ADVANCED DATA ENCRYPTION AND STEGANOGRAPHY

GROUP MEMBERS	**:**	**SABINA BHADRISH (805)** **SAYANTAN CHAKRABORTY (819)** **DOLLY LEE (820)**
STREAM	**:**	**B.Sc. COMPUTER SCIENCE - 3RD YEAR (SEMESTER VI).**
COLLEGE	**:**	**ST. XAVIER'S COLLEGE, KOLKATA.**
PROJECT MENTOR	**:**	**PROF. SANKAR DAS,** **COMPUTER SCIENCE DEPARTMENT,** **SXC, KOLKATA.**

ST. XAVIER'S COLLEGE, KOLKATA

PROJECT MENTOR: PROF. SANKAR DAS.

This is to certify that the following students with registration numbers:

- C4548 of 2008
- C4561 of 2008
- C4562 of 2008

of B.Sc., final year, Computer Science Honours of St. Xavier's College, Kolkata under the affiliation of Calcutta University (CU) have prepared themselves thoroughly for the project topic **"Advanced Data Encryption and Steganography"** and have completed the project and developed an executable software successfully using Visual Basic 2010 and some other technologies. The project mentor was **Prof. Sankar Das**.

CERTIFICATE OF COMPLETION

This is to certify that the project report entitled **"Advanced Data Encryption and Steganography"** submitted to the St. Xavier's College, Kolkata is an original work carried out by

- Sabina Bhadrish
- Sayantan Chakraborty
- Dolly Lee

of B.Sc. Computer Science Department, St. Xavier's College, Kolkata session 2010-2011 under my guidance.

The matter embodied in this project is a genuine work done by the students and has not been submitted whether to this University or to any other University/Institute of the fulfillment of the requirement of any course of study.

---------- ----------

(Signature of Student) (Signature of Project Mentor)

---------- ----------

(Signature of Student) (Signature of Head of Department)

---------- ----------

(Signature of Student) (Signature of External Examiner)

Date: Tuesday, April 12, 2011

ACKNOWLEDGEMENTS

We would like to thank the following people for their help and encouragement.

- **Prof. Sankar Das,**
  Our project mentor, for guiding us through the process.

- **Prof. Anal Acharya,**
  Our Head of Department, for his encouragement, words of advice and help.

- **Prof. Asoke Nath,**
  For his invaluable support and words of encouragement without which the project would not have been completed successfully.

- All the professors of the B.Sc. Computer Science Department.

CONTENTS

Project Objective

The primary objective of the product is to provide means to hide critical and confidential data within other files, thus rendering them undetectable by unauthorized individuals. The hidden information can be a text, audio, image or video file. The carrier file also can be of any uncompressed file format. The information to be hidden is first encrypted using a new advanced encryption algorithm, the output of which is then stored within the carrier file, ensuring high level of security. This document describes the scope, functionalities, software requirements and viability of the project undertaken.

Introduction

The massive development in internet technology in the last few years now it is a real challenge for the sender to send confidential data from one computer to another computer. There is no guarantee that between sender and receiver there is no one is intercepting those confidential data provided the data is not encrypted or properly protected. The security originality of data has now become a very important issue in data communication network. One cannot send any confidential or important message in raw form from one computer to another computer as any hacker can intercept that confidential message or important message. Now it is a common practice that the teachers are sending question papers over the mail. Now it is not a difficult job for a hacker to intercept that mail and retrieve the question paper if it is not encrypted. This may be very dangerous when someone is sending some confidential matter over the mail such as Bank transaction, Bank statement or any other confidential matter. There is no guarantee that the message will not be intercepted by anyone. This may be further worse during e-banking or e-commerce where the real data should not be intercepted by any hacker. When a client is sending some confidential matter from client machine to another client machine or from client machine to server then that data should not be intercepted by someone. The data should be protected from any unwanted intruder otherwise any massive disaster may happen all on a sudden. The disaster may happen if a sales manager of a company is sending some crucial data to his Managing Director related sales issue over the e-mail. Suppose some intruder has intercepted that data from the internet and pass it to some other rival company. It is possible when the data moving from one computer to other computer is totally unprotected or not encrypted. To get rid of this problem one has to send the encrypted text or cipher text from client to server or to another client. Because of this hacking problem now a days network security and cryptography is an emerging research area where the people are trying to develop some good encryption algorithm so that no intruder can intercept the encrypted message. The so called classical cryptographic algorithm can be classified into two categories:

(i) Symmetric key cryptography where one key is used for both encryption and decryption purpose.

(ii) Public key cryptography where two different keys are used one for encryption and the other for decryption purpose.

The merits of symmetric key cryptography are that the key management is very simple as one key is used for both encryption as well as for decryption purpose. In case of symmetric key cryptography the key must be secret. In public key cryptography the encryption key remains as public but the decryption key should be kept as secret key. The public key methods have got both merits as well as demerits. The problem of Public key cryptosystem is that one has to do massive computation for encrypting any plain text. Moreover in some public key cryptography the size of encrypted message may increase. Due to massive computation the public key crypto system may not be suitable in security of data in sensor networks. So the security problem in sensor node is a real problem.

Cryptography – A Brief History

Cryptography is the science of writing in secret code and is an ancient art; the first documented use of cryptography in writing dates back to circa 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any un-trusted medium, which includes just about any network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

- ***Authentication:*** The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
- ***Privacy/confidentiality:*** Ensuring that no one can read the message except the intended receiver.
- ***Integrity:*** Assuring the receiver that the received message has not been altered in any way from the original.
- ***Non-repudiation:*** A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as ***plaintext***. It is encrypted into ***ciphertext***, which will in turn (usually) be decrypted into usable plaintext.

Cryptography can be defined as the conversion of data into a scrambled code that can be deciphered and sent across a public or private network. Cryptography uses two main styles or forms of encrypting data; symmetrical and asymmetrical. Symmetric encryptions, or algorithms, use the same key for encryption as they do for decryption. Other names for this type of encryption are secret-key, shared-key, and private-key. The encryption key can be loosely related to the decryption key; it does not necessarily need to be an exact copy.

Symmetric cryptography is susceptible to plain text attacks and linear cryptanalysis meaning that they are hack-able and at times simple to decode. With careful planning of the coding and functions of the cryptographic process these threats can be greatly reduced. Asymmetric cryptography uses different encryption keys for encryption and decryption. In this case an end user on a network, public or private, has a pair of keys; one for encryption and one for decryption. These keys are labeled or known as a public and a private key; in this instance the private key cannot be derived from the public key.

In a symmetric cipher, both parties must use the same key for encryption and decryption. This means that the encryption key must be shared between the two parties before any messages can be decrypted. Symmetric systems are also known as shared secret systems or private key systems.

The **asymmetrical cryptography** method has been proven to be secure against computationally limited intruders. The security is a mathematical definition based upon the application of said encryption. Essentially, asymmetric encryption is as good as its applied use; this is defined by the method in which the data is encrypted and for what use. The most common form of asymmetrical encryption is in the application of sending messages where the sender encodes and the receiving party decodes the message by using a random key generated by the public key of the sender.

In an asymmetric cipher, the encryption key and the decryption keys are separate. In an asymmetric system, each person has two keys. One key, the public key, is shared publicly. The second key, the private key, should never be shared with anyone. When you send a message using asymmetric cryptography, you encrypt the message using the recipients public key. The recipient then decrypts the message using his private key. That is why the system is called asymmetric. Because asymmetric ciphers tend to be significantly more computationally intensive, they are usually used in combination with symmetric ciphers to implement effect public key cryptography. The asymmetric cipher is used to encrypt a session key and the encrypted session key is then used to encrypt the actual message. This gives the key-exchange benefits of asymmetric ciphers with the speed of symmetric ciphers.

In cryptography, a **Caesar cipher** is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals. The encryption step performed by a Caesar cipher is often incorporated as part of more complex schemes, such as the Vigenère cipher, and still has modern application in the ROT13 system. As with all single alphabet substitution ciphers, the Caesar cipher is easily broken and in practice offers essentially no communication security.

Example: The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places (the shift parameter, here 3, is used as the key):

```
Plain:    ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:   DEFGHIJKLMNOPQRSTUVWXYZABC
```

When encrypting, a person looks up each letter of the message in the "plain" line and writes down the corresponding letter in the "cipher" line. Deciphering is done in reverse.

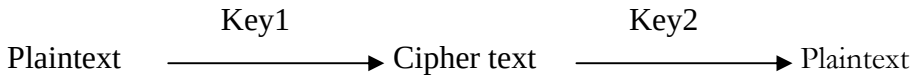
```
Ciphertext: WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
Plaintext:  the quick brown fox jumps over the lazy dog
```

There are several ways of classifying cryptographic algorithms. For purposes of this paper, they will be categorized based on the number of keys that are employed for

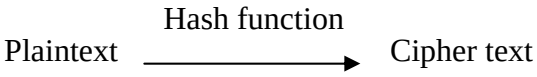
encryption and decryption, and further defined by their application and use. The three types of algorithms that will be discussed are (Figure 1):

- **Secret Key Cryptography (SKC):** Uses a single key for both encryption and decryption

- **Public Key Cryptography (PKC):** Uses one key for encryption and another for decryption



- **Hash Functions:** Uses a mathematical transformation to irreversibly "encrypt" information



Software Requirement Specification

1. Introduction

1.1 Product Purpose

The primary objective of the product is to provide means to hide critical and confidential data within other files, thus rendering them undetectable by unauthorized individuals. The hidden information can be a text, audio, image or video file. The carrier file also can be of any uncompressed file format. The information to be hidden is first encrypted using a new advanced encryption algorithm, the output of which is then stored within the carrier file, ensuring high level of security. This document describes the scope, functionalities, software requirements and viability of the project undertaken.

1.2 Document Conventions

The document has been prepared in accordance with IEEE standards. The primal components of this document are highlighted in Bold Face.

1.3 Intended Audience

End Users: They are the ones who will be operating the software product most. So, this SRS is extremely useful for the End Users as they get a clear idea about the requirements, scope, functionality and performance of the software.

Software Developers: This SRS is also helpful for developers who contrive to develop any such similar software product based on advanced data encryption and image steganography from scratch.

Software Testers: It is imperative that the professionals involved with testing this software product have a clear idea about the product, which is provided in this SRS document.

Students: The SRS can also be regarded as a practicable solution to queries regarding data encryption and image steganography.

1.4 Product Scope

The software product is able to perform encryption technique on text, audio, image, video files or just simple plaintext based messages. It also supports steganography, hiding the encrypted file in an uncompressed file. Both of these data hiding techniques is protected by secure passwords, thus being highly difficult for cryptanalysts to decode.

1.5 References

1. Symmetric key cryptography using random key generator, A.Nath, S.Ghosh, M.A.Mallik, Proceedings of International conference on SAM-2010 held at Las Vegas(USA) 12-15 July,2010, Vol-2,P-239-244
2. Modified Version of Playfair Cipher using Linear Feedback Shift Register, P. Murali and Gandhidoss Senthilkumar, UCSNS International journal of Computer Science and Network Security, Vol-8 No.12, Dec 2008.
3. Richard A. Mollin, An Introduction to Cryptography, CRC Press, 2000
4. Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, Ton Kalker, Digital Watermarking and Steganography, Morgan Kaufmann Publishers, 2008
5. Peter Wayner, Disappearing Cryptography - Information Hiding, Steganography and Watermarking, Morgan Kaufmann Publishers, 2008

2. Overall Description

2.1 Product Perspective

Steganography is the science of hiding messages within some sort of carrier files in such a way that no one apart from the sender or the receiver suspects the existence of the message. This software product applies an advanced encryption algorithm to encode data, and then in turn hide the encrypted file within the carrier file. First the cipher algorithm is applied on the input file, the output of which is hidden in an uncompressed file using bit manipulation steganographic technique. The primary advantage of this software is that it's not just limited to hiding plaintext data within a carrier file; it can be used to hide other images, audio and even video files within such hosts, in its encrypted form, thus ensuring security to the utmost degree.

2.2 Product Features

- An advanced data encryption algorithm is implemented which is the primary feature of the software.
- Multiple encryptions are supported.
- Steganography is also supported.
- The hidden object can be a simple plaintext file, an image, or even an audio/video file.
- Carrier files can be of any uncompressed file type.
- The software provides authentication check during login. Without a proper username and password the user cannot access the software.
- The software also provides high levels of security via a secure password used during encryption and during steganography without which the decryption would not be possible.

2.3 User Classes and Characteristics

The software product can be used by developers to understand, and if possible, improve the encryption standard, and can be operated by naïve end users to securely transmit confidential data.

2.4 Operating Environment

Operating System: Windows XP, Vista, 7
Software Requirements: Microsoft .NET Framework 4
Recommended Configuration: Intel Pentium 4, 2 GHz or higher, 256mb RAM or higher, sufficient disk space to install the software product and the necessary files.
Optimal Screen Resolution: 1024x768

2.5 Design and Implementation Constraints

- Although the software is portable, it cannot run on platforms other than windows. It cannot be made to run over LAN or any form of network.
- The size of the file to be hidden should preferably be approximately 20% of the size of the carrier file for efficient and fast steganographic operations.

2.6 Assumption and Dependencies

- The minimum system requirements must be satisfied for efficient software performance.
- The Operating System should be any one of the Microsoft Windows NT family.
- Relevant software must be installed to view audio/video files.

3. External Interface Requirements

3.1 User Requirements

Login Screen: It's the first screen that is shown to the user. The user should provide a valid user id and password to access the software.

Main Form: It provides the user with all the features of the software product, including option for implementing steganography and data encryption. Options are provided to reset the fields entered and to shut down the application.

3.2 Hardware Requirements

Apart from the recommended hardware specifications as mentioned, no other specific hardware is necessary to operate the software.

3.3 Software Requirements

Microsoft .NET Framework 4 should be present in the system, along with the necessary software to view the audio/video files.

4. Other Non-functional Requirements

4.1 Performance Requirements

- ✓ The RAM should be 256mb though 512mb is recommended.
- ✓ The processing unit should be fast enough to perform the encryption and steganographic functions efficiently.
- ✓ Sufficient storage space is necessary to install the software and other dependent files, along with the hidden object and carrier file.

4.2 Safety Requirements

- ➢ The size constraints for steganography are to be evaluated by the end user.
- ➢ The size of the object to be hidden depends on the carrier image.
- ➢ The software is fully visible with a screen resolution of 1024x768 or higher.
- ➢ The size of the file to be hidden should preferably be approximately 20% of the size of the carrier file for efficient and fast steganographic operations.

4.3 Security Requirements

The user must possess a valid user id and password to access the software. The user should also possess two unique passwords used for encryption and steganography to get back the original file. The password for encryption process determines the number of times encryption and decryption is performed, thus it should be same for both the processes, whereas the one used for steganography is embedded in the carrier file and is validated before retrieving the data.

5. Other Requirements

The software performs efficiently when the object to be hidden and the carrier file are in the same folder, preferably in the folder where the software is, else there is a slight decrease in performance. It should also be noted that the size of the carrier file is preferred to be much more than the source object so that it can be completely hidden in the carrier. There are no further requirements than those specified in the SRS under the different headers.

Features and Overview of Software

- An advanced data encryption algorithm is implemented which is the primary feature of the software.
- Multiple encryptions are supported.
- Steganography is also supported.
- The hidden object can be a simple plaintext file, an image, or even an audio/video file.
- Carrier files can be of any uncompressed file type.
- The software provides authentication check during login. Without a proper username and password the user cannot access the software.
- The software also provides high levels of security via a secure password used during encryption and during steganography without which the decryption would not be possible.

Overview of the softwares used in the project:

Software type	Name	Logo	Advantage
Operating System	Microsoft Windows XP, Windows 7		Support for .NET Framework 4.0
Programming Tool	Turbo C		Extreme fast compile speed, integrated development environment.
Front End Development	Microsoft Visual Basic 2010		Provides efficient interfacing with executable files generated from Turbo C

Methodology

Now we will describe our new advanced symmetric key cryptography. The present method is fully dependent on the text-key which is any string of maximum length 16 characters long. From the text-key we calculate two important parameters (i) Randomization number and (ii) Encryption number. To calculate these two parameters we use the method following method:

Step-1: Suppose key=AB
Choose the following table for calculating the place value and the power of characters of the incoming key:

Table-1: Length of key (n) vs. Base value (b)

Length of key(n)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Base value(b)	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

(i) Calculate $\text{Sum}=\sum_{m=1}^{n} \text{ASCII Code} * b^m$ ---- (1)

where n=number of characters in the input text-key.

Now we calculate the sum for key="**AB**" using equation (1). Here n=2, b=16

Sum = $65*16^1 + 66 * 16^2$

= 17936

Now we will show how we calculate the two parameters from this sum:

(ii) Randomization number(n1):

num1=1*1+7*2+9*3+3*4+6*5=84

n1=sum mod num1=17936 mod 84=**44**

Note: if n1=0 then n1=num1 and if n1>32 then n1=Mod (n1, 32)

(iii) Encryption number(n2):

num2=6*1+3*2+9*3+7*4+1*5=72

n2=sum mod num2 =17936 mod 72 =**8**

Note: if n2=0 then n2=num2 and if n2>32 then n2=Mod (n2, 32)

Step-2: Now we will describe how we perform the bit manipulation in the input stream of characters.

Step-2.1: Read 32 bytes at a time from the input file. Convert 32 bytes into 256 bits and store in some 1-dimensional array.

Step-2.2: Choose the first bit from the bit stream and also the corresponding number (n) from the key matrix. Interchange the 1st bit and the n-th bit of the bit stream.

Step-2.3: Repeat step-2.2 for 2nd bit, 3rd bit…256-th bit of the bit stream

Step-2.4: Perform right shift by one bit.

Step-2.5: Perform bit (1) XOR bit (2), bit (3) XOR bit (4)….bit (255) XOR bit (256)

Step-2.6: Repeat step-2.4 with 2 bit right, 3bit right….n2 bit right shift followed by step-2.5 after each completion right bit shift.

After performing step-2 we obtain a file which is Encrypted file and it will not be possible to decrypt until and unless one knows the exact number of encryption and exact randomized matrix. One can use this encrypted file for sending data from one terminal to another terminal or from one terminal to one server.

Step-3: Then we use this encrypted file as the input file to apply MSA encryption algorithm to encrypt this file twice to make the encryption further strong. In the next section we will describe the MSA encryption algorithm in brief.

MSA Symmetric key Cryptographic method:

In this algorithm, a symmetric key method is proposed where we have used a random key generator for generating the initial key and that key is used for encrypting the given source file. MSA method is basically a substitution method where we take 2 characters from any input file and then search the corresponding characters from the random key matrix and store the encrypted data in another file.

The key matrix contains all possible characters (ASCII code 0 to 255) in a random order. The pattern of the key matrix will depend on text key entered by the user. A long trial run has been given on text key and it is found that it is very difficult to match the three above parameters for 2 different text key which means if someone wants to break the encryption method then he/she has to know the exact pattern of the text key otherwise it will not be possible to obtain two sets of identical parameters from two different text keys.

For pure text file we can apply brute force method to decrypt small text but for any other file such as any binary file brute force method cannot be applied and it does not work.

After calculating the above two parameters then the original key matrix is created which is of size 16x16 which contains all characters from 0-255 ASCII codes. Then we make the key matrix random by applying some simple function calls one after the other and substitute the digraphs from the source file by mapping onto the randomized matrix using the rules of modified Playfair cipher.

Coding

Source Code for ENCRYPTION:

```
/*Program to perform encryption on a file in blocks of 256 bits*/

#include<stdio.h>
#include<math.h>

#define siz 16
int times,places,secure;
void keygen();
char file1[50],file2[50];
char file3[50]={'T','E','M','P','.','D','A','T','\0'};
FILE *fp1,*fp2,*fp3,*fp4;
char rshift_residual(char c,int n_r_shift);
unsigned char data2[32];
unsigned char mat[siz][siz],mat1[siz][siz];

double nbyte,nbit;
void encrypt(unsigned char ch1,unsigned char ch2,unsigned char mat[siz][siz]);
void encrypt1(unsigned char ch1,unsigned char mat[siz][siz]);

struct mytag1
{
unsigned char str[2];
}mydata1;

void msa_encryption();

void randomization();
void leftshift();
void cycling();
void upshift();
void rightshift();
void downshift();
void encrypt_bit();
void rshift(int);
void lshift(int);
int bleft[128];
int bright[128];
int n=0;

struct tag1
{
unsigned char ch[32];
}data1;

int data[256];
void bit_stream(char s[]);
void bit_wise_xor();
```

```
main(int argc,char *argv[])
{
long int i,j,k,l,n1,n2,nl;
if(argc==3)
{
clrscr();

fp1=fopen(argv[1],"rb");
fp2=fopen(file3,"wb");

fseek(fp1,0,2);
l=ftell(fp1);
n1=l/32;
n2=l%32;
fseek(fp1,0,0);
nl=0;
for(i=0;i<16;i++)
        for(j=0;j<16;j++)
        mat[i][j]= nl++;
clrscr();
keygen();
/* To envoke randomization() function secure-times */
for(i=1;i<=secure;i++)
randomization();
for(i=1;i<=n1;i++)
        {
        fread(&data1,sizeof(data1),1,fp1);
        bit_stream(data1.ch);
        encrypt_bit();
        }

if (n2!=0)
        {
                for(i=0;i<n2;i++)
                {
                fscanf(fp1,"%c",&data2[i]);
                data2[i]=rshift_residual(data2[i],5);
                fprintf(fp2,"%c",data2[i]);
                }
        }

fcloseall();
msa_encryption(file3,argv[2]);

printf("\nData encryption is over.\n\n");
getch();
}
else
printf("\n***Invalid command line arguments***\n");
}
```

```
void bit_stream(char s[])
{
int i,k,j;
int mask=128;
n=0;
for(i=0;i<32;i++)
{
j=s[i];
mask=128;
        while(mask!=0)
        {
        k=(j & mask)?1:0;
        data[n++]=k;
        mask=mask>>1;
        }
}
}

void encrypt_bit()
{
        int i,j,x,x1,k,temp=0,s1,c,num[8],p2,sum;
        /*randomization();*/

        s1=0;
        for(i=0;i<siz;i++)
        {
                for(j=0;j<siz;j++)
                {
                        x=(int)mat[i][j];
                        temp=data[s1];
                        data[s1]=data[x];
                        data[x]=temp;
                        s1=s1+1;
                        x=0;

                }
        }


        for(c=0;c<times;c++)
        {
                rshift(c);
                bit_wise_xor();
        }

        for(i=0;i<256;i=i+8)
                {
                j=0;
                k=0;
```

```
                num[j]=data[i];
                num[j+1]=data[i+1];
                num[j+2]=data[i+2];
                num[j+3]=data[i+3];
                num[j+4]=data[i+4];
                num[j+5]=data[i+5];
                num[j+6]=data[i+6];
                num[j+7]=data[i+7];
                sum=0;
                p2=1;
                for(k=7;k>=0;k--)
                        {
                        if(num[k]!=0)
                                sum=sum+p2;
                        p2=p2*2;
                        }
                fprintf(fp2,"%c",sum);
                }
}

void rshift(int d)
{
        int x,i,j;
        for(j=0;j<=d;j++)
        {
                x=data[n-1];
                for(i=n-1;i>0;i--)
                data[i]=data[i-1];
                data[0]=x;
        }

}

char rshift_residual(char c,int n_r_shift)
{
        int x,i,j,sum,p2;
        int bit_c[8];
        int k,n1;
        int mask=128;
        n1=0;
        mask=128;
        while(mask!=0)
        {
        k=(c & mask)?1:0;
        bit_c[n1++]=k;
        mask=mask>>1;
        }

        for(j=0;j<n_r_shift;j++)
        {
                x=bit_c[n1-1];
```

```
                for(i=n1-1;i>0;i--)
                bit_c[i]=bit_c[i-1];
                bit_c[0]=x;
        }

        /* To convert bits to character */

                sum=0;
                p2=1;
                for(k=7;k>=0;k--)
                        {
                        if(bit_c[k]!=0)
                                sum=sum+p2;
                        p2=p2*2;
                        }
        return (char)sum;
        }

void lshift(int d)
{
        int x,i,j;
        for(j=d;j>=0;j--)
        {
                x=data[0];
                for(i=0;i<n-1;i++)
                data[i]=data[i+1];
                data[n-1]=x;
        }
}

void bit_wise_xor()
{
        int i,x,y,z;
        for(i=0;i<n-1;i=i+2)
        {
                x=data[i];
                y=data[i+1];
                z=(x ^ y);
                data[i]=z;
        }
}

/*start of randomization*/

        void randomization()
        {
        unsigned char ch;
        int  a=0,j,k,i,check,nc1;
        int nl;

        long int nblock;
```

```
int ii,jj,ii1,jj1;

                    for(a=1;a<=times;a++)
                        {
                        check=a%5;
                            switch(check)
                            {
                            case 0 :cycling();
                                    upshift();
                                    rightshift();
                                    downshift();
                                    leftshift();
                                    downshift();
                                    rightshift();
                                    upshift();
                                    cycling();
                                    break;
                            case 1 :leftshift();
                                    cycling();
                                    upshift();
                                    rightshift();
                                    downshift();
                                    rightshift();
                                    upshift();
                                    cycling();
                                    leftshift();
                                    break;
                            case 2 :downshift();
                                    leftshift();
                                    cycling();
                                    upshift();
                                    rightshift();
                                    upshift();
                                    cycling();
                                    leftshift();
                                    downshift();
                                    break;
                            case 3 :rightshift();
                                    downshift();
                                    leftshift();
                                    cycling();
                                    upshift();
                                    cycling();
                                    leftshift();
                                    downshift();
                                    rightshift();
                                    break;
                            case 4 :upshift();
                                    rightshift();
                                    downshift();
```

```
                                                leftshift();
                                                cycling();
                                                leftshift();
                                                downshift();
                                                rightshift();
                                                upshift();
                                                break;
                                        }
                                }
}

/*** rightshift() function starts ***/

void rightshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=0;k<siz;k++)
        {
        ch1=mat[j][k];
        m=((j*siz)+k)+1;
        r=j;
        c=k+m;
                while(c>=siz)
                {
                r=r+1;
                        if(r==siz) r=0;
                        c=c-siz;
                }
                if(m>((siz*siz)/2))
                {
                c=c+1;
                        if(c==siz) r++,c=0;
                        if(r==siz) r=0;
                }
        mat1[r][c]=ch1;
        }
}
for(r=0;r<siz;r++)
{
        for(c=0;c<siz;c++)
        {
        mat[r][c]=mat1[r][c];
        }
}

}

void downshift()
```

```
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=0;k<siz;k++)
        {
        ch1=mat[k][j];
        m=((j*siz)+k)+1;
        c=j;
        r=k+m;
                while(r>=siz)
                {
                c=c+1;
                        if(c==siz) c=0;
                        r=r-siz;
                }
                if(m>((siz*siz)/2))
                {
                r=r+1;
                        if(r==siz) c++,r=0;
                        if(c==siz) c=0;
                }
        mat1[c][r]=ch1;
        }
}
for(r=0;r<siz;r++)
{
        for(c=0;c<siz;c++)
        {
        mat[r][c]=mat1[c][r];
}
}
}

void leftshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=siz-1;k>=0;k--)
        {
        ch1=mat[j][k];
        m=(j*siz)+(siz-1-k)+1;
        r=j;
        c=k-m;
                while(c<0)
                {
                r=r+1;
                        if(r==siz) r=0;
```

```
                    c=c+siz;
                }
                if(m>((siz*siz)/2))
                {
                c=c-1;
                    if(c==-1) r++,c=siz-1;
                    if(r==siz) r=0;
                }
        mat1[r][c]=ch1;
        }
}
for(r=0;r<siz;r++)
{
        for(c=0;c<siz;c++)
        {
        mat[r][c]=mat1[r][c];
        }
}
}

void upshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=siz-1;k>=0;k--)
        {
        ch1=mat[k][j];
        m=(j*siz)+(siz-1-k)+1;
        c=j;
        r=k-m;
                while(r<0)
                {
                c=c+1;
                if(c==siz) c=0;
                r=r+siz;
                }
                if(m>((siz*siz)/2))
                {
                r=r-1;
                if(r==-1) c++,r=siz-1;
                if(c==siz) c=0;
                }
        mat1[c][r]=ch1;
        }
}
        for(r=0;r<siz;r++)
        {
                for(c=0;c<siz;c++)
                {
```

```
                        mat[r][c]=mat1[c][r];
                        }
                }
        }

        void cycling()
        {
        int j=0,k=0,r,c,size,start;
        unsigned char ch1,ch2,ch3;
        size=siz;
        start=siz-size;
        while(1)
        {
                if(start==siz/2) break;
                ch1=mat[j][k];
                if(j==start && k>=j && k!=size-1)   k++,mat1[j][k]=ch1;
                else if(k==size-1 && k>j && j!=size-1)    j++,mat1[j][k]=ch1;
                else if(j==size-1 && j>=k && k!=start)    k--,mat1[j][k]=ch1;
                else if(k==start && j>k && j!=start)     j--,mat1[j][k]=ch1;
                if(j==start && k==start)      start++,size--,k=j=start;
                if(start%2==1)
                while(1)
                {
                ch1=mat[j][k];
                if(k==start && j>=k && j!=size-1)   j++,mat1[j][k]=ch1;
                else if(j==size-1 && j>k && k!=size-1)    k++,mat1[j][k]=ch1;
                else if(k==size-1 && k>=j && j!=start)    j--,mat1[j][k]=ch1;
                else if(j==start && k>j && k!=start)     k--,mat1[j][k]=ch1;
                if(j==start && k==start)      start++,size--,k=j=start;
                if(start%2==0) break;
                }
        }
        for(j=0;j<siz;j++)
        {
                for(k=0;k<siz;k++)
                {
                mat[j][k]=mat1[j][k];
                }
        }
        }

/*start of keygen function*/

        void keygen()
        {
        unsigned char key_string[50];
        long int s,s1,s2,power,p=0;
        int l,n;
        unsigned char ch;
        FILE *fp1;
```

```
s=0;
clrscr();
printf("\n\n\nEnter your secret key string(Max. 16 chars. long) :");
        while(1)
        {
        ch=getch();
        if(ch!=13)
        {
        printf("*");
        key_string[s++]=ch;
        }
        else
        {
        key_string[s]='\0';
        break;
        }
        }
s=0;
for(l=0;key_string[l]!='\0';l++)
{
power=1;
for(n=1;n<=l;n++)
{
        power=power*2;
}
s+=power*key_string[l];
}
s1=0;
s2=s;
while(s2!=0)
{
p++;
s1=s1+((s2%10)*p);
s2=s2/10;
}
secure=s%s1;
if(secure==0)
secure=s1;
for( ;secure>32; )
secure=secure-32;
if(secure>32)
secure=secure/32;
s1=0;
s2=s;
while(s2!=0)
{
s1=s1+((s2%10)*p);
s2=s2/10;
p--;
}
times=s%s1;
```

```
if(times==0)
times=s1;
for( ;times>32; )
times=times-32;
if(times>32)
times=times/32;
printf("\n\nNumber of times of the randomization process will
continue : %d\n",times);
printf("\nNumber of times of Encryption is to be done :%d\n",secure);
printf("\nPress any key to continue--->");
getch();
}


/* Function to encrypt message using MSA algorithm*/

void msa_encryption(char file3[],char file2[])
{
char file4[20]={'T','E','M','P','1','.','D','A','T','\0'};
long int i,nrec,nloop,flag;
int a;
unsigned char ch;
clrscr();
nrec=0;
fp3=fopen(file3,"rb");
fp4=fopen(file4,"wb");

while(fscanf(fp3,"%c",&ch)>0)
fprintf(fp4,"%c",ch);
fcloseall();

for(a=0;a<secure;a++)
{
fp1=fopen(file4,"rb");
fp2=fopen(file2,"wb");
while(fscanf(fp1,"%c",&ch)>0)
nrec++;
rewind(fp1);
nloop=nrec/2;
        if((nrec%2)!=0)
        flag=1;
        else
        flag=0;
        for(i=1;i<=nloop;i++)
        {
        fread(&mydata1,sizeof(mydata1),1,fp1);
                if(mydata1.str[0] == mydata1.str[1])
                {
                encrypt1(mydata1.str[0],mat);
                encrypt1(mydata1.str[1],mat);
                }
```

```
                else
                encrypt(mydata1.str[0],mydata1.str[1],mat);
        }
        if(flag==1)
        {
        fscanf(fp1,"%c",&ch);
        fprintf(fp2,"%c",ch);
        }
fcloseall();
nrec=0;
fp1=fopen(file2,"rb");
fp2=fopen(file4,"wb");
        while(fscanf(fp1,"%c",&ch)>0)
        fprintf(fp2,"%c",ch);
fcloseall();
}   /*End of Encryption Process*/
}

void encrypt(unsigned char ch1,unsigned char ch2,unsigned char mat[siz][siz])
{
int j,k,m,n,p,q,c;
for(j=0,c=0;(c<2)&&(j<siz);j++)
        for(k=0;k<siz;k++)
                if(mat[j][k] == ch1)
                        {
                                m=j;
                                n=k;
                                c++;
                        }
                else if(mat[j][k] == ch2)
                        {
                                p=j;
                                q=k;
                                c++;
                        }
                n+=q;
                q=n-q;
                n-=q;
                n++;
                q++;
                if(n==siz)
                {
                        n=0;
                        m++;
                        if(m==siz) m=0;
                }
                if(q==siz)
                {
                        q=0;
                        p++;
                        if(p==siz) p=0;
```

```
                    }
fprintf(fp2,"%c%c",mat[p][q],mat[m][n]);
}

void encrypt1(unsigned char ch1,unsigned char mat[siz][siz])
{
int j,k,m,n,q,c;
for(j=0,c=0;(c<1)||(j<siz);j++)
        for(k=0;k<siz;k++)
                if(mat[j][k] == ch1)
                        {
                                m=j;
                                n=k;
                                q=n+1;
                                c++;
                        }
if(q==siz)
{
q=0;
m=m+1;
if(m==siz) m=0;
}
fprintf(fp2,"%c",mat[m][q]);
}
```

Source code for DECRYPTION:

```
/*Program to perform decryption on a file in blocks of 256 bits*/

#include<stdio.h>
#include<math.h>

#define siz 16
char lshift_residual(char c,int d);
char file1[20],file2[20];
char file3[50]={'T','E','M','P','.','D','A','T','\0'};
int times,places,secure;

struct mytag1
{
unsigned char str[2];
}mydata1;

void decrypt();
void decrypt1();
void msa_decryption();

void keygen();
FILE *fp1,*fp2;
unsigned char data2[32];
unsigned char mat[siz][siz],mat1[siz][siz];
```

```
void randomization();
void leftshift();
void cycling();
void upshift();
void rightshift();
void downshift();
void decrypt_bit();
void rshift(int);
void lshift(int);
int bleft[128];
int bright[128];
int n=0;

struct tag1
{
unsigned char ch[32];
}data1;

int data[256];
void bit_stream(char s[]);
void bit_wise_xor();
main(int argc,char *argv[])
{
long int i,j,k,l,n1,n2,nl;
if(argc==3)
{
clrscr();

msa_decryption(argv[1],file3);

fp1=fopen(file3,"rb");
fp2=fopen(argv[2],"wb");

fseek(fp1,0,2);
l=ftell(fp1);

n1=l/32;
n2=l%32;
fseek(fp1,0,0);
nl=0;
for(i=0;i<16;i++)
        for(j=0;j<16;j++)
        mat[i][j]= nl++;
clrscr();
keygen();
/* To envoke randomization() function secure-times */
for(i=1;i<=secure;i++)
randomization();
for(i=1;i<=n1;i++)
        {
        fread(&data1,sizeof(data1),1,fp1);
```

```
		bit_stream(data1.ch);
		decrypt_bit();
		}

if (n2!=0)
		{
		for(i=0;i<n2;i++)
		{
		fscanf(fp1,"%c",&data2[i]);

		data2[i]=lshift_residual(data2[i],5);
		fprintf(fp2,"%c",data2[i]);
		}
		}

fcloseall();
printf("\nData decryption is over.\n\n");
getch();
}
else
printf("\n***Invalid command line arguments***\n");
}

void bit_stream(char s[])
{
int i,k,j;
int mask=128;
n=0;
for(i=0;i<32;i++)
{
j=s[i];
mask=128;
		while(mask!=0)
		{
		k=(j & mask)?1:0;
		data[n++]=k;
		mask=mask>>1;
		}
}
}

void decrypt_bit()
{
		int i,j,x,x1,k,temp=0,s1,c,num[8],p2,sum;

		for(c=times-1;c>=0;c--)
		{
			bit_wise_xor();
			lshift(c);
		}
```

```
	s1=255;
	for(i=siz-1;i>=0;i--)
	{
		for(j=siz-1;j>=0;j--)
		{
			x=(int)mat[i][j];
			temp=data[s1];
			data[s1]=data[x];
			data[x]=temp;
			s1=s1-1;
			x=0;
		}
	}
	for(i=0;i<256;i=i+8)
		{
		j=0;
		k=0;
		num[j]=data[i];
		num[j+1]=data[i+1];
		num[j+2]=data[i+2];
		num[j+3]=data[i+3];
		num[j+4]=data[i+4];
		num[j+5]=data[i+5];
		num[j+6]=data[i+6];
		num[j+7]=data[i+7];
		sum=0;
		p2=1;
		for(k=7;k>=0;k--)
			{
			if(num[k]!=0)
				sum=sum+p2;
			p2=p2*2;
			}
		fprintf(fp2,"%c",sum);
		}
}

void rshift(int d)
{
	int x,i,j;
	for(j=0;j<=d;j++)
	{
		x=data[n-1];
		for(i=n-1;i>0;i--)
		data[i]=data[i-1];
		data[0]=x;
	}

}

void lshift(int d)
```

```
{
        int x,i,j;
        for(j=d;j>=0;j--)
        {
                x=data[0];
                for(i=0;i<n-1;i++)
                data[i]=data[i+1];
                data[n-1]=x;
        }
}


char lshift_residual(char c,int d)
{
        int x,i,j,sum,p2;
        int bit_c[8];
        int k,n1;
        int mask=128;
        n1=0;
        mask=128;
        j=c;
        while(mask!=0)
        {
        k=(j & mask)?1:0;
        bit_c[n1++]=k;
        mask=mask>>1;
        }

        for(j=0;j<d;j++)
        {
                x=bit_c[0];
                for(i=0;i<n1-1;i++)
                bit_c[i]=bit_c[i+1];
                bit_c[n1-1]=x;
        }
      sum=0;
                p2=1;
                for(k=7;k>=0;k--)
                        {
                        if(bit_c[k]!=0)
                                sum=sum+p2;
                        p2=p2*2;
                        }
return (char)sum;
}

void bit_wise_xor()
{
        int i,x,y,z;
        for(i=0;i<n-1;i=i+2)
        {
```

```
                x=data[i];
                y=data[i+1];
                z=(x ^ y);
                data[i]=z;
        }
}

/*start of randomization*/

        void randomization()
        {
        unsigned char ch;
        int  a=0,j,k,i,check,nc1;
        int nl;

        long int nblock;
        int ii,jj,ii1,jj1;


                                        for(a=1;a<=times;a++)
                                                {
                                                check=a%5;
                                                        switch(check)
                                                        {

                                                        case 0 :cycling();
                                                                upshift();
                                                                rightshift();
                                                                downshift();
                                                                leftshift();
                                                                downshift();
                                                                rightshift();
                                                                upshift();
                                                                cycling();
                                                                break;
                                                        case 1 :leftshift();
                                                                cycling();
                                                                upshift();
                                                                rightshift();
                                                                downshift();
                                                                rightshift();
                                                                upshift();
                                                                cycling();
                                                                leftshift();
                                                                break;
                                                        case 2 :downshift();
                                                                leftshift();
                                                                cycling();
                                                                upshift();
                                                                rightshift();
                                                                upshift();
```

```
                                        cycling();
                                        leftshift();
                                        downshift();
                                        break;
                                case 3 :rightshift();
                                        downshift();
                                        leftshift();
                                        cycling();
                                        upshift();
                                        cycling();
                                        leftshift();
                                        downshift();
                                        rightshift();
                                        break;
                                case 4 :upshift();
                                        rightshift();
                                        downshift();
                                        leftshift();
                                        cycling();
                                        leftshift();
                                        downshift();
                                        rightshift();
                                        upshift();
                                        break;

                                }
                        }
}

/*** rightshift() function starts ***/

void rightshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=0;k<siz;k++)
        {
        ch1=mat[j][k];
        m=((j*siz)+k)+1;
        r=j;
        c=k+m;
                while(c>=siz)
                {
                r=r+1;
                        if(r==siz) r=0;
                        c=c-siz;
                }
                if(m>((siz*siz)/2))
                {
```

```
                c=c+1;
                        if(c==siz) r++,c=0;
                        if(r==siz) r=0;
                }
        mat1[r][c]=ch1;
        }
}
for(r=0;r<siz;r++)
{
        for(c=0;c<siz;c++)
        {
        mat[r][c]=mat1[r][c];
        }
}

}

void downshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=0;k<siz;k++)
        {
        ch1=mat[k][j];
        m=((j*siz)+k)+1;
        c=j;
        r=k+m;
                while(r>=siz)
                {
                c=c+1;
                        if(c==siz) c=0;
                        r=r-siz;
                }
                if(m>((siz*siz)/2))
                {
                r=r+1;
                        if(r==siz) c++,r=0;
                        if(c==siz) c=0;
                }
        mat1[c][r]=ch1;
        }
}
for(r=0;r<siz;r++)
{
        for(c=0;c<siz;c++)
        {
        mat[r][c]=mat1[c][r];
}
}
```

```
}

void leftshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=siz-1;k>=0;k--)
        {
        ch1=mat[j][k];
        m=(j*siz)+(siz-1-k)+1;
        r=j;
        c=k-m;
                while(c<0)
                {
                r=r+1;
                        if(r==siz) r=0;
                        c=c+siz;
                }
                if(m>((siz*siz)/2))
                {
                c=c-1;
                        if(c==-1) r++,c=siz-1;
                        if(r==siz) r=0;
                }
        mat1[r][c]=ch1;
        }
}
for(r=0;r<siz;r++)
{
        for(c=0;c<siz;c++)
        {
        mat[r][c]=mat1[r][c];
        }
}
}

void upshift()
{
unsigned char ch1,ch2,ch3;
int j,k,m,r,c;
for(j=0;j<siz;j++)
{
        for(k=siz-1;k>=0;k--)
        {
        ch1=mat[k][j];
        m=(j*siz)+(siz-1-k)+1;
        c=j;
        r=k-m;
                while(r<0)
```

```
                {
                c=c+1;
                if(c==siz) c=0;
                r=r+siz;
                }
                if(m>((siz*siz)/2))
                {
                r=r-1;
                if(r==-1) c++,r=siz-1;
                if(c==siz) c=0;
                }
        mat1[c][r]=ch1;
        }
}
        for(r=0;r<siz;r++)
        {
                for(c=0;c<siz;c++)
                {
                mat[r][c]=mat1[c][r];
                }
        }
}



void cycling()
{
int j=0,k=0,r,c,size,start;
unsigned char ch1,ch2,ch3;
size=siz;
start=siz-size;
while(1)
{
        if(start==siz/2) break;
        ch1=mat[j][k];
        if(j==start && k>=j && k!=size-1)   k++,mat1[j][k]=ch1;
        else if(k==size-1 && k>j && j!=size-1)    j++,mat1[j][k]=ch1;
        else if(j==size-1 && j>=k && k!=start)    k--,mat1[j][k]=ch1;
        else if(k==start && j>k && j!=start)     j--,mat1[j][k]=ch1;
        if(j==start && k==start)      start++,size--,k=j=start;
        if(start%2==1)
        while(1)
        {
        ch1=mat[j][k];
        if(k==start && j>=k && j!=size-1)   j++,mat1[j][k]=ch1;
        else if(j==size-1 && j>k && k!=size-1)    k++,mat1[j][k]=ch1;
        else if(k==size-1 && k>=j && j!=start)    j--,mat1[j][k]=ch1;
        else if(j==start && k>j && k!=start)     k--,mat1[j][k]=ch1;
        if(j==start && k==start)      start++,size--,k=j=start;
        if(start%2==0) break;
        }
```

```
        }
        for(j=0;j<siz;j++)
        {
                for(k=0;k<siz;k++)
                {
                mat[j][k]=mat1[j][k];
                }
        }
        }

/*start of keygen function*/

        void keygen()
        {
        unsigned char key_string[50];
        long int s,s1,s2,power,p=0;
        int l,n;
        unsigned char ch;
        FILE *fp1;

        s=0;
        clrscr();
        printf("\n\n\nEnter your secret key string(Max. 16 chars. long) :");
                while(1)
                {
                ch=getch();
                if(ch!=13)
                {
                printf("*");
                key_string[s++]=ch;
                }
                else
                {
                key_string[s]='\0';
                break;
                }
                }
        s=0;
        for(l=0;key_string[l]!='\0';l++)
        {
        power=1;
        for(n=1;n<=l;n++)
        {
                power=power*2;
        }
        s+=power*key_string[l];
        }
        s1=0;
        s2=s;
        while(s2!=0)
        {
```

```
p++;
s1=s1+((s2%10)*p);
s2=s2/10;
}
secure=s%s1;
if(secure==0)
secure=s1;
for( ;secure>32; )
secure=secure-32;
if(secure>32)
secure=secure/32;
s1=0;
s2=s;
while(s2!=0)
{
s1=s1+((s2%10)*p);
s2=s2/10;
p--;
}
times=s%s1;
if(times==0)
times=s1;
for( ;times>32; )
times=times-32;
if(times>32)
times=times/32;
printf("\n\nNumber of times of the randomization process will
 continue :%d\n",times);
printf("\nNumber of times of Decryption is to be done :%d\n",secure);
printf("\nPress any key to continue--->");
getch();
}

/* Function To decrypt an encrypted file using MSA algorithm */

void msa_decryption(char file1[],char file3[])
{
char file4[50]={'T','E','M','P','1','.','D','A','T','\0'};
long int i,nrec,nloop,flag,n1;
int a;
unsigned char ch;
clrscr();
nrec=0;

fp1=fopen(file1,"rb");
fp2=fopen(file4,"wb");
n1=0;
while(fscanf(fp1,"%c",&ch)>0)
{
fprintf(fp2,"%c",ch);
n1++;
```

```
}
fcloseall();
for(a=0;a<secure;a++)
{
fp1=fopen(file4,"rb");
fp2=fopen(file3,"wb");
        while(fscanf(fp1,"%c",&ch)>0)
        nrec++;
rewind(fp1);
nloop=nrec/2;
        if((nrec%2)!=0)
        flag=1;
        else
        flag=0;
        for(i=1;i<=nloop;i++)
        {
        fread(&mydata1,sizeof(mydata1),1,fp1);
                if(mydata1.str[0] == mydata1.str[1])
                {
                decrypt1(mydata1.str[0],mat);
                decrypt1(mydata1.str[1],mat);
                }
                else
                decrypt(mydata1.str[0],mydata1.str[1],mat);
        }
        if(flag==1)
        {
        fscanf(fp1,"%c",&ch);
        fprintf(fp2,"%c",ch);
        }
fcloseall();
nrec=0;
fp1=fopen(file3,"rb");
fp2=fopen(file4,"wb");
        while(fscanf(fp1,"%c",&ch)>0)
        fprintf(fp2,"%c",ch);
fcloseall();
}  /*End of Decryption Process*/
}

void decrypt(unsigned char ch1,unsigned char ch2,unsigned char mat[siz][siz])
{
int j,k,m,n,p,q,c;
for(j=0,c=0;(c<2)&&(j<siz);j++)
        for(k=0;k<siz;k++)
                if(mat[j][k] == ch1)
                        {
                                m=j;
                                n=k;
                                c++;
                        }
```

```
                else if(mat[j][k] == ch2)
                        {
                                p=j;
                                q=k;
                                c++;
                        }
                n--;
                q--;
                if(n==-1)
                {
                        n=siz-1;
                        m--;
                        if(m==-1) m=siz-1;
                }
                if(q==-1)
                {
                        q=siz-1;
                        p--;
                        if(p==-1) p=siz-1;
                }
                n+=q;
                q=n-q;
                n-=q;
fprintf(fp2,"%c%c",mat[p][q],mat[m][n]);
}

void decrypt1(unsigned char ch1,unsigned char mat[siz][siz])
{
int j,k,m,n,q,c;
for(j=0,c=0;(c<1)||(j<siz);j++)
        for(k=0;k<siz;k++)
                if(mat[j][k] == ch1)
                        {
                                m=j;
                                n=k;
                                q=n-1;
                                c++;
                        }
if(q==-1)
{
q=siz-1;
m=m-1;
if(m==-1) m=siz-1;
}
fprintf(fp2,"%c",mat[m][q]);
}
/*  End of MSA Algorithm  */
```

Testing

We ran tests with the software with all possible files and we found that the encryption algorithm is working perfectly and the encrypted file embedded in the carrier can also be extracted successfully followed by subsequent decryption. Here we are giving some of our experimental findings.

In all our experiments we have used same text-key=”45”. The calculated randomization number=18 and encryption number=11.

Case-1: File type: Text/ASCII

Original File Name:sample.txt (Size=2,992 bytes)

Cryptography is the science of analyzing and deciphering codes, ciphers and cryptograms. In today’s world information security is of utmost importance. And so are the means to ensure information security. But these methods which promise security can also be used to intrude on private data by exploiting certain points which are known only to the creator. Thus such techniques to hide information or render them undetectable are always greeted with mixed emotions. This is another such attempt to present a pretty simple block cipher technique to encrypt data with amazingly decent outcomes.
2. Paper Outline

First we state the basic structure of a Feistel cipher routine, and then describe one of the popular block cipher algorithms, DES. We further isolate the regions where changes might be possible to repel cryptanalysis. Then we propose the new encryption algorithm by modifying the Feistel structure as required.

3. Feistel Cipher Structure

Feistel cipher is a symmetric structure which is used in the construction of most block ciphers. It is named after the German physicist and cryptographer Horst Feistel who was associated with IBM for a significant part of his life. A large number of block ciphers use this scheme, including the Data Encryption Standard. The primary advantage of the Feistel cipher is that the encryption and decryption operations are almost identical. Thus the size of the circuitry to implement such a cipher is greatly reduced. They were first implemented in IBM’s Lucifer cipher. Feistel networks gained popularity when the US Federal Government adopted the Data Encryption Standard.
We depict the Feistel structure using the following diagram:
This is the most widely used encryption scheme until recently. It was adopted in 1977 by the National Bureau of Standards, as Federal Information Processing Standard 46. The algorithm itself is referred to as the Data Encryption Algorithm (DEA). In this algorithm, data is encrypted as 64-bit blocks, using a 56-bit key. It consists of 16 successive rounds, each involving confusion and diffusion. With a key length of 56 bits, there are 256 possible keys. Thus, a brute-force attack does seem impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

Though there were suspicions that the NSA have installed a backdoor in the algorithm that only they can exploit, thus able to decrypt any text encrypted with DES, it was adopted as a federal standard after about four years of critical analysis. DES was also primarily responsible in developing and improving different cryptanalytic attacks. According to renowned cryptographer Bruce Schneier, "DES did more to galvanize the field of cryptanalysis than anything else. Now there was an algorithm to study." There probably has never been a symmetric key block cipher that was never compared to the DES.

Encrypted File: sample1.txt (size=2,992 bytes) (It is the encrypted form of sample.txt)

‰†ÍBZ"• ©«◘ ¡ç'J^la┘ gIA3]Iy3ë¶‼ É
«┘ õËß˜&½Ó┘ ájšÉ^`H&)◀ ³ À»*⇳ ö\b™Ÿ,¤åÑÀª"© ©r*f*L«MeJM
‼ ù43û" tÙ=Š
mHWÐ%©Í-;*f*m T6[ªKd↕ mZ~Kïò┬ m /ËýbE • ©⇳ @mNÃL*f*Íd ÌM•,\ª
:Ë¦┴ ãýïì¦ÕBFš&«o !ëžH©daY# D³|€«8Ë¾Ü` ?¨┘ åXI•─ ¹HE-
b€É‰^pJ³] 3þ├ ùyOº¶ ü©
'}KK'└ ©%DiOšÈ^\'Ik €*f*Ý`ʃZÿ² d˜ ◘ 8å@B–└ ±•
¡[ʃî¨Æ9d³J 2nXªZÏ²┴ l½/(▏ ÅaÆ·─ « ©âòÍ‹_`H-├ À¢^HªJ_¦"wÝ)
—õiÆ • «!┘ ëNŠNʃDqhš? *f*û,=┐ Ÿ² MÉ?+ õèŽž"¹ëD+┐ ÒL
dl²M◀ "<É
Ã`+²┬ âÉ?i• ÝbFº┐ ©┘ ʃbšJ¡L,┘ ZIq"ü±8K_ò}m)/Î…ýRCº└ »3@án®J
D,↑ « %³\P²└ ¯º B‰-©UÁG™#©fDãjÂÌ«T4`®[À³žÒ%↕ {º
}MKŠ¶ å@Cž└ «D◘ ãÎ†L F`HžO┼ "-ÃÙJC² r9 ª▏ }KC˜#©se[,M¢┤ 0h*)▏↕ xø?
⇳ ²UoÉ `ªÕ`F´&»└ ©c¢ÍŠguMv├┘ ‼ Ý@b oú|ò‰Ýn,ÙaS—#=v┘ {ºÌ¢@T
tº*Q'YËùBG²=Lè
A▏ äSË" ù1
¡…┬ l+Ö0d'K`┐ öI└◀ïê|Ò =Bå`W^#©% ʃF„j©Ìd,^ …3^s¢;ëº-
tÉ©ÊåéBº• «q ¡-ÆÌ TpPã◀└ Ý◘ Þ┐ ⇳
º¶ ð‰olŽÝcR°&¿← ◘ ¡BÚOªæ(P3← a↕ L(&┐ O¶5l-úÀ<ä@Ã − ©U B¢Î┐ äaH'
G▏ '÷zʃ1Oº5v‰¸O┐ ýâÊµ• ¹i@!h H¡FaV6=€"\hp└ ïúÜt©œ
┬ ÍbG"«%D¡Lšl€ÝtB? ¤‼ NHã2Î²ßb½^ íXVµ&-)@ └ *f*oʃÔxp»-0"x┼ 4
_þ=xÍ?j$ýé€˜• ± ┘)B*f*Ì¡Lss¦)
↕ ├ Ò?kÛò5c +j¶ íá┤ 'ço@/Ä,Œ€4|◘ G┴ *f*Zè';oº7 É«h┴ õZ└ 8┘ s.
ëy;¨~d┘ 2├ ±↕ Ìp±cI²Ì+ HÝJÂ"ëI◘ ʃÇ¶I¨Gb → Lp‼ ÉrBó²ÿc‰;
&írÀ°└ é ◘ «fÆNʃl6eÒ ‘³[├:2Cº┤ DéOÀ¥MbB,• ©ÿ▏ !DóL*f*Í&T*[…3]P¤
3îº_|Ý
"õÐG'#¼î┘ çJ†N©Fd!⇳┘ "^A¤← k²¶ r-ý6åh[˜#é
áøÊ,¢GsA˜] 3lKâBÃ¦œ`¹ÿjœåxB³┐ -eHé@šŒ«æ(cB
'‼
VRâ¶^b‰-N…åIV€"û└
ãRÎL«§d,┐ ┼'<É3RWö0L¹½n åZÂ€"Ù/@!E›N¢gqt,?Ð*f*üA,┐ Çâ
Üb™O*®}pÚ └íOAécÂÌ¢D4L353È'"→ Ãö¶ l©mMªäPB¶┐ ¨ ┘ !y"NŠNmDsY
└ nÁ$ZÌº┤ n
õHÖ³─ -û af²N¨Åa,Ã⇳ ³^C ◘ Ûê=¦™M ┬ å@C'&½- ‰└
,m*f*Trq;M¶ ‼ \↕ –→ Kþ èÉ½Ì–ýKB¸#»@AeÛ'm© `
*f*A↕ \B©zC¶¶ ãéjIý#R˜└ «┘
*f*ÂÊOª.}e,80"Üº9┐ Oú¶ t-.J$õXÏ'─ ù*▏ ¥ðÞÌ*f*Ubd←]À3^JppÏº

TÀÍ-$íâÀ¼– í‼ @gbÚl*ƒ*– z@Ã├ *ƒ*Ø
)Ë²•t‰/◘ ¼ÍÃÊ´● »/ ¡BªL nd┼ └ m5#Ìá¶
← j²◀ n - üPJº ¢*ƒ* -Ã3Ê©^$b2-*ƒ*#þÁ

xkº├ Ví¥õQR ┐ µ◘ %S³É©W+@W Å‼ ÝÙâr÷ê péýŠ…½bD │©A¡*┐ ì└ Å*
DóOa“^€¬
Ëþ?e J´åj┐ ^┐ áO │ àÏŒ£│E"┬ q ‼ ts³P[¾œ2ÝoŽ┘ -RD©● ©⛧ ┘ oq–l©D)fb
◀ 3^Ð5C[ºt@├ ← ,/íhV └ -Ï e2ÒNªÕe&«-◀ “ÞËn┐ _¾¶ <Ø?,¥í@V€#«‡L¡†
ŻK©D}Hš├¡“žA3Rÿ² VI«│ 'õH×↑ − é│ %¸çN¨U0dŠ⛧ º“ÜZt‼ Ïþ r^}N(ÝrB
š└ íL
¡J’È¨m-HæL│ ↕ (x→ bOvÕh‰⛧ È55HÔ¶&{ ek.ÌªF1H¢
²*ƒ*%← ò┴ pÌŸÉ┘ õùÂ±● ¹ã énCl dqg&k ↕ ?@SÏv•úµ;J†íBF ┐ Ù
)¶ †L¢T;E{OQ"LB4PçöTM¹V.%eÀF ┐ ý$ %ë’h¡Ä$zš
€3lP0zë¾Œb- C◘ "D
”
é↑ │ %oµÌ¨^-X&. “\s Ûò]x KO„ícG´#·# e«¢è l r
t└ î ¨ZK²”`Œ-ëíA‡ª– £à@ [Þì¡┬ d
┐ →┼ “YQdKâ¶ Þ) ‹5õÒL¿&©I obŠê¢^}@O├◀ ³ÚøS+»º]tè,®-ýÊBš#¹Æ
çR‹N & ¢ð↕ \Ð¬Jïö┴ ûI?D'U@^š– ñw┘ ¡ËÂl¢œ
&I'◘ ‼ ìq3ß– ré⛧ ↕ m`F¨"¹! O“n¡¶ <cC
$↕ Y9m
⛧ ¾×Ð-] åØÁ‘└ ÉM◘ mr’Î£Fa}k
a£\È:Ãú│ÀøOC%õbÓ™#ím┘ -jÓì£f4J¶ OP“]◀ "↕ Óî¶ z├ -
¬õBÚ ┐ -‡
AÎ¶L«§!)*ƒ*€‼ l@"BGª┤ f ^– ÕbÂ¹└ í9@!F“H©Æ$◘ [┘ ‼ ,Ø²2ç²¶n¨⛧ Â¬uëB
9– 9┤ │ eÄ*ƒ*Ì£Æ0qßà“\R° ë° 6˜ÿ¯„î`Þ²'«Ó◘ áê’Ê‰dpCV 3?Épj‹
¶┤ ã™^„Õ@B’● ©ã┘ ©ÒM^MqN^┼ ‼ ŸX]‹²Üò ½¨ éÈA←¸U│ …b¢L¢D⛧ $/
M 2_Û6j[¶4A©*L⛧ ýb→ ¶ ● »%HãF,ÌŠV(\Ê P’\¶↕ Kò4©Í
(ÅàÂ¾#©ã «**ƒ**N¨│ d ⛧ 0‼ ÙH‼ ÿºVr
J¥åaÖ ● §…EëB–Ì£V!I¢9±‼ ŠÊ!↕ ëþ┴ J©»A„ÅHF°└ ¹% v¢O‰duKr
!£Þ@ä→ Cº│áÙ{
†áBJ”┐¸©H│ ┐ ,Ï^LYJF[ð‼ Ý›€← O¢6â -Ê┘ ýàD¸"ñ
¡B,ì Ä(LÓ⛧
3ÎPõRkº<@ Ž¶ åÂCž– ¿£ ©Ïªn¨┤ 2H¢M!‼ \‰NA ò┬ PÍ⛧ ê µaÚœ● -?
£G²Nª ÅtXJ
LÀ‼ Î¹Ç šT`‰)– Ý@Ú¶● «ù ¡w¸J fsT#
ñ3]Àè← O¶┤ am=Ê│ ýkß‘– ½Ù◘ ● Z“Í¡wi`ŠkA‼ }H=B⛧ ö┤ p©íh„ýÀÎž&©7◘ 'f'Î£¥
pL*ƒ*}ð3m◀ 'C‰¶ E ß
‰¶ áÐZ´● á£ å`–€eadW├ A3]I_3Î³┬ À…,ê”ý@Ä’
·M
aZ,Ì¡Ìrg¶ ‼ ↑ ØrPK²[åÝ]n„íùÊ·"¿#A¡bÂL V4pZ ‼ Ã·┼ ãú4À©è¶ eÓP,● ¿WAáf
ŠÍ¨ì1d↕ ↕ ºØ³bûò¶ qÍÝx¶ ¥ÈA•┐ ¡⛧ @-J,J T=d– K 3ÿS‹Pçþ│Z‘<ê†åPÈ¹└ í

¥Gšl«De$┐ iP³ÜòK← ëª•p9mª┴ åkZž└ ©ã┘ ¥S’M¨O`dRNa↕ \I‹*¦¶ ð™ £{ £C+
"*šqhPhP

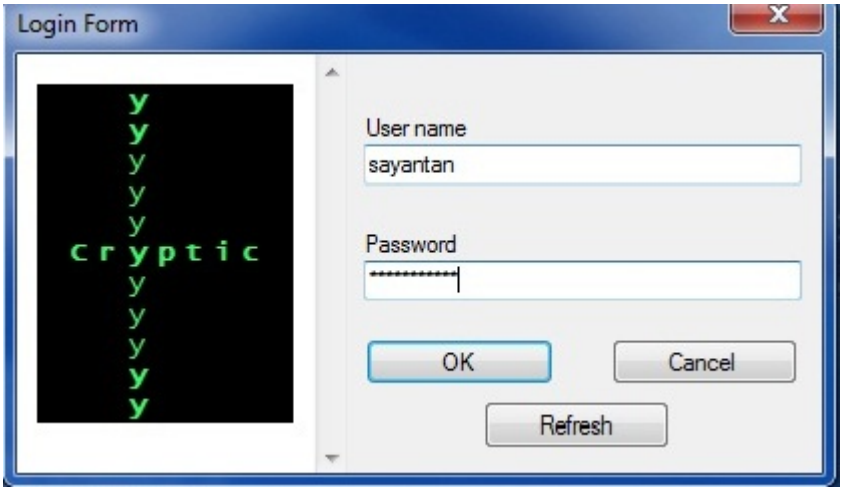
We apply our encryption and decryption algorithm on various other files and we found in every case it was working perfectly ok. In the following table the different type of files are given below:

Table: Consolidated report on encryption and decryption algorithm

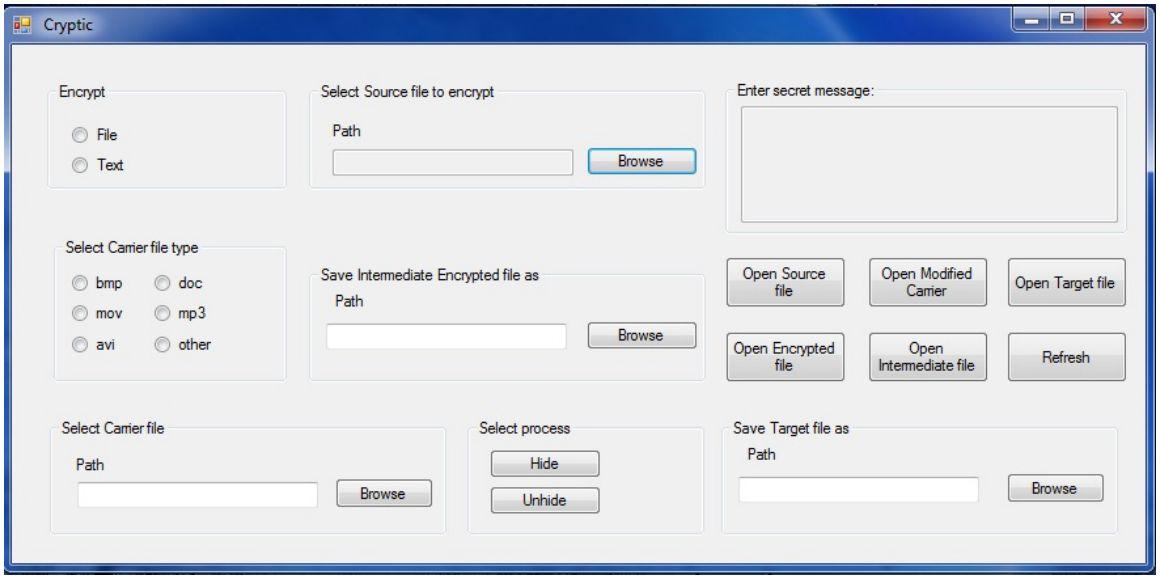
Sl. No.	Original File(size of File in byte)	Encrypted File(size of File in byte)	Decrypted File(size of File in byte)	Remarks
1.	Sample.doc(22,016 bytes)	Sample1doc((22,016 bytes)	Sample2.doc(22,016 bytes)	Original file and the decrypted file identical
2.	Sample.avi(82,944 bytes)	Sample1.avi(82,944 bytes)	Sample2.avi(82,944 bytes)	Original file and the decrypted file identical
3.	Sample.bmp(1,440,054 bytes)	Sample1.bmp(1,440,054 bytes)	Sample2.bmp(1,440,054 bytes)	Original file and the decrypted file identical
4.	Sample.com(69,886 bytes)	Sample1.com(69,886 bytes)	Sample2.com(69,886 bytes)	Original file and the decrypted file identical
5.	Sample.exe(1,26,976 bytes)	Sample1.exe(1,26,976 bytes)	Sample2.exe(1,26,976 bytes)	Original file and the decrypted file identical
6	Sample.jpeg(7,189 bytes)	Sample1.jpeg(7,189 bytes)	Sample2.jpeg(7,189 bytes)	Original file and the decrypted file identical
7.	Sample.mp3(5,184 bytes)	Sample1.mp3(5,184 bytes)	Sample2.mp3(5,184 bytes)	Original file and the decrypted file identical
8.	Sample.pdf(76,864 bytes)	Sample1.pdf(76,864 bytes)	Sample2.pdf(76,864 bytes)	Original file and the decrypted file identical
9.	Sample.wav(9,01,164 bytes)	Sample1.wav(9,01,164 bytes)	Sample2.wav(9,01,164 bytes)	Original file and the decrypted file identical
10.	Sample.xls(20,480 bytes)	Sample1.xls(20,480 bytes)	Sample2.xls(20,480 bytes)	Original file and the decrypted file identical
11.	Sample.ppt(37,84,192 bytes)	Sample1.xls(37,84,192 bytes)	Sample2.xls(37,84,192 bytes)	Original file and the decrypted file identical
12.	Sample.txt(2,992 bytes)	Sample1.txt(2,992 bytes)	Sample2.txt(2,992 bytes)	Original file and the decrypted file identical

Screenshots of the Software

Login Screen:

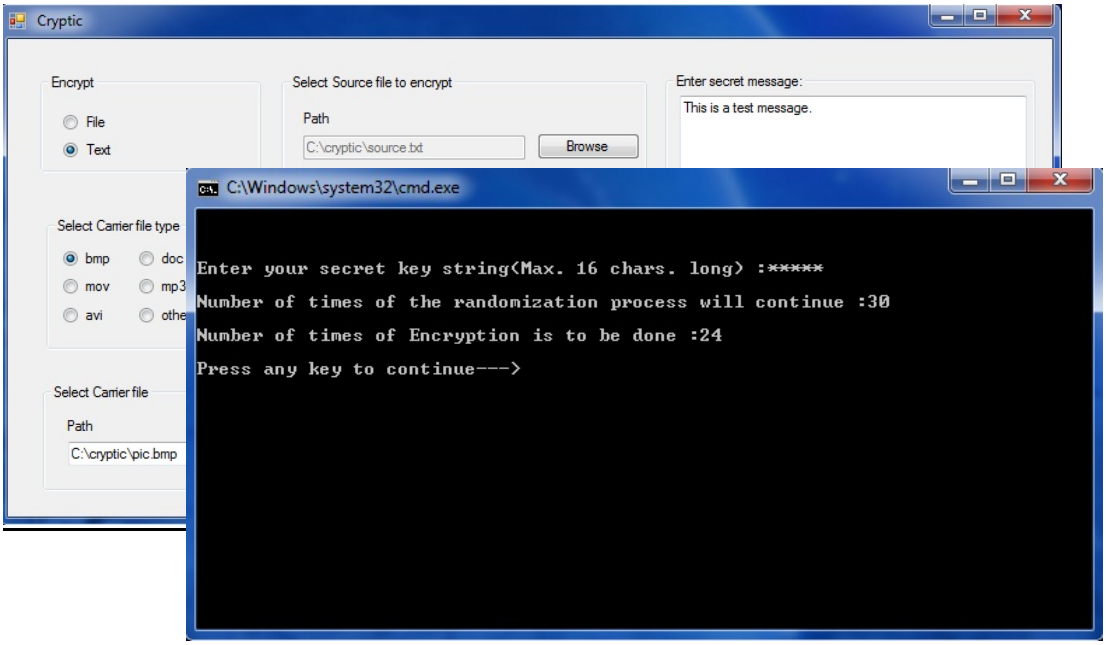


Front End:

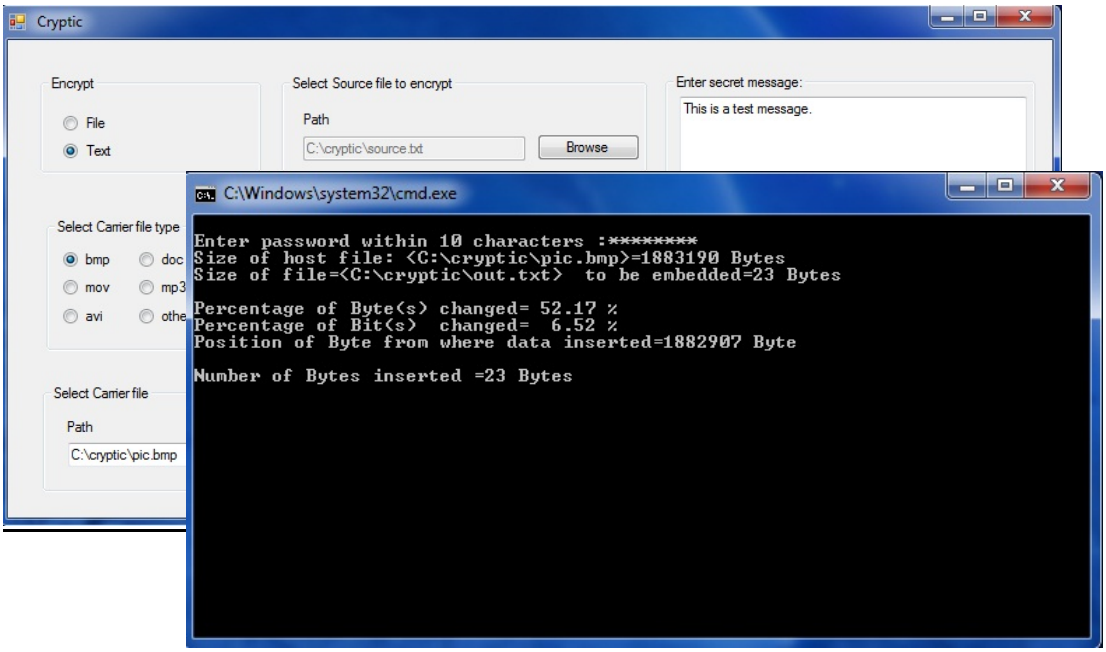


- **Data Hiding**

Data Encryption:



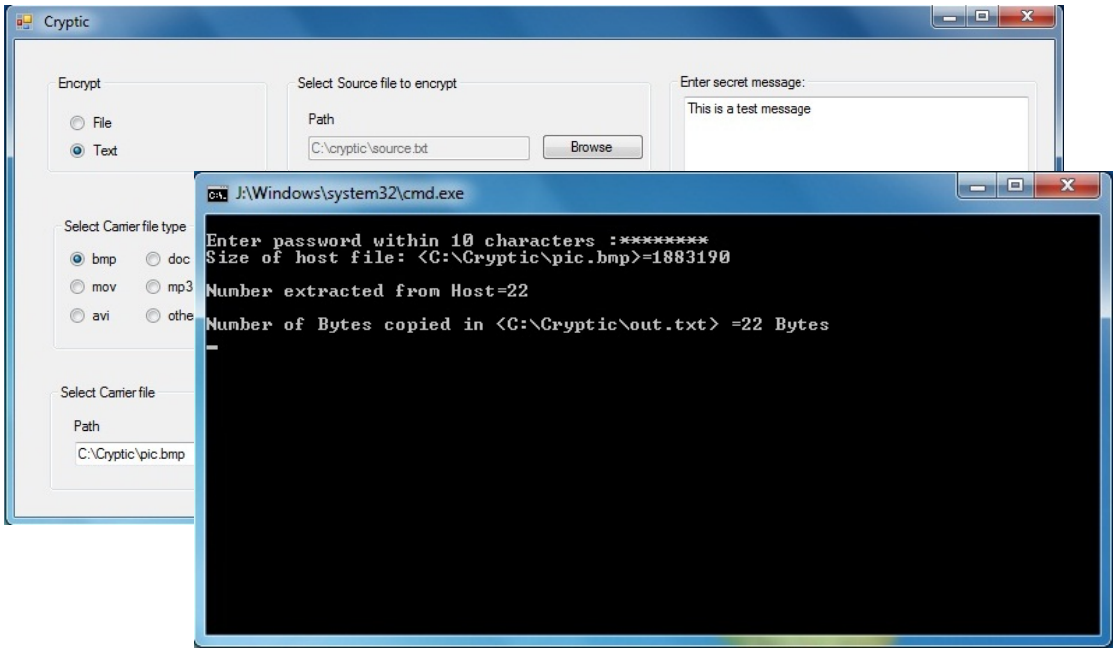
Data embedding in carrier file:



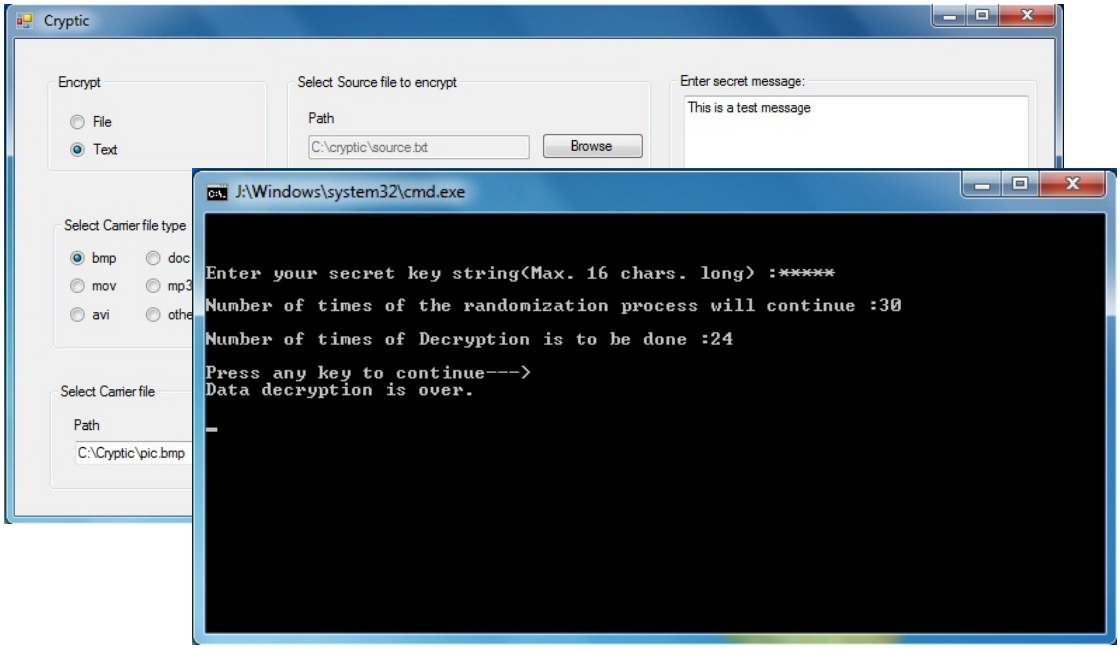
- **Data Retrieval**

Modified carrier file:

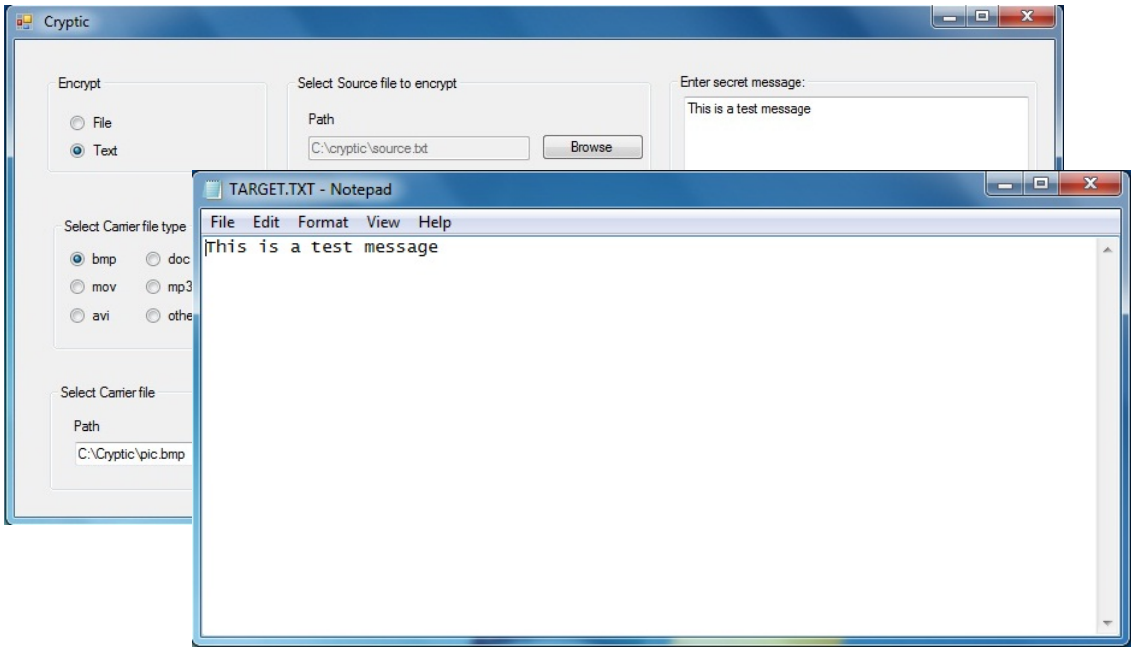
Data extraction from carrier file:



Data Decryption:



Decrypted text:



Advantages

- The software provides unparalleled level of security, by combining encryption and steganography.
- An advanced data encryption algorithm is implemented.
- Multiple encryptions are supported.
- The hidden object can be a simple plaintext file, an image, or even an audio/video file.
- Carrier files can be of any uncompressed file type.
- The software provides authentication check during login. Without a proper username and password the user cannot access the software.
- The software also provides high levels of security via a secure password used during encryption and during steganography without which the decryption would not be possible.

Limitations

- Although the software is portable, it cannot run on platforms other than windows. It cannot be made to run over LAN or any form of network.
- The size of the file to be hidden should preferably be approximately 20% of the size of the carrier file for efficient and fast steganographic operations.
- The steganography technique applied works well only on uncompressed files. If files like jpeg are used, the steganographic technique fails. Though it can be taken care of by using another advanced steganographic technique involving DCT and FFT.

Bibliography

1. Symmetric key cryptography using random key generator, A.Nath, S.Ghosh, M.A.Mallik, Proceedings of International conference on SAM-2010 held at Las Vegas(USA) 12-15 July,2010, Vol-2,P-239-244.
2. Modified Version of Playfair Cipher using Linear Feedback Shift Register, P. Murali and Gandhidoss Senthilkumar, UCSNS International journal of Computer Science and Network Security, Vol-8 No.12, Dec 2008.
3. SSB-4 System of Steganography using bit 4, J.M. Rodrigues, J.R. Rios and W. Puech Laboratoire, Universit´e Montpellier II, France, Compute Department, Universidade Federal do Cear´a, Brazil.
4. Fridrich, J. Applications of Data Hiding in Digital Images. In ISPACS'98 Conference,1998
5. Neil F. Johnson, Zoran Duric, S. G. J. Information Hiding: Steganography and Watermarking - Attacks and Countermeasures (Advances in Information Security, Volume 1). Kluwer Academic Publishers, February 15, 2001.
6. Stefan Katzenbeisser, Fabien, A. P. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House, Boston - London, February 2000.
7. Richard A. Mollin, An Introduction to Cryptography, CRC Press, 2000.
8. Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, Ton Kalker, Digital Watermarking and Steganography, Morgan Kaufmann Publishers, 2008.
9. Peter Wayner, Disappearing Cryptography - Information Hiding, Steganography and Watermarking, Morgan Kaufmann Publishers, 2008.
10. John Talbot, Dominic welsh, Complexity and Cryptography An Introduction, Cambridge University Press, 2006.
11. David Kahn, The Codebreakers - The Story of Secret Writing, 1967.
12. http://www.invisiblesecrets.com
13. http://www.webkclub.com/tte
14. http://wbstego.wbailer.com
15. http://en.wikipedia.org/wiki/Cryptography
16. http://www.cryptographyworld.com/

Reviewer's notes and comments