

EXPERIMENT 4

Aim: Write a program to perform Encryption/Decryption using Playfair techniques.

Theory:

The technique encrypts pairs of letters (digraphs), instead of single letters as in the simple substitution cipher. The Playfair is significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. Frequency analysis can still be undertaken, but on the $25 \times 25 = 625$ possible digraphs rather than the 25 possible monographs. Frequency analysis thus requires much more ciphertext in order to work.

Algorithm:**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.

For example:

The key is "monarchy"

Thus, the initial entries are: 'm', 'o', 'n', 'a', 'r', 'c', 'h', 'y'

followed by remaining characters of a-z (except 'j') in that order.

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

- 2. 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.

For example:

Plain Text: "instruments"

After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'

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).

For example:

Diagraph: "me"

Encrypted Text: cl

Encryption: m -> c; e -> l

- 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).

For example:

Diagraph: "st"

Encrypted Text: tl

Encryption: s -> t

t -> l

- 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.

For example:

Diagraph: "nt"

Encrypted Text: rq

Encryption: n -> r
t -> q

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

For example:

Plain Text: "instrumentsz"

Encrypted Text: gatlmzclrqtx

Encryption:

i -> g m -> c
n -> a e -> l
s -> t n -> r
t -> lt -> q
r -> m s -> t
u -> z z -> x

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 50
void toLowerCase(char plain[], int ps) {
    int i;
    for (i = 0; i < ps; i++) {
        if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
    }
}
int removeSpaces(char* plain, int ps) {
    int i, count = 0;
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            plain[count++] = plain[i];
    plain[count] = '\0';
    return count;
}
void generateKeyTable(char key[], int ks, char keyT[5][5]) {
    int i, j, k, flag = 0, *dicty;
    dicty = (int*)calloc(26, sizeof(int));
    for (i = 0; i < ks; i++) {
        if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
    }
}
```

```
dicty['j' - 97] = 1;
i = 0;
j = 0;
for (k = 0; k < ks; k++) {
    if (dicty[key[k] - 97] == 2) {
        dicty[key[k] - 97] -= 1;
        keyT[i][j] = key[k];
        j++;
        if (j == 5) {
            i++;
            j = 0;
        }
    }
}
for (k = 0; k < 26; k++) {
    if (dicty[k] == 0) {
        keyT[i][j] = (char)(k + 97);
        j++;
        if (j == 5) {
            i++;
            j = 0;
        }
    }
}
}

void search(char keyT[5][5], char a, char b, int arr[]) {
    int i, j;
    if (a == 'j')
        a = 'i';
    else if (b == 'j')
        b = 'i';
    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            if (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            }
            else if (keyT[i][j] == b) {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}

int mod5(int a) {
    return (a % 5);
}

int prepare(char str[], int ptrs) {
    if (ptrs % 2 != 0) {
        str[ptrs++] = 'z';
        str[ptrs] = '\0';
    }
}
```

```
    }
    return ptrs;
}

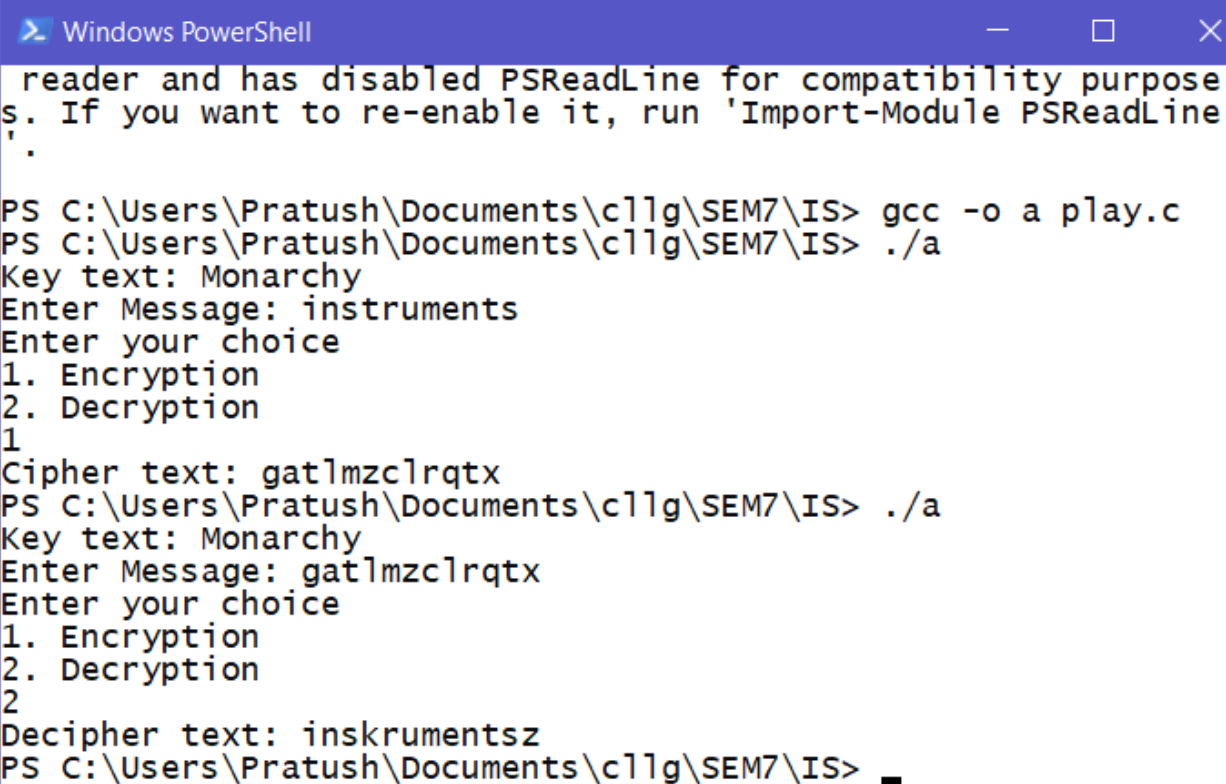
void encrypt(char str[], char keyT[5][5], int ps) {
    int i, a[4];
    for (i = 0; i < ps; i += 2) {
        search(keyT, str[i], str[i + 1], a);
        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] + 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] + 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}

void encryptByPlayfairCipher(char str[], char key[]) {
    char ps, ks, keyT[5][5];
    ks = strlen(key);
    ks = removeSpaces(key, ks);
    toLowerCase(key, ks);
    ps = strlen(str);
    toLowerCase(str, ps);
    ps = removeSpaces(str, ps);
    ps = prepare(str, ps);
    generateKeyTable(key, ks, keyT);
    encrypt(str, keyT, ps);
}

void decrypt(char str[], char keyT[5][5], int ps)
{
    int i, a[4];
    for (i = 0; i < ps; i += 2) {
        search(keyT, str[i], str[i + 1], a);
        if (a[0] == a[2]) {
            str[i] = keyT[a[0]][mod5(a[1] - 1)];
            str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];
        }
        else if (a[1] == a[3]) {
            str[i] = keyT[mod5(a[0] - 1)][a[1]];
            str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];
        }
        else {
            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}
```

```
}  
void decryptByPlayfairCipher(char str[], char key[]) {  
    char ps, ks, keyT[5][5];  
    ks = strlen(key);  
    ks = removeSpaces(key, ks);  
    toLowerCase(key, ks);  
    ps = strlen(str);  
    toLowerCase(str, ps);  
    ps = removeSpaces(str, ps);  
    generateKeyTable(key, ks, keyT);  
    decrypt(str, keyT, ps);  
}  
int main() {  
    char str[SIZE], key[SIZE], msg[50];  
    int ch;  
    strcpy(key, "Monarchy");  
    printf("Key text: %s\n", key);  
    printf("Enter Message: ");scanf("%s",&str);  
    //strcpy(str, msg);  
    //strcpy(str, "instruments");  
    printf("Enter your choice \n1. Encryption \n2. Decryption \n");  
    scanf("%d",&ch);  
    if(ch==1){  
        encryptByPlayfairCipher(str, key);  
        printf("Cipher text: %s\n", str);  
    }  
    else if(ch==2){  
        decryptByPlayfairCipher(str, key);  
        printf("Decipher text: %s\n", str);  
    }  
    return 0; }
```

Output:



```
Windows PowerShell
reader and has disabled PSReadLine for compatibility purpose
s. If you want to re-enable it, run 'Import-Module PSReadLine'
'.

PS C:\Users\Pratush\Documents\c11g\SEM7\IS> gcc -o a play.c
PS C:\Users\Pratush\Documents\c11g\SEM7\IS> ./a
Key text: Monarchy
Enter Message: instruments
Enter your choice
1. Encryption
2. Decryption
1
Cipher text: gatlmzclrqtx
PS C:\Users\Pratush\Documents\c11g\SEM7\IS> ./a
Key text: Monarchy
Enter Message: gatlmzclrqtx
Enter your choice
1. Encryption
2. Decryption
2
Decipher text: inskrumentsz
PS C:\Users\Pratush\Documents\c11g\SEM7\IS> _
```