Cryptography and Network Security Lab

Assignment 4
Implementation and Understanding of Playfair Cipher

2019BTECS00058
Devang K

Batch: B2

Title: Implementation and Understanding of Playfair Cipher

Aim: To Study, Implement and Demonstrate the Playfair Cipher Algorithm

Theory:

Playfair Cipher is a manual symmetric encryption technique invented by Wheatstone and brought in use by Playfair. The technique encrypts pairs of letters – diagrams and the algorithm is harder to break. It requires the Plaintext and the Encryption Key.

To encrypt in Playfair Cipher, we take the key and remove any ‘J’ instance. Then, in the plaintext, if we have an instance of ‘J’ we replace it with ‘I’. Then, we construct a PlayfairMatrix a 5*5 square – which starts by putting all the unique characters of the key in that order. Then, it is filled by all the characters not in the key (except ‘J’).

After this, the plaintext is divided into set of diagraphs, with condition that both characters have to be unique, if not, bogus character ‘Z’ is added. Same to be done at the end to balance all the diagraphs.

Then, for each plaintext diagraph, we shall generate the cipher as:
We take the characters on diagraphs and plot it on the PlayfairMatrix. Then, if:

Both are in the same column - Take the letter below each one (going back to the top if at the bottom).

Both 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 all other cases - Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

Then, combining the generated diagraphs would give us our cipher-text.

To decrypt the cipher, we again generate the PlayfairMatrix based off the key. Then we divide our encrypted text in a list of diagraphs. Then, for each diagraph if:

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

Both 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).

If neither - Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

Basically, doing the opposite.

Then, we finally combine these digraphs to generate the plain-text.

Illustration.

Say we wish to encrypt ‘ALLFORCESATTACK’ with key ‘ELIZABETH’.

We start by making the Playfair square. Filling with unique characters of the key eliminating ‘J’:

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

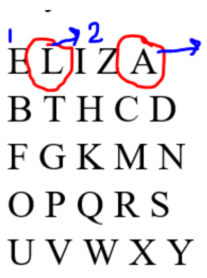
Then, we make the Playfair Diagraphs of the Plaintext.
First, we replace all ‘J’ with ‘I’. Then,

We take ‘AL’, they are unique and therefore can be diagraphs
Next, we take ‘LF’, ‘OR’ and so on. A case we encounter at ‘TTACK’, ‘TT’ cannot be taken, so we convert it to ‘TX’ and ‘TA’. Therefore, we get:

‘AL’, ‘LF’, ‘OR’, ‘CE’, ‘SA’, ‘TX’, ‘TA’, ‘CK’

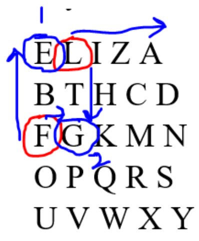
Then, for every diagraph, we encrypt using our Playfair Matrix.

Start with ‘AL’, same row



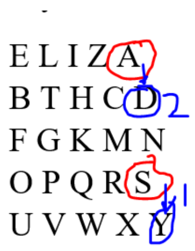
Therefore, the diagraph becomes ‘EI’.

Next, we find, ‘LF’, different rows and columns:



The encrypted diagraph becomes ‘EG’.

Similarly, we keep going till ‘SA’ – same column.



The encrypted diagraph becomes ‘YD’.

Thus, all the diagraphs can be encrypted and resultant ciphertext would be – ‘EIEGPSBZYDCVDLHM’.

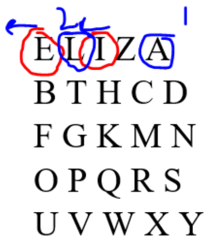
Now to decrypt, we again make the Playfair Matrix from the key and divide our ciphertext into diagraphs.

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

‘EI’, ‘EG’, ‘PS’, ‘BZ’, ‘YD’, ‘CV’, ‘DL’, ‘HM’.

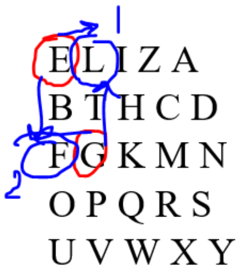
Now let us take a look at decryption cases of ‘EI’, ‘EG’ and ‘YD’ to cover our 3 cases:

For ‘EI’,



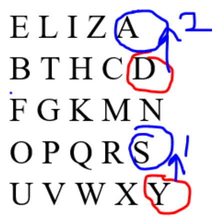
The Plaintext diagraph becomes ‘AL’.

For ‘EG’,



The Plaintext diagraph becomes ‘LF’

Finally, for ‘YD’,



The Plaintext diagraph becomes ‘SA’

Our Plaintext diagraphs are: ‘AL’, ‘LF’, ‘OR’, ‘CE’, ‘SA’, ‘TX’, ‘TA’, ‘CK’

Thus, our final plaintext would be: ‘ALLFORCESATXTACK’.

Code:

The crux of the application is in the functions that compute the Playfair Matrix and the Diagraph Generator. Rest of the functions are straightforward.

Code for Encryption:

```
alphabets = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

alphabetsList = []
for a in alphabets:
    alphabetsList.append(a)

def printMatrix(fenceMatrix):
    for i in range(len(fenceMatrix)):
        for j in range(len(fenceMatrix[0])):
            print(fenceMatrix[i][j], end=' ')
        print()


def generatekeyMatrix(key):
    # We are to make a 5*5 matrix

    # Splitting the key to a list
    keySet = []
    for k in key:
        keySet.append(k)
    keySet = list(dict.fromkeys(keySet))
    try:
        keySet.remove('J')
    except:
        pass
    for a in alphabetsList:
        if a != 'J':
            if a not in keySet:
                keySet.append(a)

    keyMatrix = []
    keyMatrix.append(keySet[0:5])
    keyMatrix.append(keySet[5:10])
    keyMatrix.append(keySet[10:15])
```

```python
    keyMatrix.append(keySet[15:20])
    keyMatrix.append(keySet[20:25])

    return keyMatrix

def generatePlainTextDiagraphs(plaintext):
    # Replace all J with I
    plaintext = plaintext.replace("J", "I")
    lstOfDraphs = []
    aDiagraph = ""
    for i in range(len(plaintext)):
        if len(aDiagraph) == 0:
            aDiagraph+=plaintext[i]
        elif len(aDiagraph) == 1:
            if plaintext[i] == aDiagraph:
                aDiagraph += "X"
                lstOfDraphs.append(aDiagraph)
                aDiagraph = plaintext[i]
            else:
                aDiagraph += plaintext[i]
                lstOfDraphs.append(aDiagraph)
                aDiagraph = ""
    if len(aDiagraph) == 1:
        aDiagraph+="Z"
        lstOfDraphs.append(aDiagraph)
    return lstOfDraphs

def getPositionOfACharInKeyMatrix(keyMatrix, c):
    for i in range(5):
        for j in range(5):
            if keyMatrix[i][j] == c:
                return i,j

print("Playfair Cipher")
print("Enter Plaintext: ", end='')
plaintext = input()
print("Enter Key: ", end='')
key = input()

keyMatrix = generatekeyMatrix(key)

print("\nKey Matrix")
print(keyMatrix)

plainTextDiagraphs = generatePlainTextDiagraphs(plaintext)

print("\nPlaintext Diagraphs")
encryptedDigraphs = []
```

```python
print(plainTextDiagraphs)
printMatrix(keyMatrix)

for ptd in plainTextDiagraphs:
    c1Row, c1Col = getPositionOfACharInKeyMatrix(keyMatrix, ptd[0])
    c2Row, c2Col = getPositionOfACharInKeyMatrix(keyMatrix, ptd[1])

    newDiagraph = ""

    # if both are in same column
    if c1Col == c2Col:
        newDiagraph+=keyMatrix[(c1Row+1)%5][c1Col]
        newDiagraph+=keyMatrix[(c2Row+1)%5][c2Col]
        encryptedDigraphs.append(newDiagraph)

    # if both are in same row
    elif c1Row == c2Row:
        newDiagraph+=keyMatrix[c1Row][(c1Col+1)%5]
        newDiagraph+=keyMatrix[c2Row][(c2Col+1)%5]
        encryptedDigraphs.append(newDiagraph)

    # else
    else:
        newDiagraph+=keyMatrix[c1Row][c2Col]
        newDiagraph+=keyMatrix[c2Row][c1Col]
        encryptedDigraphs.append(newDiagraph)

print("Plaintext: "+plaintext)
print("Encrypted Text: "+ "".join(encryptedDigraphs))
```

Code for Decryption:

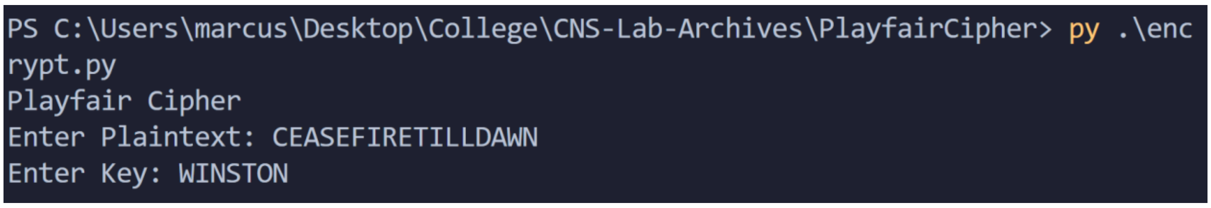
```python
alphabets = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

alphabetsList = []
for a in alphabets:
    alphabetsList.append(a)

def printMatrix(fenceMatrix):
    for i in range(len(fenceMatrix)):
        for j in range(len(fenceMatrix[0])):
            print(fenceMatrix[i][j], end=' ')
        print()

def generatekeyMatrix(key):
    # We are to make a 5*5 matrix
    # Splitting the key to a list
    keySet = []
    for k in key:
        keySet.append(k)
    keySet = list(dict.fromkeys(keySet))
    try:
        keySet.remove('J')
    except:
        pass
    for a in alphabetsList:
        if a != 'J':
            if a not in keySet:
                keySet.append(a)
    keyMatrix = []
    keyMatrix.append(keySet[0:5])
    keyMatrix.append(keySet[5:10])
    keyMatrix.append(keySet[10:15])
    keyMatrix.append(keySet[15:20])
    keyMatrix.append(keySet[20:25])
    return keyMatrix

def generatePlainTextDiagraphs(plaintext):
    # Replace all J with I
    plaintext = plaintext.replace("J", "I")
    lstOfDraphs = []
    aDiagraph = ""
    for i in range(len(plaintext)):
        if len(aDiagraph) == 0:
            aDiagraph+=plaintext[i]
        elif len(aDiagraph) == 1:
            if plaintext[i] == aDiagraph:
                aDiagraph += "X"
                lstOfDraphs.append(aDiagraph)
                aDiagraph = plaintext[i]
            else:
                aDiagraph += plaintext[i]
                lstOfDraphs.append(aDiagraph)
                aDiagraph = ""
    if len(aDiagraph) == 1:
        aDiagraph+="Z"
        lstOfDraphs.append(aDiagraph)
    return lstOfDraphs

def getPositionOfACharInKeyMatrix(keyMatrix, c):
    for i in range(5):
        for j in range(5):
            if keyMatrix[i][j] == c:
                return i,j
```

```python
print("Playfair Cipher")
print("Enter Encrypted Text: ", end='')
enctext = input()
print("Enter Key: ", end='')
key = input()

keyMatrix = generatekeyMatrix(key)

print("\nKey Matrix")
printMatrix(keyMatrix)

encTextDiagraphs = generatePlainTextDiagraphs(enctext)
print("\nEnc Text Diagraphs")
print(encTextDiagraphs)

decryptedDigraphs = []

for ptd in encTextDiagraphs:
    c1Row, c1Col = getPositionOfACharInKeyMatrix(keyMatrix, ptd[0])
    c2Row, c2Col = getPositionOfACharInKeyMatrix(keyMatrix, ptd[1])

    newDiagraph = ""

    # if both are in same column
    if c1Col == c2Col:
        newDiagraph+=keyMatrix[(c1Row-1)%5][c1Col]
        newDiagraph+=keyMatrix[(c2Row-1)%5][c2Col]
        decryptedDigraphs.append(newDiagraph)

    # if both are in same row
    elif c1Row == c2Row:
        newDiagraph+=keyMatrix[c1Row][(c1Col-1)%5]
        newDiagraph+=keyMatrix[c2Row][(c2Col-1)%5]
        decryptedDigraphs.append(newDiagraph)

    # else
    else:
        newDiagraph+=keyMatrix[c1Row][c2Col]
        newDiagraph+=keyMatrix[c2Row][c1Col]
        decryptedDigraphs.append(newDiagraph)

print("Encrypted Text: "+enctext)
print("Plain Text: "+ "".join(decryptedDigraphs))
```

We now solve some examples with the code.

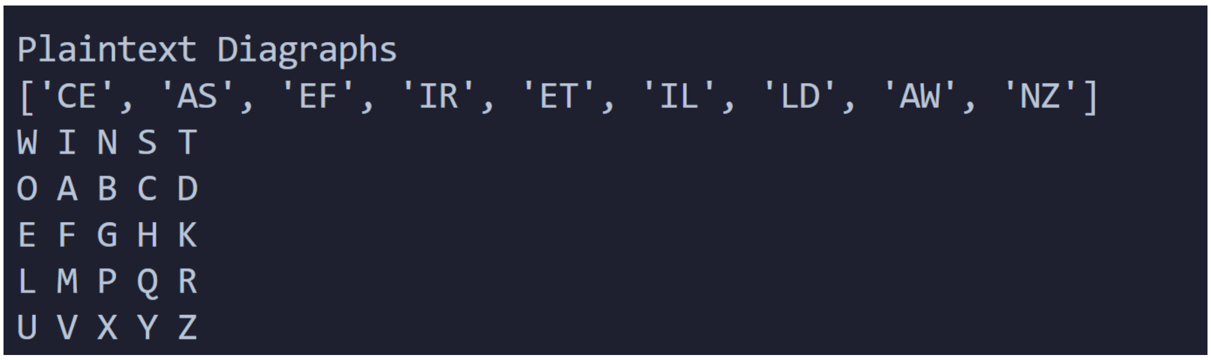
Say we wish to encrypt: ‘CEASEFIRETILLDAWN’
and we take our key as ‘WINSTON’

Running our code for encryption:



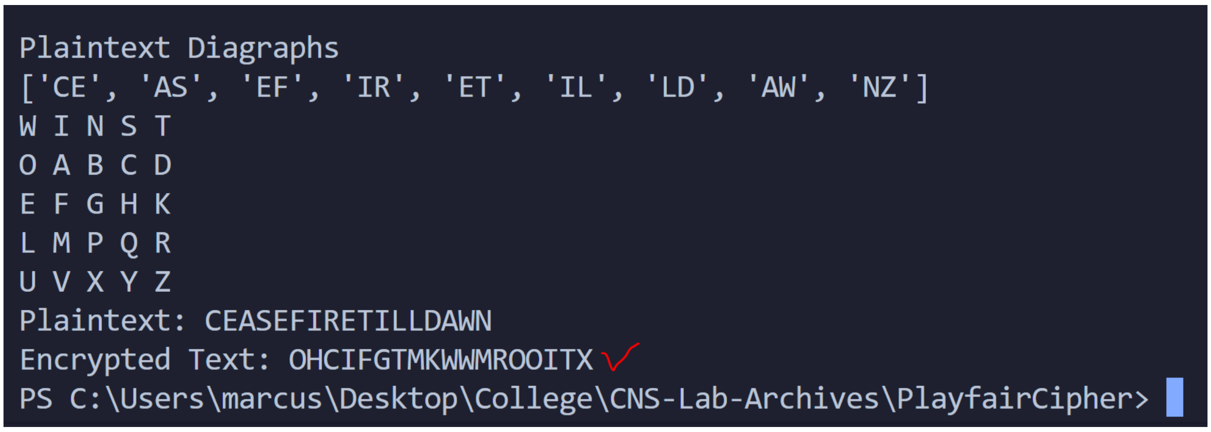
```
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher> py .\enc
rypt.py
Playfair Cipher
Enter Plaintext: CEASEFIRETILLDAWN
Enter Key: WINSTON
```

We generate the Playfair Matrix and Plaintext Diagraphs with respect to our conditions:



```
Plaintext Diagraphs
['CE', 'AS', 'EF', 'IR', 'ET', 'IL', 'LD', 'AW', 'NZ']
W I N S T
O A B C D
E F G H K
L M P Q R
U V X Y Z
```

Then, checking with the matrix for every plaintext diagraph, we get:

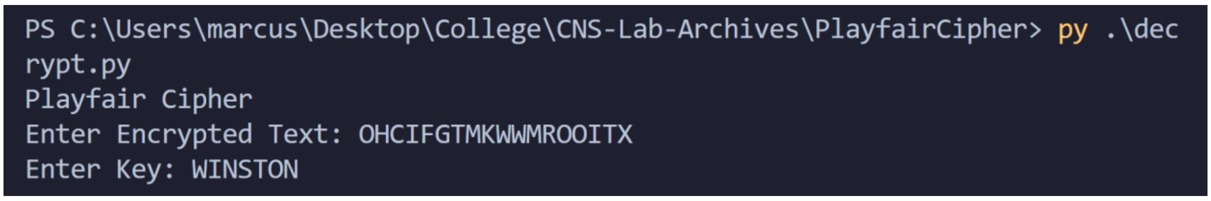


```
Plaintext Diagraphs
['CE', 'AS', 'EF', 'IR', 'ET', 'IL', 'LD', 'AW', 'NZ']
W I N S T
O A B C D
E F G H K
L M P Q R
U V X Y Z
Plaintext: CEASEFIRETILLDAWN
Encrypted Text: OHCIFGTMKWWMROOITX ✓
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher>
```

Thus, our ciphertext is ‘OHCIFGTMKWWMROOITX’

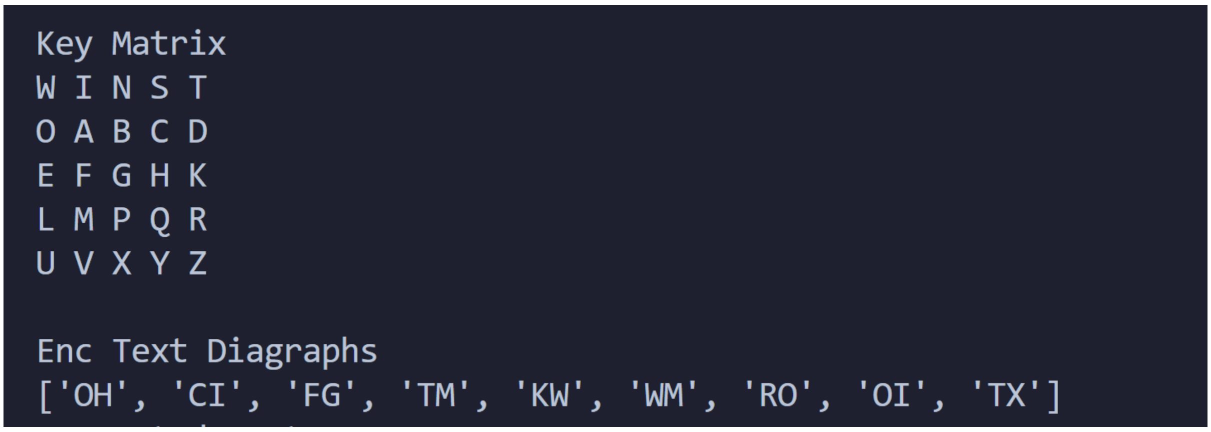
Now, to decrypt:

With encrypted text – ‘OHCIFGTMKWWMROOITX’ and key – ‘WINSTON’



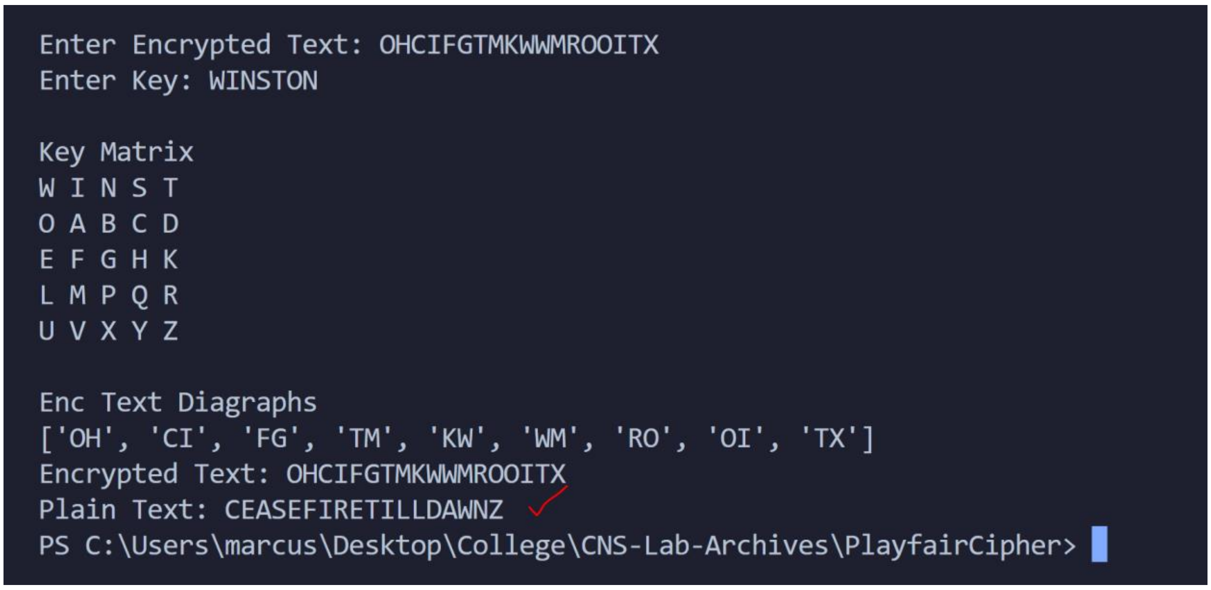
```
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher> py .\dec
rypt.py
Playfair Cipher
Enter Encrypted Text: OHCIFGTMKWWMROOITX
Enter Key: WINSTON
```

We generate the key matrix and the encrypted text diagraphs:



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

Enc Text Diagraphs
['OH', 'CI', 'FG', 'TM', 'KW', 'WM', 'RO', 'OI', 'TX']
```

Then,

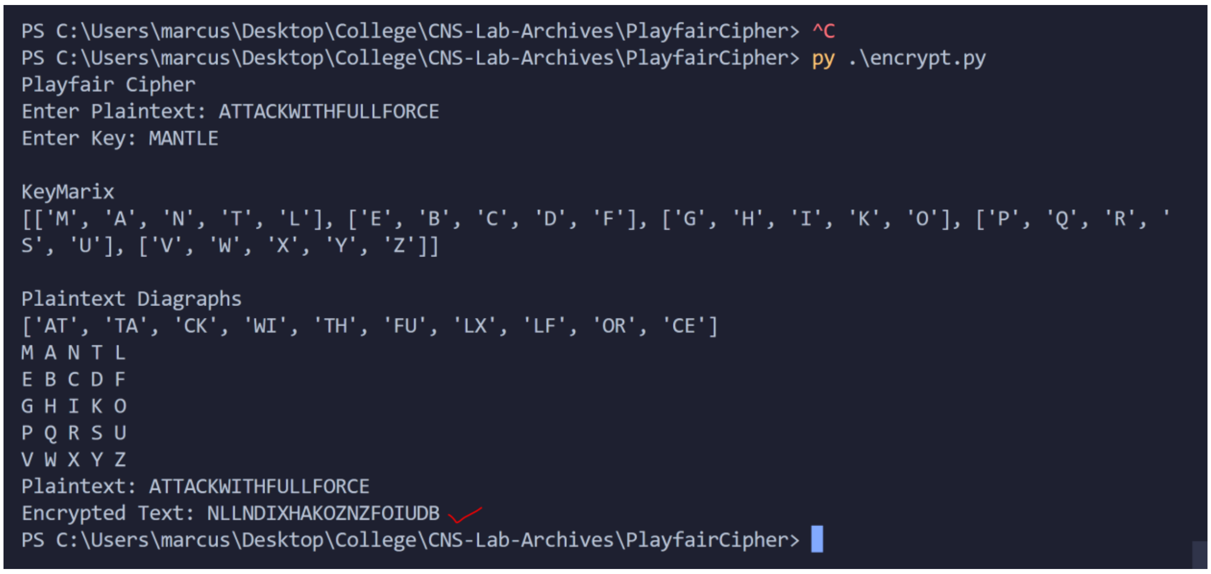


```
Enter Encrypted Text: OHCIFGTMKWWMROOITX
Enter Key: WINSTON

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

Enc Text Diagraphs
['OH', 'CI', 'FG', 'TM', 'KW', 'WM', 'RO', 'OI', 'TX']
Encrypted Text: OHCIFGTMKWWMROOITX
Plain Text: CEASEFIRETILLDAWNZ ✓
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher>
```

Thus, we get our plaintext back.

We take another example:

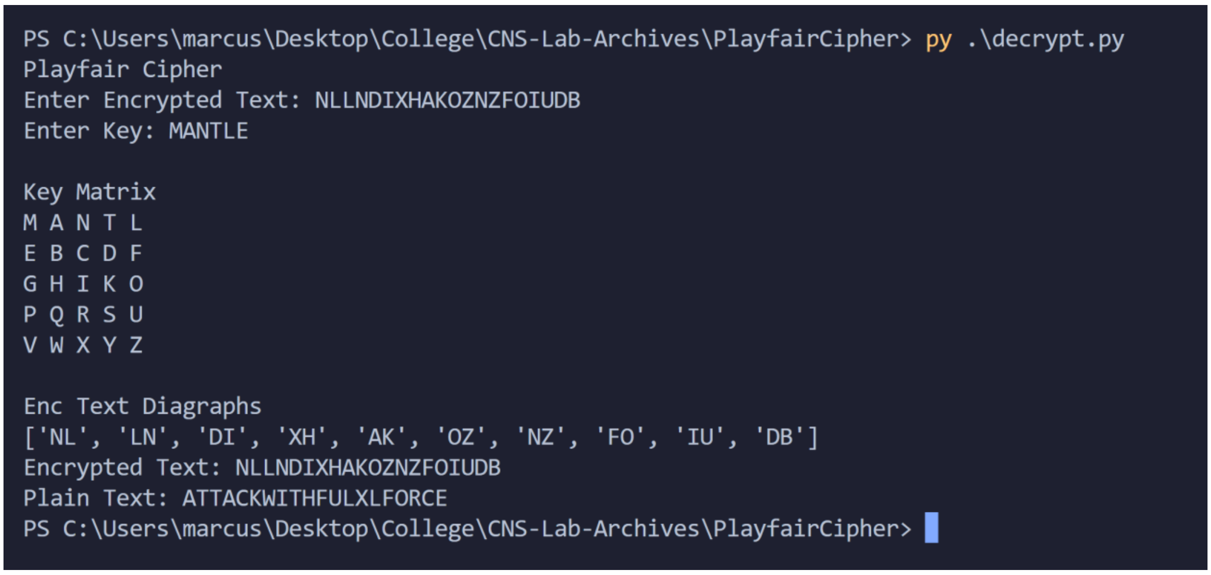
Say we wish to encrypt ‘ATTACKWITHFULLFORCE’ with key ‘MANTLE’.



```
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher> ^C
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher> py .\encrypt.py
Playfair Cipher
Enter Plaintext: ATTACKWITHFULLFORCE
Enter Key: MANTLE

KeyMarix
[['M', 'A', 'N', 'T', 'L'], ['E', 'B', 'C', 'D', 'F'], ['G', 'H', 'I', 'K', 'O'], ['P', 'Q', 'R', '
S', 'U'], ['V', 'W', 'X', 'Y', 'Z']]

Plaintext Diagraphs
['AT', 'TA', 'CK', 'WI', 'TH', 'FU', 'LX', 'LF', 'OR', 'CE']
M A N T L
E B C D F
G H I K O
P Q R S U
V W X Y Z
Plaintext: ATTACKWITHFULLFORCE
Encrypted Text: NLLNDIXHAKOZNZFOIUDB
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher>
```

Encrypted Text – ‘NLLNDIXHAKOZNZFOIUDB’.

To decrypt:



```
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher> py .\decrypt.py
Playfair Cipher
Enter Encrypted Text: NLLNDIXHAKOZNZFOIUDB
Enter Key: MANTLE

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

Enc Text Diagraphs
['NL', 'LN', 'DI', 'XH', 'AK', 'OZ', 'NZ', 'FO', 'IU', 'DB']
Encrypted Text: NLLNDIXHAKOZNZFOIUDB
Plain Text: ATTACKWITHFULXLFORCE
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\PlayfairCipher>
```

We get our plaintext back.

Thus, we demonstrated the working of the code with examples.

Conclusion:

Thus, the Playfair Cipher algorithm was studied and demonstrated with the code. The algorithm is observed to be difficult to break. Bogus characters can come in the middle of the decrypted plaintext.