Rashtriya Raksha University

School of Information Technology, Artificial Intelligence & Cyber Security (SITAICS)

At- Lavad, Dahegam, Gandhinagar, Gujarat-382305

Practical File

(Introduction to Cryptography)

Name:	Sarthak Sanay
Enrollment No:	230031101611051
Subject Name:	Introduction to Cryptography
Subject Code:	G4A19ITC
Program:	B.Tech CSE (with specialization in Cyber Security)
Year:	2nd year (Semester-IV)

This is certifying that Mr. Sarthak Sanay has satisfactorily completed all experiments in the practical work prescribed by SITAICS in the ITC laboratory.

Dr. Ashish Revar

SUBJECT INCHARGE

PRACTICAL - 2

AIM: TO IMPLEMENT PLAYFAIR CIPHER

BRIEF :-

The Playfair cipher, invented by Charles Wheatstone and popularized by Lord Playfair in the mid-19th century, is an early digraph substitution cipher. Instead of substituting single letters, it operates on pairs of letters (“digraphs”), using a 5×5 key matrix (merging I/J) to determine substitutions. By encrypting in pairs and using positional rules (same row, same column, or rectangle corners), it obscures letter frequencies better than simple monoalphabetic ciphers. While it was once used by militaries for its relative ease of use and improved security, it is now considered insecure by modern standards, but remains a valuable pedagogical example of polygraphic substitution.

Because it processes two letters at once, the Playfair cipher resists simple frequency analysis of individual letters and introduces complexity with filler characters (commonly “X”) when duplicates appear or an odd-length digraph remains. Its manual-friendly grid makes it suitable for field use in pre-computer eras, yet its fixed 5x5 matrix and predictable rules leave it vulnerable to modern cryptanalysis.

ALGORITHM / PSEUDOCODE :-

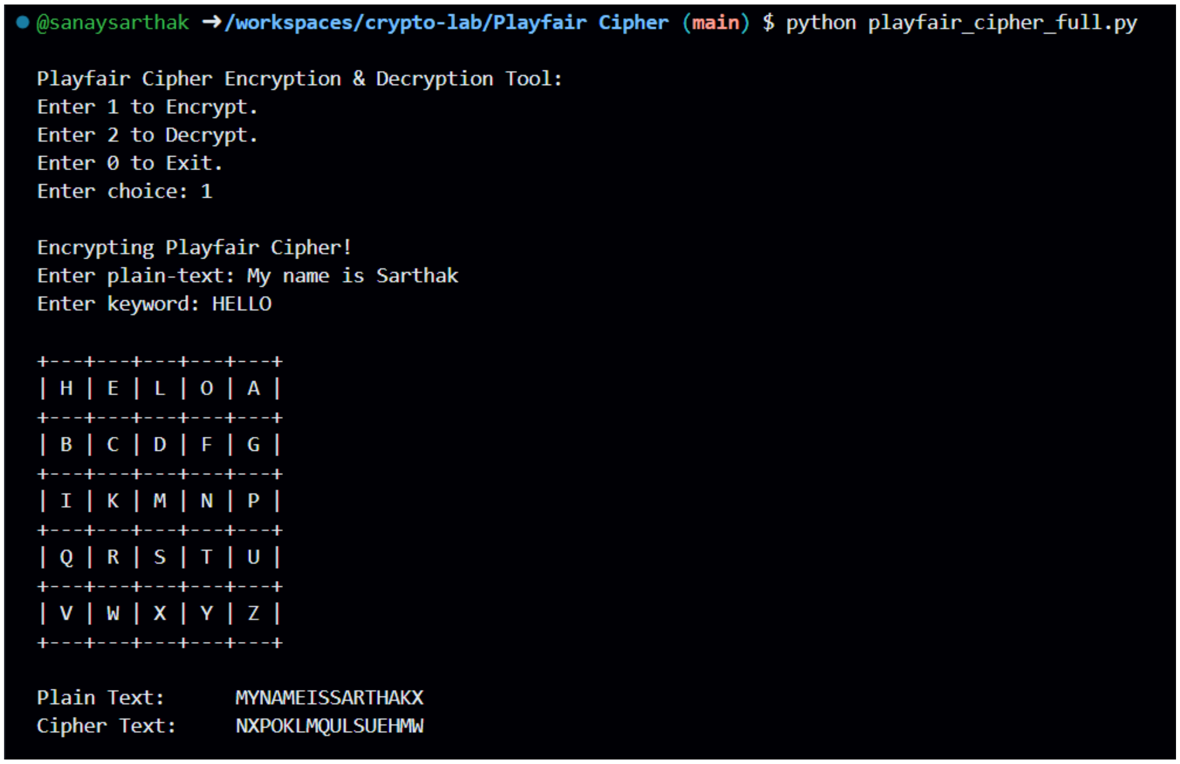
```
1. Build 5×5 Key Matrix:
   - Uppercase keyword, replace J→I, remove duplicates.
   - Fill matrix left→right, top→bottom with keyword letters.
   - Fill remaining cells with A to Z (skip J and used letters).

2. Prepare Plaintext:
   - Uppercase, replace J→I, remove spaces.
   - Insert 'X' between repeated letters in each pair.
   - If the length is odd, append 'X'.

3. Encrypt Digraphs:
   For each pair (A, B) in prepared text:
     - Find (r1,c1) for A, (r2,c2) for B in the matrix.
     - If same row: replace with letters to their right (wrap around).
     - Else if same column: replace with letters below (wrap around).
     - Else: replace with letters at the opposite corners of the rectangle.

4. Decrypt Digraphs:
   For each pair (C, D) in cipher text:
     - Find positions as above.
     - If the same row: replace with letters to their left (wrap around).
     - Else if same column: replace with letters above (wrap around).
     - Else: swap columns as in encryption.

5. Menu Loop:
   - 1: Encrypt → input plaintext & keyword → show matrix → output cipher.
   - 2: Decrypt → input ciphertext & keyword → show matrix → output plaintext.
   - 0: Exit.
```

CODE :-

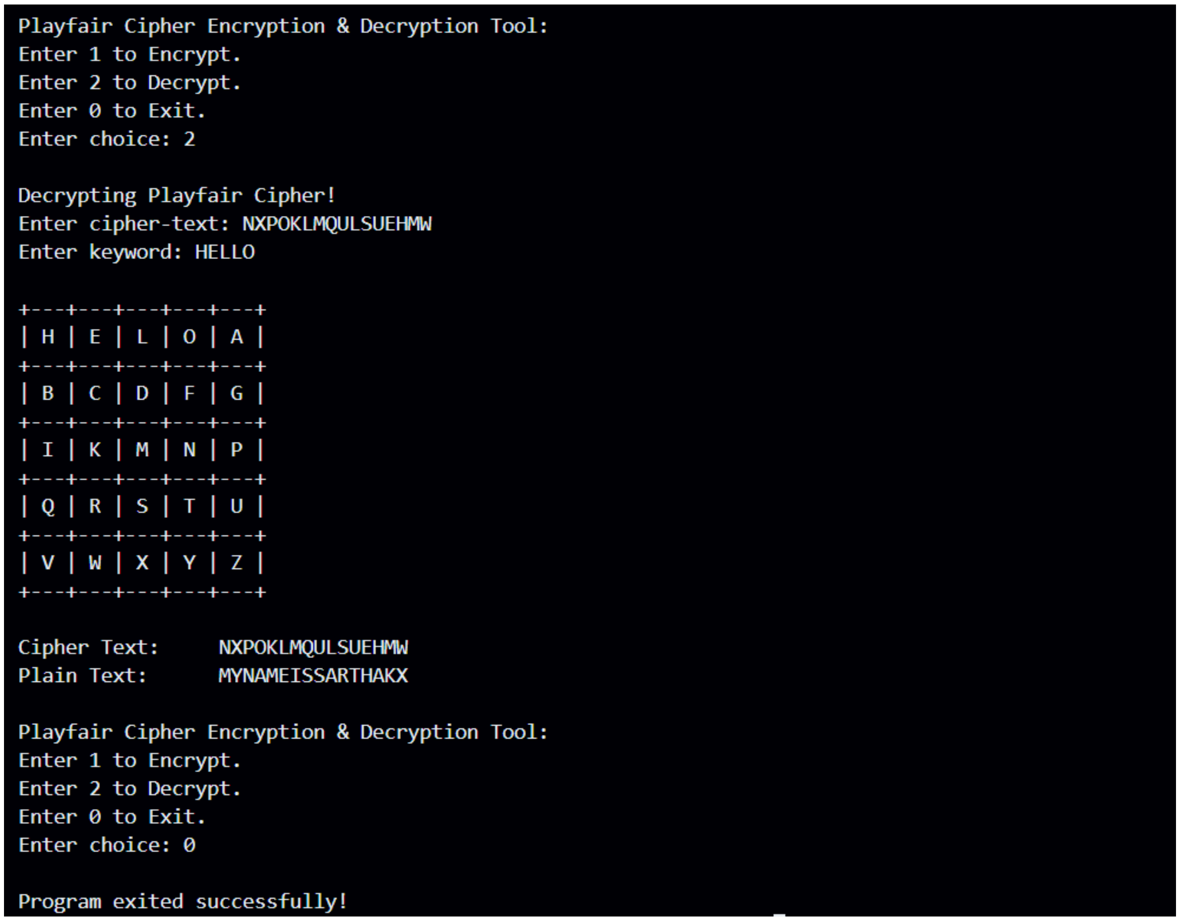
```
def create_matrix(keyword):

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

    matrix = [[], [], [], [], []]
    used_char = []
    alphabets = []  # ['A', 'B', 'C', ... , 'Z']
    for i in range(65, 91):
        if chr(i) == 'J':
            continue
        alphabets.append(chr(i))

    i, j = 0, 0
    for char in keyword:
        if j == 5:
            j = 0
            i += 1
        if char in used_char:
            continue
        if char == 'J':
            matrix[i].insert(j, char)
            used_char.append('I')
            j += 1
        else:
            matrix[i].insert(j, char)
            used_char.append(char)
            j += 1

    for char in alphabets:
        if j == 5:
            j = 0
            i += 1
        if char in used_char:
            continue
        if char == 'J':
            matrix[i].insert(j, char)
            used_char.append('I')
            j += 1
        else:
            matrix[i].insert(j, char)
            used_char.append(char)
            j += 1

    return matrix
```

```
def display_matrix(matrix):
    print("\n+---+---+---+---+---+")
    for row in matrix:
        row_string = "|"
        for num in row:
            row_string += f" {num} " + "|"
        print(row_string)
        print("+---+---+---+---+---+")


def prepare_text_encrypt(plain_text):
    plain_text = plain_text.upper().replace('J', 'I').replace(" ", "")

    i = 0
    while i < len(plain_text):
        substring = plain_text[i:i+2]
        if len(substring) == 2 and substring[0] == substring[1]:
            plain_text = plain_text[:i+1] + "X" + plain_text[i+1:]
        i += 2
        if i+1 == len(plain_text):
            plain_text += "X"
            break
    return plain_text


def encrypt_playfair(plain_text, keyword):
    matrix = create_matrix(keyword)
    display_matrix(matrix)

    plain_text = prepare_text_encrypt(plain_text)

    i = 0
    cipher_text = ""

    while i < len(plain_text):
        char1 = plain_text[i]
        char2 = plain_text[i+1]

        row1 = col1 = row2 = col2 = -1
        for r in range(5):
            for c in range(5):
                if matrix[r][c] == char1:
                    row1, col1 = r, c
                if matrix[r][c] == char2:
                    row2, col2 = r, c

        if row1 == row2:
            cipher_text += matrix[row1][(col1 + 1) % 5]
            cipher_text += matrix[row2][(col2 + 1) % 5]
        elif col1 == col2:
            cipher_text += matrix[(row1 + 1) % 5][col1]
            cipher_text += matrix[(row2 + 1) % 5][col2]
        else:
            cipher_text += matrix[row1][col2]
            cipher_text += matrix[row2][col1]

        i += 2

    return plain_text, cipher_text


def decrypt_playfair(cipher_text, keyword):
    matrix = create_matrix(keyword)
    display_matrix(matrix)

    cipher_text = cipher_text.upper().replace(" ", "")

    i = 0
    plain_text = ""

    while i < len(cipher_text):
        char1 = cipher_text[i]
        char2 = cipher_text[i+1]

        row1 = col1 = row2 = col2 = -1
        for r in range(5):
            for c in range(5):
                if matrix[r][c] == char1:
                    row1, col1 = r, c
                if matrix[r][c] == char2:
                    row2, col2 = r, c

        if row1 == row2:
            plain_text += matrix[row1][(col1 - 1) % 5]
            plain_text += matrix[row2][(col2 - 1) % 5]
        elif col1 == col2:
            plain_text += matrix[(row1 - 1) % 5][col1]
            plain_text += matrix[(row2 - 1) % 5][col2]
        else:
            plain_text += matrix[row1][col2]
            plain_text += matrix[row2][col1]

        i += 2

    return plain_text


choice = None
while choice != 0:
    print("\nPlayfair Cipher Encryption & Decryption Tool:")
    print("Enter 1 to Encrypt.")
    print("Enter 2 to Decrypt.")
    print("Enter 0 to Exit.")
    try:
        choice = int(input("Enter choice: "))
    except ValueError:
        print("Invalid input. Please enter 1, 2, or 0.")
        continue

    if choice == 1:
        print("\nEncrypting Playfair Cipher!")
        pt = input("Enter plain-text: ")
        key = input("Enter keyword: ")
        prepared, ct = encrypt_playfair(pt, key)
        print("\nPlain Text: \t", prepared)
        print("Cipher Text:\t", ct)
    elif choice == 2:
        print("\nDecrypting Playfair Cipher!")
        ct_input = input("Enter cipher-text: ")
        key = input("Enter keyword: ")
        pt_out = decrypt_playfair(ct_input, key)
        print("\nCipher Text:\t", ct_input.replace(' ', '').upper())
        print("Plain Text: \t", pt_out)
    elif choice == 0:
        print("\nProgram exited successfully!")
    else:
        print("\nEnter correct choice!")
```

OUTPUT :-



```
● @sanaysarthak ➜ /workspaces/crypto-lab/Playfair Cipher (main) $ python playfair_cipher_full.py

  Playfair Cipher Encryption & Decryption Tool:
  Enter 1 to Encrypt.
  Enter 2 to Decrypt.
  Enter 0 to Exit.
  Enter choice: 1

  Encrypting Playfair Cipher!
  Enter plain-text: My name is Sarthak
  Enter keyword: HELLO

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

  Plain Text:      MYNAMEISSARTHAKX
  Cipher Text:     NXPOKLMQULSUEHMW
```



```
Playfair Cipher Encryption & Decryption Tool:
Enter 1 to Encrypt.
Enter 2 to Decrypt.
Enter 0 to Exit.
Enter choice: 2

Decrypting Playfair Cipher!
Enter cipher-text: NXPOKLMQULSUEHMW
Enter keyword: HELLO

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

Cipher Text:      NXPOKLMQULSUEHMW
Plain Text:       MYNAMEISSARTHAKX

Playfair Cipher Encryption & Decryption Tool:
Enter 1 to Encrypt.
Enter 2 to Decrypt.
Enter 0 to Exit.
Enter choice: 0

Program exited successfully!
```