Ishita Hardasmalani

C14-2103058

EXPERIMENT NO. 3

Aim: Implementation of Playfair Cipher

Theory :

The Playfair Cipher is a cryptographic technique that encrypts pairs of letters in plaintext to produce ciphertext. It is a more advanced substitution cipher providing a higher level of security. The Playfair Cipher was widely used before the advent of modern encryption methods. The best-known multiple-letter encryption cipher is the Playfair, which treats digraphs in the plaintext as single units and translates these units into ciphertext digraphs.

1. Key Generation:

- The Playfair Cipher uses a key table, often a 5x5 grid containing a key phrase or keyword without repeating letters. If a letter is repeated in the key phrase, only the first occurrence is used, and the rest of the alphabet is filled in the remaining spaces.

- The key table is used to create digraphs (pairs of letters) for encryption and decryption.

Key Table Generation:

Key Table Rules: The key table is a 5x5 grid, traditionally filled with a keyword or key phrase without repeating letters. If the keyword has repeating letters, only the first occurrence is used, and the rest of the alphabet is filled in the remaining spaces.

- The letters 'I' and 'J' are often treated as the same letter in the key table.

2. Filling the Key Table: Begin by filling the table with the letters of the keyword. Then, add the remaining letters of the alphabet (excluding 'J' if 'I' is used). The resulting key table is used for both encryption and decryption.

Handling Odd-Length Messages:

1. Adding a Dummy Letter: When encrypting, if a digraph contains identical letters or if the message has an odd length, a dummy letter (usually 'X') is inserted between them before applying the encryption rules.

- For example, the word "HI" becomes "HI" in the Playfair Cipher, but "HIRED" becomes "HI RE DX."

2. Handling Dummy Letters During Decryption:

- In decryption, if a dummy letter was added between two identical letters, it is removed from the decrypted result.

2. Encryption: To encrypt a message, the plaintext is divided into pairs of letters (digraphs).

-If the two letters of a digraph are in the same row of the key table, they are replaced with the letters to their immediate right, wrapping around to the leftmost position if needed.

- If the two letters of a digraph are in the same column of the key table, they are replaced with the letters immediately below, wrapping around to the top if needed.

- If the two letters are in different rows and columns, they form a rectangle in the key table. In this case, each letter is replaced by the letter in the same row but in the column of the other letter.

- If a digraph contains identical letters, a dummy letter (usually 'X') is inserted between them before applying the encryption rules.

3. Decryption: Decryption in the Playfair Cipher involves applying the inverse of the encryption rules.

- If the two letters of a digraph are in the same row, they are replaced with the letters to their immediate left.

- If the two letters of a digraph are in the same column, they are replaced with the letters immediately above.

- If the two letters are in different rows and columns, they form a rectangle, and each letter is replaced by the letter in the same row but in the column of the other letter.

- If a dummy letter was added during encryption, it is removed from the decrypted result.

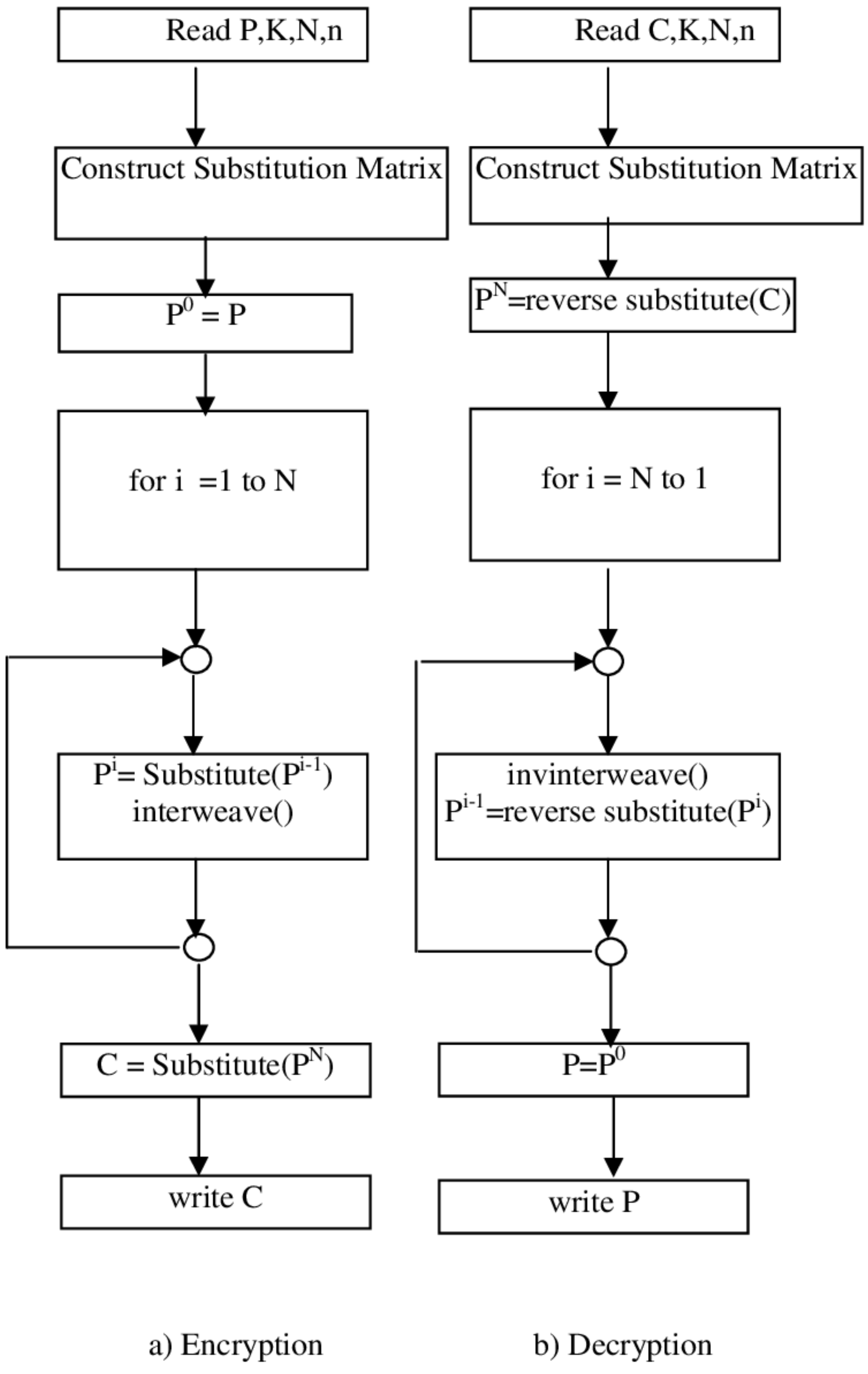
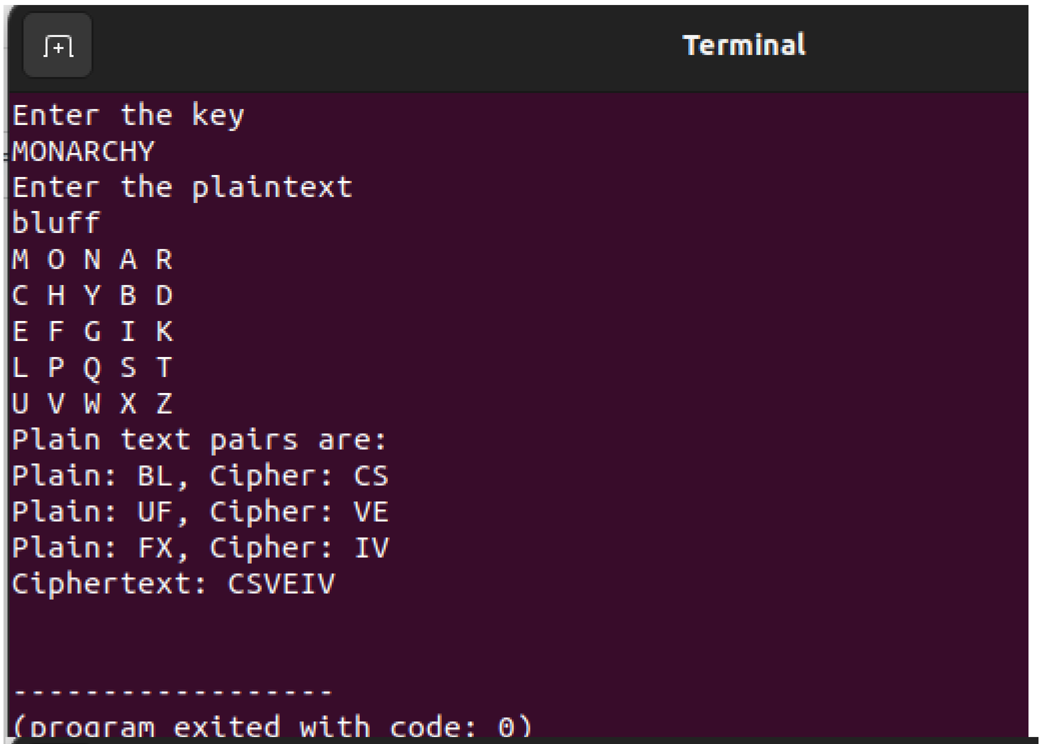


Fig. 9.1. Schematic diagram of the cipher

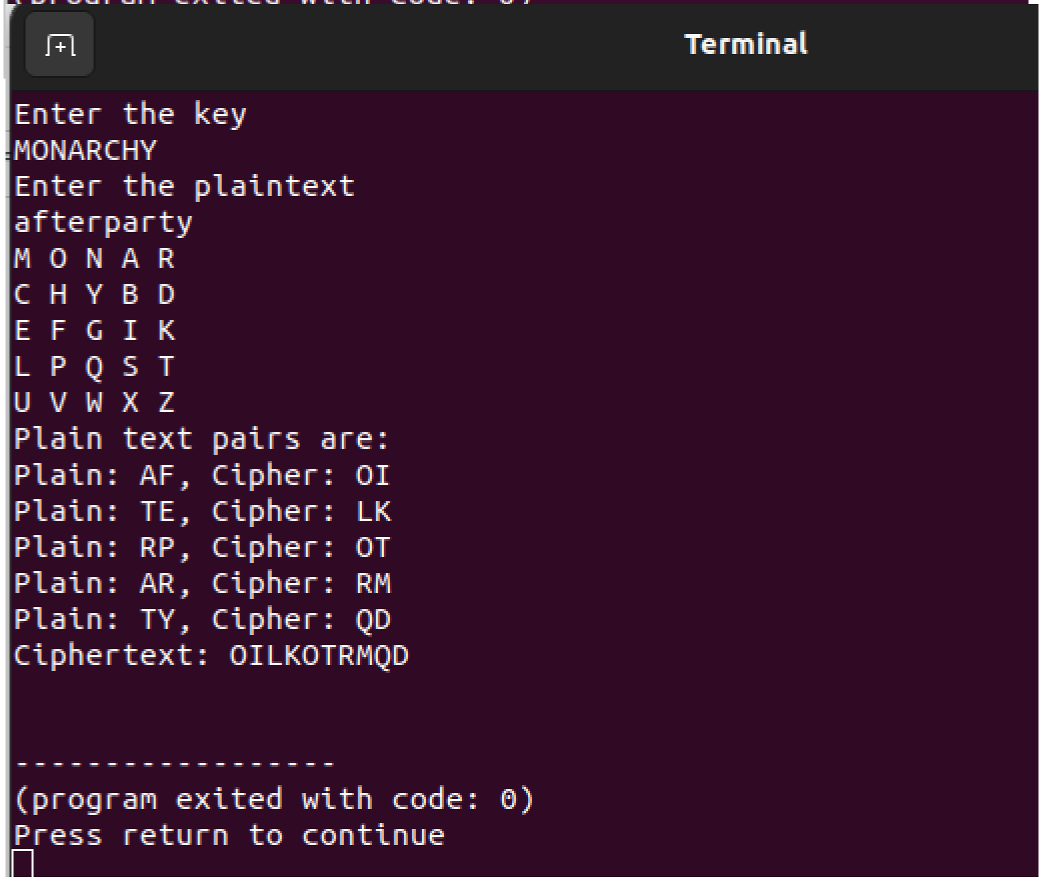
Code:

```
import java.io.*;
import java.util.*;

public class PlayFair {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the key");
        String key = sc.nextLine().toUpperCase();
        System.out.println("Enter the plaintext");
        String text = sc.nextLine().toUpperCase();

        String modKey = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";
        modKey = removeDuplicate(modKey);

        char[][] matrix = new char[5][5];
        int index = 0;
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                matrix[i][j] += modKey.charAt(index++);
            }
        }

        for(int i=0;i<5;i++) {
            for(int j=0;j<5;j++) {
                System.out.print(matrix[i][j]+ " " );
            }
            System.out.print("\n");
        }
        System.out.println("Plain text pairs are:");
        text = prepareText(text);
        int txtLength = text.length();
        StringBuilder cipherText = new StringBuilder();
        for (int i = 0; i < txtLength; i += 2) {
            char firstChar = text.charAt(i);
            char secondChar;
            if (i + 1 < txtLength) {
                secondChar = text.charAt(i + 1);
            } else {
                secondChar = 'X';
            }
            if (firstChar == secondChar) {
                secondChar = 'X';
                i--;
            }
            String cipherPair = encryptPair(matrix, firstChar, secondChar);
            cipherText.append(cipherPair);
            System.out.println("Plain: " + firstChar + "" + secondChar + ", Cipher: " +
cipherPair);
        }

        System.out.println("Ciphertext: " + cipherText.toString());
    }

    private static String removeDuplicate(String key) {
        String result = "";
        for (int i = 0; i < key.length(); i++) {
            char character = key.charAt(i);
            if (character != 'J' && result.indexOf(character) == -1) {
                result += character;
            }
        }
        return result;
    }
    private static String prepareText(String text) {
        StringBuilder preparedText = new StringBuilder();
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                preparedText.append(c);
            }
        }
        if (preparedText.length() % 2 != 0) {
            preparedText.append('X');
        }
        return preparedText.toString();
    }

    private static String encryptPair(char[][] matrix, char firstChar, char secondChar) {
        int[] firstPos = findPosition(matrix, firstChar);
        int[] secondPos = findPosition(matrix, secondChar);
        if (firstPos[0] == secondPos[0]) {
            return String.valueOf(matrix[firstPos[0]][(firstPos[1] + 1) % 5]) +
                    matrix[secondPos[0]][(secondPos[1] + 1) % 5];
        } else if (firstPos[1] == secondPos[1]) {
            return String.valueOf(matrix[(firstPos[0] + 1) % 5][firstPos[1]]) +
                    matrix[(secondPos[0] + 1) % 5][secondPos[1]];
        } else {
            return String.valueOf(matrix[firstPos[0]][secondPos[1]]) +
                    matrix[secondPos[0]][firstPos[1]];
        }
    }
    private static int[] findPosition(char[][] matrix, char ch) {
        int[] pos = new int[2];
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (matrix[i][j] == ch) {
                    pos[0] = i;
                    pos[1] = j;
                    return pos;
                }
            }
        }
        return pos;
    }
}
```



Terminal

```
Enter the key
MONARCHY
Enter the plaintext
bluff
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
Plain text pairs are:
Plain: BL, Cipher: CS
Plain: UF, Cipher: VE
Plain: FX, Cipher: IV
Ciphertext: CSVEIV


------------------
(program exited with code: 0)
```



Terminal

```
Enter the key
MONARCHY
Enter the plaintext
afterparty
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
Plain text pairs are:
Plain: AF, Cipher: OI
Plain: TE, Cipher: LK
Plain: RP, Cipher: OT
Plain: AR, Cipher: RM
Plain: TY, Cipher: QD
Ciphertext: OILKOTRMQD


------------------
(program exited with code: 0)
Press return to continue
```