CNS: EXPERIMENT - 2

Aim: Write a program in Java or Python to perform Cryptanalysis or decoding of Playfair Cipher**.**

Theory:

The Playfair Cipher is a symmetric encryption technique that was designed to improve upon the security limitations of the simple Caesar cipher and other substitution ciphers. It employs a polygraphic substitution method, which means that blocks of letters are encrypted at a time. The key feature of the Playfair Cipher is the use of a 5x5 matrix (also called a key square) containing a mixed alphabet, where each letter appears only once. The matrix is usually generated using a keyword provided by the user.

Encryption:

Key Square Generation: Generate a key square (matrix) from a given keyword. This matrix will be used as the reference for encrypting and decrypting messages. The matrix consists of a unique combination of letters from the keyword and the remaining letters of the alphabet (excluding 'J' or replacing 'J' with 'I').

Formatting the Message: The plaintext message is prepared for encryption. It is usually converted to uppercase, and non-alphabetic characters are removed. If there are repeating letters, they are separated by a filler, often 'X'. If the message length is odd, an 'X' might be added at the end to make it even.

Letter Pairing: The formatted message is divided into pairs of letters.

Encryption Process: For each letter pair:

- If the letters are in the same row of the key square, each letter is replaced with the letter to its right (circularly).
- If the letters are in the same column of the key square, each letter is replaced with the letter below it (circularly).
- If the letters are neither in the same row nor in the same column, they form the vertices of a rectangle. Replace each letter with the letter located at the opposite corner of the rectangle formed by the two letters.

Encrypted Message: The resulting encrypted pairs of letters make up the encrypted message.

Decryption:

Key Square Generation: Same as in encryption, generate the key square from the provided keyword.

Decryption Process: For each letter pair in the encrypted message:

- If the letters are in the same row, replace each letter with the letter to its left (circularly).
- If the letters are in the same column, replace each letter with the letter above it (circularly).
- If the letters form a rectangle, replace each letter with the letter located at the opposite corner of the rectangle.

Decrypted Message: The resulting decrypted pairs of letters make up the decrypted message.

Answer in brief for the below questions:

1. **What is the primary weakness of playfair cipher?**
   The primary weakness of the Playfair cipher is that it encrypts digraphs (pairs of letters) rather than single letters, which makes it vulnerable to frequency analysis based on digraphs. Let us look at each weakness of the playfair cipher in detail:
   - Digraph Frequency Analysis:
     Since the Playfair cipher encrypts letter pairs, an attacker can use the frequency of common digraphs in the English language (like "TH", "HE", "IN", etc.) to break the cipher, especially if a large amount of ciphertext is available.
   - Preserves Letter Frequencies (to some extent):
     Although it hides single-letter frequency, patterns in digraph frequencies are still somewhat preserved, giving cryptanalysts clues.
   - Not Resistant to Known Plaintext Attacks:
     If an attacker knows even a small amount of the plaintext and corresponding ciphertext, it's easier to deduce parts of the key square.
   - Fixed Pattern Rules:
     The encryption rules (same row, column, rectangle swap) are rigid and create predictable transformations, which can be reverse-engineered.
   - No Diffusion Across Whole Message:
     Each pair of letters is encrypted independently, so there is no diffusion across multiple characters — a small change in plaintext affects only one digraph.

2. **How does the Playfair cipher differ from simpler substitution ciphers?**
   The Playfair cipher differs from simpler substitution ciphers in the following key ways:
   - Substitution of Digraphs Instead of Single Letters: A simple substitution cipher replaces each individual letter with another letter. The Playfair cipher substitutes pairs of letters (digraphs), making it more complex and harder to break with frequency analysis.
   - More Complex Encryption Rules: The Playfair cipher uses a 5x5 matrix filled with a keyword and the rest of the alphabet (I and J are usually combined). The encryption depends on the relative positions of the letter pair in the matrix, applying different rules if they are in the same row, column, or form a rectangle.
   - Preserves Letter Frequencies Less Clearly: Because it encrypts digraphs, the frequency of single letters is obscured, making frequency analysis less effective.

- No Letter Encrypts to Itself: In simple substitution, a letter might map to itself. In Playfair, due to the digraph rules, no letter in a pair can remain unchanged.
- More Resistant to Simple Attacks: The increased complexity and use of digraphs make the Playfair cipher more secure than monoalphabetic substitution ciphers, though it's still not strong by modern standards.

3. **Encrypt the <Your_Name> with playfair cipher using Key "Power". Explain the steps involved in it. Decrypt the encrypted text back to Plain text.**

Construct the Playfair Matrix
Key = "POWER" (remove duplicates and combine I/J as "I").
Matrix filling order: Key letters first, then remaining alphabet (A-Z excluding J, skipping duplicates).
Matrix:

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

Encryption Steps

Step 2: Prepare Plaintext
"SHAKTI" → Split into digraphs: SH, AK, TI (no duplicates or odd length).

Step 3: Apply Playfair Rules

Digraph SH:
- S: Row 3, Col 3 (Matrix[3][3])
- H: Row 2, Col 1 (Matrix[2][1])

Different row/column → Rectangle Rule: Output: NK

Digraph AK:
- A: Row 1, Col 0 (Matrix[1][0])
- K: Row 2, Col 3 (Matrix[2][3])

Different row/column → Rectangle Rule: Output: DG

Digraph TI:
- T: Row 3, Col 4 (Matrix[3][4])
- I: Row 2, Col 2 (Matrix[2][2])

Different row/column → Rectangle Rule: Output: QL

Step 4: Final Ciphertext

Combine digraphs: NK + DG + QL → NKDGQL.

Decryption Steps

Step 1: Prepare Ciphertext

"NKDGQL" → Split into digraphs: NK, DG, QL.

Step 2: Apply Playfair Rules (Reverse)
Digraph NK:

N: Row 3, Col 1 (Matrix[3][1])

K: Row 2, Col 3 (Matrix[2][3])

Different row/column → Rectangle Rule: Output: SH

Digraph DG:

- D: Row 1, Col 3 (Matrix[1][3])
- G: Row 2, Col 0 (Matrix[2][0])

Different row/column → Rectangle Rule: Output: AK

Digraph QL:

- Q: Row 3, Col 2 (Matrix[3][2])
- L: Row 2, Col 4 (Matrix[2][4])

Different row/column → Rectangle Rule: Output: TI

Step 3: Final Plaintext
Combine digraphs: SH + AK + TI → SHAKTI.

4. **What is a digraph in the context of the Playfair cipher?**
   In the context of the Playfair cipher, a digraph is a pair of letters that are encrypted together as a single unit. The plaintext message is first divided into digraphs (two-letter groups). If a pair contains the same letter (like "LL"), a filler letter such as "X" is inserted between them to separate the repeated letters (e.g., "LX"). If the message has an odd number of letters, an "X" is also added at the end to complete the final pair.
   For example, the word "HELLO" would be divided into the digraphs: HE, LX, and OX. These digraphs are then encrypted using the Playfair cipher rules. This method makes the Playfair cipher more secure than simple monoalphabetic substitution ciphers, as it hides letter frequencies more effectively.

5. **How many possible keys does the playfair cipher have? Ignore the fact that some keys might produce identical encryption results. Express your answer as an approximate power of 2**.

   The Playfair cipher uses a 5×5 matrix of letters (25 letters, typically omitting 'J'), and the key is determined by the arrangement of these letters in the matrix. We calculate the number of ways to arrange 25 distinct letters:

   $$\text{Number of possible keys} = 25! \approx 2^{83.7} \approx 2^{84}$$

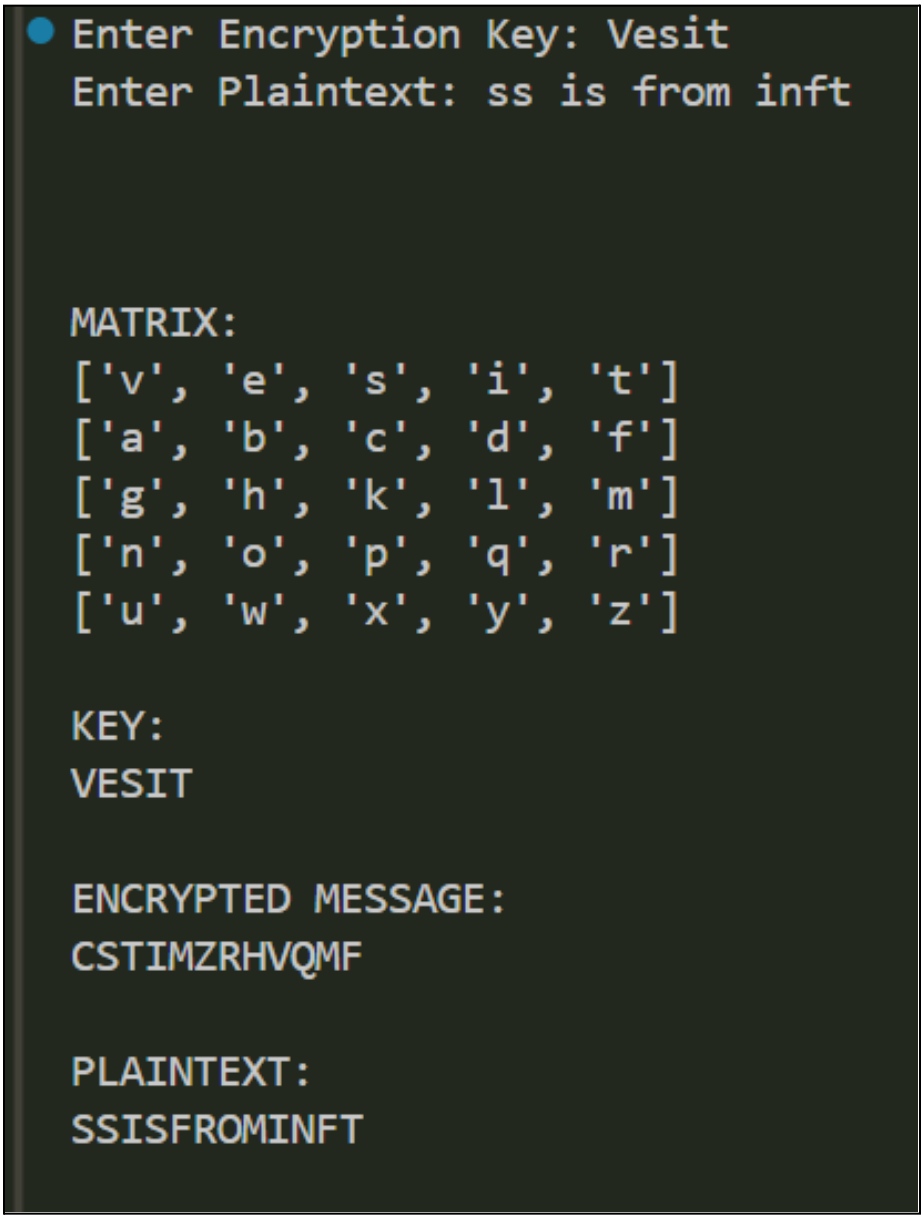
   Hence, the Playfair cipher has approximately 2^{84} possible keys.

CODE

```
key = input("Enter Encryption Key: ").lower()
plaintext = input("Enter Plaintext: ").lower()

plaintext = plaintext.replace(" ", "").replace("j", "i")
key = key.replace("j", "i")
alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i/j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

matrix = list(key)
keySet = set(key)

for char in alpha:
    if char == 'i/j':
        if 'i' not in keySet:
            matrix.append(char)
    elif char not in keySet:
        matrix.append(char)

row1 = matrix[:5]
row2 = matrix[5:10]
row3 = matrix[10:15]
row4 = matrix[15:20]
row5 = matrix[20:25]
matrix_rows = [row1, row2, row3, row4, row5]

if len(plaintext) % 2 != 0:
    plaintext += "x"

digraphs = []
for i in range(0, len(plaintext), 2):
    if i+1 < len(plaintext):
        if plaintext[i] == plaintext[i+1]:
            digraphs.append(plaintext[i] + "x")
        else:
            digraphs.append(plaintext[i] + plaintext[i+1])
def findPosition(char):
    char = 'i' if char == 'j' else char
    for i in range(5):
        for j in range(5):
            cell = matrix_rows[i][j]
            if cell == 'i/j' and char == 'i':
                return (i, j)
```

```
            elif cell == char:
                return (i, j)
    return None
cipherText = ""
for digraph in digraphs:
    a, b = digraph[0], digraph[1]
    rowA, colA = findPosition(a)
    rowB, colB = findPosition(b)

    if rowA == rowB:
        cipherText += matrix_rows[rowA][(colA+1)%5] + matrix_rows[rowB][(colB+1)%5]
    elif colA == colB:
        cipherText += matrix_rows[(rowA+1)%5][colA] + matrix_rows[(rowB+1)%5][colB]
    else:
        cipherText += matrix_rows[rowA][colB] + matrix_rows[rowB][colA]
print()
print()
print()
print("MATRIX:")
for i in matrix_rows:
    print(i)
print()
print("KEY:")
print(key.upper())
print()
print("ENCRYPTED MESSAGE:")
print(cipherText.upper())
print()
print("PLAINTEXT:")
print(plaintext.upper())
print()
print()
print()
```

OUTPUT


Enter Encryption Key: Vesit
Enter Plaintext: ss is from inft
MATRIX:
['v', 'e', 's', 'i', 't']
['a', 'b', 'c', 'd', 'f']
['g', 'h', 'k', 'l', 'm']
['n', 'o', 'p', 'q', 'r']
['u', 'w', 'x', 'y', 'z']
KEY:
VESIT
ENCRYPTED MESSAGE:
CSTIMZRHVQMF
PLAINTEXT:
SSISFROMINFT