CSE2005- OS PROJECT REPORT
(Jth Component)

FILE SYSTEM PROTECTION USING ENCRYPTION METHOD

Submitted by:
Sushant Srivastav (19BCE2662)
Bhavya Harchandani (19BCE2016)
Srivathsan Nayak(19BCE2015)
BATCH ID: A1

in partial fulfillment for the award of the degree of

B.Tech

IN

COMPUTER SCIENCE ENGINEERING
(SCOPE)

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Under the guidance of:
Prof. **Deepa. K**
Assistant Professor (Senior)
SITE

DATE- 5 June 2021

Abstract: Storage systems are increasingly vulnerable to cyber-attacks. Cryptographic document frameworks reduce the risk of information disclosure by utilizing encryption and integrity security strategies, ensuring end-to-end security for their customers. The value of data stored on digital platforms is rapidly increasing. Furthermore, most information systems are networked and contain a large number of resources such as data, software applications, and business logics, all of which are vulnerable to attacks. This project describes a generic cryptographic file system design and its implementation in a distributed storage-area network (SAN) file system. Many cryptographic algorithms are available to provide security to such information systems. The most well-known and widely used cryptographic scheme is the **Data Encryption Standard**, which is a symmetric key block cipher algorithm. DES was a popular cryptosystem for encrypting sensitive data transmissions. **Simplified DES (SDES)** was created solely for educational purposes, to assist students in learning about modern cryptanalytic techniques.

Keywords: Data Encryption Standard (DES), Simplified DES (SDES), Encryption, Decryption, Cryptography, Operating System

1. Introduction

Data encryption is the process of converting data from a readable **(plaintext)** format to an unreadable, encoded format **(ciphertext).** Data that has been encrypted can only be read or processed after it has been decrypted with a decryption key or password. The decryption key should only be accessible to the sender and recipient of the data. The **Data Encryption Standard (DES)** is the most widely used cryptosystem for information protection around the world. **DES** is a Feistel Cipher implementation. It employs a 16-round Feistel structure. The **DES algorithm** is a 64-bit block cipher with a key of 56 bits. Although the key length is 64 bits, DES has an effective key length of 56 bits because the encryption algorithm does not use 8 of the 64 bits of the key (function as check bits only). **SDES** has similar properties and structure to DES but has been simplified to make encryption and decryption much easier to perform by hand with pencil and paper. Some argue that learning **SDES** provides insight into DES and other block ciphers, as well as various cryptanalytic attacks against them.

2. Objective

1. Understanding the importance of data security and integrity in the design of a distributed operating system.

2. Types of cryptographic modules that are used to store and authorize users in different operating system data.

3. For the **Windows** platform, use DES to implement a file encryption technique.

4. Comparing the time and space characteristics of decrypted and encrypted files

3. System Requirements

Recommended **RAM** Size: 4 GB.

Recommended **Hard Drive** Space: 20 GB

OS: Windows 8.1 or higher.

Recommended **Processor** Type: Intel Core i3 or higher.

5. Literature Review

1. **Seung-Jo Han, Jongan Park, Heang-Soo Oh, "The improved data encryption standard (DES) algorithm", Proceedings of ISSSTA'95 International Symposium on Spread Spectrum Techniques and Applications, IEEE, 25 Sept. 1996**

   This article proposes the design of an improved DES cryptosystem. The new algorithm is improved DES. They show that the improved DES is stronger than the DES for cryptographic security differential cryptanalysis. They divide a single data block (96 bits) into three 32-bit sub blocks, then perform different f-function functions on each of the three sub blocks, then increase the S/sub 1/-S/sub 8/ of the S-boxes to S/sub 1/-S/sub 16/ and the coefficient of correlation (p/sub I j/) to meet the strict avalanche criteria (SAC: p/sub I j/). Finally, the key length is told to be increased to 112 bits. The analysis showed that the UD in the improved DES is higher than the DES UD.

2. **A M H Pardede, Anggi Pramana, Muhammad Zarlis, Akbar Iskandar, Rosida Tiurma Manurung, S Sriadhi, Citra Kurniawan, Abdurrozzaq Hasibuan, Nurintan Asyiah Siregar, Ayu Esteka Sari, Indrya Mulyaningsih, Teguh Hidayat Iskandar Alam, Ibrahim, Tri Listyorini, Edy Winarno and Tulus, "Designing the Application of Security Text Messages Into Audio Files Using Data Encryption Standard (DES) Algorithms Using the End Of File (EOF) Method", IOP Publishing, 1363 (2019) 012078**

   The science and art of hiding information/messages in the media is steganography in a way that other parties that do not have a right to information do not determine its existence. Cryptography instead disguises the meaning of a message but does not hide the message as the file is suspected. The technique of steganography used was End of File (EOF). EOF describes at the end of the image file how to add data or files. The data or files that are being hidden for this technique could be larger than the image file size. The hidden data will be appended to the end of the file, where it will have no effect on the image. In addition, at the time of inserting data that functions as a generator code and encrypts data, this steganography application has a cryptographing function (Data Encryption Standard (DES)), so as to provide more protection to and from data security in the file against those not allowed to know the data.

3. **Kuntal Patel, "Performance analysis of AES, DES and Blowfish cryptographic algorithms on small and large data files", Int. j. inf. tecnol. (2019), 1 January 2019**

   This paper examines three widely used cryptographic algorithms: AES, DES, and Blowfish. These are well-known symmetric key cryptographic algorithms that can be used to secure IT systems. The primary goal of this research paper is to compare the performance of these algorithms on small and large data sets. The execution time and memory used by these algorithms during implementation are used to compare their performance. The experimental results and reports show which algorithm is better suited to small and large data files. Analytical results also indicate which algorithm is best suited for time and memory constrained systems.

4. **Lu Chang and Xu GuangMing - "Research and Implementation of File Encryption and Decryption", Zhejiang Ocean University, China -2012**

   In the process of design of file encryption and decryption, the system of password is an important guarantee for the security of files. This paper shows the effective encryption and decryption of the contents of file operations. The DES and MD5 algorithms have been used to encrypt and decrypt the keys used in the process of transmission and preservation. The DES algorithm converts 64-bit plain text into 64-bit ciphertext and splits it into left and right parts of each 32 bits. Then 16 rounds are made in which the key is combined with the text. After that, left and right are obtained through the replacement of the inverse of the initial replacement, that is, the ciphertext. To conclude, this system can easily encrypt and decrypt a large variety of files like txt, doc, etc., and precisely restore the content of encrypted file, protected personal information to a degree.

5. **Purvi Garg, Shivangi Varshney, Manish Bhardwaj - "Cryptanalysis of Simplified Data Encryption Standard Using Genetic Algorithm" April 24, 2015, SRM University, Modinagar, India**

   Cryptography is basically the science and art of transforming messages to make them secure and immune to attacks. But we cannot completely immune ciphers from different attacks and this attacking or breaking of ciphers is termed as cryptanalysis. In this process, the intruder converts the ciphertext into plaintext with or without knowing the key. Cryptographic systems have a finite key space so that the intruder can easily search for a key, but still the system remains secure because of the size of the key search space. And hence, optimization techniques have got significant importance in finding the solution (key), by optimizing key search space. In this paper, the working of the cryptanalysis of Simplified Data Encryption Standard (SDES), using Genetic Algorithm and Brute Force has been explained.

6. **Manjula K G, M N Ravikumar, "Color Image Encryption and Decryption using DES Algorithm", IRJET, July 2016**

Data Encryption Standard (DES) algorithm is a popular cryptographic scheme, and it is a type of symmetric key block cipher algorithm. It can be used for securing classified data transmissions as well. Here, DES algorithm is used for image file encryption and decryption. In encryption method, two inputs are considered - the encryption secret key and the encrypted image file. The decryption method gives the output as a decrypted image file, or the original image file. Through experimental results it is explained how DES algorithm could be utilized as a secure algorithm.

7. **Indumathi Saikumar, "DES - Data Encryption Standard", IRJET, March 2017**

DES is an extensively used encryption algorithm standard. Encryption and decryption are parts of cryptography, and cryptography techniques are used to hide information by using DES, which applies cipher algorithm to each data block of the code. This paper deals with the implementation of DES algorithm. DES has increased security levels due to multiple rounds of operation, which makes it difficult for the unauthorized party to attack and crack.

8. **Harshali D Zodpe, Prakash W Wani, Rakesh R Mehta, "Design and Implementation of Algorithm for DES Cryptanalysis", International Conference of Hybrid Intelligent Systems, 2012**

Cryptanalysis of block ciphers involves many computations which are independent of each other and can be instantiated simultaneously so that the solution space is explored at a faster rate. This paper covers the design for hardware implementation of DES cryptanalysis on Field Programmable Gate Arrays (FPGAs) using exhaustive key search. Two architectures of DES are implemented - iterative and loop unrolled. The objective is to make cryptanalysis more accurate.

5. Scope

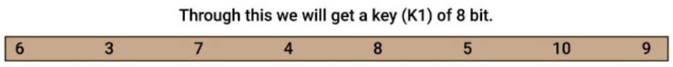
The scope of this project is to provide a platform to our users for securing their texts by encrypting them and storing them in files which can be decrypted any time they want.

First of all, users fulfill the agreement form of this program, then user insert a text message or include text message file. Encryption key as an input as a result, this program will encrypt and decrypt this text massage and save it in a file.

6. Analysis of SDES Algorithm

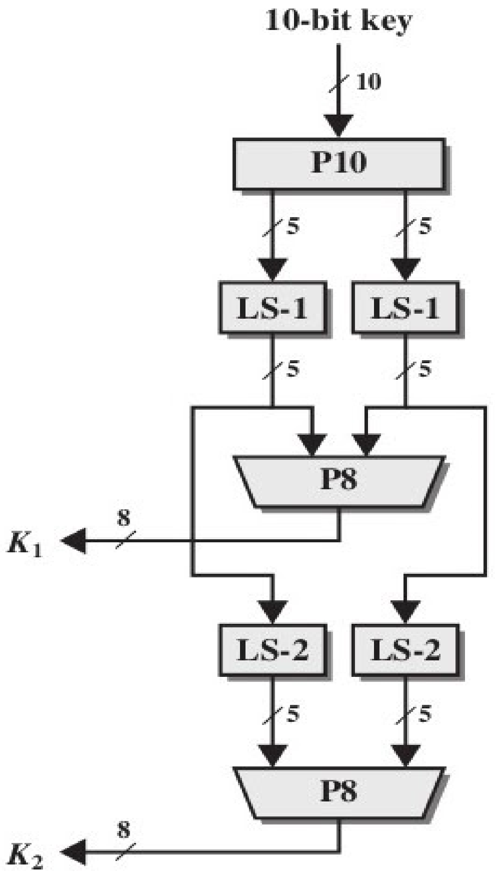
KEY GENERATOR

An input key (encryption key) is taken from the user, and this key is then converted into 10-bit binary key. After that, this 10-bit binary key is divided into 5-5 bits. Then we proceed with LS-1(LEFT [1 bit]) of these 5-5 bits. Then these bits are merged and arranged according to the following rule:


Through this we will get a key (K1) of 8 bit.

6	3	7	4	8	5	10	9

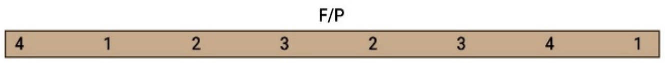
Now, from these 5-5 bits of (LS-1), along with the generating the key (K1) , LS-2(left shift [2bit]) is performed with the help of above rule (P8), we have merged(LS-2), and we get the key(key2).



Flow Chart for Key Generation

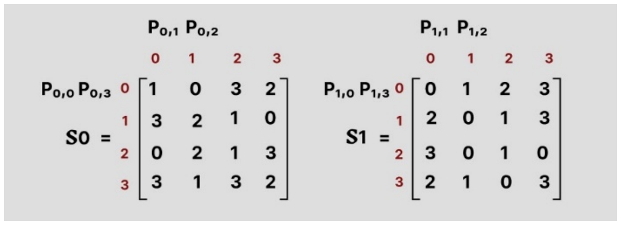
ENCRYPTION DETAIL

For encryption, let us take any alphabet or integer from the user and convert it into an 8 bit (IP BINARY), then divide these 8-bit binary into 4- 4 bit binary as left 4 bit and right 4-bit IP. The right 4-bit binary is then converted into 8 bits binary and arranged according to the rule:


F/P

4	1	2	3	2	3	4	1

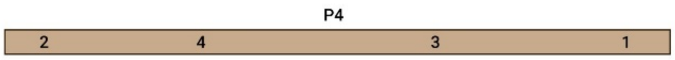
Now, this will be added with key 1 which was generated earlier. Then these 8 bits are divided into 4- 4 bits. From the first 4 bits we get S0 and from the remaining 4 bits we get S1.



$$P_{0,1}\,P_{0,2} \qquad\qquad P_{1,1}\,P_{1,2}$$

$$P_{0,0}\,P_{0,3}\quad S0 = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 0 & 3 & 2 \\ 1 & 3 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 & 3 \\ 3 & 3 & 1 & 3 & 2 \end{array} \qquad P_{1,0}\,P_{1,3}\quad S1 = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 1 & 3 \\ 2 & 3 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 & 3 \end{array}$$

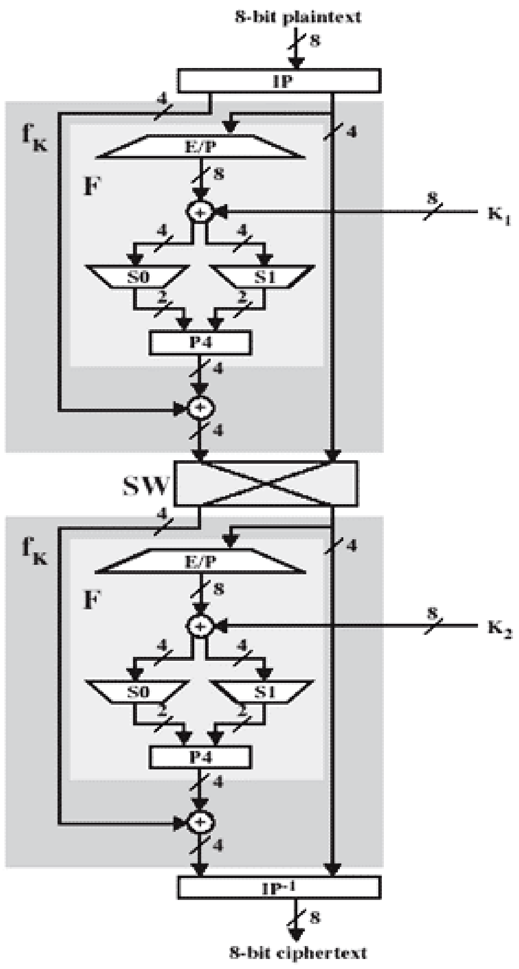
After getting S0 and S1, we will get 2-2 bit binary and arrange them according to Rule P4.



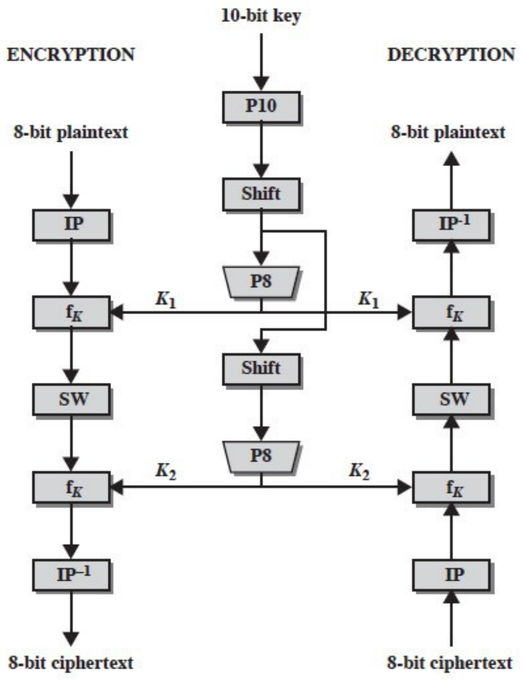
P4

2	4	3	1

Now, this left 4 bit is added with the arranged 2-2 bit binary. This method is repeated by using a switch function. Here we use key 2 instead of key 1. Therefore, we can encrypt the user's message.



STRUCTURE OF ENCRYPTION AND DECRYPTION



7. Code

```
#include <iostream>
#include <fstream>

int f99=0,f98=0,f97=0;
using namespace std;
void binary(int num,int bin[])
{

int i,loop=9;
while(loop>=0)
{
bin[loop]=num%2;
num=num/2;
loop--;
}
}
void p10_arrange(int bin[],int p10[])
{
   int i;
   p10[0]=bin[2];
   p10[1]=bin[4];
   p10[2]=bin[1];
   p10[3]=bin[6];
   p10[4]=bin[3];
   p10[5]=bin[9];
   p10[6]=bin[0];
   p10[7]=bin[8];
   p10[8]=bin[7];
   p10[9]=bin[5];
   if(f98==0)
   {

   cout<<"After Permutation ";
   for(i=0;i<=9;i++)
   {
      cout<<bin[i];
   }
   cout<<endl<<endl;
   f98=1;
   }
}
void LS1(int p10[],int left_LS1[],int right_LS1[])
{
      int i;
      left_LS1[0]=p10[1];
      left_LS1[1]=p10[2];
      left_LS1[2]=p10[3];
      left_LS1[3]=p10[4];
      left_LS1[4]=p10[0];
      right_LS1[0]=p10[6];
      right_LS1[1]=p10[7];
      right_LS1[2]=p10[8];
      right_LS1[3]=p10[9];
      right_LS1[4]=p10[5];
      if(f97==0)
      {
      cout<<"After Left Part Left Shift ";
      for(i=0;i<=4;i++)
      {
         cout<<left_LS1[i];
      }
      cout<<endl<<endl;
      cout<<"After Right Part Left Shift ";
      for(i=0;i<=4;i++)
      {
         cout<<right_LS1[i];
      }
      cout<<endl<<endl;
      f97=1;
      }
}
void p8(int left_LS1[],int right_LS1[],int subkey1[])
{

   subkey1[0]=right_LS1[0];
   subkey1[1]=left_LS1[2];
   subkey1[2]=right_LS1[1];
   subkey1[3]=left_LS1[3];
   subkey1[4]=right_LS1[2];
   subkey1[5]=left_LS1[4];
   subkey1[6]=right_LS1[4];
   subkey1[7]=right_LS1[3];
}
void LS2(int left_LS1[],int right_LS1[],int left_LS2[],int right_LS2[])
{
   left_LS2[0]=left_LS1[2];
   left_LS2[1]=left_LS1[3];
   left_LS2[2]=left_LS1[4];
   left_LS2[3]=left_LS1[0];
```

```
   left_LS2[4]=left_LS1[1];
   right_LS2[0]=right_LS1[2];
   right_LS2[1]=right_LS1[3];
   right_LS2[2]=right_LS1[4];
   right_LS2[3]=right_LS1[0];
   right_LS2[4]=right_LS1[1];
}
void key_generator(int num,int subkey1[],int subkey2[])
{
   int i,bin[10];
   binary(num,bin);
   if(f99==0)
   {
   for(i=0;i<=9;i++)
   {
         cout<<bin[i]<<" ";
   }
   f99=1;
         cout<<endl<<endl;
   }

   int p10[10];
   p10_arrange(bin,p10);
   int left_LS1[5],right_LS1[5];
   LS1(p10,left_LS1,right_LS1);
   p8(left_LS1,right_LS1,subkey1);
   int left_LS2[5],right_LS2[5];
   LS2(left_LS1,right_LS1,left_LS2,right_LS2);
   p8(left_LS2,right_LS2,subkey2);
}
   void IP_arrange(int bin[],int ip_arrange[])
{
      ip_arrange[0]=bin[3];
      ip_arrange[1]=bin[7];
      ip_arrange[2]=bin[4];
      ip_arrange[3]=bin[2];
      ip_arrange[4]=bin[5];
      ip_arrange[5]=bin[9];
      ip_arrange[6]=bin[6];
      ip_arrange[7]=bin[8];
}
void seperate(int ip_arrange[],int left_ip[],int right_ip[])
{
   left_ip[0]=ip_arrange[0];
   left_ip[1]=ip_arrange[1];
   left_ip[2]=ip_arrange[2];
   left_ip[3]=ip_arrange[3];
   right_ip[0]=ip_arrange[4];
   right_ip[1]=ip_arrange[5];
   right_ip[2]=ip_arrange[6];
   right_ip[3]=ip_arrange[7];
}
void FP_arrange(int right_ip[],int fp[])
{

   fp[0]=right_ip[3];
   fp[1]=right_ip[0];
   fp[2]=right_ip[1];
   fp[3]=right_ip[2];
   fp[4]=right_ip[1];
   fp[5]=right_ip[2];
   fp[6]=right_ip[3];
   fp[7]=right_ip[0];
}
void zor(int fp[],int subkey1[],int left_zor[4],int right_zor[4])
{
if(fp[3]==subkey1[0])
   left_zor[0]=0;
else if (fp[3]!=subkey1[0])
   left_zor[0]=1;
if(fp[0]==subkey1[1])
   left_zor[1]=0;
else if (fp[0]!=subkey1[1])
   left_zor[1]=1;
if(fp[1]==subkey1[2])
   left_zor[2]=0;
else if (fp[1]!=subkey1[2])
   left_zor[2]=1;
if(fp[2]==subkey1[3])
   left_zor[3]=0;
else if (fp[2]!=subkey1[3])
   left_zor[3]=1;
///// right_zor /////
if(fp[1]==subkey1[4])
   right_zor[0]=0;
else if (fp[1]!=subkey1[4])
   right_zor[0]=1;
if(fp[2]==subkey1[5])
   right_zor[1]=0;
else if (fp[2]!=subkey1[5])
   right_zor[1]=1;
if(fp[3]==subkey1[6])
```

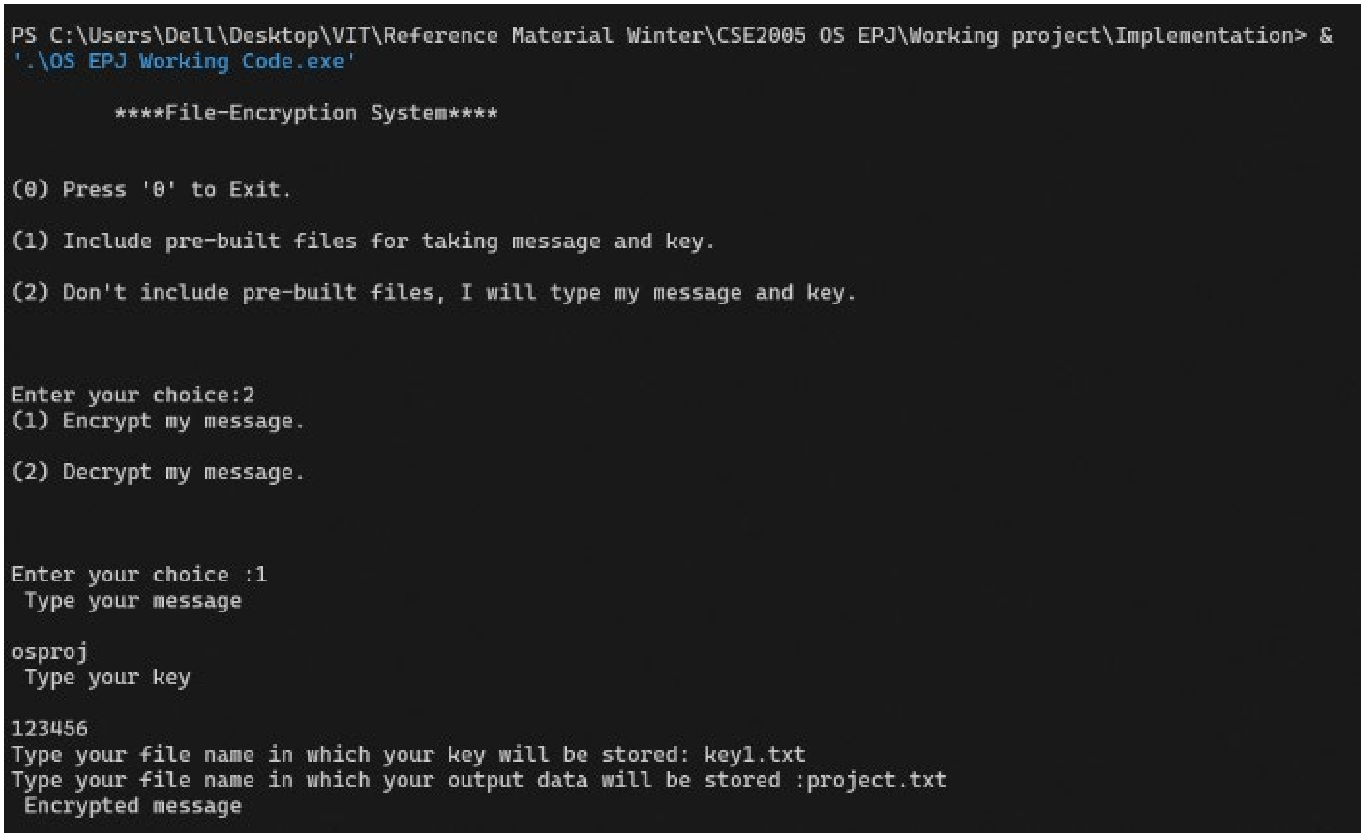
```
    right_zor[2]=0;
else if (fp[3]!=subkey1[6])
    right_zor[2]=1;
if(fp[0]==subkey1[7])
    right_zor[3]=0;
else if (fp[0]!=subkey1[7])
    right_zor[3]=1;
}
void S_box(int left_zor[],int right_zor[],int s0[] ,int s1[] )
{
    int s_box0[4][4]={{1,0,3,2},
    {3,2,1,0},{0,2,1,3},{3,1,3,2}};
    int row=(left_zor[0]*2)+(left_zor[3]*1),col=(left_zor[1]*2)+(left_zor[2]*1);
    int num;
    num=s_box0[row][col];
    for(int l=1;l>=0;l--)
    {
        s0[l]=num%2;
        num=num/2;
    }
    int s_box1[4][4]={{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
    int row1=(right_zor[0]*2)+(right_zor[3]*1),col1=(right_zor[1]*2)+(right_zor[2]*1);
    num=s_box1[row1][col1];
    for(int l=1;l>=0;l--)
    {
        s1[l]=num%2;
        num=num/2;
    }
}
void P4_arrange(int s0[],int s1[],int p4[])
{
    p4[0]=s0[1];
    p4[1]=s1[1];
    p4[2]=s1[0];
    p4[3]=s0[0];
}
void P4_zor(int left_ip[],int p4[],int sw_right_ip[])
{
for(int l=0;l<4;l++)
{
    if(left_ip[l]==p4[l])
        sw_right_ip[l]=0;
    if(left_ip[l]!=p4[l])
        sw_right_ip[l]=1;
}
}
```

```
void Ip_invers(int sw_zor[],int sw_right_ip[],int ip_inv[])
{
   ip_inv[0]=sw_zor[3];
   ip_inv[1]=sw_zor[0];
   ip_inv[2]=sw_zor[2];
   ip_inv[3]=sw_right_ip[0];
   ip_inv[4]=sw_right_ip[2];
   ip_inv[5]=sw_zor[1];
   ip_inv[6]=sw_right_ip[3];
   ip_inv[7]=sw_right_ip[1];
}
int E(int leter_num,int key_num )
{
   int subkey1[8],subkey2[8];
   key_generator(key_num,subkey1,subkey2);
   int bin[10];
   binary(leter_num,bin);
   int ip_arrange[8];
   IP_arrange(bin,ip_arrange);
   int left_ip[4],right_ip[4];
   seperate(ip_arrange,left_ip,right_ip);
   int fp[8];
   FP_arrange(right_ip,fp);
   int left_zor[4],right_zor[4];
   zor(fp,subkey1,left_zor,right_zor);
   int s0[2],s1[2];
   S_box(left_zor,right_zor,s0,s1);
   int p4[4];
   P4_arrange(s0,s1,p4);
   int sw_right_ip[4];
   P4_zor(left_ip,p4,sw_right_ip);
   int sw_fp[8];
   FP_arrange(sw_right_ip,sw_fp);
   int sw_left_zor[4],sw_right_zor[4];
   zor(sw_fp,subkey2,sw_left_zor,sw_right_zor);
   int sw_s0[2],sw_s1[2];
   S_box(sw_left_zor,sw_right_zor,sw_s0,sw_s1);
   int sw_p4[4];
   P4_arrange(sw_s0,sw_s1,sw_p4);
   int sw_zor[4];
   P4_zor(right_ip,sw_p4,sw_zor);
   int ip_inv[8];
   Ip_invers(sw_zor,sw_right_ip,ip_inv);
   int sum;
```

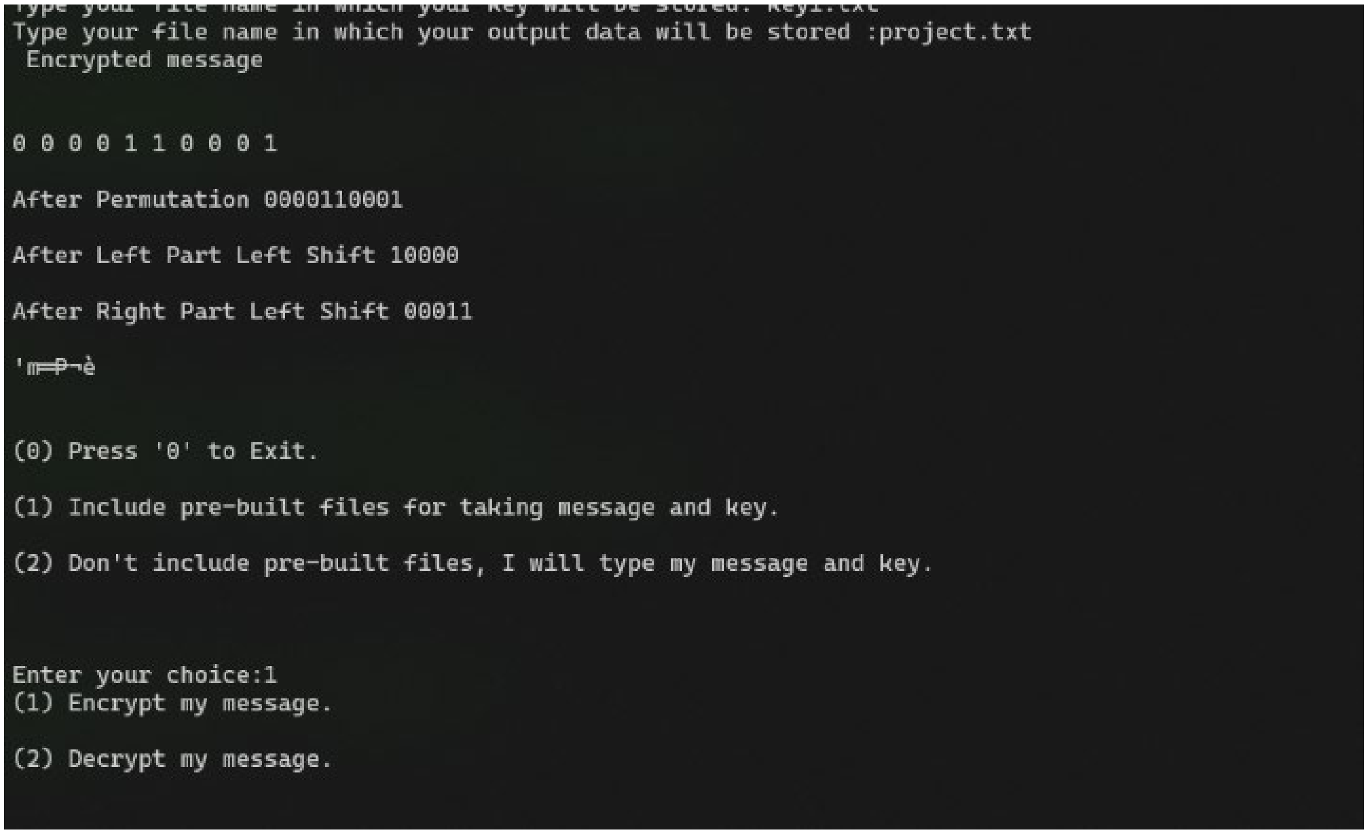
```
sum=(ip_inv[0]*128)+(ip_inv[1]*64)+(ip_inv[2]*32)+(ip_inv[3]*16)+(ip_inv[4]*8)+(ip_inv[5]
*4)+(ip_inv[6]*2)+(ip_inv[7]*1);
   return(sum);
}
int D(int leter_num,int key_num )
{
   int subkey1[8],subkey2[8];
   key_generator(key_num,subkey1,subkey2);
   int bin[10];
   binary(leter_num,bin);
   int ip_arrange[8];
   IP_arrange(bin,ip_arrange);
   int left_ip[4],right_ip[4];
   seperate(ip_arrange,left_ip,right_ip);
   int fp[8];
   FP_arrange(right_ip,fp);
   int left_zor[4],right_zor[4];
   zor(fp,subkey2,left_zor,right_zor);
   int s0[2],s1[2];
   S_box(left_zor,right_zor,s0,s1);
   int p4[4];
   P4_arrange(s0,s1,p4);
   int sw_right_ip[4];
   P4_zor(left_ip,p4,sw_right_ip);
   int sw_fp[8];
   FP_arrange(sw_right_ip,sw_fp);
   int sw_left_zor[4],sw_right_zor[4];
   zor(sw_fp,subkey1,sw_left_zor,sw_right_zor);
   int sw_s0[2],sw_s1[2];
   S_box(sw_left_zor,sw_right_zor,sw_s0,sw_s1);
   int sw_p4[4];
   P4_arrange(sw_s0,sw_s1,sw_p4);
   int sw_zor[4];
   P4_zor(right_ip,sw_p4,sw_zor);
   int ip_inv[8];
   Ip_invers(sw_zor,sw_right_ip,ip_inv);
   int sum;

sum=(ip_inv[0]*128)+(ip_inv[1]*64)+(ip_inv[2]*32)+(ip_inv[3]*16)+(ip_inv[4]*8)+(ip_inv[5]
*4)+(ip_inv[6]*2)+(ip_inv[7]*1);
   return(sum);
}
void get_file_data(char message_file[],char key_file[],char user_message[],char user_key[])
{
   fstream myline(message_file,ios::out | ios::app);
   myline.close();
   int loop=0;
   fstream gline(message_file,ios::in);
   if(!gline)
   cout<<"There is some error so that's why file can not be open .\n\n";
   else
   {
   while(!gline.eof())
   {
   user_message[loop]=gline.get();
   loop++;
   }

   }
   gline.close();
   loop=0;
   fstream key(key_file,ios::in);
   if(!key)
   cout<<"There is some error so that's why file can not be open .\n\n";
   else
   {
   while(!key.eof())
   {
   user_key[loop]=key.get();
   loop++;
   }
   }
   key.close();
}

int edchoice()
{

   char cchoice;
   int ichoice;
   cout<<"(1) Encrypt my message.\n\n";
   cout<<"(2) Decrypt my message.\n\n";
   cout<<"\n\nEnter your choice :";
   cin>>cchoice;
   ichoice=cchoice;
   return(ichoice);
}

void key_store(char user_key[])
{
   char key_file[500];
   int loop=0;
   cout<<"Type your file name in which your key will be stored: "; cin.getline(key_file,500);
   fstream myfile(key_file,ios::out | ios:: app);
   while(user_key[loop]!=-52)
   {
   myfile<<user_key[loop];
   loop++;
   }
   myfile.close();
}

void Encryption_and_Decryption(char user_message[],char user_key[],char ED_choice)
{

   char output_file[500];
   cout<<"Type your file name in which your output data will be stored :";
   cin.getline(output_file,500);
   int line;
   int key;
   char output[99999];
   int loop=0;
   char ch;
   if(ED_choice==49)
   {

      cout<<" Encrypted message \n\n\n";
      while((user_message[loop]!='\0')&&(user_key[loop]!='\0')&&(user_message[loop]!=-
      52)&&(user_key[loop]!=-52))
      {
         line=user_message[loop];
         key=user_key[loop];
         ch=E(line,key);
         output[loop]=ch;
         fstream save_file(output_file,ios::out | ios::app);
         save_file<<ch;
         save_file.close();
         cout<<ch;
         loop++;
      }

   }
   loop=0;
   if(ED_choice==50)
   {
      cout<<" Decrypted message \n\n\n";
      while((user_message[loop]!='\0')&&(user_key[loop]!='\0')&&(user_message[loop]!=-
      52)&&(user_key[loop]!=-52))
      {
         line=user_message[loop];
         key=user_key[loop];
         if(line<0)
         line=line+256;
         ch=D(line,key);
         fstream key(output_file,ios::out | ios::app);
         key<<ch;
         key.close();
         cout<<ch;
         loop++;
      }
   }
}
void message_and_key(int ichoice)
{
   char user_message[99999];
   char user_key[99999];
   char message_file[500];
   char key_file[500];
   int ED_choice;
   ED_choice=edchoice();
   if(ichoice==49)

   {
      cout<<"Enter your message file name :";
      cin.sync();
      cin.getline(message_file,500);
      cout<<"\n\n";
      cout<<"Enter your key file name :";
      cin.getline(key_file,500);
      get_file_data(message_file,key_file,user_message,user_key);
      Encryption_and_Decryption(user_message,user_key,ED_choice);
   }

   else if(ichoice==50)
   {
      cout<<" Type your message\n\n";
      cin.sync();
      cin.getline(user_message,99999);
      cout<<" Type your key\n\n";
      cin.getline(user_key,99999);
      key_store(user_key);
      Encryption_and_Decryption(user_message,user_key,ED_choice);
   }
```

```
}
void file_choice()
{
   char cchoice;
   int ichoice;
   cout<<"\n\t****FILE SYSTEM PROTECTION USING ENCRYPTION
METHOD****\n\n";

   cout<<"\n(0) Press '0' to Exit.\n\n";
   cout<<"(1) Include pre-built files for taking message and key.\n\n";
   cout<<"(2) Don't include pre-built files, I will type my message and key.\n\n";
   cout<<"\n\nEnter your choice:";
   cin>>cchoice;
   while (cchoice!='0')
{

   ichoice=cchoice;
   message_and_key(ichoice);
   cout<<"\n\n\n(0) Press '0' to Exit.\n\n";
   cout<<"(1) Include pre-built files for taking message and key.\n\n";
   cout<<"(2) Don't include pre-built files, I will type my message and key.\n\n";
   cout<<"\n\nEnter your choice:";

   cin>>cchoice;
}
}
int main()
{
   char cchoice;
   int ichoice;
   cchoice='1';
   ichoice=cchoice;
   if(ichoice==49)
   {
      file_choice();
   }
   return 0;
}
```

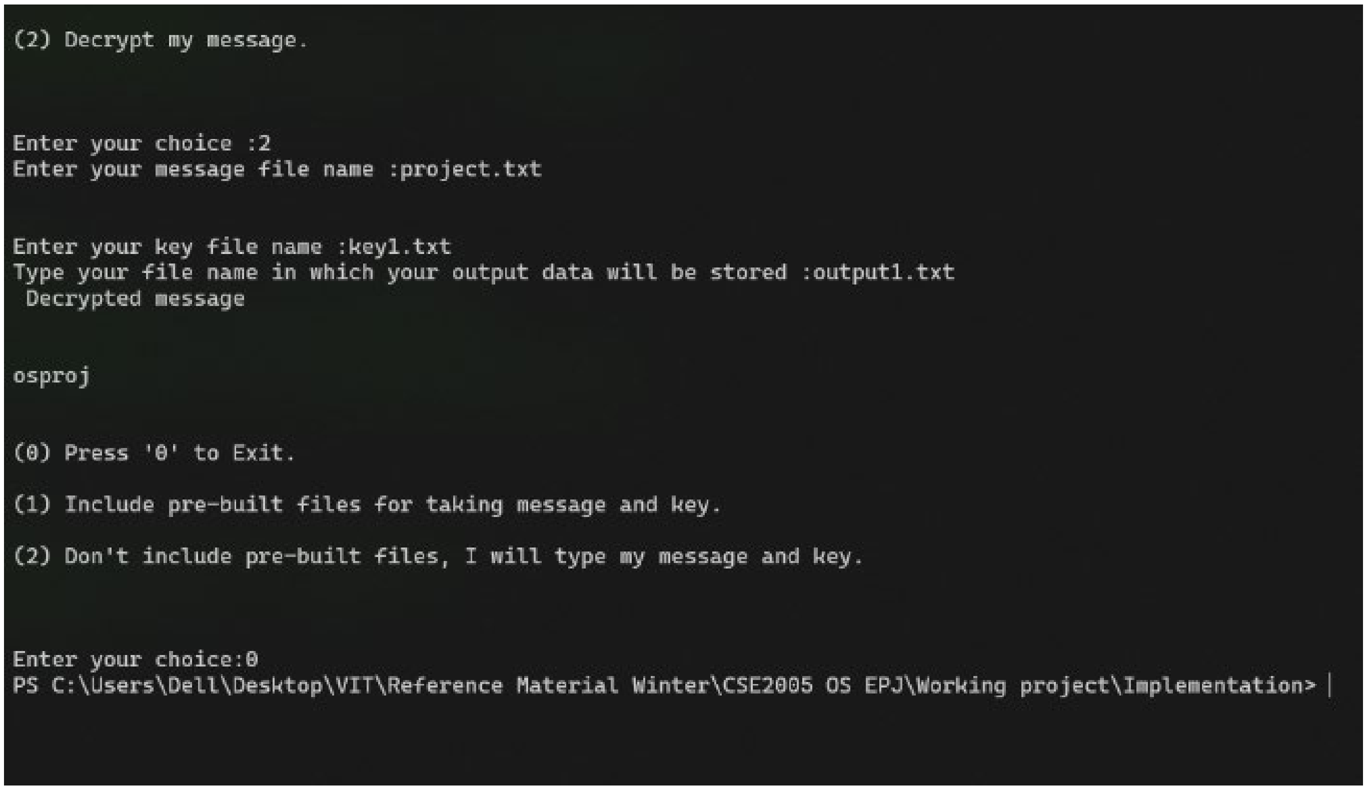
8. Output



```
PS C:\Users\Dell\Desktop\VIT\Reference Material Winter\CSE2005 OS EPJ\Working project\Implementation> &
'.\OS EPJ Working Code.exe'

        ****File-Encryption System****


(0) Press '0' to Exit.

(1) Include pre-built files for taking message and key.

(2) Don't include pre-built files, I will type my message and key.



Enter your choice:2
(1) Encrypt my message.

(2) Decrypt my message.



Enter your choice :1
 Type your message

osproj
 Type your key

123456
Type your file name in which your key will be stored: key1.txt
Type your file name in which your output data will be stored :project.txt
 Encrypted message
```



```
Type your file name in which your output data will be stored :project.txt
 Encrypted message


0 0 0 0 1 1 0 0 0 1

After Permutation 0000110001

After Left Part Left Shift 10000

After Right Part Left Shift 00011

'╓═P¬è


(0) Press '0' to Exit.

(1) Include pre-built files for taking message and key.

(2) Don't include pre-built files, I will type my message and key.



Enter your choice:1
(1) Encrypt my message.

(2) Decrypt my message.
```



```
(2) Decrypt my message.


Enter your choice :2
Enter your message file name :project.txt


Enter your key file name :key1.txt
Type your file name in which your output data will be stored :output1.txt
 Decrypted message


osproj


(0) Press '0' to Exit.

(1) Include pre-built files for taking message and key.

(2) Don't include pre-built files, I will type my message and key.



Enter your choice:0
PS C:\Users\Dell\Desktop\VIT\Reference Material Winter\CSE2005 OS EPJ\Working project\Implementation> |
```

9. Future Work

The project implemented will help us understand the in-depth features of the **Data Encryption Standard**. However, we will ensure the following changes in the near future:

1. Learning about other improved cryptographic systems as the older encryption standards cannot be used forever due to developing cryptanalytic attacks.
2. Implementation of the latest cryptographic systems on different elements such as file systems in order to test the cryptographic scheme used.
3. Discovering a more efficient approach to encrypt and decrypt file systems.

10. Conclusion

During the course of this project, we have understood the vital nature of data security and cryptography, and its role in creating a safer environment for the file systems. The **Simplified Data Encryption Standard** (SDES) cryptographic scheme has been used in such a way that it is easier to learn about the principals involved in cryptography. This knowledge can be used and applied in our further works involving cryptography.

11. Bibliography

1) https://ieeexplore.ieee.org/document/563518 - "The improved data encryption standard (DES) algorithm"

2) https://iopscience.iop.org/article/10.1088/1742-6596/1363/1/012078/meta - "Designing the Application of Security Text Messages into Audio Files Using Data Encryption Standard (DES) Algorithms Using the End of File (EOF) Method."

3) https://www.researchgate.net/publication/330077146_Performance_analysis_of_AES_DES_and_Blowfish_cryptographic_algorithms_on_small_and_large_data_files - "Performance analysis of AES, DES and Blowfish cryptographic algorithms on small and large data files"

4) https://www.researchgate.net/publication/297613039_Research_and_Implementation_of_File_Encryption_and_Decryption - "Research and Implementation of File Encryption and Decryption"

5) https://www.researchgate.net/publication/282526872_Cryptanalysis_of_Simplified_Data_Encryption_Standard_Using_Genetic_Algorithm - "Cryptanalysis of Simplified Data Encryption Standard Using Genetic Algorithm."

6) https://www.irjet.net/archives/V3/i7/IRJET-V3I7322.pdf - "Color Image Encryption and Decryption using DES Algorithm."

7) https://www.irjet.net/archives/V4/i3/IRJET-V4I3489.pdf - "DES - Data Encryption Standard"

8) https://ieeexplore.ieee.org/document/6421347 - "Design and Implementation of Algorithm for DES Cryptanalysis"

9) https://en.wikipedia.org/wiki/Data_Encryption_Standard

10) https://www.geeksforgeeks.org/simplified-data-encryption-standard-key-generation/

THE END