**SCHOOL OF COMPUTER SCIENCE ENGINEERING AND APPLICATION**

**BCA TY SEM VI**

**SUBJECT NAME: INFORMATION SECURITY**

**LAB- ASSIGNMENT NO 3**

**AIM: IMPLEMENT PLAY FAIR ALGORITHM. (ENCRYPTION AND DECRYPTION)**

**PLAY FAIR:**

The Playfair cipher is a classical substitution cipher that encrypts pairs of letters at a time, instead of individual letters like in simpler substitution ciphers. It's named after its creator, Sir Charles Wheatstone. The cipher uses a 5x5 grid of letters, typically constructed from a keyword.

**Example:**

Suppose we want to encrypt the plaintext "HELLO" using the keyword "PLAYFAIR".

Step 1: Generate Key Table

mathematica

Copy code

P L A Y F

I R B C D

E G H K M

N O Q S T

U V W X Z

Step 2: Prepare Plaintext

Break "HELLO" into digraphs: "HE", "LX", "LO"

Step 3: Encrypt

"HE" becomes "PK".

"LX" becomes "BK".

"LO" becomes "QM".

So, the encrypted ciphertext is "PKBKBQM".

**ALGORITHM:**

**Encryption Algorithm:**

**1)Key Table Generation**:

Start with a keyword

Construct a 5x5 grid (key table) of letters.

Fill the grid with the unique letters of the keyword, omitting duplicates and the letter 'J'.

Fill the remaining cells with the remaining letters of the alphabet, omitting 'J'.

**2)Preparing the Plaintext:**

Break the plaintext into digraphs (pairs of letters).

If there's an odd number of letters, add a filler letter (commonly 'X') at the end.

**3)Encrypting Digraphs:**

For each digraph:

Find the positions of the two letters in the key table.

If the letters are in the same row, replace them with the letters to their immediate right (wrapping around if necessary).

If the letters are in the same column, replace them with the letters immediately below them (wrapping around if necessary).

If the letters form a rectangle, replace each letter with the letter at the opposite corner of the rectangle.

If none of the above conditions apply, form a rectangle with the two letters and replace each letter with the letter at the opposite corner of the rectangle.

**4)Output:**

The resulting digraphs form the encrypted ciphertext.

**Decryption Algorithm:**

**1)Key Table Generation**:

Use the same method as encryption to generate the key table from the keyword.

**2)Preparing the Ciphertext**:

Break the ciphertext into digraphs.

**3)Decrypting Digraphs**:

For each digraph:

Find the positions of the two letters in the key table.

Use the reverse process of encryption to determine the original plaintext letters.

**4)Output**:

The resulting digraphs form the decrypted plaintext.

**PSUEDO CODE IN PYTHON FOR PLAY FAIR ENCRYPTION AND DECRYPTION WITH OUTPUT:**

def generate\_key\_table(keyword):

    # Initialize key table with keyword

    key\_table = [['' for \_ in range(5)] for \_ in range(5)]

    keyword = keyword.replace('J', 'I')  # Replace 'J' with 'I' to avoid confusion

    keyword += 'ABCDEFGHIKLMNOPQRSTUVWXYZ'  # Append remaining alphabet without 'J'

    keyword = ''.join(dict.fromkeys(keyword))  # Remove duplicates while preserving order

    # Fill key table with keyword

    k = 0

    for i in range(5):

        for j in range(5):

            key\_table[i][j] = keyword[k]

            k += 1

    return key\_table

def find\_position(key\_table, letter):

    # Find position of letter in key table

    for i in range(5):

        for j in range(5):

            if key\_table[i][j] == letter:

                return i, j

def encrypt(plaintext, key\_table):

    # Prepare plaintext

    plaintext = plaintext.upper().replace('J', 'I')  # Convert to uppercase and replace 'J' with 'I'

    plaintext = [plaintext[i:i+2] for i in range(0, len(plaintext), 2)]  # Break into digraphs

    ciphertext = ''

    # Encrypt each digraph

    for digraph in plaintext:

        # Find positions of letters in key table

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

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

        # Encrypt digraph

        if row1 == row2:

            ciphertext += key\_table[row1][(col1 + 1) % 5] + key\_table[row2][(col2 + 1) % 5]

        elif col1 == col2:

            ciphertext += key\_table[(row1 + 1) % 5][col1] + key\_table[(row2 + 1) % 5][col2]

        else:

            ciphertext += key\_table[row1][col2] + key\_table[row2][col1]

    return ciphertext

def decrypt(ciphertext, key\_table):

    # Prepare ciphertext

    ciphertext = ciphertext.upper()

    ciphertext = [ciphertext[i:i+2] for i in range(0, len(ciphertext), 2)]  # Break into digraphs

    plaintext = ''

    # Decrypt each digraph

    for digraph in ciphertext:

        # Find positions of letters in key table

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

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

        # Decrypt digraph

        if row1 == row2:

            plaintext += key\_table[row1][(col1 - 1) % 5] + key\_table[row2][(col2 - 1) % 5]

        elif col1 == col2:

            plaintext += key\_table[(row1 - 1) % 5][col1] + key\_table[(row2 - 1) % 5][col2]

        else:

            plaintext += key\_table[row1][col2] + key\_table[row2][col1]

    return plaintext

# Example usage

keyword = "PLAYFAIR"

plaintext = input("enter any plain text:")

key\_table = generate\_key\_table(keyword)

encrypted\_text = encrypt(plaintext, key\_table)

decrypted\_text = decrypt(encrypted\_text, key\_table)

print("Plaintext:", plaintext)

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

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

PS C:\Users\T480> & C:/Users/T480/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/T480/Desktop/sem\_6/IS/assignments/lab assignments/PLAYFAIR.PY"

enter any plain text:omkarpatil

Plaintext: omkarpatil

Encrypted Text : TGHYILFQRP

Decrypted Text: OMKARPATIL