**ST. FRANCIS INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SECURITY LAB**

**Experiment – 2: Implementation of Playfair Cipher**

**Aim:** Write a program to implement Playfair Cipher Technique and understand cryptanalysis of the same.

**Objective:** After performing the experiment, the students will be able to –

* To understand the encryption and decryption using polyalphabetic substitution technique.
* To understand that secure encryption is not possible with a small key space.

**Lab objective mapped:** L502.1: Students should be able to apply the knowledge of symmetric cryptography to implement simple ciphers.

**Prerequisite:** Basic knowledge of cryptography.

**Requirements:** PYTHON

**Pre-Experiment Theory:**

**Playfair Cipher**:

1. WHAT IS PLAYFAIR CIPHER?

Playfair cipher is the first and best-known digraph substitution cipher, which uses the technique of symmetry encryption. Invented in 1854 by Charles Wheatstone, the cipher got its name from Lord Playfair, who promoted its use. Unlike single letters in a simple substitution cipher, the Playfair cipher technique encrypts digraphs or parts of letters.

The Playfair cipher is relatively fast and doesn’t require special equipment. British Forces used it for tactical purposes during World War I and the Second Boer War, and Australians utilized it during World War II. The primary use of the cipher was for protecting vital but non-critical secrets during actual combat. By the time the enemy cryptanalysts could decrypt the information, it was useless for them.

2. PLAYFAIR CIPHER’S RELEVANCE

The Playfair cipher gained significant popularity during World War I and II due to its complexity relative to other available ciphers of that era. Additionally, it required no special tools or equipment for encryption or decryption. However, with the advent of computers, the Playfair cipher quickly became obsolete, as computers could easily break the cipher within seconds using code-breaking techniques. Consequently, with the advancement of digital encryption methods, the Playfair cipher became an unacceptable means of encoding messages due to the increased risk of data being compromised. Therefore, the Playfair cipher cannot be recommended for use by business organizations.

3. ALGORITHM

**Generating the Key Square**

* The ‘key square’ is a 5×5 grid consisting of alphabets that helps encrypt the plain text.
* All these 25 letters should be unique.
* Since the grid can accommodate only 25 characters, there is no ‘J’ in this table. Any ‘J’ in the plaintext is replaced by ‘I’.
* Remove any characters or punctuation that are not present in the key square. Instead, spell out the numbers, punctuations, and any other non-alphabetic text.
* The key square will start with the key’s unique alphabet in the order of appearance, followed by the alphabet’s remaining characters in order.

Ref: <https://www.youtube.com/watch?v=U_J2xnhblPg>

**Rules for Playfair Cipher Encryption:**

* **Case I – Both the letters in the digraph are in the same row –**Consider the letters right of each alphabet. Thus, if one of the digraph letters is the rightmost alphabet in the grid, consider the leftmost alphabet in the same row.
* **Case II – Both the letters in the digraph are in the same column –**Consider the letters below each alphabet. Thus, if one of the digraph letters is the grid’s bottommost letter, consider the topmost alphabet in the same column.
* **Case III – Neither Case I or II is true** – Form a rectangle with the two letters in the digraph and consider the rectangle’s horizontal opposite corners.

Ref: <https://www.youtube.com/watch?v=O8MxWNfrzho>

Ref: <https://www.youtube.com/watch?v=66K1tplwYqg>

**Rules for Playfair Cipher Decryption:**

* **Case I – Both the letters in the digraph are in the same row –**Consider the letters left of each alphabet. Thus, if one of the digraph letters is the leftmost letter in the grid, consider the rightmost alphabet in the same row.
* **Case II – Both the letters in the digraph are in the same column –**Consider the letters above each alphabet. Thus, if one of the digraph letters is the topmost letter in the grid, consider the bottommost alphabet in the same column.
* **Case III – Neither Case I or II is true** – Form a rectangle with the two letters in the digraph and consider the rectangle’s horizontal opposite corners.

Ref: <https://www.youtube.com/watch?v=2PUInSjhxNs>

**Breaking of Playfair Cipher:**

Ref: <https://www.youtube.com/watch?v=n6ljkcJXVFY>

**Output:**

1. Attach complete program (with detailed comments explaining important steps) performing encryption and decryption of playfair cipher.
2. Display the key matrix created out of a given keyword.
3. Display the output of code for plaintext “the key is hidden under the door pad” with keyword “GUIDANCE” and its decryption.
4. Display the output of code (encryption as well as decryption) for plaintext given in post experiment exercise.

**Post Experimental Exercise-**

Solve the following exercise on the journal sheets.

1. Encrypt message using playfair cipher : “The world of Cryptography”

Key: DECRYPTION *[theoretical result and code output attached should match]*

**Conclusion:**

The basic features of classical cryptographic technique: Playfair Cipher are understood by implementing a code for encryption and decryption when key is known and also understood breaking of key when key space is very small by performing cryptanalysis of ciphertext.

**Output:**

1. **Attach complete program (with detailed comments explaining important steps) performing encryption and decryption of playfair cipher.**

**CODE:**

import re

def generate\_key\_square(keyword):

# Generate a 5x5 key square matrix for the Playfair cipher

#Normalize the keyword

keyword = keyword.upper()

keyword = keyword.replace('J', 'I')

#Create a unique list of characters for the key square

key\_square = []

seen = set()

for char in keyword:

if char not in seen and char.isalpha():

seen.add(char)

key\_square.append(char)

#Add the remaining characters from the alphabet

for char in 'ABCDEFGHIKLMNOPQRSTUVWXYZ':

if char not in seen:

key\_square.append(char)

#Create a 5x5 matrix

key\_square = [key\_square[i:i + 5] for i in range(0, 25, 5)]

return key\_square

def print\_key\_square(key\_square):

#Print the 5x5 key square matrix

print("Key Square:")

for row in key\_square:

print(" ".join(row))

print()

def format\_text(text):

#removing non-alphabetic characters and replacing 'J' with 'I'.

text = text.upper()

text = text.replace('J', 'I')

text = re.sub(r'[^A-Z]', '', text)

return text

def prepare\_digraphs(text):

#Prepare digraphs for encryption/decryption.

digraphs = []

i = 0

while i < len(text):

if i + 1 < len(text):

if text[i] == text[i + 1]:

digraphs.append(text[i] + 'X')

i += 1

else:

digraphs.append(text[i] + text[i + 1])

i += 2

else:

digraphs.append(text[i] + 'X')

i += 1

return digraphs

def find\_position(key\_square, char):

#Find the (row, col) position of a character in the key square.

for r, row in enumerate(key\_square):

if char in row:

return r, row.index(char)

return None

def encrypt\_digraph(key\_square, digraph):

#Encrypt a single digraph using the Playfair cipher rules.

row1, col1 = find\_position(key\_square, digraph[0])

row2, col2 = find\_position(key\_square, digraph[1])

if row1 == row2:

# Same row

return key\_square[row1][(col1 + 1) % 5] + key\_square[row2][(col2 + 1) % 5]

elif col1 == col2:

# Same column

return key\_square[(row1 + 1) % 5][col1] + key\_square[(row2 + 1) % 5][col2]

else:

# Rectangle

return key\_square[row1][col2] + key\_square[row2][col1]

def decrypt\_digraph(key\_square, digraph):

#Decrypt a single digraph using the Playfair cipher rules.

row1, col1 = find\_position(key\_square, digraph[0])

row2, col2 = find\_position(key\_square, digraph[1])

if row1 == row2:

# Same row

return key\_square[row1][(col1 - 1) % 5] + key\_square[row2][(col2 - 1) % 5]

elif col1 == col2:

# Same column

return key\_square[(row1 - 1) % 5][col1] + key\_square[(row2 - 1) % 5][col2]

else:

# Rectangle

return key\_square[row1][col2] + key\_square[row2][col1]

def playfair\_cipher(text, key, mode='encrypt'):

#Encrypt or decrypt the given text using the Playfair cipher.

key\_square = generate\_key\_square(key)

print\_key\_square(key\_square)

text = format\_text(text)

digraphs = prepare\_digraphs(text)

if mode == 'encrypt':

return ''.join(encrypt\_digraph(key\_square, digraph) for digraph in digraphs)

elif mode == 'decrypt':

return ''.join(decrypt\_digraph(key\_square, digraph) for digraph in digraphs)

def main():

#Main function to handle user input and perform encryption/decryption.

print("Welcome to the Playfair Cipher Program!")

# Take input from user

keyword = input("Enter the keyword for the Playfair cipher: ").strip()

plaintext = input("Enter the plaintext to encrypt: ").strip()

# Perform encryption

encrypted\_text = playfair\_cipher(plaintext, keyword, mode='encrypt')

print("\nEncrypted Text:", encrypted\_text)

# Perform decryption

decrypted\_text = playfair\_cipher(encrypted\_text, keyword, mode='decrypt')

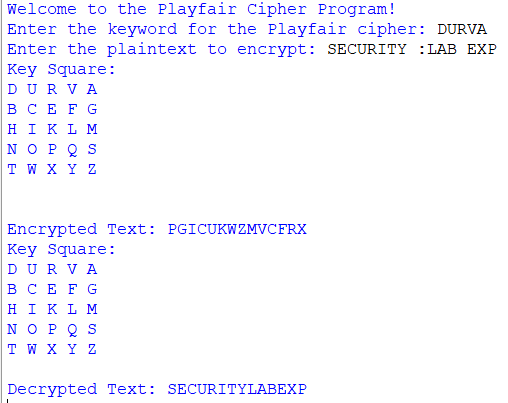
print("Decrypted Text:", decrypted\_text)

if \_\_name\_\_ == "\_\_main\_\_":

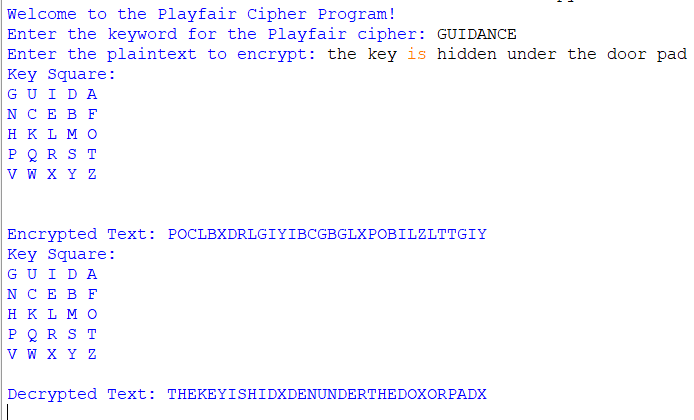
main()

1. **Display the key matrix created out of a given keyword.**

OUTPUT:

****

1. **Display the output of code for plaintext “the key is hidden under the door pad” with keyword “GUIDANCE” and its decryption.**

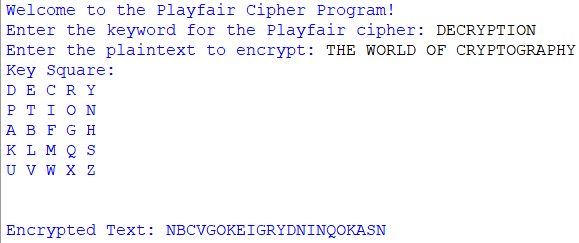
****

Here, the given plaintext is “THE KEY IS HIDDEN UNDER THE DOOR PAD”.

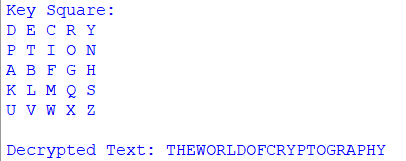
When we use the program to encrypt and decrypt the text using the keyword ‘GUIDANCE’. Initially a key square is made out of the given key word and the text is encrypted using that.

1. **Display the output of code (encryption as well as decryption) for plaintext given in post experiment exercise.**

**ENCRYPTION:**

****

**DECRYPTION:**

****