ID : VU4F2223028 Experiment No.:02 Security Lab

**Aim:** Implementation of Playfair Cipher.

**Program:**

def generate\_matrix(key):

matrix = []

seen = set()

key = key.upper().replace('J', 'I') # Replace J with I

key = ''.join(filter(str.isalpha, key)) # Remove non-alphabetic characters

for char in key:

if char not in seen:

seen.add(char)

matrix.append(char)

# Add remaining letters of the alphabet

for char in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':

if char not in seen and char != 'J':

matrix.append(char)

# Generate 5x5 matrix

return matrix

def get\_coordinates(char, matrix):

index = matrix.index(char)

return index // 5, index % 5

def get\_character(row, col, matrix):

return matrix[row \* 5 + col]

def preprocess\_text(text):

text = text.upper().replace('J', 'I') # Replace J with I

text = ''.join(filter(str.isalpha, text)) # Remove non-alphabetic characters

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 encrypt(text, key):

matrix = generate\_matrix(key)

digraphs = preprocess\_text(text)

ciphertext = ''

for pair in digraphs:

r1, c1 = get\_coordinates(pair[0], matrix)

r2, c2 = get\_coordinates(pair[1], matrix)

if r1 == r2:

ciphertext += get\_character(r1, (c1 + 1) % 5, matrix)

ciphertext += get\_character(r2, (c2 + 1) % 5, matrix)

elif c1 == c2:

ciphertext += get\_character((r1 + 1) % 5, c1, matrix)

ciphertext += get\_character((r2 + 1) % 5, c2, matrix)

else:

ciphertext += get\_character(r1, c2, matrix)

ciphertext += get\_character(r2, c1, matrix)

return ciphertext

def decrypt(text, key):

matrix = generate\_matrix(key)

digraphs = preprocess\_text(text)

plaintext = ''

for pair in digraphs:

r1, c1 = get\_coordinates(pair[0], matrix)

r2, c2 = get\_coordinates(pair[1], matrix)

if r1 == r2:

plaintext += get\_character(r1, (c1 + 4) % 5, matrix)

plaintext += get\_character(r2, (c2 + 4) % 5, matrix)

elif c1 == c2:

plaintext += get\_character((r1 + 4) % 5, c1, matrix)

plaintext += get\_character((r2 + 4) % 5, c2, matrix)

else:

plaintext += get\_character(r1, c2, matrix)

plaintext += get\_character(r2, c1, matrix)

return plaintext.rstrip('X') # Remove trailing Xs

def print\_matrix(matrix):

print('Playfair Cipher Matrix:')

for i in range(5):

print(' '.join(matrix[i \* 5:(i + 1) \* 5]))

def main():

key = input('Enter the key for encryption/decryption: ')

plaintext = input('Enter the plaintext to be encrypted: ')

matrix = generate\_matrix(key) # Generate matrix for printing

encrypted = encrypt(plaintext, key)

decrypted = decrypt(encrypted, key)

print(f'Key: {key}')

print(f'Plaintext: {plaintext}')

print(f'Encrypted: {encrypted}')

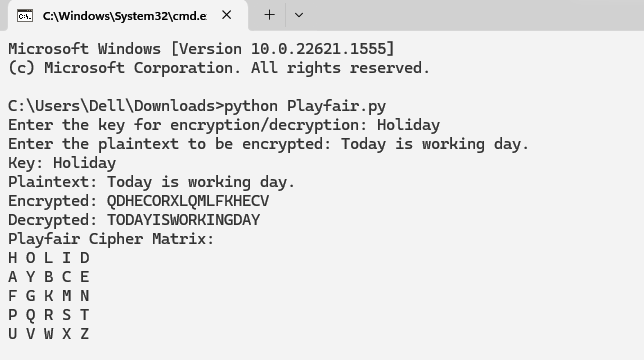
print(f'Decrypted: {decrypted}')

print\_matrix(matrix) # Print the matrix

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Output:**



**Conclusion:**

We have suscessfully implemented Playfair Cipher.