**Government College of Arts, Science and Commerce Quepem-Goa**

**Practical Record in the subject of**

**DSE – CSD105 – NETWORK SECURITY**

**Name of the Student: Shreyash Devali**

**Roll No : S21-626**

**Class/Semester :** TYBSc / Sem. VI

**Batch : 1**

**Academic Year: 2023-2024**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SR** | **Experiment Name** | **Date of Completion** | **Page No** | **Sign. Teacher Incharge** |
| 01 | Caesar cipher | 08-01-2024 | 03 |  |
| 02 | Vigenere Cipher | 05-02-2024 | 06 |  |
| 03 | Playfair Cipher | 05-02-2024 | 08 |  |
| 04 | Railfence Cipher | 12-02-2014 | 11 |  |
| 05 | Vernam Cipher | 26-02-2024 | 13 |  |
| 06 | Stegnography with DOS commands | 09-02-2024 | 15 |  |
| 07 | OpenStego | 04-03-2024 | 17 |  |
| 08 | OpenSSL | 01-04-2024 | 20 |  |

**Experiment No:01 DOE:08-01-2024**

**Caesar cipher**

Aim: Implementation of Ceasar Cipher

Theory:

It's a type of substitution cipher where each letter in the plaintext is replaced by a letter a certain number of positions down the alphabet. For example, with a shift of 3, "hello" would become "khoor."

The modified Caesar cipher is a method for encrypting messages by replacing each alphabet with an alphabet K places down or up the line. The only difference between the Caesar cipher and the modified Caesar cipher is that in the Caesar cipher, alphabets are replaced by 3 places down or up the line.

Encryption: E(x) = (x + k) mod 26, where x is the position of the letter in the alphabet and k shift

value(Key).

Decryption: D(x) = (x - k) mod 26 ,where x is the position of the letter in the alphabet and k is the

shift value(Key).

**Programs:**

**Encryption**

#include<stdio.h>

#include<ctype.h>

int main() {

    char text[500], ch;

    int key;

    // Taking user input.

    printf("Enter a message to encrypt: ");

    scanf("%s", text);

    printf("Enter the key: ");

    scanf("%d", & key);

    // Visiting character by character.

    for (int i = 0; text[i] != '\0'; ++i) {

        ch = text[i];

        // Check for valid characters.

        if (isalnum(ch)) {

            //Lowercase characters.

            if (islower(ch)) {

                ch = (ch - 'a' + key) % 26 + 'a';

            }

            // Uppercase characters.

            if (isupper(ch)) {

                ch = (ch - 'A' + key) % 26 + 'A';

            }

            // Numbers.

            if (isdigit(ch)) {

                ch = (ch - '0' + key) % 10 + '0';

            }

        }

        // Invalid character.

        else {

            printf("Invalid Message");

        }

        // Adding encoded answer.

        text[i] = ch;

    }

    printf("Encrypted message: %s", text);

    return 0;

}

**Output:-**

Enter a message to encrypt: abcdefghijklmnopqrstuvwxyz

Enter the key: 3

Encrypted message: defghijklmnopqrstuvwxyzabc

**Decryption:-**

#include<stdio.h>

#include<ctype.h>

int main() {

char text[500], ch;

int key;

// Taking user input.

printf("Enter a message to decrypt: ");

scanf("%s", text);

printf("Enter the key: ");

scanf("%d", & key);

// Visiting each character.

for (int i = 0; text[i] != '\0'; ++i) {

ch = text[i];

// Check for valid characters.

if (isalnum(ch)) {

//Lowercase characters.

if (islower(ch)) {

ch = (ch - 'a' - key + 26) % 26 + 'a';

}

// Uppercase characters.

if (isupper(ch)) {

ch = (ch - 'A' - key + 26) % 26 + 'A';

}

// Numbers.

if (isdigit(ch)) {

ch = (ch - '0' - key + 10) % 10 + '0';

}

}

// Invalid characters.

else {

printf("Invalid Message");

}

// Adding decoded character back.

text[i] = ch;

}

printf("Decrypted message: %s", text);

return 0;

}

**Output:-**

Enter a message to decrypt: defghijklmnopqrstuvwxyzabc

Enter the key: 3

Decrypted message: abcdefghijklmnopqrstuvwxyz

**Conclusion:** Implementation of Caesar Cipher was done successfully.

**Experiment No:02 DOE:05-02-2024**

**Vigenere Cipher**

Aim: Implementation of Vigenere Cipher

Theory:

The Vigenère cipher is a more sophisticated way to encrypt messages compared to the Caesar cipher and Rail Fence cipher. It falls under the category of polyalphabetic substitution ciphers, meaning it uses multiple Caesar ciphers in sequence to encrypt the message, making it more complex to break. Here's a breakdown of the Vigenère cipher:

**Programs:**

**Encryption & Decryption:-**

#include <stdio.h>

#include<conio.h>

#include <ctype.h>

#include <string.h>

void encipher();

void decipher();

void main()

{

int choice;

while(1)

{

 printf("\n1. Encrypt Text");

 printf("\t2. Decrypt Text");

 printf("\t3. Exit");

 printf("\n\nEnter Your Choice : ");

 scanf("%d",&choice);

 if(choice == 3)

  exit (0) ;

 else if(choice == 1)

  encipher();

 else if(choice == 2)

  decipher();

 else

  printf("Please Enter Valid Option.");

}

}

void encipher()

{

unsigned int i,j;

char input[50],key[10];

printf("\n\nEnter Plain Text: ");

scanf("%s",input);

printf("\nEnter Key Value: ");

scanf("%s",key);

printf("\nResultant Cipher Text: ");

for(i=0,j=0;i<strlen(input);i++,j++)

/\* i will be the plain text and represent the column of table and

j will repeate/reset the after key length \*/

 {

  if(j>=strlen(key)) { j=0;}

  printf("%c",65+(((toupper(input[i])-65)+(toupper(key[j])-65))%26));

//65+(   (  (toupper(input[i])-65)   +  (toupper(key[j])-65) )   %26     )

 }

}

void decipher()

{

unsigned int i,j;

char input[50],key[10];

int value;

printf("\n\nEnter Cipher Text: ");

scanf("%s",input);

printf("\n\nEnter the key value: ");

scanf("%s",key);

for(i=0,j=0;i<strlen(input);i++,j++)

 {

  if(j>=strlen(key)) { j=0; }

  value = (toupper(input[i])-64)-(toupper(key[j])-64);

  if( value < 0) { value = value \* -1;}

printf("%c",65 + (value % 26));

 }

}

Output:-

1. Encrypt Text 2. Decrypt Text 3. Exit

Enter Your Choice : 1

Enter Plain Text: abcdefghijklmnopqrstuvwxyz

Enter Key Value: keyword

Resultant Cipher Text: KFAZSWJRMHGZDQYTONGKXFAVUN

1. Encrypt Text 2. Decrypt Text 3. Exit

Enter Your Choice : 2

Enter Cipher Text: KFAZSWJRMHGZDQYTONGKXFAVUN

Enter the key value: keyword

Decrypt: ABYDEFGHIRQLONOPKJIHUFEDCB

**Conclusion:** Implementation of Vigenere Cipher was done successfully.

**Experiment No:03 DOE:05-02-2024**

**Playfair Cipher**

Aim: Implementation of Playfair Cipher

Theory:

The Playfair cipher, a manual encryption technique, uses a 5x5 letter grid to substitute pairs of letters (bigrams) instead of single letters. To encrypt a message, you create a key with the grid, split the message into bigrams, and then apply rules based on the bigrams' position in the grid to create an encrypted message. Despite its historical importance, the Playfair cipher is not considered secure for modern cryptography.

**Programs:**

**Encryption & Decryption:-**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define SIZE 5

char\* prepare\_input(char\* text) {

int i, j;

char\* filtered\_text = (char\*)malloc(strlen(text) \* sizeof(char));

int index = 0;

for (i = 0; i < strlen(text); i++) {

if (isalpha(text[i])) {

if (text[i] == 'J')

text[i] = 'I';

filtered\_text[index++] = toupper(text[i]);

}

}

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

if (strlen(filtered\_text) % 2 != 0) {

filtered\_text[index++] = 'X';

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

}

return filtered\_text;

}

void generate\_key\_matrix(char\* key, char key\_matrix[SIZE][SIZE]) {

int i, j, k;

char alphabet[] = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

strcat(key, alphabet);

int index = 0;

for (i = 0; i < SIZE; i++) {

for (j = 0; j < SIZE; j++) {

while (strchr(key\_matrix[i], key[index]) || key[index] == '\0')

index++;

key\_matrix[i][j] = key[index];

index++;

}

}

}

void find\_position(char key\_matrix[SIZE][SIZE], char letter, int\* row, int\* col) {

int i, j;

for (i = 0; i < SIZE; i++) {

for (j = 0; j < SIZE; j++) {

if (key\_matrix[i][j] == letter) {

\*row = i;

\*col = j;

return;

}

}

}

}

void encrypt(char\* text, char\* key, char key\_matrix[SIZE][SIZE], char\* encrypted\_text) {

char\* filtered\_text = prepare\_input(text);

int len = strlen(filtered\_text);

int i, j, index = 0;

for (i = 0; i < len; i += 2) {

char letter1 = filtered\_text[i];

char letter2 = filtered\_text[i + 1];

int row1, col1, row2, col2;

find\_position(key\_matrix, letter1, &row1, &col1);

find\_position(key\_matrix, letter2, &row2, &col2);

if (row1 == row2)

encrypted\_text[index++] = key\_matrix[row1][(col1 + 1) % SIZE];

else if (col1 == col2)

encrypted\_text[index++] = key\_matrix[(row1 + 1) % SIZE][col1];

else {

encrypted\_text[index++] = key\_matrix[row1][col2];

encrypted\_text[index++] = key\_matrix[row2][col1];

}

}

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

free(filtered\_text);

}

void decrypt(char\* text, char\* key, char key\_matrix[SIZE][SIZE], char\* decrypted\_text) {

int len = strlen(text);

int i, j, index = 0;

for (i = 0; i < len; i += 2) {

char letter1 = text[i];

char letter2 = text[i + 1];

int row1, col1, row2, col2;

find\_position(key\_matrix, letter1, &row1, &col1);

find\_position(key\_matrix, letter2, &row2, &col2);

if (row1 == row2)

decrypted\_text[index++] = key\_matrix[row1][(col1 - 1 + SIZE) % SIZE];

else if (col1 == col2)

decrypted\_text[index++] = key\_matrix[(row1 - 1 + SIZE) % SIZE][col1];

else {

decrypted\_text[index++] = key\_matrix[row1][col2];

decrypted\_text[index++] = key\_matrix[row2][col1];

}

}

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

}

int main() {

char key[SIZE\*SIZE];

char key\_matrix[SIZE][SIZE];

char text[100];

char encrypted\_text[100];

char decrypted\_text[100];

printf("Enter the key (no spaces, all uppercase, J replaced by I): ");

scanf("%s", key);

generate\_key\_matrix(key, key\_matrix);

printf("Enter the text to encrypt: ");

scanf("%s", text);

encrypt(text, key, key\_matrix, encrypted\_text);

printf("Encrypted: %s\n", encrypted\_text);

decrypt(encrypted\_text, key, key\_matrix, decrypted\_text);

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

return 0;

}

Output:-

Enter the key (no spaces, all uppercase, J replaced by I): test

Enter the text to encrypt: matrix

Encrypted: LBWOMO

Decrypted: MATRIR

**Conclusion:** Implementation of Playfair Cipher was done successfully.

**Experiment No:04 DOE:12-02-2014**

**Railfence Cipher**

Aim: Implementation of Railfence Cipher

Theory:

The railfence cipher, also known as the zigzag cipher, is a classic type of transposition cipher. It derives its name from the way in which encryption is performed, in analogy to a fence built with horizontal rails.

**Programs:**

**Encryption & Decryption:-**

#include<stdio.h>

#include<string.h>

int main()

{

int i,j,k,l;

char a[20],c[20],d[20];

printf("\n\t\t RAIL FENCE TECHNIQUE");

printf("\n\nEnter the input string : ");

gets(a);

l=strlen(a);

/\*Ciphering\*/

for(i=0,j=0;i<l;i++)

{

if(i%2==0)

c[j++]=a[i];

}

for(i=0;i<l;i++)

{

if(i%2==1)

c[j++]=a[i];

}

c[j]='\0';

printf("\nCipher text after applying rail fence :");

printf("\n%s",c);

/\*Deciphering\*/

if(l%2==0)

k=l/2;

else

k=(l/2)+1;

for(i=0,j=0;i<k;i++)

{

d[j]=c[i];

j=j+2;

}

for(i=k,j=1;i<l;i++)

{

d[j]=c[i];

j=j+2;

}

d[l]='\0';

printf("\nText after decryption : ");

printf("%s",d);

}

**Output:-**

RAIL FENCE TECHNIQUE

Enter the input string : abcdefghijklmnopqrstuvwxyz

Cipher text after applying rail fence : acegikmoqsuwybdfhjlnprtvxz

Text after decryption : abcdefghijklmnopqrstuvwxyz

**Conclusion:** Implementation of Rail Fence Technique was done successfully.

**Experiment No:05 DOE:26-02-2024**

**Vernam Cipher**

Aim: Implementation of Vernam Cipher

Theory:

The Vernam Cipher, also known as the One-Time Pad (OTP), is a theoretically unbreakable encryption technique. It works by XORing a random key, as long as the message itself, with the message to create a scrambled ciphertext. This ciphertext can only be decrypted back to the original message by using the same exact random key, making it secure as long as the key is truly random, never reused, and kept secret.

**Programs:**

**Encryption**

#include<bits/stdc++.h>

using namespace std;

int main(){

int t,n,i,j,k,sum=0;

string m;

cout<<"Enter the message"<<'\n';

cin>>m;

string key;

cout<<"Enter the key"<<'\n';

cin>>key;

int mod = key.size();

j=0;

for(i=key.size();i<m.size();i++){

key+=key[j%mod];

j++;

}

string ans="";

for(i=0;i<m.size();i++){

ans += (key[i]-'A'+m[i]-'A')%26+'A';

}

cout<<"Encrypted message: "<<ans<<'\n';

return 0;

}

Output:-

Enter the message :- godspeed

Enter the key:- QWERTYUI

Encrypted message:- WKHJICYL

**Decryption:-**

#include<bits/stdc++.h>

using namespace std;

int main(){

int t,n,i,j,k,sum=0;

string m;

cout<<"Enter the message"<<'\n';

cin>>m;

string key;

cout<<"Enter the key"<<'\n';

cin>>key;

int mod = key.size();

j=0;

for(i=key.size();i<m.size();i++){

key+=key[j%mod];

j++;

}

string ans="";

for(i=0;i<m.size();i++){

ans += (m[i]-key[i]+26)%26+'A';

}

cout<<"Decrypted message: "<<ans<<'\n';

return 0;

}

Output:-

Enter the message:- WKHJICYL

Enter the key:- QWERTYUI

Decrypted message:- GODSPEED

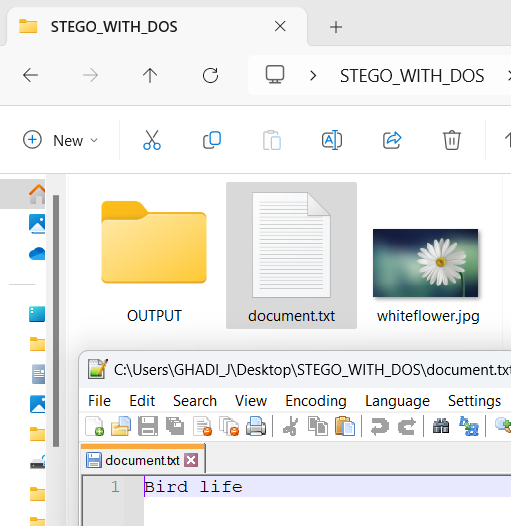
**Conclusion:** Implementation of Vernam Cipher was done successfully.

**Experiment No:06 DOE:09-02-2024**

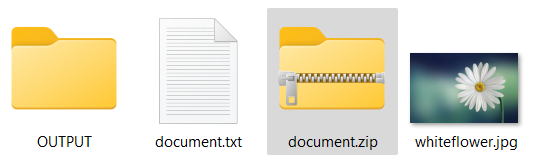
**Stegnography with DOS commands**

Aim: Implementation of Stegnography with DOS commands

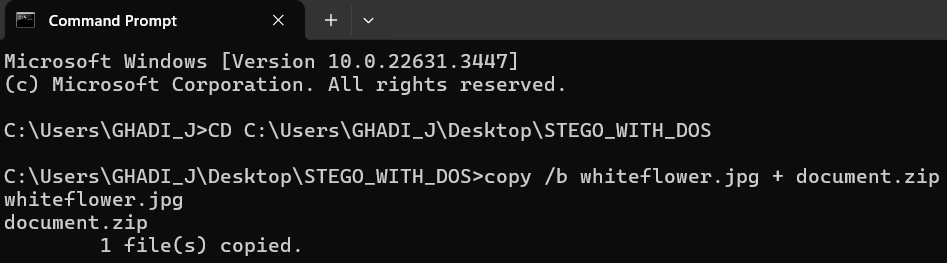
Original text file before hiding behind an image file



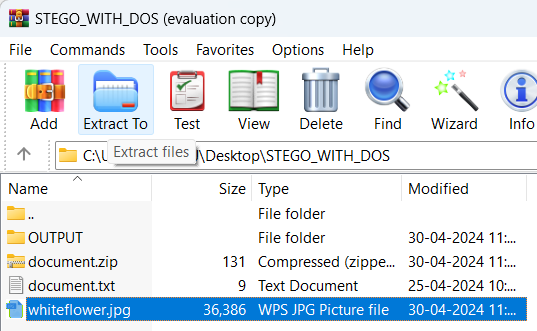
Create zip file of the text document



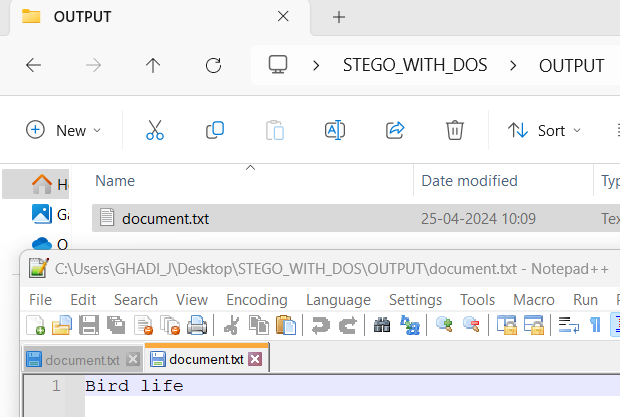
Cmd command to copy the content of the zip into the image file



Extraction of hidden file from the image



Displaying the content of the file after extraction into an output folder



**Conclusion:** Implementation of Stegnography with DOS commands was done successfully.

**Experiment No:07 DOE:04-03-2024**

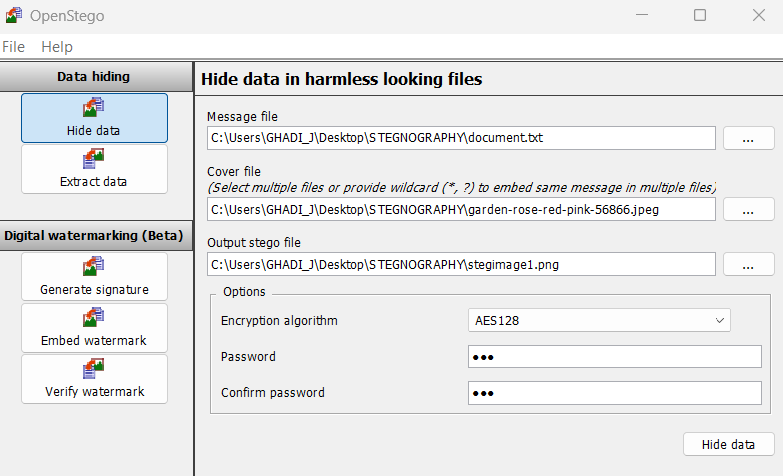
**OpenStego**

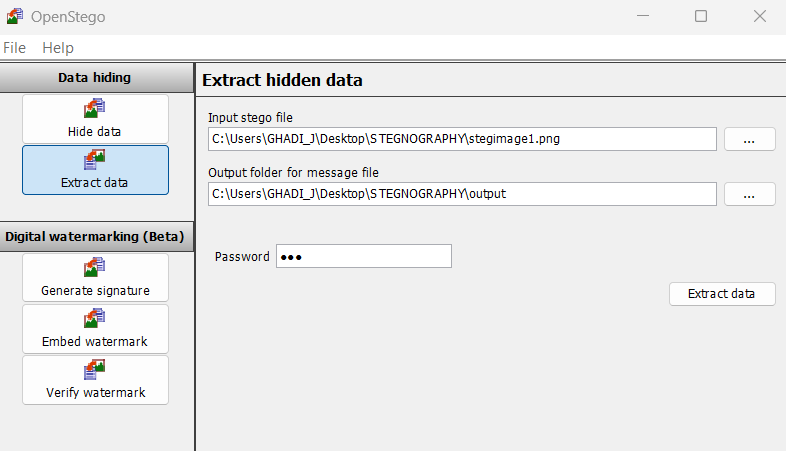
Aim: Implementation of OpenStego to perform:

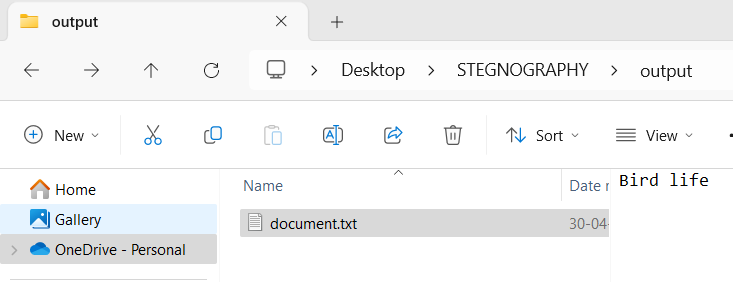
I. Stegnography with OpenStego

II. Digital watermarking with OpenStego

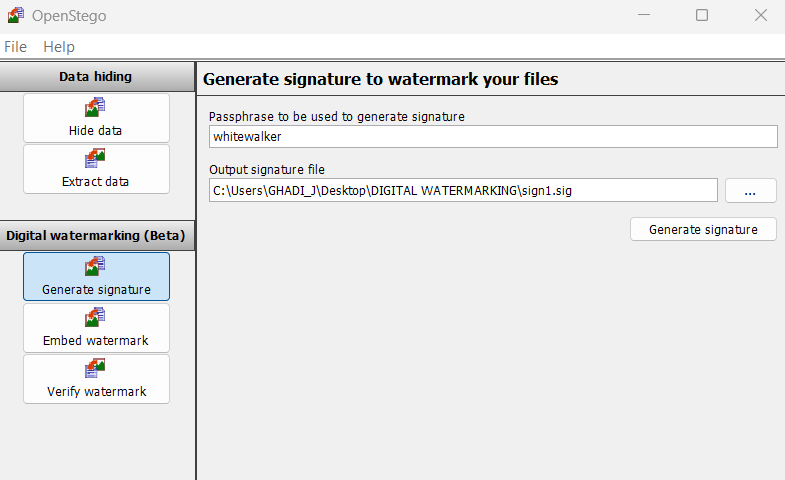
I. Stegnography with OpenStego

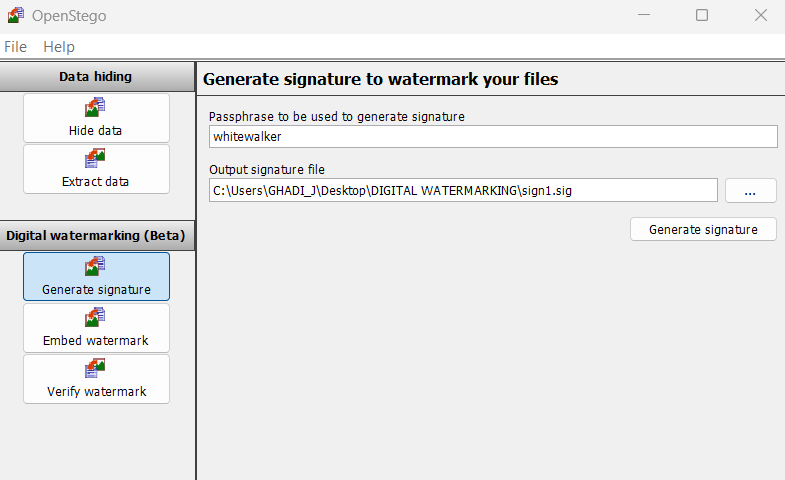


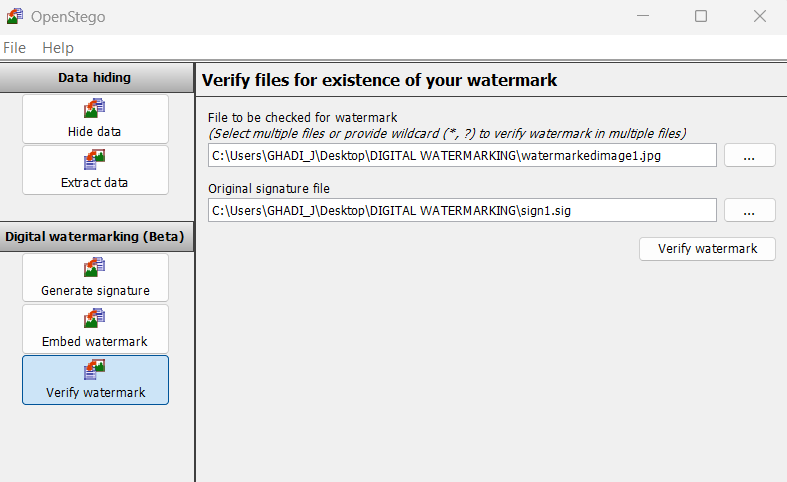


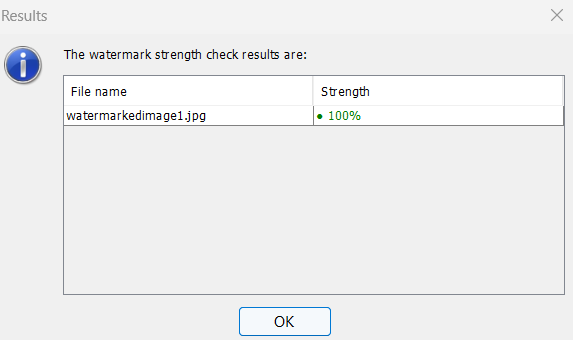


II. Digital watermarking with OpenStego









**Conclusion:** Implementation of OpenStego to perform Stegnography with OpenStego,Digital watermarking with OpenStego was done successfully.

**Experiment No:08 DOE:01-04-2024**

**OpenSSL**

**Aim: Implementation of OpenSSL**

1. CHECKING OPENSSL TOOL VERSION

**openssl version**

**openssl version -a**

1. GENERATE KEY PAIR

#to generate a key and name it as assymkey.key of size 2048

**openssl genrsa -out assym\_key1.key 2048**

#to show the directory

**dir**

1. Encrypting a file (warning displayed)

#this will generate a encrypted file document.enc

C:\Users\GHADI\_J>

**openssl aes-256-cbc -in C:\Users\GHADI\_J\Desktop\NS STEGO\document.txt -out C:\Users\GHADI\_J\Desktop\NS STEGO\document.enc -e -kfile assym\_key1.key**

Simpler version if are in the same folder:

C:\Users\GHADI\_J\Desktop\NSSTEGO>

**openssl aes-256-cbc -in document.txt -out document.enc -e -kfile assym\_key1.key**

1. Decrypting an encrypted file (warning displayed)

#this will generate a decrypted file document.dec

**openssl aes-256-cbc -in document.enc -out document.dec -d -kfile assym\_key1.key**

1. GENERATE CERTIFICATE SIGNING REQUEST

#this is a request to generate CSR file using the key

**openssl req -new -key qwerty.key -out qwerty.csr**

1. VERIFYING DETAILS OF CSR

#this is a request to verify the .csr file

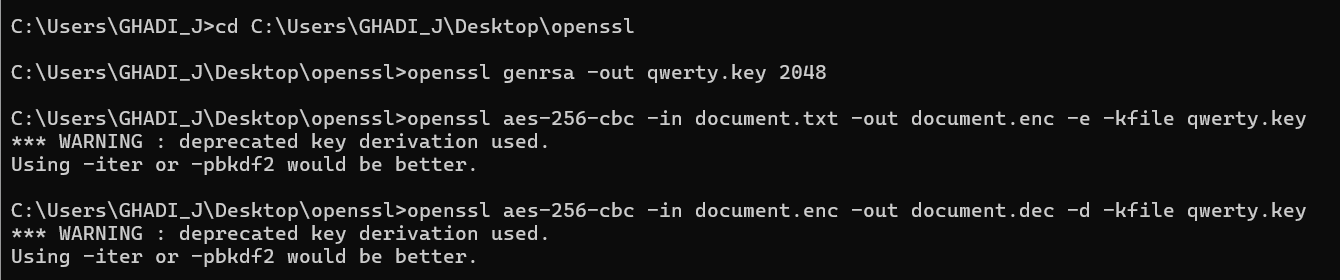
**openssl req -text -in qwerty.csr -noout -verify**

1. GENERATE SELF SIGNING CERTIFICATE

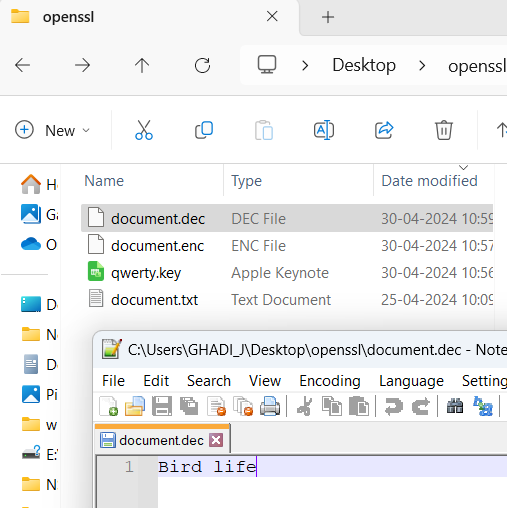
#this generates a self signing certificate using the .csr file and sign with the key valid for certain days

**openssl x509 -in qwerty.csr -out qwerty.crt -req -signkey qwerty.key -days 365**

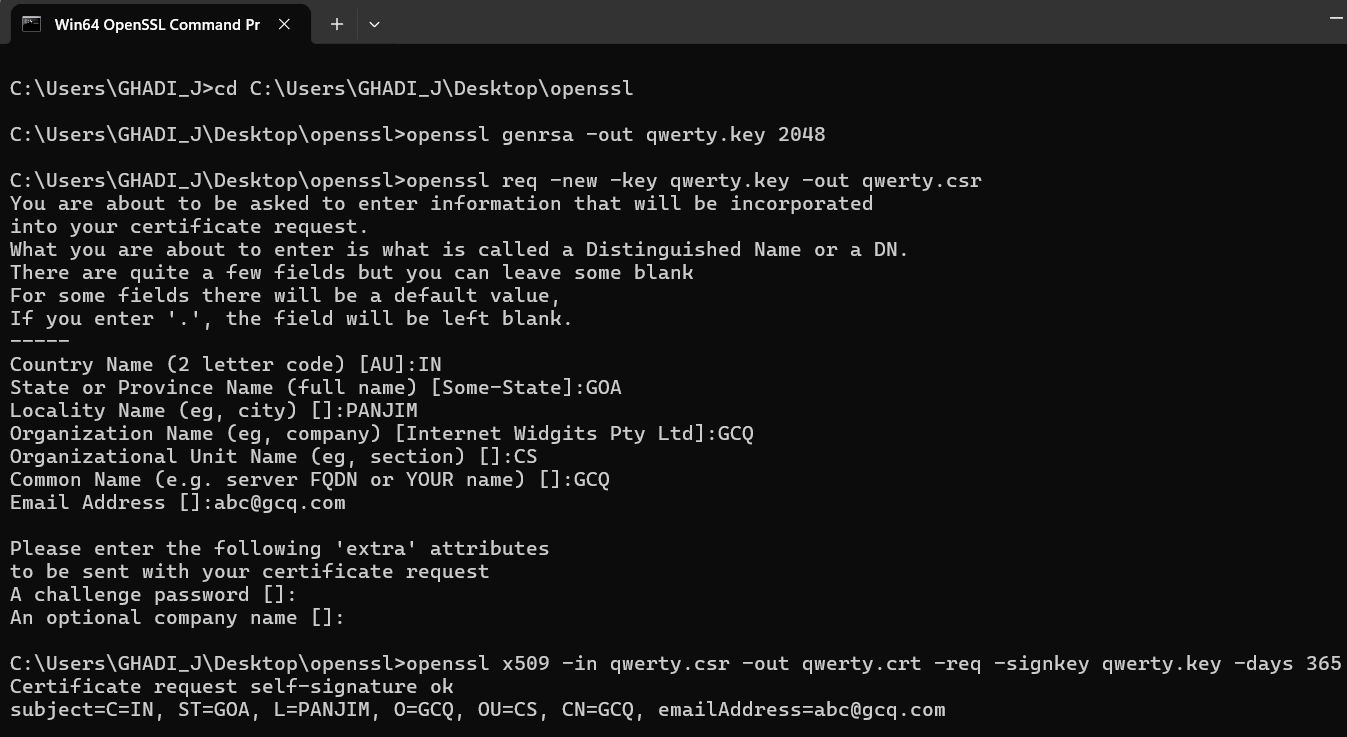
Q. Key creation, Encryption, Decryption



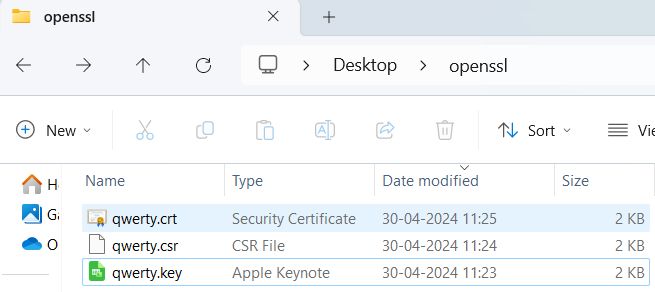
OUTPUT FOLDER:



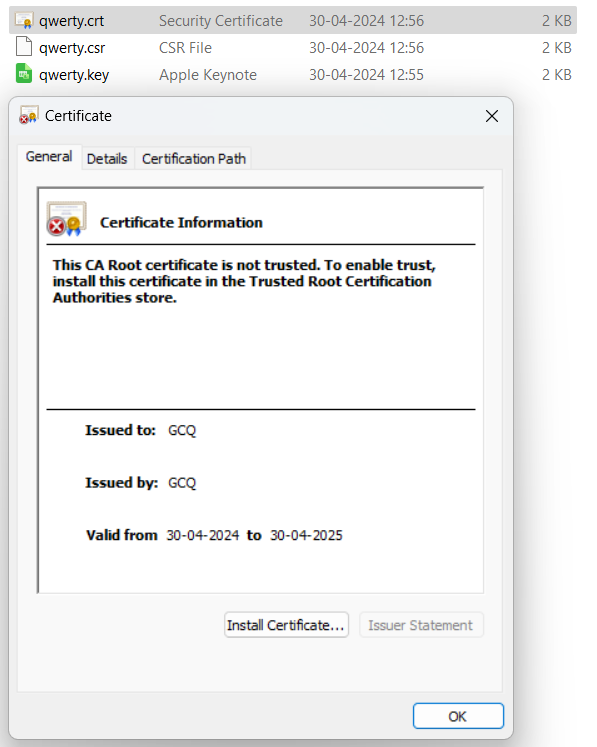
Q. Key creation, Certificate Signing request generation, self signing certificate generation



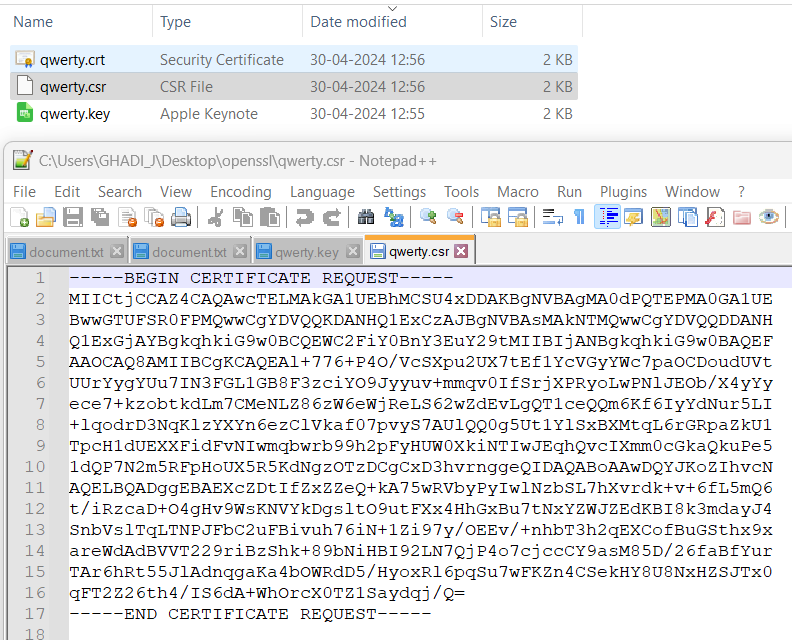
OUTPUT FOLDER:



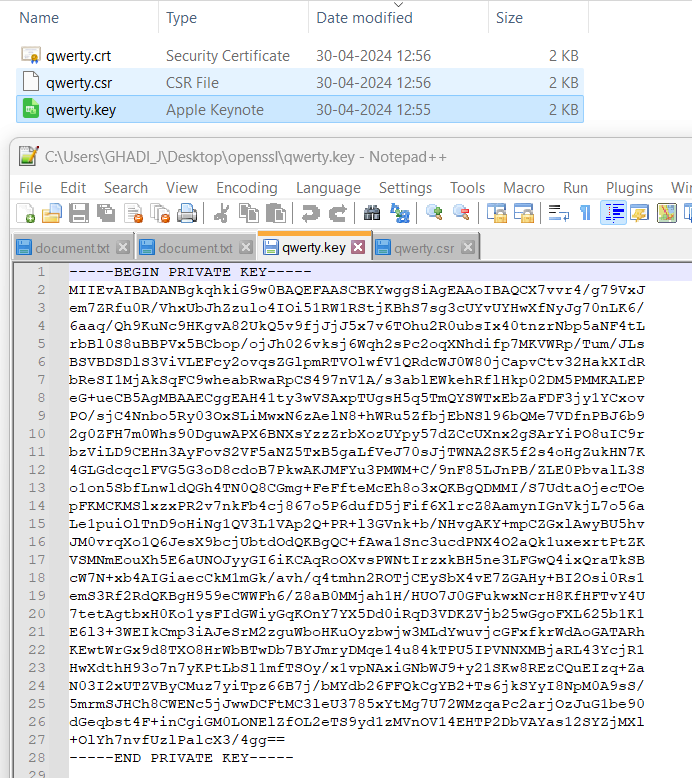
Content of qwerty.crt



Content of qwert.csr file



Content of qwerty.key



**Conclusion:** Implementation of OpenSSL was done successfully.