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.

Name: Vishal Rajesh Mahajan SL EXP 2
Class: TE IT A Roll No: 62

In Lab Exercise (Implementation of Playfair Cipher Technique):

```
# Function to Generate Key Matrix
def generate_key_mat(key):
    """
    Generate a 5x5 key matrix for Playfair cipher.

    Args:
        key (str): The key string used to generate the matrix.

    Returns:
        list: A 5x5 matrix represented as a list of lists.
    """
    # Remove spaces and convert key to uppercase and to replace 'J'
with 'I' as per Playfair cipher rules
    key = key.replace(" ", "").upper()
    key = key.replace("J", "I")

    # Define the alphabet without 'J'
    letter = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
    key_mat = []

    # Add characters from key and remaining letters of the alphabet
    for char in key + letter:
        if char not in key_mat:
            key_mat.append(char)

    # Arrange the characters into a 5x5 matrix and returning it
    key_mat = [key_mat[i:i+5] for i in range(0, 25, 5)]
    return key_mat
```

```python
# Function that returns the position of a character in the matrix
def get_pos(matrix, char):
    """
    Find the position (row, column) of a character in the key matrix.

    Args:
        matrix (list): The key matrix.
        char (str): The character to find.

    Returns:
        tuple: Row and column index of the character in the matrix.

    Raises:
        ValueError: If the character is not found in the matrix.
    """
    for i in range(5):
        for j in range(5):
            if matrix[i][j] == char:
                return i, j
    raise ValueError(f"Character {char} not found in matrix.")

# Function to preprocess the text
def preprocess_text(text):
    """
    Preprocess the text by removing spaces, converting to uppercase,
    replacing 'J' with 'I', and adding padding if necessary.

    Args:
        text (str): The input text to preprocess.

    Returns:
        str: The processed text ready for encryption or decryption.
    """
    text = text.replace(" ", "").upper().replace("J", "I")

    # Add padding if the length of the text is odd
    if len(text) % 2 != 0:
        text += 'X'
    return text
```

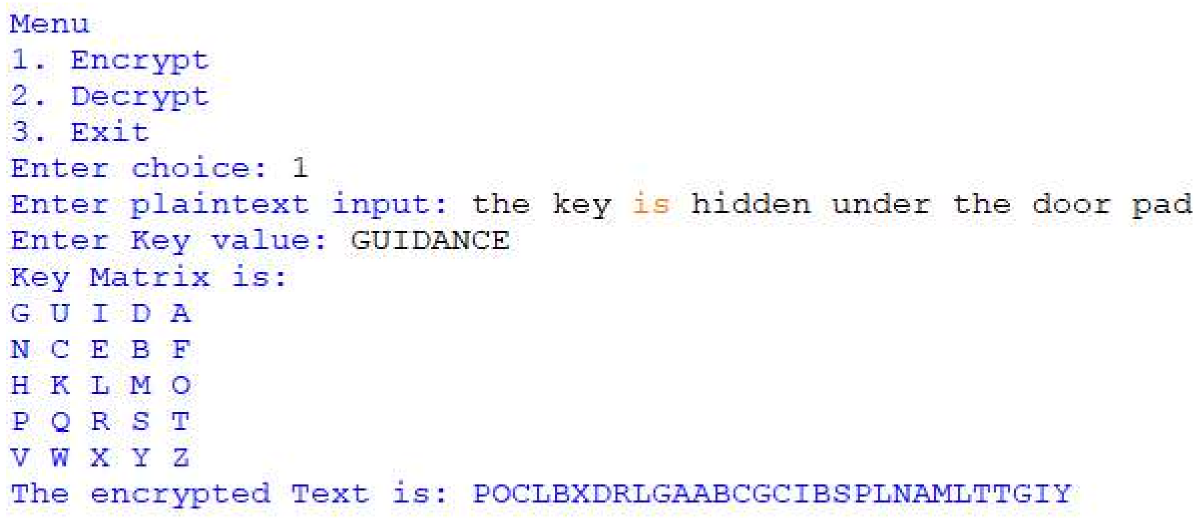
```python
# Function to encrypt the plaintext using the key matrix
def encrypt(plain_text, key_mat):
    """
    Encrypt the plaintext using Playfair cipher.

    Args:
        plain_text (str): The plaintext to encrypt.
        key_mat (list): The 5x5 key matrix.

    Returns:
        str: The encrypted text.
    """
    plain_text = preprocess_text(plain_text)
    encr_text = ""
    for i in range(0, len(plain_text) - 1, 2):
        char1, char2 = plain_text[i], plain_text[i+1]
        row1, col1 = get_pos(key_mat, char1)
        row2, col2 = get_pos(key_mat, char2)
        # Same row: replace with characters to the right
        if row1 == row2:
            encr_text += key_mat[row1][(col1 + 1) % 5]
            encr_text += key_mat[row2][(col2 + 1) % 5]
        # Same column: replace with characters below
        elif col1 == col2:
            encr_text += key_mat[(row1 + 1) % 5][col1]
            encr_text += key_mat[(row2 + 1) % 5][col2]
        # Rectangle: swap columns
        else:
            encr_text += key_mat[row1][col2]
            encr_text += key_mat[row2][col1]
    return encr_text

# Function to decrypt the encrypted text using the key matrix
def decrypt(encr_text, key_mat):
    """
    Decrypt the encrypted text using Playfair cipher.

    Args:
        encr_text (str): The encrypted text to decrypt.
        key_mat (list): The 5x5 key matrix.
    Returns:
        str: The decrypted text.
    """
    decr_text = ""
    for i in range(0, len(encr_text) - 1, 2):
        char1, char2 = encr_text[i], encr_text[i+1]
        row1, col1 = get_pos(key_mat, char1)
        row2, col2 = get_pos(key_mat, char2)
        # Same row: replace with characters to the left
        if row1 == row2:
            decr_text += key_mat[row1][(col1 - 1) % 5]
            decr_text += key_mat[row2][(col2 - 1) % 5]
        # Same column: replace with characters above
        elif col1 == col2:
            decr_text += key_mat[(row1 - 1) % 5][col1]
            decr_text += key_mat[(row2 - 1) % 5][col2]
        # Rectangle: swap columns
        else:
            decr_text += key_mat[row1][col2]
            decr_text += key_mat[row2][col1]
    # Remove padding character 'X' if present
    return decr_text.rstrip('X')

# Function to print the matrix
def display_matrix(key_mat):
    """
    Display the key matrix in a 5x5 grid format.

    Args:
        key_mat (list): The 5x5 key matrix.
    """
    for row in key_mat:
        print(" ".join(row))
```

```
# Menu Driven Program Loop
while True:
    print("\nMenu")
    print("1. Encrypt")
    print("2. Decrypt")
    print("3. Exit")
    # Get user choice
    n = int(input("Enter choice: "))

    # Encryption
    if n == 1:
        text = input("Enter plaintext input: ")
        key = input("Enter Key value: ")
        matrix = generate_key_mat(key)
        print("Key Matrix is:")
        display_matrix(matrix)
        encr_text = encrypt(text, matrix)
        print("The encrypted Text is:", encr_text.upper())

    # Decryption
    elif n == 2:
        text = input("Enter Encrypted Text: ")
        key = input("Enter Key value: ")
        matrix = generate_key_mat(key)
        print("Key Matrix is:")
        display_matrix(matrix)
        decr_text = decrypt(text, matrix)
        print("The decrypted Text is:", decr_text.lower())

    # Exit
    elif n == 3:
        break

    # Invalid Choice
    else:
        print("Input is invalid")
```

Output:

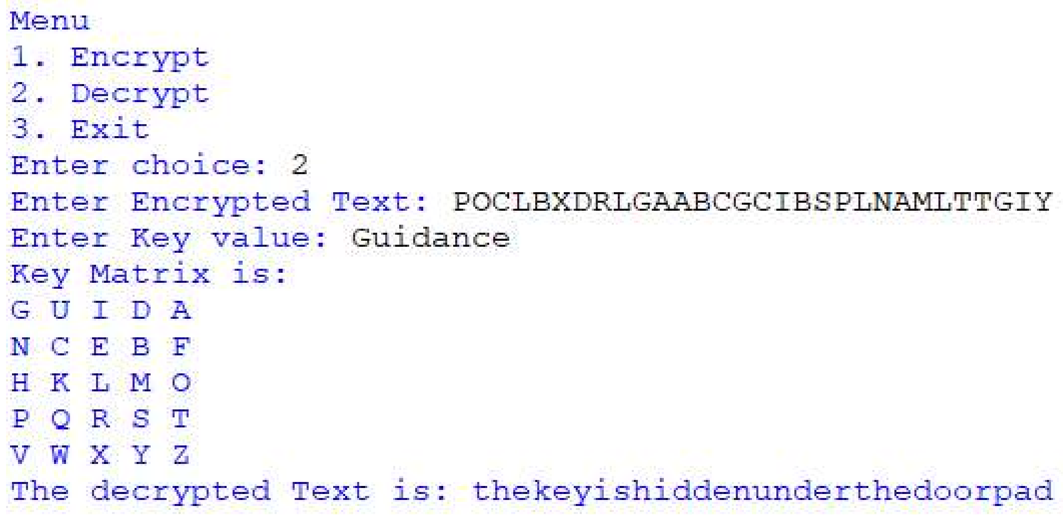
Plaintext: the key is hidden under the door pad
key: GUIDANCE

Encryption:



```
Menu
1. Encrypt
2. Decrypt
3. Exit
Enter choice: 1
Enter plaintext input: the key is hidden under the door pad
Enter Key value: GUIDANCE
Key Matrix is:
G U I D A
N C E B F
H K L M O
P Q R S T
V W X Y Z
The encrypted Text is: POCLBXDRLGAABCGCIBSPLNAMLTTGIY
```

Decryption:

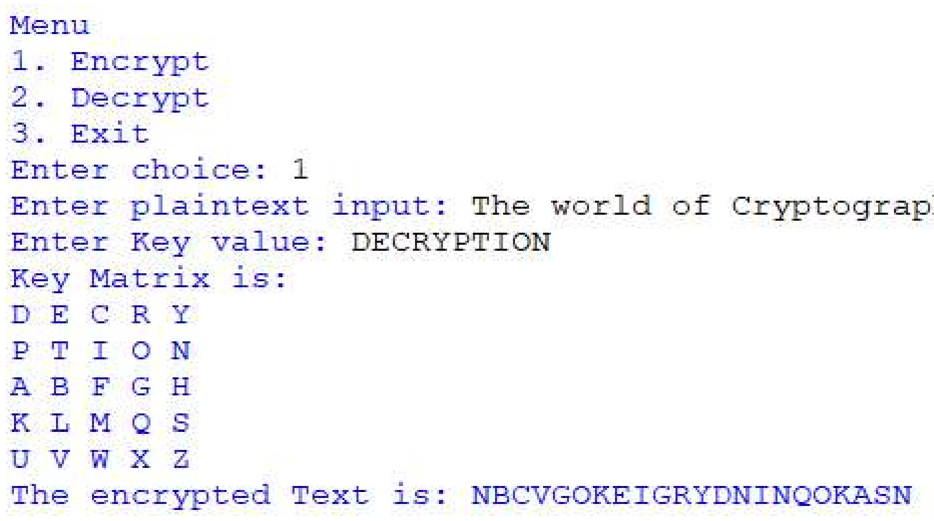


```
Menu
1. Encrypt
2. Decrypt
3. Exit
Enter choice: 2
Enter Encrypted Text: POCLBXDRLGAABCGCIBSPLNAMLTTGIY
Enter Key value: Guidance
Key Matrix is:
G U I D A
N C E B F
H K L M O
P Q R S T
V W X Y Z
The decrypted Text is: thekeyishiddenunderthedoorpad
```

POST LAB EXERCISE:

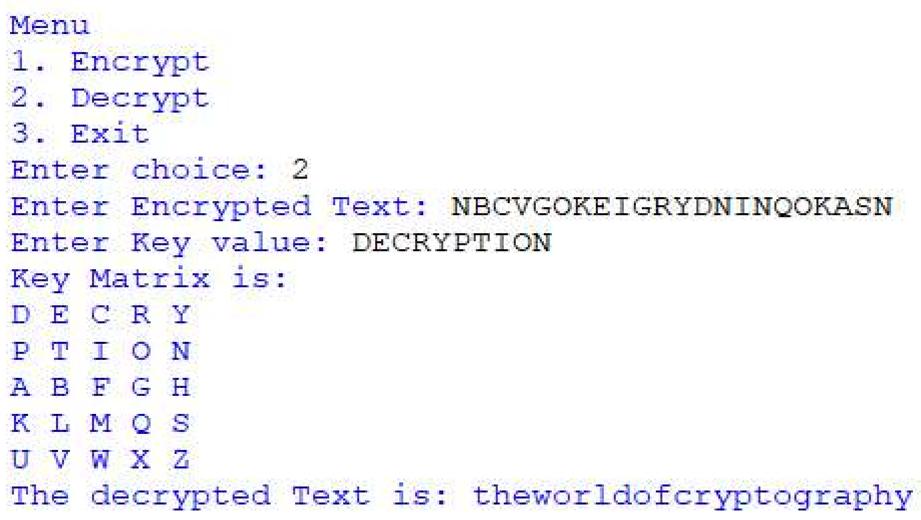
Encrypt message using playfair cipher : “The world of Cryptography”
Key: DECRYPTION

Encryption:



```
Menu
1. Encrypt
2. Decrypt
3. Exit
Enter choice: 1
Enter plaintext input: The world of Cryptography
Enter Key value: DECRYPTION
Key Matrix is:
D E C R Y
P T I O N
A B F G H
K L M Q S
U V W X Z
The encrypted Text is: NBCVGOKEIGRYDNINQOKASN
```

Decryption:



```
Menu
1. Encrypt
2. Decrypt
3. Exit
Enter choice: 2
Enter Encrypted Text: NBCVGOKEIGRYDNINQOKASN
Enter Key value: DECRYPTION
Key Matrix is:
D E C R Y
P T I O N
A B F G H
K L M Q S
U V W X Z
The decrypted Text is: theworldofcryptography
```