Experiment - 4

Aim → Implementation of symmetric cryptographic ~~algori~~ process for secured communication of information in computer networking.

Objective 1 → An overview on Playfair and Vigenere cryptographic algorithm.

Playfair Cipher →

- Type → Diagraph substitution cipher
- Key → 5×5 matrix based on a keyword (I & J treated as one)
- Encryption → i) Same row → Replace with the letter to the right.
  ii) Same column → Replace with the letter below.
  iii) Rectangle → Swap with opposite corners.
- Security → More secure than simple substitution but vulnerable to digraph analysis.

Vigenère Cipher →

- Type → Polyalphabetic substitution cipher
- Key : Repeating keyword determines shifts.
- Encryption → i) Uses a Vigenère table to shift letters based on the keyword.
  ii) Formula → $C_i = (P_i + K_i) \bmod 26$
- Security → Stronger than Playfair but breakable using frequency analysis.

Objective 2 → Execution of Playfair Algorithm for encryption and decryption of text message.

Python Program →

```
import random
import strings
def generate_key_matrix(key):
    key = "".join(sorted(set(key), key = key.index))
    key = key.replace('J', 'I')
    alphabet = "ABCDEFGHIKLMNOPQRSTUVXYZ"
    matrix = key + alphabet
    matrix = "".join(sorted(set(matrix), key = matrix.index))
    key_matrix = [matrix[i:i+5] for i in range(0, len(matrix),5)]
    return key_matrix
```

Name: _____ Regd No: _____

```
def create_digraphs(plaintext):
    plaintext = ''.join(filter(str.isalpha, plaintext)).upper()
    plaintext = plaintext.replace('J', 'I')
    digraphs = []
    i = 0
    while i < len(plaintext):
        if i+1 < len(plaintext) and plaintext[i] == plaintext[i+1]:
            digraphs.append(plaintext[i] + 'X')
            i += 1
        else:
            digraphs.append(plaintext[i:i+2])
            i += 2
    return digraphs

def find_positions(letter, key_matrix):
    for i in range(5):
        for j in range(5):
            if key_matrix[i][j] == letter:
                return i, j

def encrypt_digraph(digraph, key_matrix):
    r1, c1 = find_position(digraph[0], key_matrix)
    r2, c2 = find_position(digraph[1], key_matrix)
    if r1 == r2:
        return key_matrix[r1][(c1+1)%5] + key_matrix[r2][(c2+1)%5]
    elif c1 == c2:
        return key_matrix[(r1+1)%5][c1] + key_matrix[(r2+1)%5][c2]

def decrypt_digraph(digraph, key_matrix):
    r1, c1 = find_position(digraph[0], key_matrix)
    r2, c2 = find_position(digraph[1], key_matrix)
    if r1 == r2:
        return key_matrix[r1][(c1-1)%5] + key_matrix[r2][(c2-1)%5]
    elif c1 == c2:
```

```
            return key_matrix[(r1-1)%5][c1] + key_matrix[(r2-1)%5][c2]
    else:
        return key_matrix[r1][c2] + key_matrix[r2][c1]

def playfair_encrypt(plaintext, key):
    key_matrix = generate_key_matrix(key)
    digraphs = create_digraphs(plaintext)
    ciphertext = ''.join([encrypt_digraph(d, key_matrix) for d in digraphs])
    return ciphertext.

def playfair_decrypt(ciphertext, key):
    key_matrix = generate_key_matrix(key)
    digraphs = create_digraphs[ciphertext[i:i+2] for i in range(0, len(ciphertext), 2)]
    plaintext = ''.join([decrypt_digraph(d, key_matrix) for d in digraphs])
    return plaintext

def generate_random_plaintext(length):
    return ''.join(random.choices(string.ascii_uppercase, k=length))

key = "KEYWORD"
random_plaintext = generate_random_plaintext(random.randint(5, 10))
encrypted = playfair_encrypt(random_plaintext, key)
decrypted = playfair_decrypt(encrypted, key)
print(f"Generated Plaintext: {random_plaintext}")
print(f"Encrypted: {encrypted}")
print(f"Decrypted: {decrypted}")
```

Output →

Generated Plaintext: CYSUQC
Encrypted: AONZSB
Decrypted: CYSUQC

Objective 3 → Execution of Vignére algorithm for encryption and decryption of text message.

Python Program →

```
def Vignére_encrypt (plaintext, keyword):
    plaintext = plaintext.replace (" ", "").upper()
    keyword = keyword.upper()
    extended_keyword = (keyword *(len(plaintext) // len(keyword))) +
                       keyword[:len(plaintext) % len(keyword)]
    ciphertext = []
    for p, k in zip(plaintext, extended_keyword):
        encrypted_letter = chr(((ord(p)-65 + ord(k)-65) % 26)+65)
        ciphertext.append(encrypted_letter)
    return ''.join(ciphertext)
def vignerce_decrypt(ciphertext, keyword):
    ciphertext = ciphertext.replace(" ", "").upper()
    keyword = keyword.upper()
    extended_keyword = (keyword *(len(ciphertext) // len(keyword))) + keyword
                       [:len(ciphertext) % len(keyword)]
    plaintext = []
    for c, k in zip(ciphertext, extended_keyword):
        decrypted_letter = chr((( ord(c)-65 - (ord(k)-65)) % 26)+65)
        plaintext.append(decrypted_letter)
    return ''.join(plaintext)

keyword = "KEY"
plaintext = "HELLO VIGNERE CIPHER"
encrypted = vignere_encrypt(plaintext, keyword)
decrypted = vignere_decrypt(encrypted, keyword)
print(f"Plaintext: {plaintext}")
print(f"Encrypted: {encrypted}")
```

print(f"Decrypted: {decrypted}")

Output → Plaintext → HELLO VIGNERE CIPHER

Encrypted → RIJVSTSKCXIPOGGSZLCB

Decrypted → HELLOVIGNERECIPHER

Conclusion → The Playfair and Vignère ciphers are classic cryptographic techniques that introduced complexity beyond simple substitution ciphers. While Playfair uses digraph substitution for increased security against frequency analysis, Vignère's polyalphabetic approach offers stronger protection by varying shifts based on keyword.

Exercise →

1) Encrypt "ALL IS WELL" using Vignère Cipher with the keyword "CAKE".

Ans → $C_i = (P_i + K_i) \bmod 26$

Plaintext	A L L I S W E L L
Keyword	C A K E C A K E C
Shift Value	2 0 10 4 2 0 10 4 2
Encrypted	C L V M U W O P N

Ciphertext → CLVMUWOPN

2) Decrypt "DCTXS WWVVVAMZ" using Vignere cipher with the keyword "BEST".

Ans → $P_i = (C_i - k_i + 26) \bmod 26$

Ciphertext	D C T X S W W V V V A M Z
Keyword	B E S T B E S T B E S T B
Shift Value	1 4 18 19 1 4 18 19 1 4 18 19 1
Decrypted	C Y B E R S E C U R I T Y

Decrypted → CYBERSECURITY

Name: _____ Regd No: _____

3) Encrypt "INSTRUMENTS" using Playfair ciphering with key word "MONARCHY" assuming the letters I & J share the the same cell in key matrix.

Ans → Playfair Matrix →

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

Plaintext → IN ST RU ME NT SX

Using play fair rules → IN → GA
ST → TL
RU → MZ
ME → CL
NT → RQ
SX → XA

Encrypted → GA TL MZ CL RQ XA

4) Decrypt "HBTIDBHKMO" using Playfair Ciphering with keyword "DIAMOND" assuming the letters I & J are in the same cell in key matrix.

Ans → Playfair Matrix →

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

Cipher
~~Plain~~text → HB TI DB HK MO

Using Playfair rule → HB → BI
TI → RM
DB → IN
HK → GH
MO → AM

Decrypted → BIRMINIGHAM

Name: _____ Regd No: _____