School of Computer Science and Engineering

OPERATING SYSTEM

PROJECT COMPONENT

PROTECTION OF FILE SYSTEM IN OPERATING SYSTEM USING ENCRYPTION METHOD

SUBMITTED BY:

Manali Singh

1918444

ABSTRACT:

Storage systems are increasingly subject to attacks. Cryptographic document frameworks alleviate the threat of uncovering information by utilizing encryption and integrity security strategies and ensure end-to-end security for their customers. This paper describes a generic design for cryptographic file systems and its realization in a distributed storage-area network (SAN) file system. Key administration is incorporated with the meta-information administration of the SAN file system. The two systems have been executed in the customer record framework driver. Benchmarks illustrate that the overhead is recognizable for some falsely developed utilize cases, however that it is little for normal document framework applications.

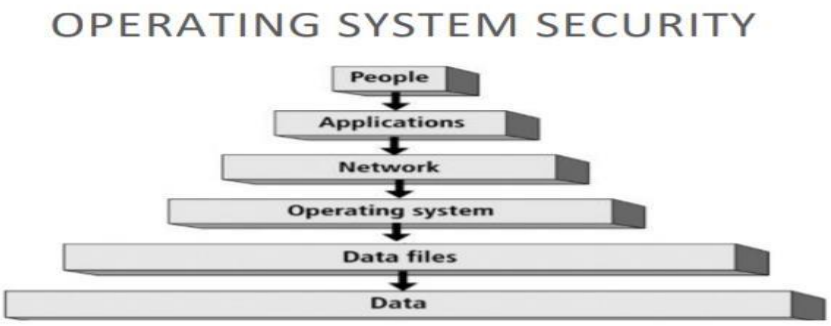
INTRODUCTION:

Data Security has always remained an integral and indispensable part of operating system. OS security refers to specified steps or measures used to protect the OS from threats, viruses, worms, malware or remote hacker intrusions. OS security encompasses all preventive-control techniques, which safeguard any computer assets capable of being stolen, edited or deleted if OS security is compromised. So, a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.

- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

The basis of OS protection is separation. The separation can be of four different kinds:

– Physical: physical objects, such as CPU's, printers, etc.
– Temporal: execution at different times
– Logical: domains, each user gets the impression
– Cryptographic: hiding data, so that other users cannot understand them



Key areas that need to security measures in operating system:

1. User Accounts
2. Account Policies
3. File System
4. Network Services

If we develop some basic cryptosystem that has the objective to encrypt and decrypt the given data then few fundamentals that must be kept in mind are as follows:

- **Plaintext:** It is the data to be protected during transmission.
- **Encryption Algorithm:** It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
- **Ciphertext:** It is the scrambled version of the plaintext produced by the encryption algorithm using a specific the encryption key. The ciphertext is not guarded. It flows on public channel. It can be intercepted or compromised by anyone who has access to the communication channel.
- **Decryption Algorithm:** It is a mathematical process, that produces a unique plaintext for any given ciphertext and decryption key. It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.

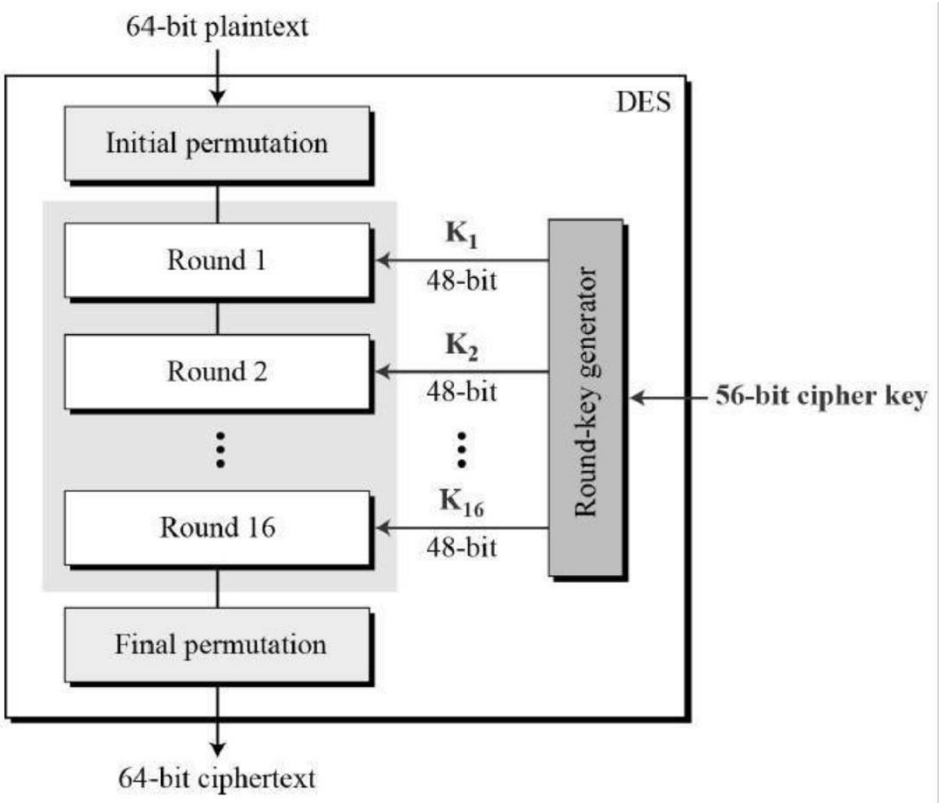
- **Encryption Key:** It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
- **Decryption Key:** It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.

Objective:

- Understanding the need of data security and integrity to be included during design of distributed operating system.
- Types of cryptography modules used by different operating system store data and authorize users.
- Implement a file encryption technique using DES for the windows platform.
- Comparing the time and space aspects of decrypted file and encrypted file.

ABOUT DES

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).



SCOPE:

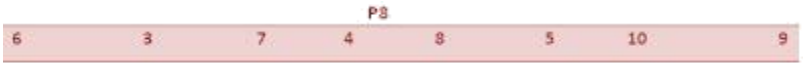
- First of all user fulfill the agreement form of this program then user insert a text massage or include text massage file.
- Encryption key as an input as a result, this program will encrypt and decrypt this text massage and save it in a file.

ANALYSIS OF S-DES Algorithm:

Key generator

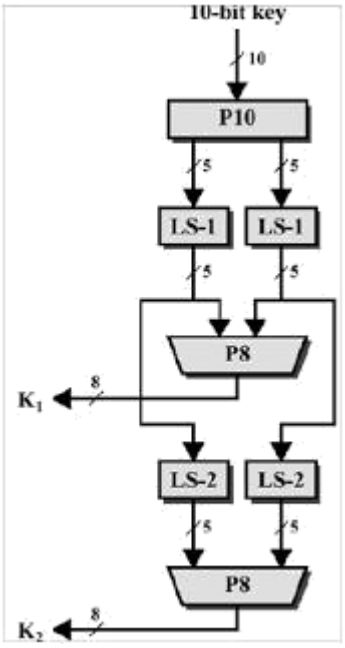
-

We have analyzed the project that we will take a an input (encryption key) from user ,we will convert this key in to 10 bit binary. Then we will divide the 10 bit binary key in 5-5 bits .we will do LS-1(LEFT [1bit]) of these 5-5 bits .we will merge and arrange these bits according to the following rule



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

From those 5-5 bits of (LS-1),along with generating the key (K1) ,we will do LS-2(left shift [2bit]) by the help of above rule (P8) we have merged(LS-2),we will get key(key2).



Flow chart of key generator

Encryption detail

We will take any alphabet or integer from the user and will convert into an 8 bit (IP BINARY),we will divide the this 8 bit binary in to 44 bit as left 4bit and right 4bit **IP.** We will convert right 4bit binary into 8 bit binary and arrange it according to the following rule.

F/P							
4	1	2	3	2	3	4	1

After applying this rule ,we will add (key 1)with it which we have already generated .we will divide these 8 bits into 4-4 bits from first 4 bots we will find so and from remaining 4 bits ,we will find S1.

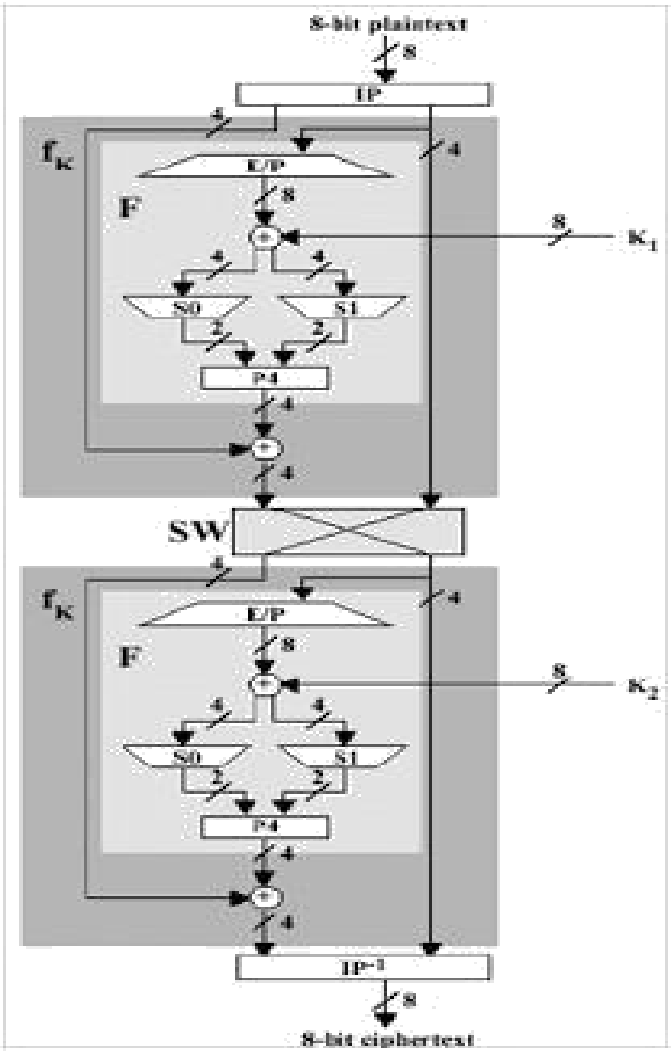
$$S0 = \begin{array}{c} \\ P_{0,0}\,P_{0,3} \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} P_{0,1}\,P_{0,2} & & & \\ 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array}$$

$$S1 = \begin{array}{c} \\ P_{1,0}\,P_{1,3} \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} P_{1,1}\,P_{1,2} & & & \\ 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array} \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array}$$

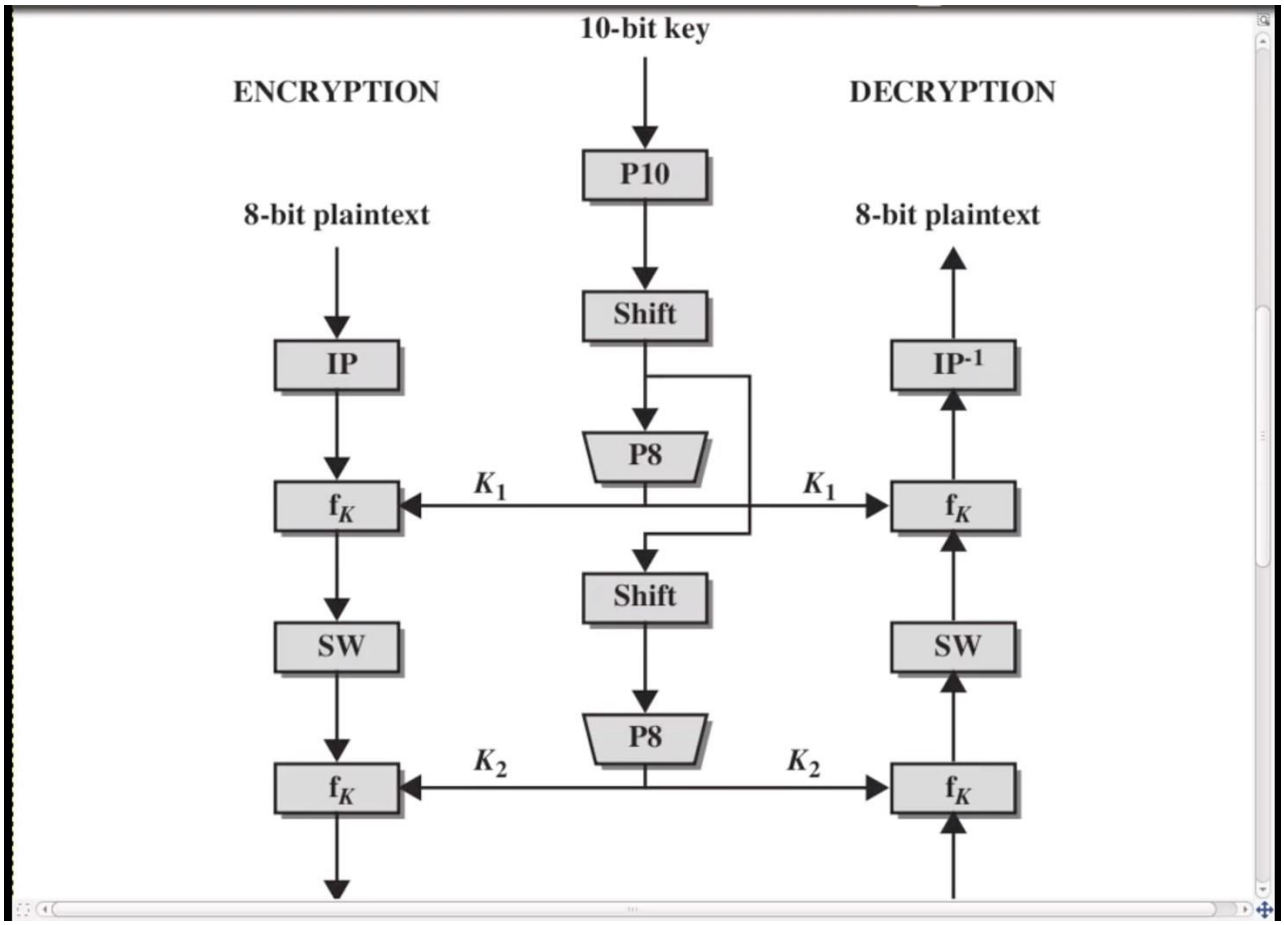
After solving S0 and S1 ,we will get 2-2 bit binary and arrange them through RULE P 4.

P4			
2	4	3	1

Now we will add the left 4 bit IP with this arranged 2-2 bit binary. by using switch function we will repeat this method and use key 2 instead of key 1 as a result of this whole scheme the user ‘s given massage will encrypt.



STRUCTURE OF ENCRYPTION AND DECRYPTION



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[])
{
        i
        n
        t
        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<<rig
                                ht_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(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_nu
m,bin);  int
ip_arrange[8];
        IP(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_rig
        ht_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_le
        ft_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)+(i
        p_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(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,s
        w_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)+(i
        p_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 way 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 way 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
```

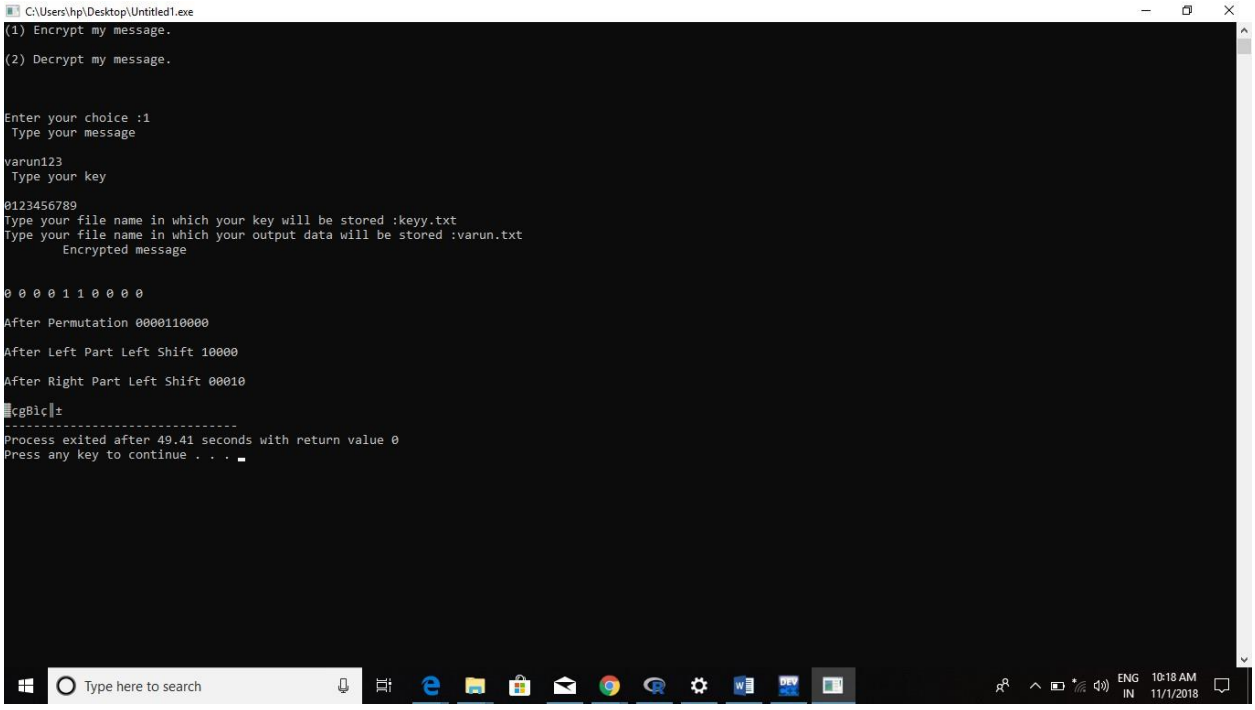
```
        :"; cin.getline(key_file,500);
fstream myfile(key_file,ios::out  | ios::
app);
                while(user_key[loop]!=-52)
                        {
        myfile<<user_key[loop];    loop++;
                        }
        myfile.close();
}

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
        cch
        oic
        e;
        int ichoice;
        cout<<"(1) Include my files for taking message and key.\n\n";
cout<<"(2) Don't include my  files, I will  type my message and key.\n\n";
        cout<<"\n\nEnter your choice:";
        cin>>cchoice;
        ichoice=cchoice;
        message_and_key(ichoice);
}
```

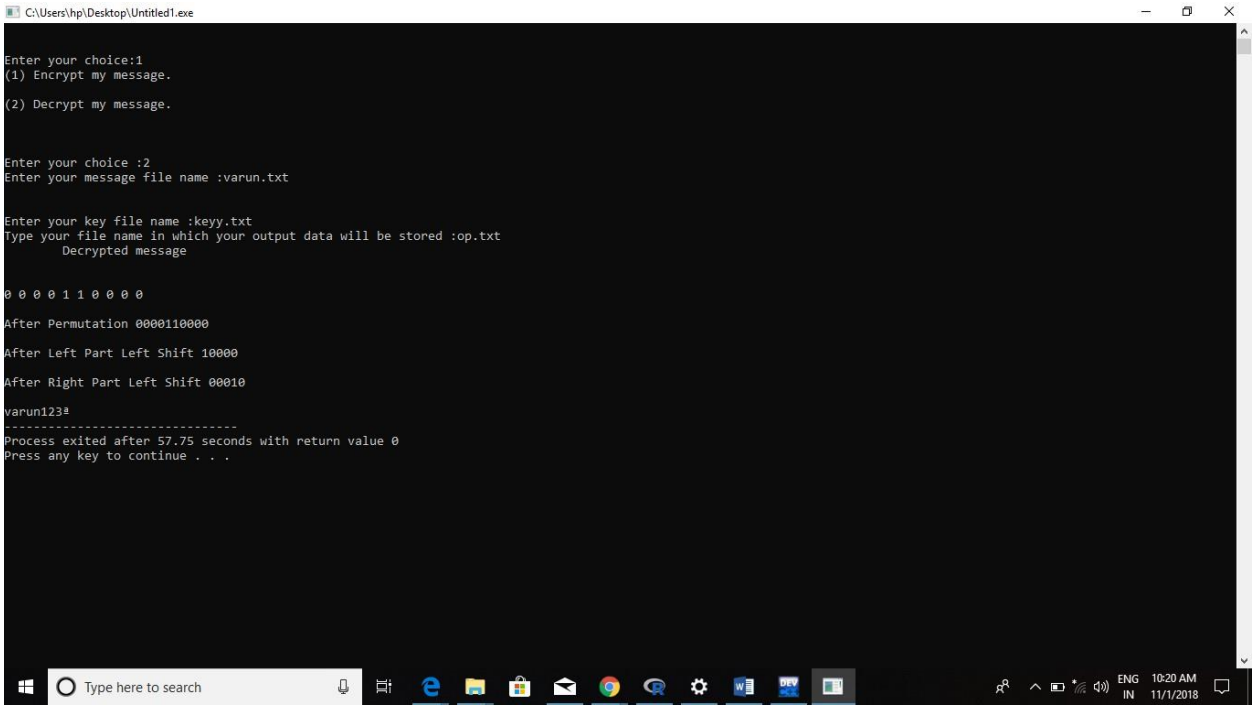
```
int main()
{
        char
cchoice;
int ichoice;
cchoice='1';
ichoice=cchoice
;
        if(ichoice==49)
        {
                file_choice();
        }
        return 0;
}
```

OUTPUT:

1. Encryption Result:



2. Decryption Result:



BIBLIOGRAPHY:

- https://www.codeproject.com/
- www.google.com
- https://www.tutorialspoint.com
- www.stackoverflow.com
- www.geeksforgeeks.com