encryption algorithms, are systems for encrypting and decrypting data. A cipher converts the original message, called plaintext, into ciphertext using a key to determine how it is done.

2. What is Playfair Cipher?

Playfair is a substitution cipher.Playfair ciphers are an approach of block cipher and the ciphertext character that restores a specific plaintext character in the encryption will rely upon an element on an contiguous character in the plaintext.

Encryption is adept using a square array of characters, built from the encryption key. Because the group of plaintext characters is the 26-letter English alphabet. This array would be 5 × 5, with 2 of the 26 characters appearing in an individual position in the array.Generally, these two characters are i and j, because usually it can be simply to categorize from the context which of these two letters was pre-determined in the plaintext. The encryption key for a Playfair cipher is a word through a finite order of characters taken from the group of plaintext characters.

The Playfair cipher is a manual symmetric encryption technique and was the first literal diagram substitution cipher. The technique encrypts pairs of letters, 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 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.

First, a plaintext message is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Let us say we want to encrypt the message “hide money”. It will be written as:

HI DE MO NE YZ

Rules of encryption:

If both the letters are in the same column, take the letter below each one (going back to the top if at the bottom)

T U O R I

A L S B C

D E F G H

K M N P Q

V W X Y Z

('H' and 'I' are in the same column, hence take the letter below them to replace. HI → QC) If both letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

T U O R I

A L S B C

D E F G H

K M N P Q

V W X Y Z

('D' and 'E' are in the same row, hence take the letter to the right of them to replace. DE → EF) If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

Using these rules, the result of the encryption of 'hide money' with the key of 'tutorials' would be:

QC EF NU MF ZV

Decrypting the Playfair cipher is as simple as doing the same process in reverse. Receiver has the same key and can create the same key table, and then decrypt any messages made using that key. The Playfair cipher was used mainly to protect important, yet non-critical secrets, as it is quick to use and requires no special equipment.

Algorithm:

STEP 1: Read the plain text from the user.
STEP 2: Read the keyword from the user.
STEP 3: Arrange the keyword without duplicates in a 5*5 matrix in the row order and fill the remaining cells with missed out letters in alphabetical order. Note that ' i' and ' j' take the same cell.
STEP 4: Group the plain text in pairs and match the corresponding corner letters by forming a rectangular grid.
STEP 5: Display the obtained cipher text.

Code:

```
import java.awt.Point;
import java.util.Scanner;
public class PlayfairCipher
{
private int length = 0;
private String [][] table;
public static void main(String args[])
{
PlayfairCipher pf = new PlayfairCipher();
}
private PlayfairCipher()
{
System.out.print("Enter the key for playfair cipher: \n");
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: \n");
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: \n");
System.out.println(encipher);
System.out.println();
System.out.print("Decrypted Message: \n");
System.out.println(dec);
}
}
```

Output:

Case 1: Key has a repeated letter

Here key is - Ishika

```
java -cp /tmp/MNjMPDajnL PlayfairCipher
Enter the key for playfair cipher:
Ishika
Enter the plaintext to be encipher:
The Secret Message is Hello
Playfair Cipher Key Matrix:
I S H K A
B C D E F
G L M N O
P Q R T U
V W X Y Z

Encrypted Message:
RKCKFDTDRNCKHINBSHKDMWMG

Decrypted Message:
THESECRETMESSAGEISHELXLO
```

Case 2: Key doesn't has a repeated letter
Here key is - Omkar

```
java -cp /tmp/wO5XAnv9qf PlayfairCipher
Enter the key for playfair cipher:
Omkar
Enter the plaintext to be encipher:
The Secret Message is Hello
Playfair Cipher Key Matrix:

O M K A R
B C D E F
G H I L N
P Q S T U
V W X Y Z

Encrypted Message:
QLDTFDAFQADTTKLBSXLCIYGA

Decrypted Message:
THESECRETMESSAGEISHELXLO
```

Conclusion:
We have performed cryptanalysis and decoded the playfair cipher and vigenere cipher by using a python program.