SUB: Information Security

AY 2023-24 (Semester-V)

Experiment No: 5

60009210105 **Amitesh Sawarkar**

D 12

Aim: Design and implement Encryption and Decryption Algorithm using Play fair Cipher.

Theory:

1. Playfair Cipher

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 we encrypt a pair of alphabets(digraphs) instead of a single alphabet.

It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.

Playfair cipher is an encryption algorithm to encrypt or encode a message. It is the same as a traditional cipher. The only difference is that it encrypts a **digraph** (a pair of two letters) instead of a single letter.

It initially creates a key-table of 5*5 matrix. The matrix contains alphabets that act as the key for encryption of the plaintext. Note that any alphabet should not be repeated. Another point to note that there are 26 alphabets and we have only 25 blocks to put a letter inside it. Therefore, one letter is excess so, a letter will be omitted (usually J) from the matrix. Nevertheless, the plaintext contains J, then **J** is replaced by **I**. It means treat I and J as the same letter, accordingly.

Since Playfair cipher encrypts the message **digraph by digraph**. Therefore, the Playfair cipher is an example of a **digraph substitution cipher**.

SUB: Information Security

Example:

1) **Plaintext: ATTACK**
Keyword: MONARCHY

SUB: Information Security

```
def construct_playfair_matrix(key):
    key = key.replace(" ", "").upper()
    matrix = [['' for _ in range(5)] for _ in range(5)]
    alphabet = 'ABCDEFGHIKLMNOPQRSTUVWXYZ'

    key_set = set()
    row, col = 0, 0

    for char in key:
        if char not in key_set:
            matrix[row][col] = char
            key_set.add(char)
            col += 1
            if col == 5:
                col = 0
                row += 1

    for char in alphabet:
        if char not in key_set:
            matrix[row][col] = char
            col += 1
            if col == 5:
                col = 0
                row += 1

    return matrix
```

SUB: Information Security

```
def print_playfair_matrix(matrix):
    for row in matrix:
        print(" ".join(row))

def preprocess_text(text):
    text = text.replace(" ", "").upper()
    text_pairs = [text[i:i+2] for i in range(0, len(text), 2)]

    for i in range(len(text_pairs)):
        if len(text_pairs[i]) == 1:
            text_pairs[i] += 'X'

    return text_pairs
```

```
def encrypt(plaintext, key):
    matrix = construct_playfair_matrix(key)
    plaintext = preprocess_text(plaintext)

    ciphertext = []
    for pair in plaintext:
        a, b = pair[0], pair[1]
        a_row, a_col, b_row, b_col = 0, 0, 0, 0

        for i in range(5):
            for j in range(5):
                if matrix[i][j] == a:
                    a_row, a_col = i, j
                if matrix[i][j] == b:
                    b_row, b_col = i, j

        if a_row == b_row:
            ciphertext.append(matrix[a_row][(a_col + 1) % 5] + matrix[b_row][(b_col + 1) % 5])
        elif a_col == b_col:
            ciphertext.append(matrix[(a_row + 1) % 5][a_col] + matrix[(b_row + 1) % 5][b_col])
        else:
            ciphertext.append(matrix[a_row][b_col] + matrix[b_row][a_col])

    return "".join(ciphertext)
```

SUB: Information Security

```
def decrypt(ciphertext, key):
    matrix = construct_playfair_matrix(key)
    ciphertext = preprocess_text(ciphertext)

    plaintext = []
    for pair in ciphertext:
        a, b = pair[0], pair[1]
        a_row, a_col, b_row, b_col = 0, 0, 0, 0

        for i in range(5):
            for j in range(5):
                if matrix[i][j] == a:
                    a_row, a_col = i, j
                if matrix[i][j] == b:
                    b_row, b_col = i, j

        if a_row == b_row:
            plaintext.append(matrix[a_row][(a_col - 1) % 5] + matrix[b_row][(b_col - 1) % 5])
        elif a_col == b_col:
            plaintext.append(matrix[(a_row - 1) % 5][a_col] + matrix[(b_row - 1) % 5][b_col])
        else:
            plaintext.append(matrix[a_row][b_col] + matrix[b_row][a_col])

    return "".join(plaintext)
```

```
def main():
    key = input("Enter the key: ")
    matrix = construct_playfair_matrix(key)
    print("Playfair Matrix:")
    print_playfair_matrix(matrix)

    plaintext = input("Enter the plaintext: ")
    ciphertext = encrypt(plaintext, key)
    print("Encrypted text:", ciphertext)

    decrypted_text = decrypt(ciphertext, key)
    print("Decrypted text:", decrypted_text)

if __name__ == "__main__":
    main()
```

SUB: Information Security

```
Enter the key: MONARCHY
Playfair Matrix:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
Enter the plaintext: ATTACK
Encrypted text: RSSRDE
Decrypted text: ATTACK
```

Conclusion:

The Playfair cipher experiment involved encoding and decoding messages using a Playfair cipher, a symmetric key substitution cipher. This experiment aimed to assess its effectiveness in encrypting and decrypting messages.

The conclusion of the experiment is that the Playfair cipher is a relatively secure method for simple encryption, making it suitable for basic communication security. However, it may not withstand modern cryptographic attacks, and its security largely depends on the choice of the key and key management. Despite its historical significance, the Playfair cipher is not recommended for highly sensitive or critical data in contemporary security contexts.