

**Integrantes:**

Briones Bedoy Dayana Yareli

Gómez González Josué Esaud

Perez Rodríguez Laura Georgina

Orellan Silva Luis Adolfo Mateo

**Materia:**

Fundamentos de Sistemas Operativos.

**Fecha:**

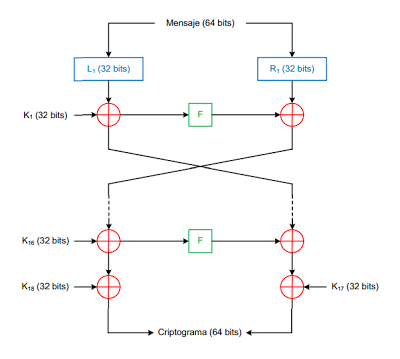
06 de Noviembre del 2017

**Objetivo**

La implementación de algoritmos de encriptación.

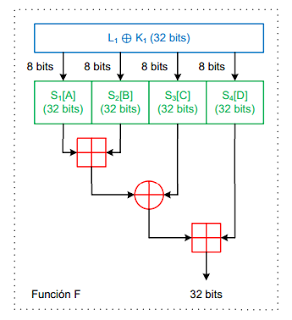
**Marco teórico**

Blowfish es un algoritmo de cifrado por bloques simétrico libre de patente creado por Bruce Schneier en 1993 como una alternativa para reemplazar a DES como estándar de cifrado. Este algoritmo está compuesto por 18 semiclaves (K) y 4 cajas (subtitution boxes S). Es un proceso relativamente simple y altamente seguro ya que a la fecha no se conoce ningún tipo de criptoanálisis efectivo contra este algoritmo de cifrado.

[](http://2.bp.blogspot.com/-2o0QTPxZ3VQ/Udy3th_V2OI/AAAAAAAAAnM/ESpiLW5ax9Q/s1600/Captura+de+pantalla+de+2013-07-09+19:52:30.png)

Para poder entender el algoritmo primero tendremos que describir la función F, la cual se encarga de sustituir los valores resultantes de las operaciones XOR utilizando las cajas de sustitución (subtitution boxes).

La función F divide el grupo de 32bits en 4 grupos de 8bits el bloque a y bloque b se buscan en las cajas sustitución(representadas con la letra "S" en el diagrama), el valor representado por el primer octeto de la primer caja se suma al valor representado del segundo octeto de la segunda caja y al resultado se saca el módulo de 2³², posteriormente a este resultado se aplica una operación XOR con el valor representado por el tercer bloque en la tercer caja y al resultado de esto se le suma el valor representado por el cuarto octeto en la cuarta caja y al final se vuelve a aplicar el módulo 2³².

[](http://3.bp.blogspot.com/-0s-qUlbTR3M/UdzDzPqDvwI/AAAAAAAAAnk/BLuoguv1i9A/s1600/Captura+de+pantalla+de+2013-07-09+19:52:42.png)

Scheier describe la función F como:

*F* (*Li* ⊕ *Ki*) =((((*S*1[*A*]+*S*2[*B*]) mod232) ⊕*S*3[*C*]) +*S*4[*D*]) mod232

Una vez entendido que es la función F del Blowfish paso a describir el algoritmo de cifrado:

* Un bloque de 64bits se divide en dos bloques de 32bits (L y R)
* Se realiza una operación XOR a los primeros 32bits(L) con la primera subclave (K) y se realiza la función F con el resultado de la operación.
* Se realiza una operación XOR con la segunda parte de los 32 bits(R) con el resultado de la función F(L).
* Se intercambian posiciones (R pasa a ser L y L pasa a ser R) y se repite el proceso anterior por 16 iteraciones.
* Al terminar la última iteración no se realizará el intercambio.
* Se realiza la operación XOR entre el valor alojado en L y la subclave 18.
* Se realiza la operación XOR entre el valor alojado en R y la subclave 17.
* Se unen los dos fragmentos del bloque para generar nuevamente un bloque de 64bits ya cifrado.

Las subclaves y las cajas de substitución no corresponden a valores arbitrarios, estos valores son el resultado de un proceso previo a la encriptación o des encriptación con Blowfish.

**Procedimiento**

* Se define un arreglo de 18 secciones las cuales deben de poder almacenar 32bits cada una ya que estas alojaran las subclaves, además se generan 4 arreglos de 256 posiciones también de 32bits cada una, estas últimas son las cajas de substitución.
* Cada sección de los arreglos (subclaves y cajas) son inicializados con una cadena fija, esta cadena son los dígitos en hexadecimal de *π* a excepción de su parte entera. El orden de inicialización es *P*1, *P*2, *P*3, …, *P*18, *S*1, *S*2, *S*3, *S*4.
* A cada una de las subclaves se le aplica un XOR con 32bits de la clave de usuario.
* Una vez inicializadas las cajas y las subclaves, se cifra un mensaje nulo (lleno de ceros)
* El paso anterior tiene 2 funciones, primero sustituye las primeras dos subclaves y segundo entra a Blowfish para ser cifrada (con las subclaves sustituidas).
* Se repite el paso anterior pero esta vez se sustituyen las siguientes dos subclaves. Este paso se repite hasta que las 18 subclaves y el contenido de las 4 cajas hayan sido sustituidas por las salidas correspondientes del cifrado de la palabra nula.

En total se realizarán 521 iteraciones para obtener los valores que se utilizaran en el cifrado de información.

**Codigo**

#include <iostream> //LIBRERIA QUE PERMITE LA ENTRADA Y SALIDA DE DATOS

#include<fstream> //LIBRERIA QUE PERMITE EL MANEJO DE FICHEROS

using namespace std; //IMPLEMENTACION DEL ESPACION DE NOMBRES ESTANDAR

int main()

{

char text; /\*VARIABLE PARA ALMACENAR EL CARACTER QUE SE ESTA LEYENDO DE EL ARCHIVO, ESTE MISMO CARACTER ESEL QUE SE ENCRIPTARA\*/

char encript = ''; //CLAVE DE ENCRIPTACION/DESENCRIPTACION

cout<<"1)Encriptar"<<endl<<"2)Desencriptar"<<endl; /\*PREGUNTA SI SE QUIERE ENCRIPTAR O DESENCRIPTAR\*/

int opc; //ALMACENA LA OPCION ELEJIDA (ENCRIPTAR O DESENCRIPTAR)

cin>>opc;

if(opc==1) //SI ELIJE ENCRIPTAR

{

ifstream original("Original.txt"); /\*ABRE UN ARCHIVO txt Y LO DECLARA COMO ARCHIVO DE LECTURA PARA OBTENER EL TEXTO A ENCRIPTAR\*/

ofstream encriptado("Encriptado.txt"); /\*CREA UN ARCHIVO txt Y LO DECLARA COMO ARCHIVO DE ESCRITURA PARA ESCRIBIR EL TEXTO ENCRIPTADO\*/

if(original.fail()) /\*SI FALLA LA APERTURA DEL ARCHIVO DE LECTURA SE INDICA QUE NO EXISTE\*/

{

cout<<"NO EXISTE EL ARCHIVO Original.txt";

system("pause");

return 0;

}

original.get(text); /\*OBTIENE UN CARACTER DEL ARCHIVO DE LECTURA Y LO ALMACENA EN LA VARIABLE text\*/

while(!original.eof()) //MIENTRAS NO SE LLEGE AL FINAL DEL CONTENIDO

{

text ^= encript; /\*APLICA EL OPERADOR LOGICO XOR ENTRE CARACTER OBTENIDO Y LA CLAVE DE ENCRIPTACION Y EL RESULTADO LO ALMACENA DE NUEVO EN LA VARIABLE text\*/

encriptado<<text; /\*ESCRIBE EL CARACTER ENCRIPTADO EN EL ARCHIVO DE ESCRITURA\*/

original.get(text); /\*OBTIENE EL SIGUIENTE CARACTER DEL ARCHIVO DE LECTURA Y LO ALMACENA EN LA VARIABLE text\*/

}

original.close(); //CIERRA EL ARCHIVO DE LECTURA

encriptado.close(); /\*CIERRA EL ARCHIVO DE ESCRITURA (NESESARIO PARA QUE SE GUARDEN LOS CAMBIOS)\*/

}

else if(opc==2) //SI ELIJE DESENCRIPTAR

{

ifstream leerencriptado("Encriptado.txt"); /\*ABRE UN ARCHIVO txt Y LO DECLARA COMO ARCHIVO DE LECTURA PARA OBTENER EL TEXTO A DESENCRIPTAR\*/

ofstream desencriptado("Desencriptado.txt"); /\*CREA UN ARCHIVO txt Y LO DECLARA COMO ARCHIVO DE ESCRITURA PARA ESCRIBIR EL TEXTO DESENCRIPTADO\*/

if(leerencriptado.fail()) /\*SI FALLA LA APERTURA DEL ARCHIVO DE LECTURA SE INDICA QUE NO EXISTE\*/

{

cout<<"NO EXISTE EL ARCHIVO Encriptado.txt";

system("pause");

return 0;

}

leerencriptado.get(text); /\*OBTIENE UN CARACTER DEL ARCHIVO DE LECTURA Y LO ALMACENA EN LA VARIABLE text\*/

while(!leerencriptado.eof()) /\*MIENTRAS NO SE LLEGE AL FINAL DEL CONTENIDO\*/

{

text ^= encript; /\*APLICA EL OPERADOR LOGICO XOR ENTRE CARACTER OBTENIDO Y LA CLAVE DE DESENCRIPTACION\*/

desencriptado<<text; /\*ESCRIBE EL CARACTER DESENCRIPTADO EN EL ARCHIVO DE ESCRITURA\*/

leerencriptado.get(text); /\*OBTIENE EL SIGUIENTE CARACTER DEL ARCHIVO DE LECTURA Y LO ALMACENA EN LA VARIABLE text\*/

}

leerencriptado.close(); //CIERRA EL ARCHIVO DE LECTURA

desencriptado.close(); /\*CIERRA EL ARCHIVO DE ESCRITURA (NESESARIO PARA QUE SE GUARDEN LOS CAMBIOS)\*/

}

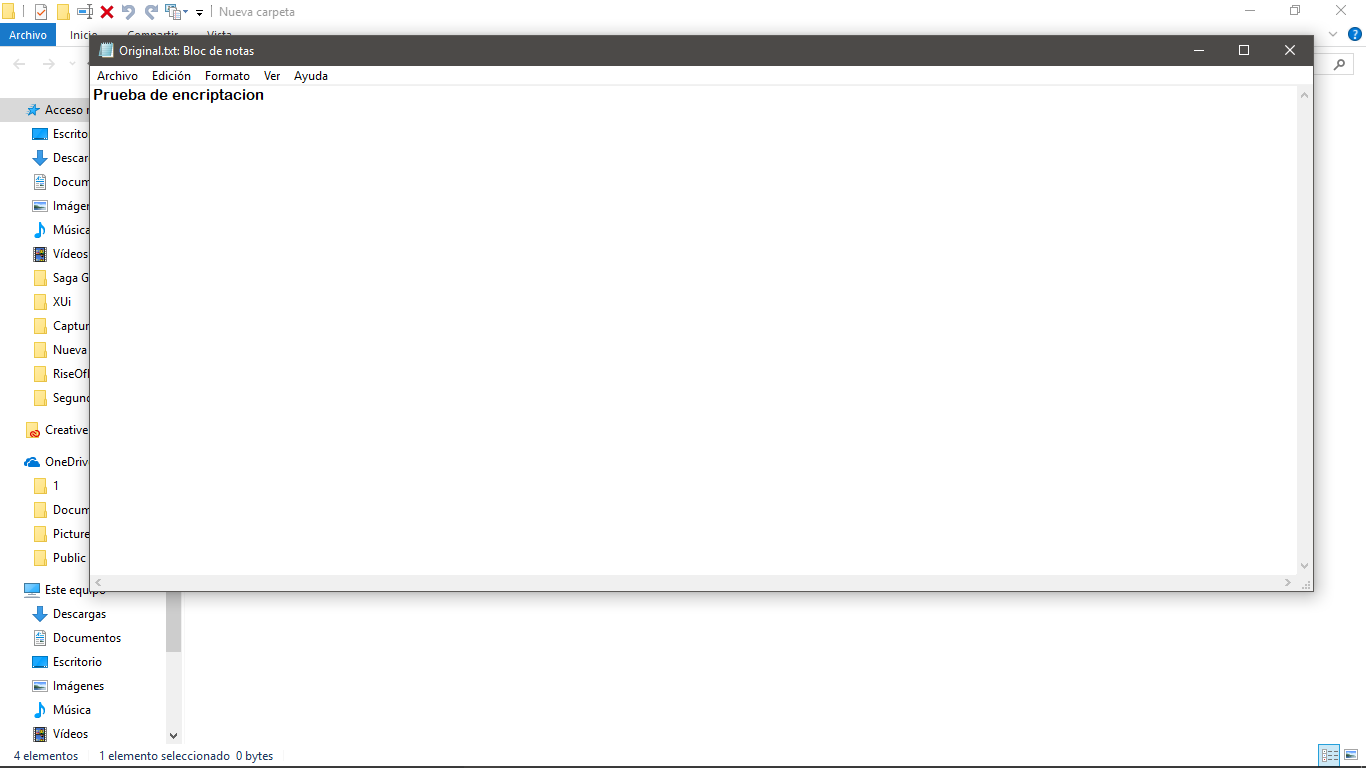
system("pause");

return 0;

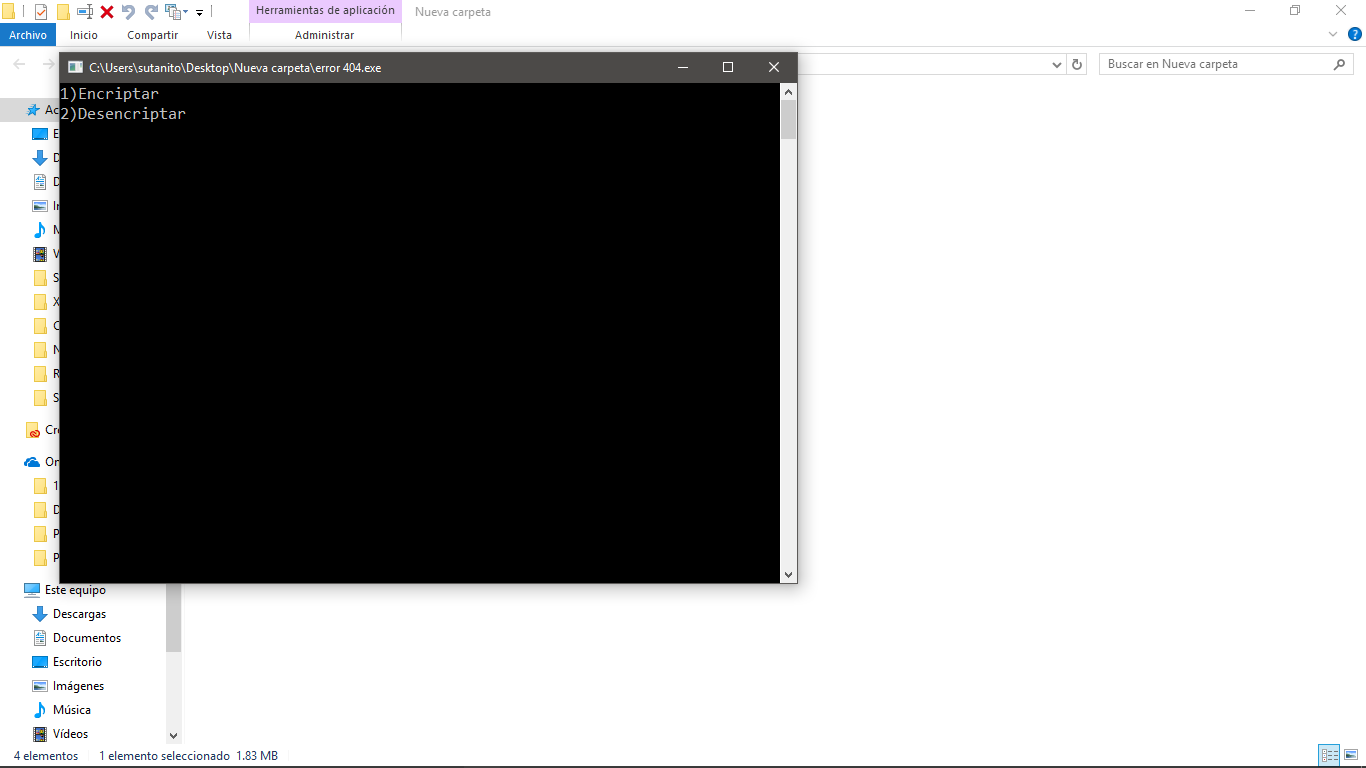
}

**Capturas**

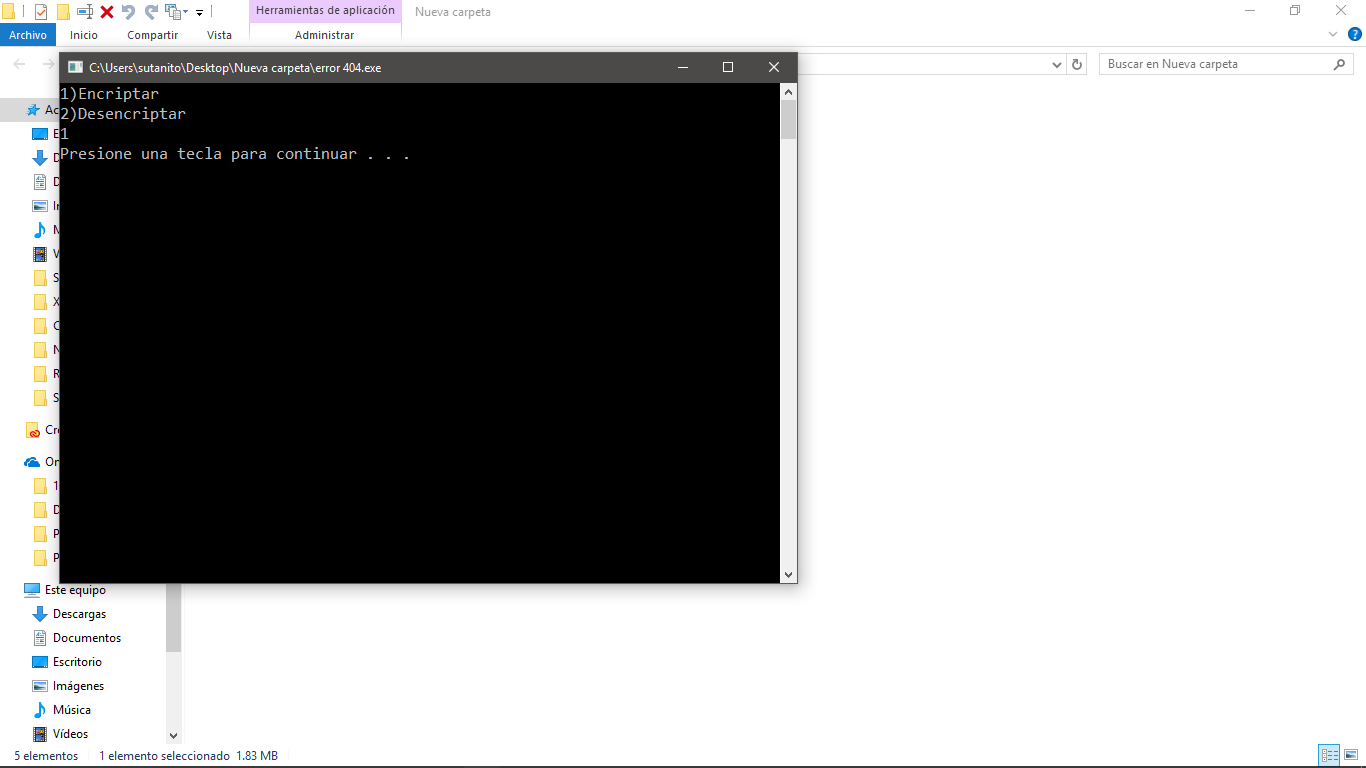
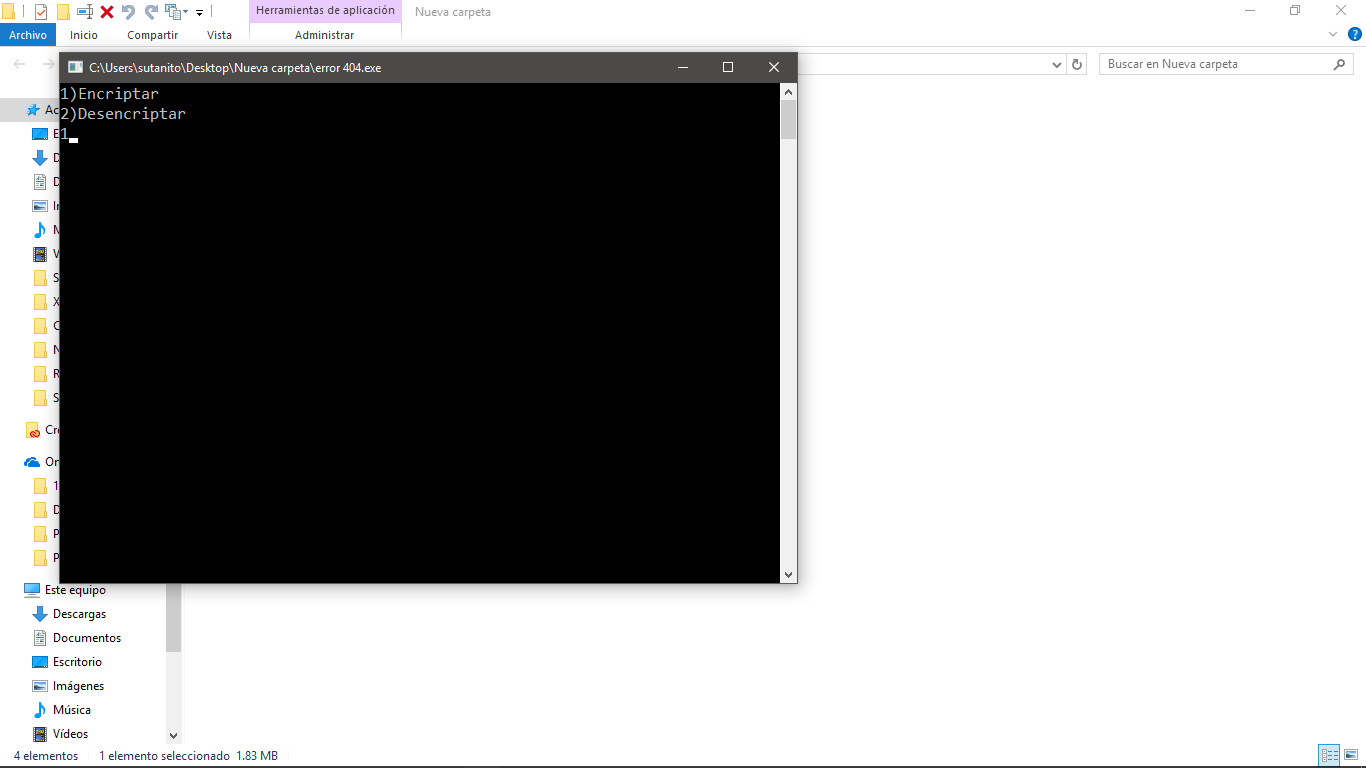
1Texto a encriptar



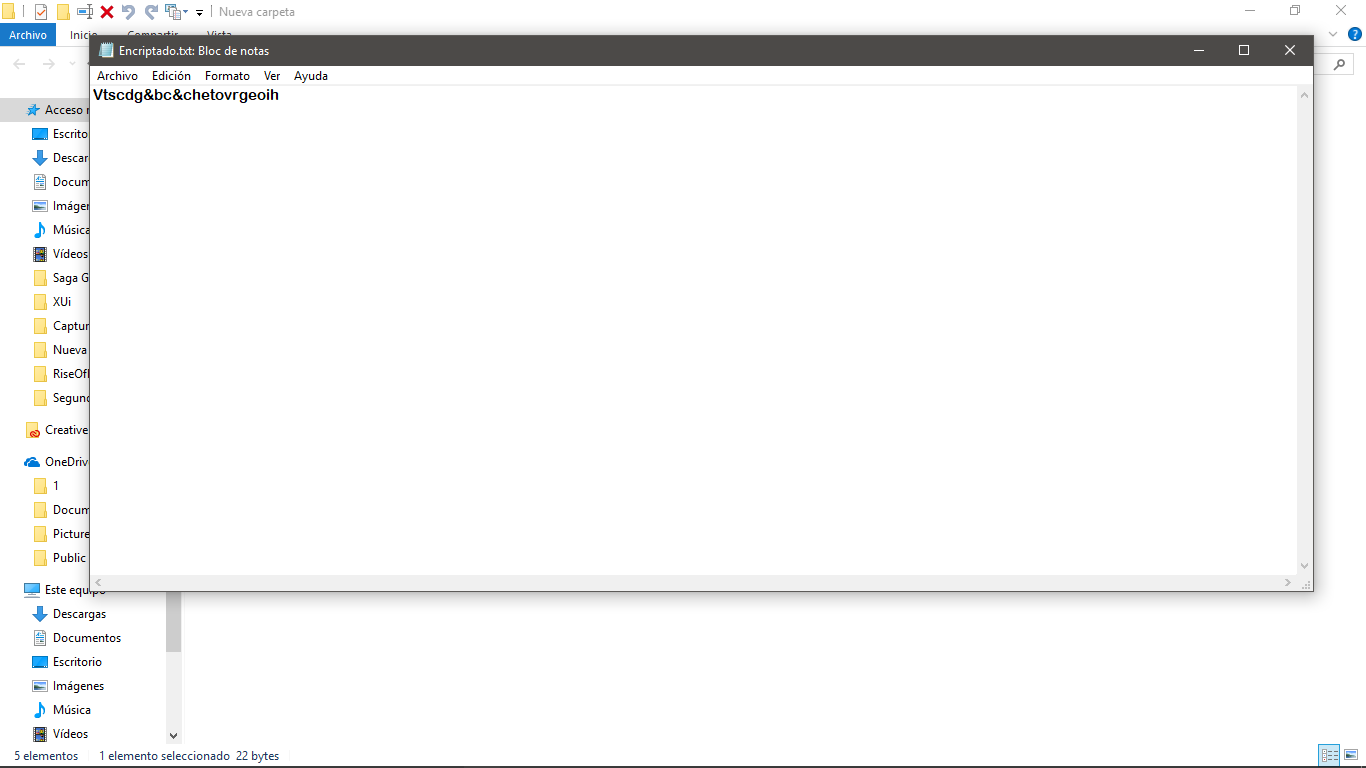
2 Aplicacion de encriptacion



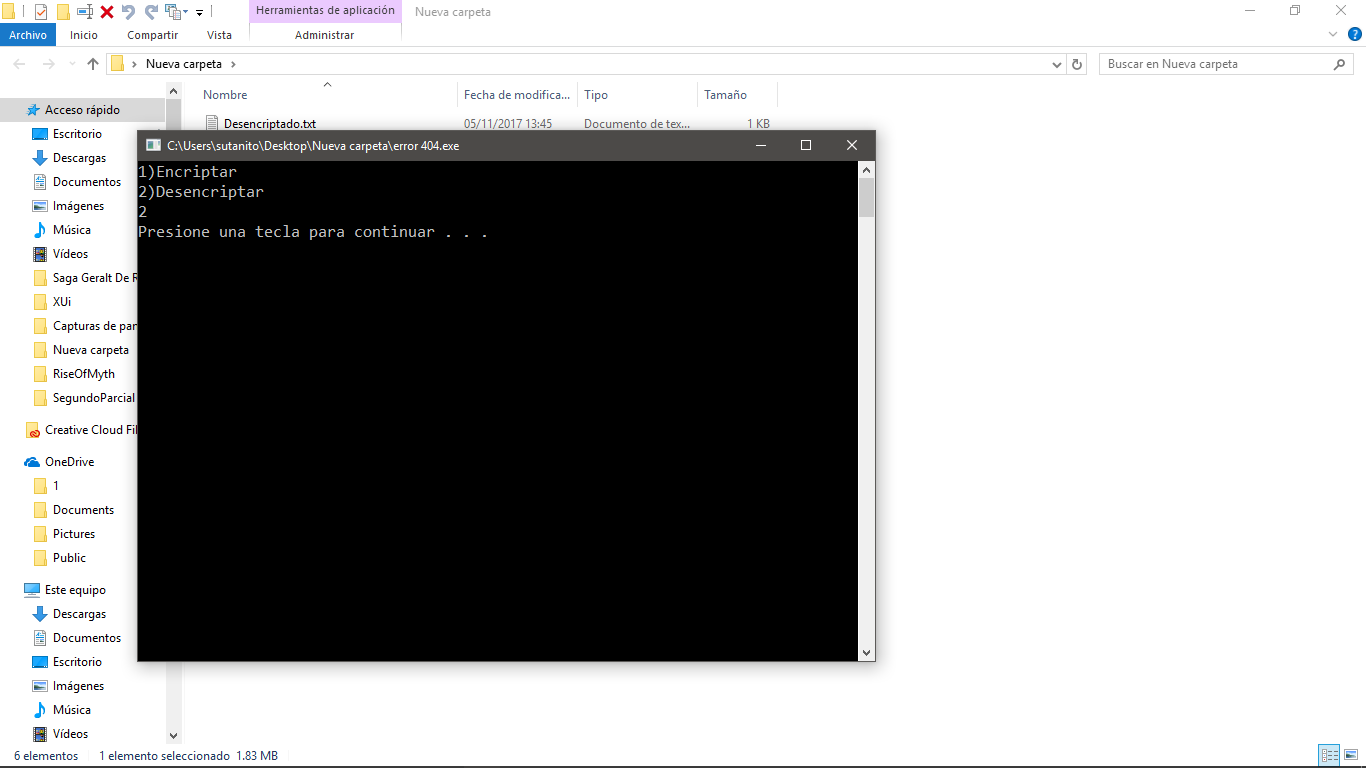
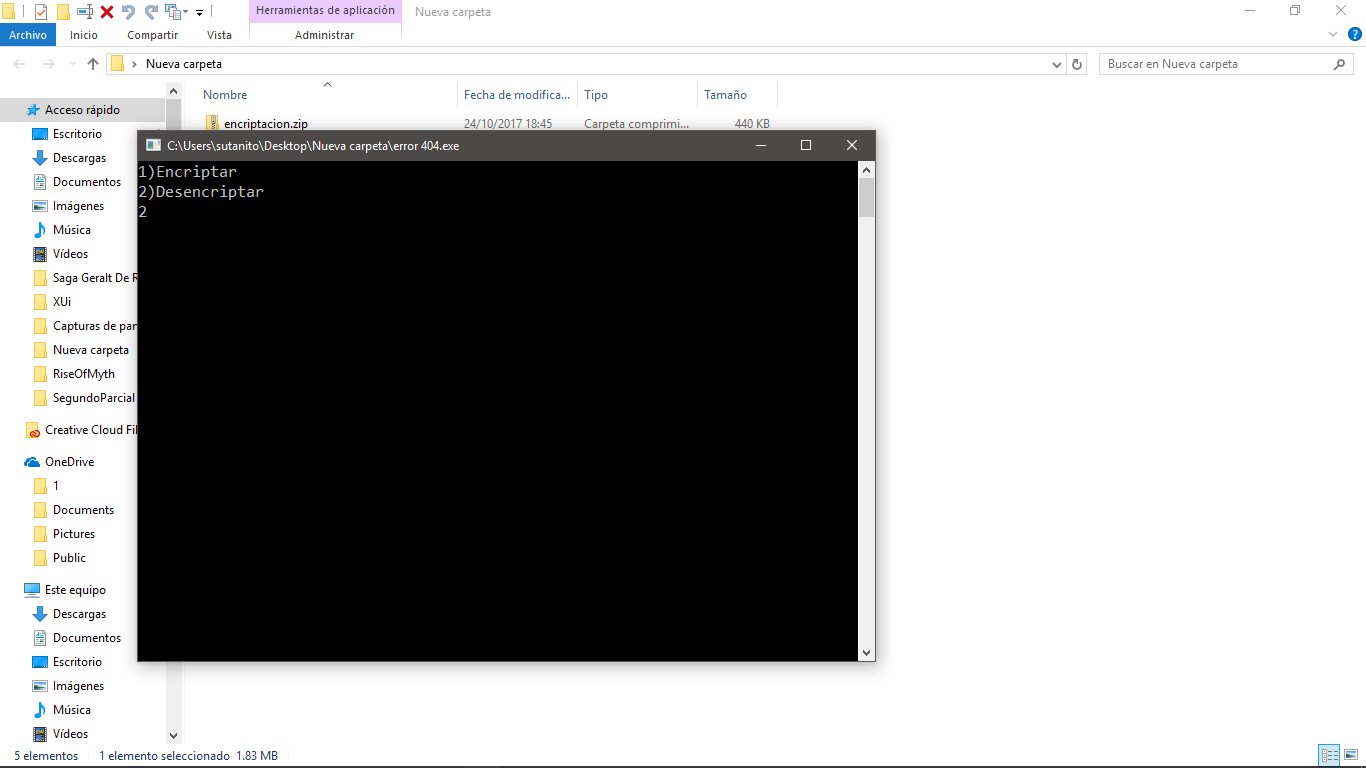
3Opcion de encriptar



4Texto Encriptado



5Opcion de desencriptar



6 Texto desencriptado

