Date: 29th July 2024

22BCP116

Experiment 2 Information Security - Lab

AIM: Study and Implement a program for Playfair Cipher

Introduction:

* Program:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define GRID\_SIZE 5

#define MAX\_INPUT\_SIZE 100

void create\_grid(char key[], char grid[GRID\_SIZE][GRID\_SIZE]) {

int alphabets[26] = {0};

int key\_len = strlen(key);

int grid\_index = 0;

for (int i = 0; i < key\_len; i++) {

key[i] = tolower(key[i]);

if (key[i] == 'j') {

key[i] = 'i';

}

if (!alphabets[key[i] - 'a']) {

alphabets[key[i] - 'a'] = 1;

}

}

for (int i = 0; i < key\_len; i++) {

if (isalpha(key[i]) && alphabets[key[i] - 'a'] == 1) {

grid[grid\_index / GRID\_SIZE][grid\_index % GRID\_SIZE] = key[i];

alphabets[key[i] - 'a'] = 2;

grid\_index++;

}

}

for (char ch = 'a'; ch <= 'z'; ch++) {

if (ch == 'j') continue;

if (alphabets[ch - 'a'] == 0) {

grid[grid\_index / GRID\_SIZE][grid\_index % GRID\_SIZE] = ch;

grid\_index++;

}

}

}

void create\_pairs(char plain\_text[], char pairs[][2], int \*num\_pairs) {

int text\_len = strlen(plain\_text);

int index = 0;

for (int i = 0; i < text\_len; i++) {

if (plain\_text[i] == ' ') continue;

if (index % 2 == 1 && plain\_text[i] == pairs[index / 2][0]) {

pairs[index / 2][index % 2] = 'x';

index++;

}

pairs[index / 2][index % 2] = tolower(plain\_text[i]);

index++;

}

if (index % 2 == 1) {

pairs[index / 2][1] = 'x';

index++;

}

\*num\_pairs = index / 2;

}

void find\_position(char letter, char grid[GRID\_SIZE][GRID\_SIZE], int \*row, int \*col) {

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

if (grid[i][j] == letter) {

\*row = i;

\*col = j;

return;

}

}

}

}

void encrypt\_pairs(char pairs[][2], int num\_pairs, char grid[GRID\_SIZE][GRID\_SIZE], char cipher\_text[]) {

int index = 0;

for (int i = 0; i < num\_pairs; i++) {

int r1, c1, r2, c2;

find\_position(pairs[i][0], grid, &r1, &c1);

find\_position(pairs[i][1], grid, &r2, &c2);

if (r1 == r2) {

cipher\_text[index++] = grid[r1][(c1 + 1) % GRID\_SIZE];

cipher\_text[index++] = grid[r2][(c2 + 1) % GRID\_SIZE];

} else if (c1 == c2) {

cipher\_text[index++] = grid[(r1 + 1) % GRID\_SIZE][c1];

cipher\_text[index++] = grid[(r2 + 1) % GRID\_SIZE][c2];

} else {

cipher\_text[index++] = grid[r1][c2];

cipher\_text[index++] = grid[r2][c1];

}

}

cipher\_text[index] = '\0';

}

void decrypt\_pairs(char cipher\_text[], char grid[GRID\_SIZE][GRID\_SIZE], char plain\_text[]) {

int text\_len = strlen(cipher\_text);

int index = 0;

for (int i = 0; i < text\_len; i += 2) {

int r1, c1, r2, c2;

find\_position(cipher\_text[i], grid, &r1, &c1);

find\_position(cipher\_text[i + 1], grid, &r2, &c2);

if (r1 == r2) {

plain\_text[index++] = grid[r1][(c1 - 1 + GRID\_SIZE) % GRID\_SIZE];

plain\_text[index++] = grid[r2][(c2 - 1 + GRID\_SIZE) % GRID\_SIZE];

} else if (c1 == c2) {

plain\_text[index++] = grid[(r1 - 1 + GRID\_SIZE) % GRID\_SIZE][c1];

plain\_text[index++] = grid[(r2 - 1 + GRID\_SIZE) % GRID\_SIZE][c2];

} else {

plain\_text[index++] = grid[r1][c2];

plain\_text[index++] = grid[r2][c1];

}

}

plain\_text[index] = '\0';

}

int main() {

char key[MAX\_INPUT\_SIZE];

char text[MAX\_INPUT\_SIZE];

char grid[GRID\_SIZE][GRID\_SIZE];

char pairs[50][2];

int num\_pairs;

char choice;

printf("Enter the key: ");

fgets(key, MAX\_INPUT\_SIZE, stdin);

key[strcspn(key, "\n")] = 0; // Remove trailing newline

printf("Do you want to encrypt (e) or decrypt (d)? ");

scanf("%c", &choice);

getchar(); // Consume the newline character left in the buffer

if (choice == 'e' || choice == 'E') {

printf("Enter the plain text: ");

fgets(text, MAX\_INPUT\_SIZE, stdin);

text[strcspn(text, "\n")] = 0; // Remove trailing newline

create\_grid(key, grid);

create\_pairs(text, pairs, &num\_pairs);

char cipher\_text[100];

encrypt\_pairs(pairs, num\_pairs, grid, cipher\_text);

printf("Cipher Text:\n%s\n", cipher\_text);

} else if (choice == 'd' || choice == 'D') {

printf("Enter the cipher text: ");

fgets(text, MAX\_INPUT\_SIZE, stdin);

text[strcspn(text, "\n")] = 0; // Remove trailing newline

create\_grid(key, grid);

char plain\_text[100];

decrypt\_pairs(text, grid, plain\_text);

printf("Decrypted Text:\n%s\n", plain\_text);

} else {

printf("Invalid choice. Please enter 'e' for encryption or 'd' for decryption.\n");

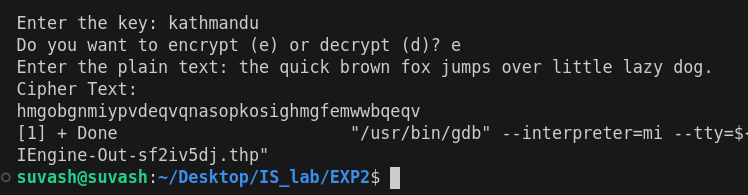
}

return 0;

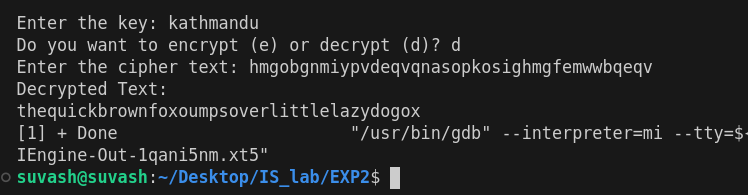
}

Output:

1. Encryption

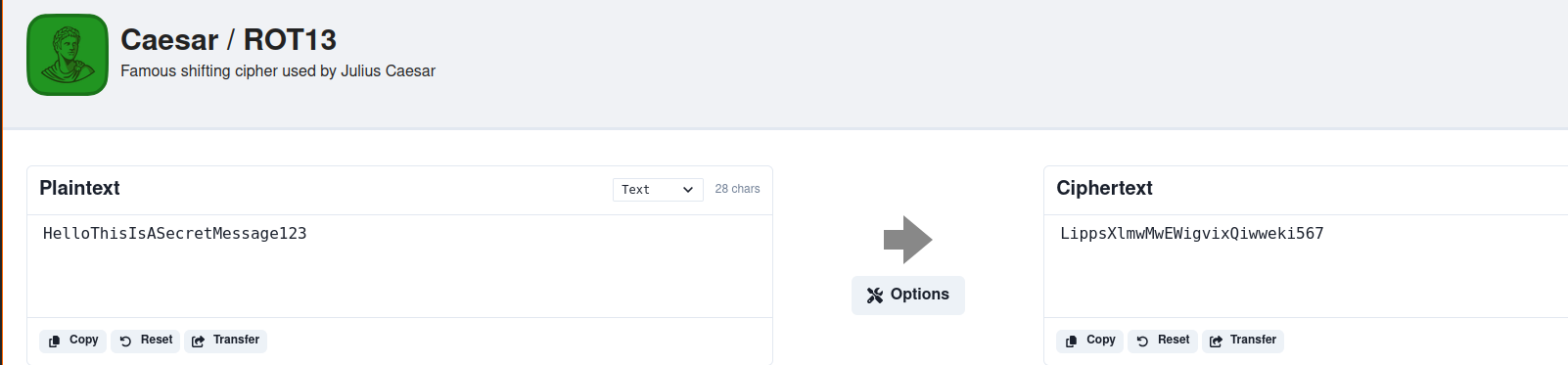


2. Decryption

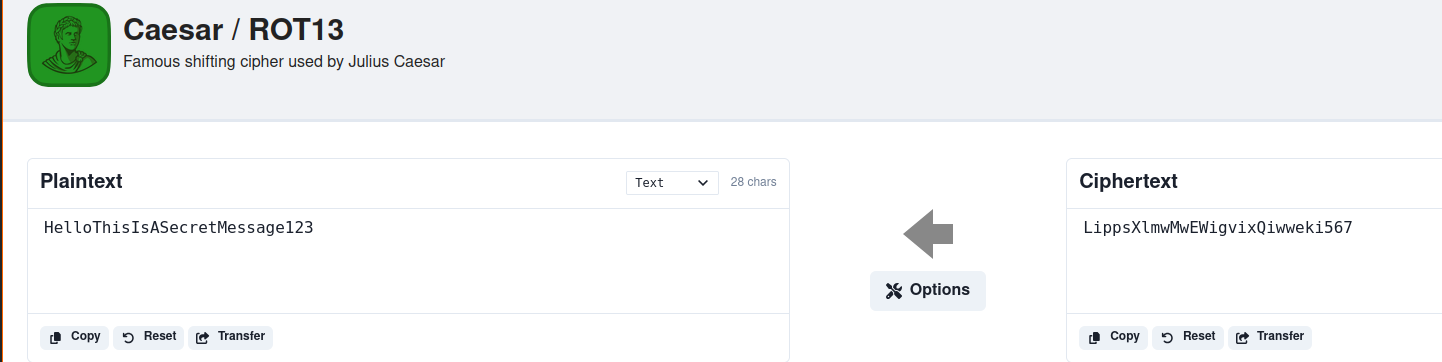


Cryptool Output:

1. Encryption



2. Decryption

****