Experiment 03

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

Roll No.	14
Name	Ishika Sanjay Devare
Class	D15-B
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 and Vigenere cipher.

Introduction:

What is a Cipher?

In cryptography, a cipher (or cypher) is an algorithm for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure.

What is a 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 Vigenere 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:

Hl DE MO NE YZ

The Playfair Cipher Encryption Algorithm:

The Algorithm consists of 2 steps:

1. **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.

2. **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.

For example-

PlainText: "instruments"

After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'

1. Pair cannot be made with the same letter. Break the letter in single and add a bogus letter to the previous letter.

Plain Text: "hello"

After Split: 'he' 'lx' 'lo'

Here **'x'** is the bogus letter.

2. If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter

Plain Text: "helloe"

AfterSplit: 'he' 'lx' 'lo' 'ez'

Here **'z'** is the bogus letter.

- **If both the letters are in the same column**: Take the letter below each one (going back to the top if at the bottom).
  For example:

Diagraph: "me"

Encrypted Text: cl

Encryption:

m -> c

e -> l

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

- **If both the letters are in the same row**: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).
  For example:

Diagraph: "st"

Encrypted Text: tl

Encryption:

s -> t

t -> l

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

- **If neither of the above rules is true**: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

Diagraph: "nt"

Encrypted Text: rq

Encryption:

n -> r

t -> q

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

Decryption Technique

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.

The Playfair Cipher Decryption Algorithm:

The Algorithm consists of 2 steps:

1. **Generate the key Square(5×5) at the receiver's end:**
   - 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.
2. **Algorithm to decrypt the ciphertext:** The ciphertext is split into pairs of two letters (digraphs).

   For example:

   CipherText: "gatlmzclrqtx"

   After Split: 'ga' 'tl' 'mz' 'cl' 'rq' 'tx'

Rules for Decryption:

- **If both the letters are in the same column**: Take the letter above each one (going back to the bottom if at the top).

  Diagraph: "cl"
  Decrypted Text: me
  Decryption:
  c -> m
  l -> e

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

- **If both the letters are in the same row**: Take the letter to the left of each one (going back to the rightmost if at the leftmost position).

  Diagraph: "tl"

  Decrypted Text: st

  Decryption:

  t -> s

  l -> t

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

- **If neither of the above rules is true**: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

  Diagraph: "rq"

  Decrypted Text: nt

  Decryption:

  r -> n

  q -> t

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

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 fi11 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
{
//length of digraph array
private int length = 0;
//creates a matrix for Playfair cipher
private String [][] table;
//main() method to test Playfair method
public static void main(String args[])
{
PlayfairCipher pf = new PlayfairCipher();
}
//main run of the program, Playfair method
//constructor of the class
private PlayfairCipher()
{
//prompts user for the keyword to use for encoding & creates tables
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);
//prompts user for message to be encoded
System.out.print("Enter the plaintext to be encipher: ");
//System.out.println("using the previously given keyword");
String input = parseString(sc);
while(input.equals(""))
input = parseString(sc);
//encodes and then decodes the encoded message
String output = cipher(input);
String decodedOutput = decode(output);
//output the results to user
this.keyTable(table);
this.printResults(output,decodedOutput);
}
//parses an input string to remove numbers, punctuation,
//replaces any J's with I's and makes string all caps
private String parseString(Scanner sc)
{
String parse = sc.nextLine();
//converts all the letters in upper case
parse = parse.toUpperCase();
//the string to be substituted by space for each match (A to Z)
parse = parse.replaceAll("[^A-Z]", "");
//replace the letter J by I
parse = parse.replace("J", "I");
```

```
return parse;
}
//creates the cipher table based on some input string (already parsed)
private String[][] cipherTable(String key)
{
//creates a matrix of 5*5
String[][] playfairTable = new String[5][5];
String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";
//fill string array with empty string
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;
}
//cipher: takes input (all upper-case), encodes it, and returns the output
private String cipher(String in)
{
length = (int) in.length() / 2 + in.length() % 2;
//insert x between double-letter digraphs & redefines "length"

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;
}
}
//------------makes plaintext of even length--------------
//creates an array of digraphs
String[] digraph = new String[length];
//loop iterates over the plaintext
for(int j = 0; j < length ; j++)
{
//checks the plaintext is of even length or not
if(j == (length - 1) && in.length() / 2 == (length - 1))
//if not addends X at the end of the plaintext
in = in + "X";
digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1);
}
```

```
//encodes the digraphs and returns the output
String out = "";
String[] encDigraphs = new String[length];
encDigraphs = encodeDigraph(digraph);
for(int k = 0; k < length; k++)
out = out + encDigraphs[k];
return out;
}
//---------------encryption logic-----------------
//encodes the digraph input with the cipher's specifications
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();
//executes if the letters of digraph appear in the same row
//in such case shift columns to right
if(r1 == r2)
{
c1 = (c1 + 1) % 5;
c2 = (c2 + 1) % 5;
}
//executes if the letters of digraph appear in the same column
//in such case shift rows down
else if(c1 == c2)
{
r1 = (r1 + 1) % 5;
r2 = (r2 + 1) % 5;
}
//executes if the letters of digraph appear in the different row and different column
//in such case swap the first column with the second column
else
{
int temp = c1;
c1 = c2;
c2 = temp;
}
//performs the table look-up and puts those values into the encoded array
encipher[i] = table[r1][c1] + "" + table[r2][c2];
}
return encipher;
}
//----------------------decryption logic---------------------
// decodes the output given from the cipher and decode methods (opp. of encoding process)
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
{
//swapping logic
int temp = c1;
c1 = c2;
c2 = temp;
}
decoded = decoded + table[r1][c1] + table[r2][c2];
}
//returns the decoded message
return decoded;
}
// returns a point containing the row and column of the letter
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;
}
//function prints the key-table in matrix form for playfair cipher
private void keyTable(String[][] printTable)
{
System.out.println("Playfair Cipher Key Matrix: ");
System.out.println();

//loop iterates for rows
for(int i = 0; i < 5; i++)
{
//loop iterates for column
for(int j = 0; j < 5; j++)
{
//prints the key-table in matrix form
System.out.print(printTable[i][j]+" ");
}
System.out.println();
}
System.out.println();
}
//method that prints all the results
private void printResults(String encipher, String dec)
{
System.out.print("Encrypted Message: ");
//prints the encrypted message
System.out.println(encipher);
System.out.println();
System.out.print("Decrypted Message: ");
//prints the decryted message
System.out.println(dec);
}
}
```

Results:

```
java -cp /tmp/dfeKBohTPi PlayfairCipher
Enter the key for playfair cipher:
Ishika
Enter the plaintext to be encipher:
My Name is Garviit
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: NXOKNDSHOIPXHVKP

Decrypted Message: MYNAMEISGARVIXIT
```

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

Encrypted Message: LZOGSMEZEZMAHR

Decrypted Message: MYNAMEISISHIKA
```

Conclusion: Hence, we successfully wrote a program in Java to perform Cryptanalysis or decoding of Playfair Cipher.