**DEPARTMENT OF INFORMATION TECHNOLOGY**

**COURSE CODE: DJS22ITL504**

**DATE:16/8/24**

**COURSE NAME: Cryptography and Network Security Laboratory CLASS: T. Y. BTech**

**Name: Anish Sharma ROLL NO: I011**

**SAP ID: 60003220045**

**EXPERIMENT NO. 2**

**CO/LO:** Design secure system using appropriate security mechanism

**AIM / OBJECTIVE:**

a. Implementation of Playfair Cipher on Alphanumeric data.

**THEORY / CONCEPT / ALGORITHM:**

* The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In playfair cipher unlike [traditional cipher](https://www.geeksforgeeks.org/caesar-cipher/) we encrypt a pair of alphabets(digraphs) instead of a single alphabet.

* The Algorithm consists of 2 steps:
  1. **Generate the key Square (5×5):**

The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table

(as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by

I. The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

* 1. **Algorithm to encrypt the plain text:** The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

Pair cannot be made with same letter. Break the letter in single and add a bogus letter to the previous letter.

If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter

Rules for Encryption:

* If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom).
* If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).
* If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

**SOURCE CODE:**

def generate\_matrix(key): # Define the alphabet and digits

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

# Create a list for the key and remove duplicates

key = "".join(sorted(set(key), key=lambda x: key.index(x)))

# Combine the key with the remaining characters

combined = key + "".join([c for c in alphabet if c not in key])

# Create a 6x6 matrix

matrix = [list(combined[i:i + 6]) for i in range(0, 36, 6)] return matrix

def find\_position(matrix, char):

for i, row in enumerate(matrix): if char in row:

return i, row.index(char)

return None

def playfair\_encrypt(plaintext, matrix): plaintext = plaintext.upper().replace(" ", "")

# Pad with '4' for odd length plaintext

if len(plaintext) % 2 != 0:

plaintext += '4'

ciphertext = ""

# Process pairs of characters for i in range(0, len(plaintext), 2): a, b = plaintext[i], plaintext[i + 1] row\_a, col\_a = find\_position(matrix, a) row\_b, col\_b = find\_position(matrix, b)

if row\_a == row\_b: # Same row

ciphertext += matrix[row\_a][(col\_a + 1) % 6] ciphertext += matrix[row\_b][(col\_b + 1) % 6]

elif col\_a == col\_b: # Same column

ciphertext += matrix[(row\_a + 1) % 6][col\_a] ciphertext += matrix[(row\_b + 1) % 6][col\_b]

else: # Rectangle

ciphertext += matrix[row\_a][col\_b] ciphertext += matrix[row\_b][col\_a]

return ciphertext

def playfair\_decrypt(ciphertext, matrix): plaintext = ""

# Process pairs of characters

for i in range(0, len(ciphertext), 2): a, b = ciphertext[i], ciphertext[i + 1] row\_a, col\_a = find\_position(matrix, a) row\_b, col\_b = find\_position(matrix, b)

if row\_a == row\_b: # Same row

plaintext += matrix[row\_a][(col\_a - 1) % 6] plaintext += matrix[row\_b][(col\_b - 1) % 6]

elif col\_a == col\_b: # Same column

plaintext += matrix[(row\_a - 1) % 6][col\_a] plaintext += matrix[(row\_b - 1) % 6][col\_b]

else: # Rectangle

plaintext += matrix[row\_a][col\_b] plaintext += matrix[row\_b][col\_a]

# Remove padding if it exists if plaintext[-1] == '4':

plaintext = plaintext[:-1]

return plaintext

# Main function if name == " main ": key = "ANALOGY"

plaintext = "INFORMATION123"

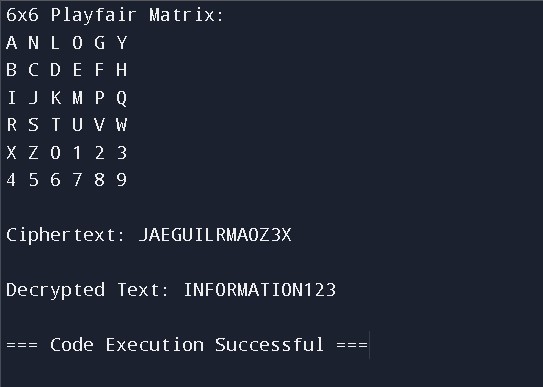
matrix = generate\_matrix(key)

print("6x6 Playfair Matrix:") for row in matrix:

print(" ".join(row))

ciphertext = playfair\_encrypt(plaintext, matrix) print(f"\nCiphertext: {ciphertext}")

decrypted\_text = playfair\_decrypt(ciphertext, matrix) print(f"\nDecrypted Text: {decrypted\_text}")



import numpy as np from PIL import Image

def generate\_key\_matrix():

# Create a 256x256 matrix with unique values (0-255) matrix = np.arange(256) np.random.shuffle(matrix)

matrix = matrix.reshape((16, 16)) return matrix

def playfair\_cipher(pixel\_value1, pixel\_value2, matrix, decrypt=False):

# Find positions of the pixel values in the matrix pos1 = np.argwhere(matrix == pixel\_value1)[0] pos2 = np.argwhere(matrix == pixel\_value2)[0]

# Implement Playfair-like rules for substitution (encryption or decryption) if pos1[0] == pos2[0]: # Same row if decrypt:

new\_value1 = matrix[pos1[0], (pos1[1] - 1) % 16] new\_value2 = matrix[pos2[0], (pos2[1] - 1) % 16] else:

new\_value1 = matrix[pos1[0], (pos1[1] + 1) % 16] new\_value2 = matrix[pos2[0], (pos2[1] + 1) % 16]

elif pos1[1] == pos2[1]: # Same column if decrypt:

new\_value1 = matrix[(pos1[0] - 1) % 16, pos1[1]] new\_value2 = matrix[(pos2[0] - 1) % 16, pos2[1]] else:

new\_value1 = matrix[(pos1[0] + 1) % 16, pos1[1]] new\_value2 = matrix[(pos2[0] + 1) % 16, pos2[1]] else:

# Rectangle swap

new\_value1 = matrix[pos1[0], pos2[1]] new\_value2 = matrix[pos2[0], pos1[1]]

return new\_value1, new\_value2

def encrypt\_image(image, matrix, decrypt=False):

encrypted\_image = np.zeros\_like(image) height, width, channels = image.shape

for ch in range(channels):

for i in range(0, height, 2):

for j in range(0, width, 2):

if i + 1 < height and j + 1 < width: pixel\_value1 = image[i, j, ch] pixel\_value2 = image[i + 1, j + 1, ch]

new\_value1, new\_value2 = playfair\_cipher(pixel\_value1, pixel\_value2, matrix,

decrypt=decrypt)

encrypted\_image[i, j, ch] = new\_value1 encrypted\_image[i + 1, j + 1, ch] = new\_value2

else:

encrypted\_image[i, j, ch] = image[i, j, ch] # Leave the last pixel as is if no pair

return encrypted\_image

def main(): # Load an image

image = Image.open('image.png')

image = np.array(image)

# Generate the key matrix (ensure this is the same as used for encryption) matrix = generate\_key\_matrix()

# Encrypt the image

encrypted\_image = encrypt\_image(image, matrix)

# Save the encrypted image

encrypted\_image\_pil = Image.fromarray(encrypted\_image) encrypted\_image\_pil.save('encrypted\_image.png')

# Decrypt the image

decrypted\_image = encrypt\_image(encrypted\_image, matrix, decrypt=True)

# Save the decrypted image

decrypted\_image\_pil = Image.fromarray(decrypted\_image) decrypted\_image\_pil.save('decrypted\_image.png')

if name == " main ": main()

IMAGE USED:



ENCRPTED IMAGE:



**DECRYPTED IMAGE**



**CONCLUSION:** We learned about playfair cipher for text and image and implemented it in python.