***STEGANOGRAPHY***

**Project Report – Software Lab**

**IV Semester 2008-09**

**Submitted by:**

ABHINAV KUMAR KUSHWAHA ALOK RANJAN SAURABH GUPTA

07000006 07000005 07000004

**Supervisor:**

*Dr. VINAYAK SRIVASTAVA*

*(Lecturer)*

*DEPARTMENT OF COMPUTER SCIENCE AND ENGINEEERING*

*INSTITUTE OF TECHNOLOGY*

*BANARAS HINDU UNIVERSITY*

*VARANASI*

 **

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**BANARAS HINDU UNIVERSITY**

**VARANASI-221005, (U.P.)**

**INDIA**

**CERTIFICATE**

This is to certify that **Mr. Abhinav Kumar Kushwaha, Mr. Alok Ranjan and Mr. Saurabh Gupta,** students of Department of Computer Science and Engineering, Institute of Technology BHU, Varanasihave worked on the topic **“STEGNOGRAPHY”** under my direct supervision and guidance for the 4th semester of B.Tech program 2008-09, the findings of which have been incorporated in this dissertation. They have worked diligently, meticulously and methodically. The report submitted by them embodies the literature from various reputed resources and materials provided by me during this period and is approved for submission.

**Dr. Vinayak Srivastava**

Lecturer

Department of Computer Science & Engineering

Institute Of Technology

Banaras Hindu University

# ACKNOWLEDGEMENT

I would like to convey my deepest gratitude to **Dr. Vinayak Srivastava**, Lecturer Computer Science Department, Institute of Technology BHU, who guided me through this project. His keen awe-inspiring personality, superb guidance, and constant encouragement are the motive force behind this project work.

I would also like to thank **Prof. A.K. Agrawal**, Head of the Department, Computer Science and Engineering, Institute of Technology BHU, for allowing us to avail all the facilities of the Department necessary for this project.

I am very thankful to all the technical and non-technical staffs of the Department of Computer Engineering for their assistance and co-operation.

Abhinav Kumar Kushwaha Alok Ranjan Saurabh Gupta

07000006 07000005 07000004

B. Tech –part II

Computer Science and engineering

Institute of Technology

Banaras Hindu University, Varanasi - 221005

**INDEX**

1. Abstract…………………………………….……5
2. Chapter 1: Introduction………………………….6
3. Chapter 2: Analysis & design…...……………....9

Context diagram……………………...9

Flow chart…………………………...10

Algorithm…........................................11

Data flow diagram…………………..12

Functions…………………………….13

1. Chapter 3: Implementation……………………..15
2. References…………...………………………….31
3. Appendix………………………………………..32

***ABSTRACT***

This project hides a secret text message into an image file using a key leaving image file practically unchanged so that no one can expect that some type of secret message is hidden behind the image file. The text message can be recovered correctly only by entering a correct key.

***KEYWORDS***

*Steganography, Cover file, Image Steganography, RSA Encryption, Image Format, LSB, J2EE*

**Chapter 1**

***Introduction***

***STEGANOGRAPHY:*** Steganography is the practice of hiding information “in plain sight”. This technique relies on a message being encoded and hidden in a transport layer in such a way as to make the existence of the message unknown to an observer. Importantly, the transport layer - the carrier file - is not secret and can therefore be viewed by observers from whom the secret message itself should be concealed. *The power of steganography is in hiding the secret message by obscurity, hiding its existence in a non-secret file. In that sense, steganography is* ***different from cryptography****, which involves making the content of the secret message unreadable while not preventing non-intended observers from learning about its existence.*

Because the success of the technique depends entirely on the ability to hide the message such that an observer would not suspect it is there at all, the greatest effort must go into ensuring that the message is invisible unless one knows what to look for. The way in which this is done will differ for the specific media that are used to hide the information. In each case, the value of a steganographic approach can be measured by how much information can be concealed in a carrier before it becomes detectable, each technique can thus be thought of in terms of its capacity for information hiding. This point is discussed in greater detail in the sections below.

***COVER FILE:*** It is the file in which we hide our secret message.

***IMAGE STEGANOGRAPHY:*** Steganography that uses an image file as cover file is termed as Image steganography.

***RSA ENCRYPTION*** ***:*** The algorithm was publicly described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT; the letters RSA are the initials of their surnames. The RSA algorithm involves three steps, key generation, encryption, and decryption. RSA involves a public key and private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated by the following way:

* Choose two distinct large random prime numbers *p* and *q*
* Compute***: η = pq***

***ׂη*** is used as the modulus for the public and private keys.

* Compute the totient: ***φ (η) = (p-1) (q-1)***.
* Choose an integer ***e*** such that ***1<e<φ (η)***, and ***e*** and ***φ (η)*** share no factor other than 1 i.e. ***e*** and ***φ (η)*** are co-prime.

***e*** is released as the public key exponent.

* Compute d to satisfy the congruence relation ***de = 1 (mod φ (η))*** i.e.

***de*** ***= 1+ k φ (η)*** for some integer k.

d is kept as the private key component.

The **public key** consists of the modulus ***η*** and the public (or encryption) exponent ***e***. The **private key** consists of the modulus ***η*** and the private (or decryption) exponent ***d*** which must be kept secret.

***IMAGE FORMAT*:** In the default RGB color model, an image consists of an array of pixels. A pixel is an integer with Alpha, Red, Green, and Blue (0xAARRGGBB). The Alpha value represents a degree of transparency for the pixel. Fully transparent is 0 and fully opaque is 255. Colors Red, Green and Blue are of 8 bits each in an order as shown above.

***LSB:*** LSB i.e. “Lowest Significant Bit” is the lowest bit of a color in a pixel such that change in its value has a very less impact on the quality of the picture.

* Consider a 24 bit picture
* Data to be inserted: character ‘*A’*: (10000011)
* Host pixels: 8 pixels will be used to store one character of 8-bits at LSB of blue color.
* The pixels which would be selected for holding the data are chosen on the basis of the key which can be a random number {Stego Key}.

Example:

00100111 11101001 11001000  
 00100111 11001000 11101001  
 11001000 00100111 11101001

Embedding *‘A’*

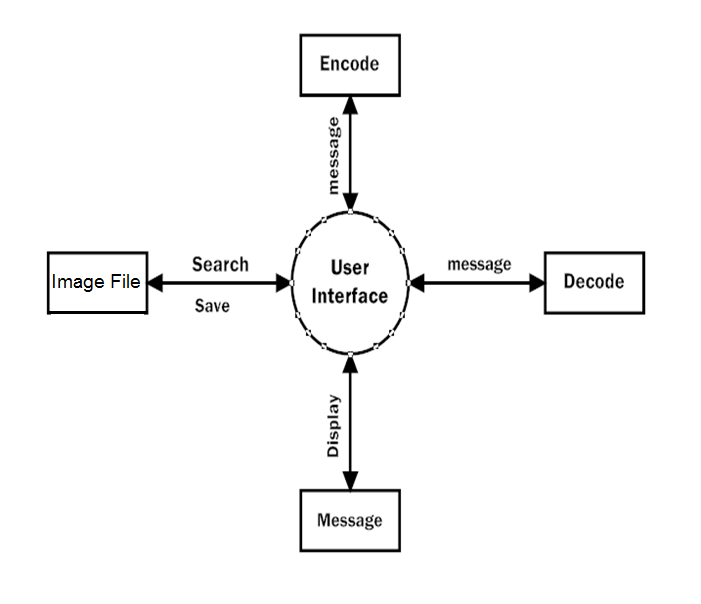
0010011**1** 1110100**0** 1100100**0**  
 0010011**0** 1100100**0** 1110100**0**  
 1100100**1** 0010011**1** 11101001

According to researchers on an average only 50% of the pixels actually change from 0-1 or 1-0.

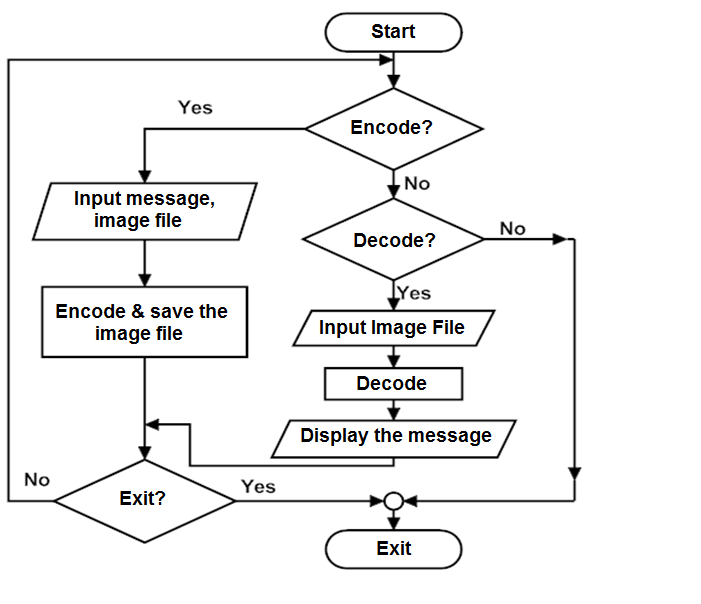
**Chapter 2**

***Analysis and design***

***CONTEXT DIAGRAM:***

**

***FLOW CHART:***

******

***ALGORITHM:***  Here Binary WAV encoding is used whose algorithm is as follows:

* **To hide message m in cover WAV file c:**

1. Read sound data from c into memory.

2. Read bitstream from m into memory.

3. Encode |m| into the cover file.

4. Modify the least significant k bits of the first n samples after the length

encoding (where |m| <=nk).

5. Write the loaded cover file c’ to disk.

* **To extract the secret message from c’:**

1. Decode the length |m| from the loaded cover.

2. Read off the |m| modified bits, given the knowledge that k bits per

sample were modified.

3. Write the bits to a file m’, where m’ = m.

***DATA FLOW DIAGRAM :***

Image file

(Cover file)

Secret message

Unique key

Encrypted image file

Secret message

Unique key

Image file

***FUNCTIONS:*** Following are the functions and their functioning :

1. **public void Audioencrypt(String message, File file, int key):** Thisfunction receives secret message, audio file(in WAV format) and a key consisting of numbers. Then it embeds the secret message into the audio file with the help of key and asks from user where to save the resulting output file.
2. **public void Audiodecrypt(File filename, int key):** This function receives the location of file which is to be decrypted and the key which is entered on decryption window. If the key is correct, correct secret message is displayed on message area. If key is not correct, some scribbled data is displayed on message area.
3. **public void actionPerformed(ActionEvent ae**): This function contains the code for action to be performed on pressing open, encode, decode and clear buttons. This functions also holds coding for sub-actions performed when a button is pressed. All functions performed by this functions are as follows:

**a). Open button:** When open button is pressed, a window appears

which asks for location where the audio file is currently stored.

Now, selecting a file and pressing open button is expected from user

to open our cover file. When user does above job, this function

checks whether the selected file is WAV file or not just by

comparing the extension name of file with wav.

If the file is not WAV, a window appears telling the user to select

only WAV file. If the file is WAV, it opens silently. Now, user is

expected to write the secret message in the space given on

right hand side and press the encode button.

**b). Encode button:** After entering the secret message in message area,

encode button is pressed by user. Now, a window appears asking for

the key for encryption. Key must be having numerals only. If key is not

entered, an error message box appears telling to enter the key. Now

user must close this error message box, again press encode button

where he/she should enter a non empty key and then press ok

button. If a non empty key is entered, another window opens asking the

name of resulting file and location where to store the encrypted file.

Then Audioencrypt function is called.

**c). Clear button:** After pressing the encode button, user should press

clear button located at bottom of message area to clear the

message.

**d). Decode button:** In order to decode an encrypted file, user should

press open button to open the encrypted file from source directory.

After this decode button must be pressed. A window appears, asking

the key. If nothing is entered, a window appears to enter the non

empty key. User should, now, press the decode button again to enter a

non empty key. After this Audiodecrypt function is called.

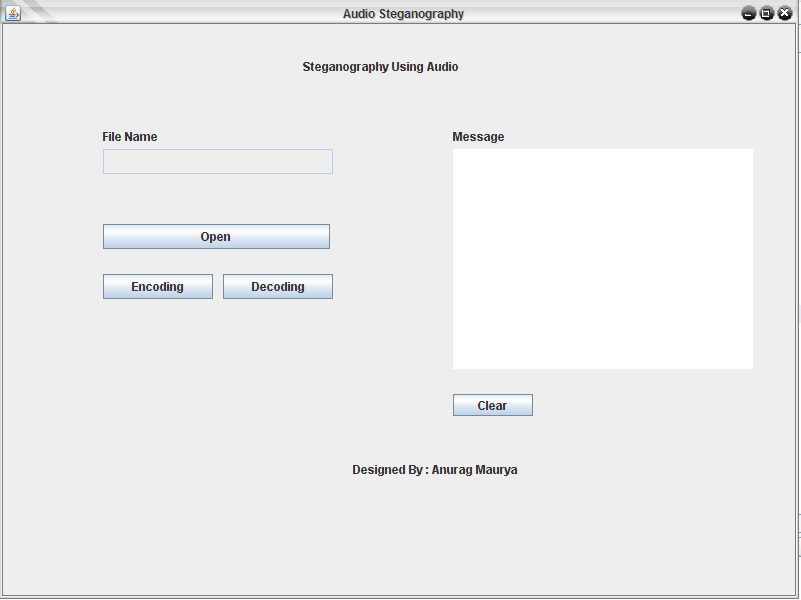
**Chapter 3**

***Implementation***

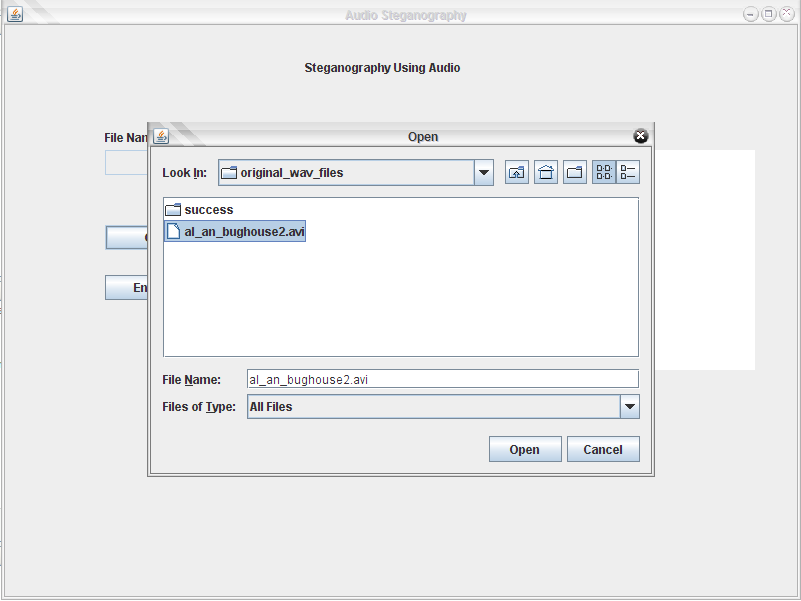
The project consists of a java source file named steghide.java .

When it is compiled and run in NETBEANS 6.0.1 , following window appears:

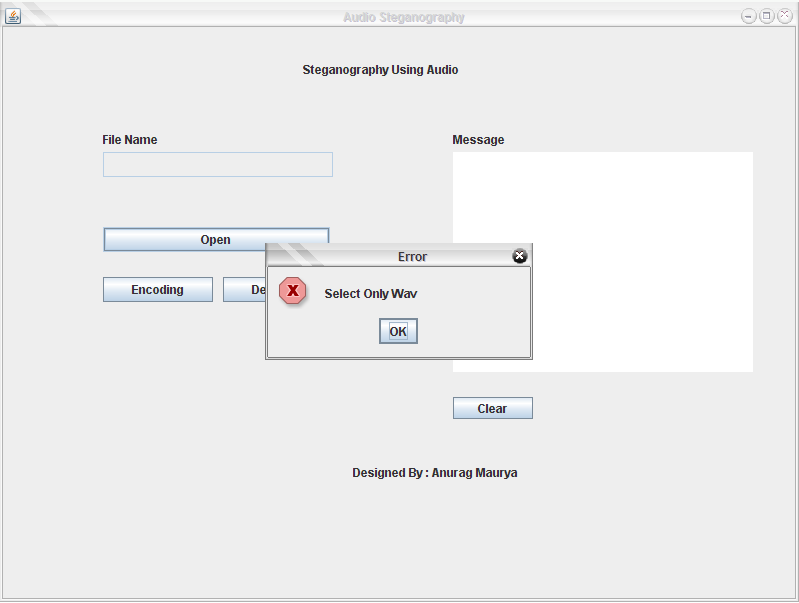
**1).** **ENCRYPTION**:



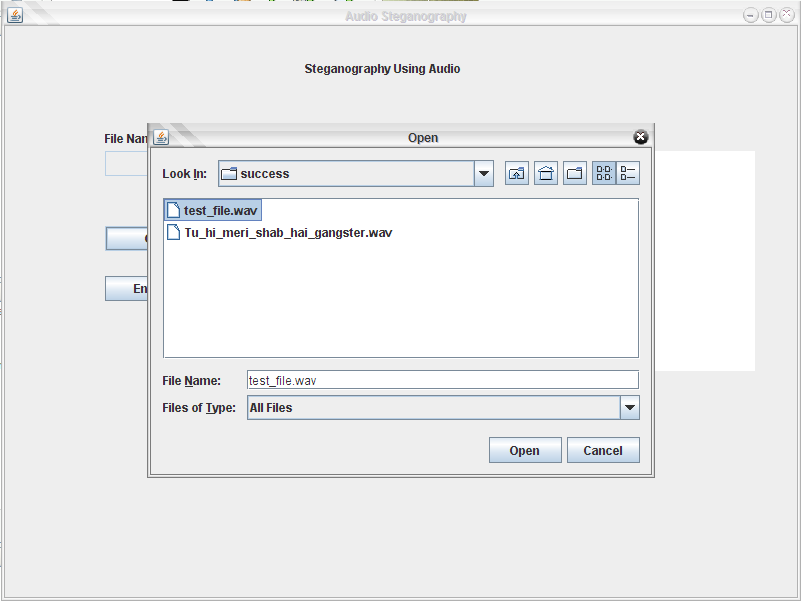
**FIGURE 3.1***: INITIAL WINDOW*



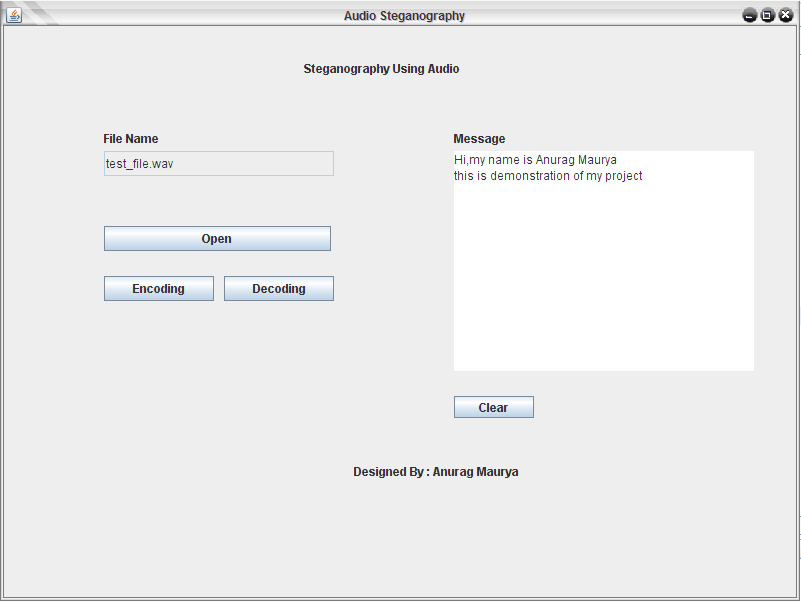
**FIGURE 3.2:** *SELECTING WRONG FORMAT FILE*



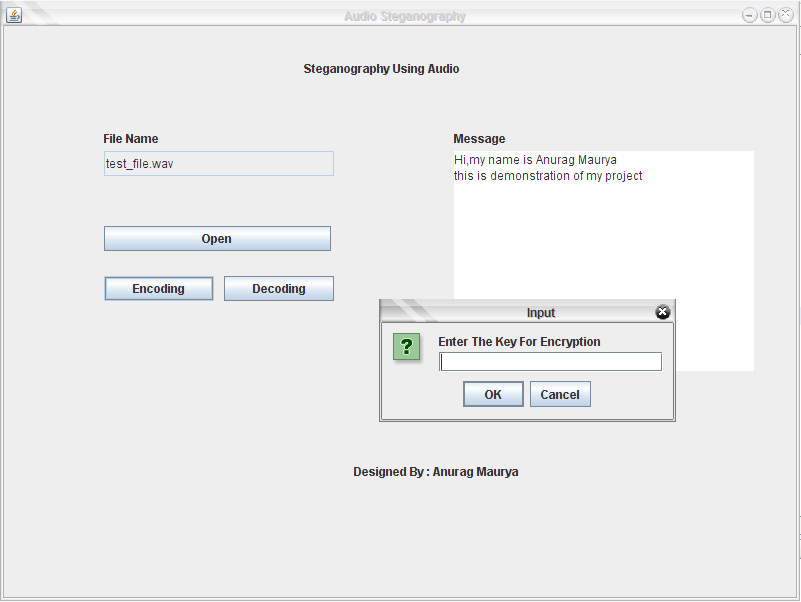
**FIGURE 3.3**: *ERROR MESSAGE ON SELECTING WRONG FORMAT FILE*



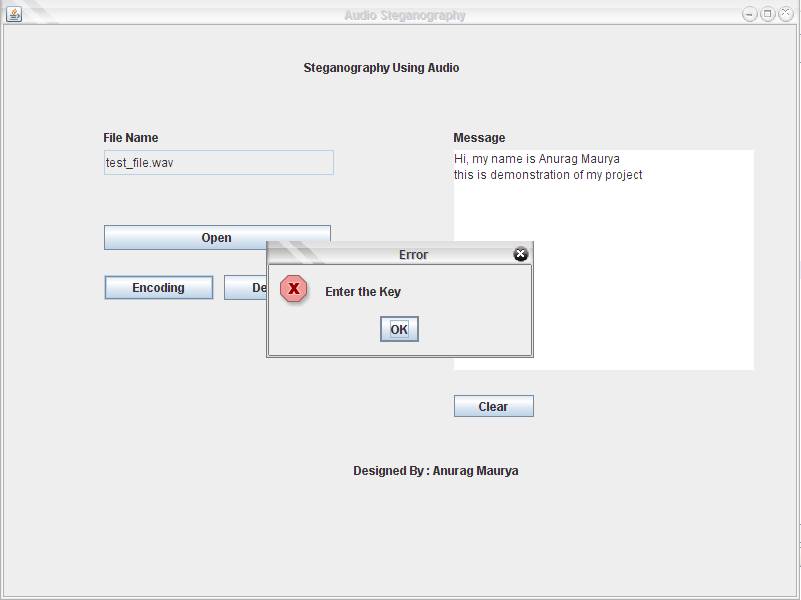
**FIGURE 3.4** : *SELECTING WAV FILE*



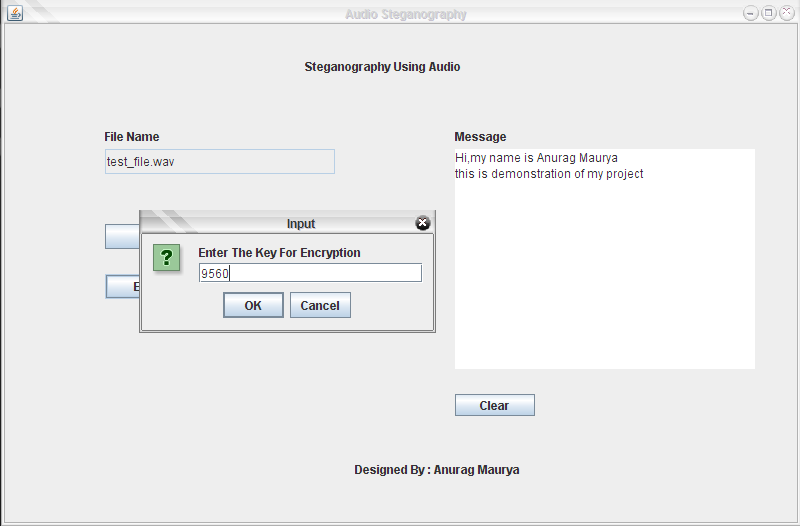
**FIGURE 3. 5**: *TYPING SECRET MESSAGE*



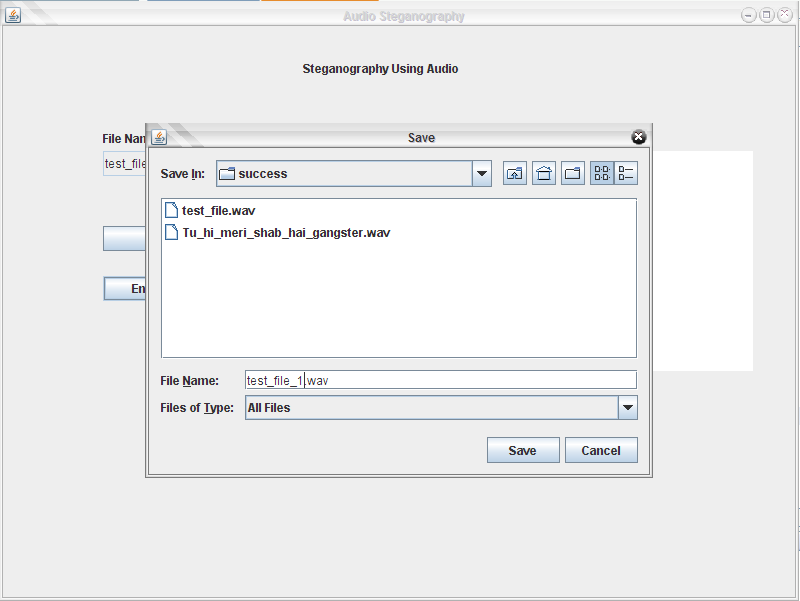
**FIGURE 3.6**: *ASKING KEY FOR ENCRYPTION*



**FIGURE 3.7**: *ERROR MESSAGE WHEN EMPTY KEY IS ENTERED*

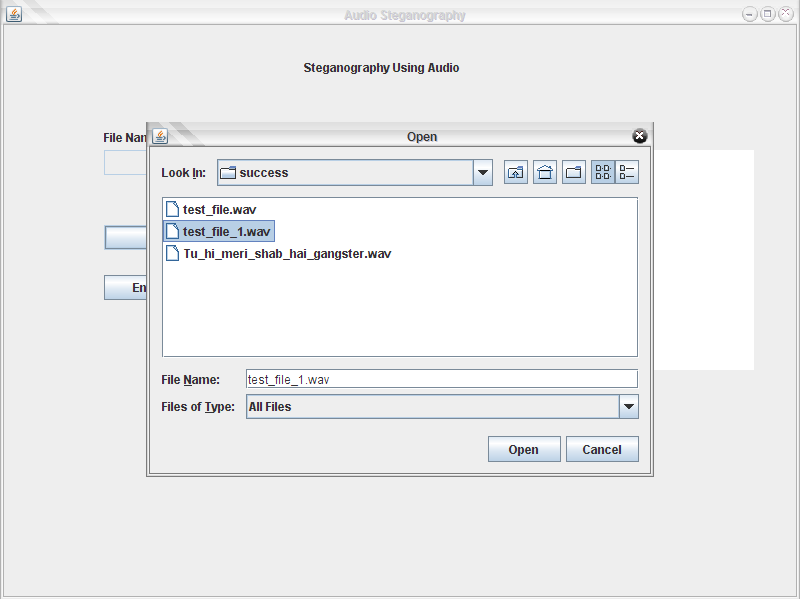


**FIGURE 3.8***: ENTERING NON-EMPTY KEY*

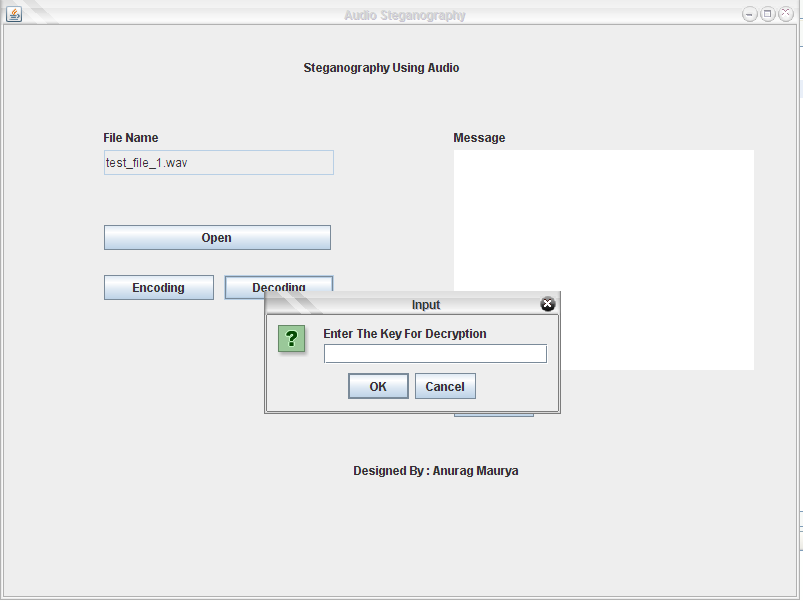


**FIGURE 3.9**: *ASKING FOR LOCATION AND NAME OF OUTPUT FILE*

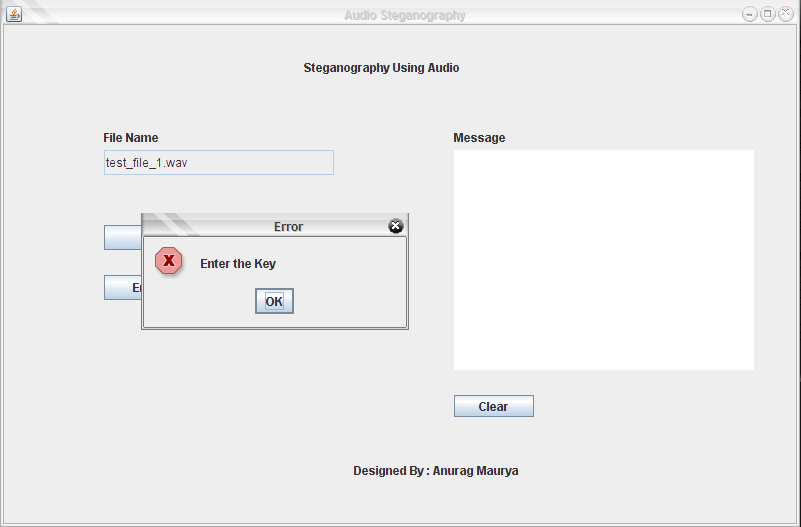
2). **DECRYPTION**:



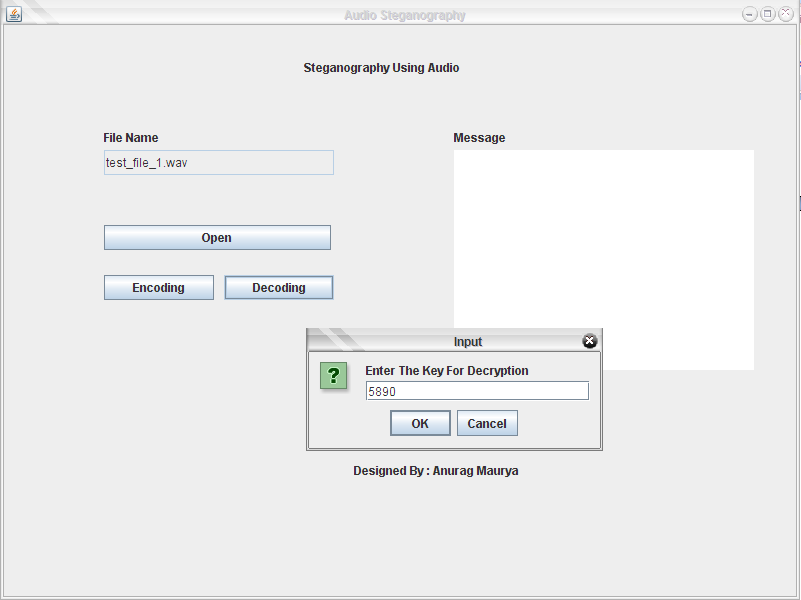
**FIGURE 3.10**: *OPENING ENCRYPTED FILE*



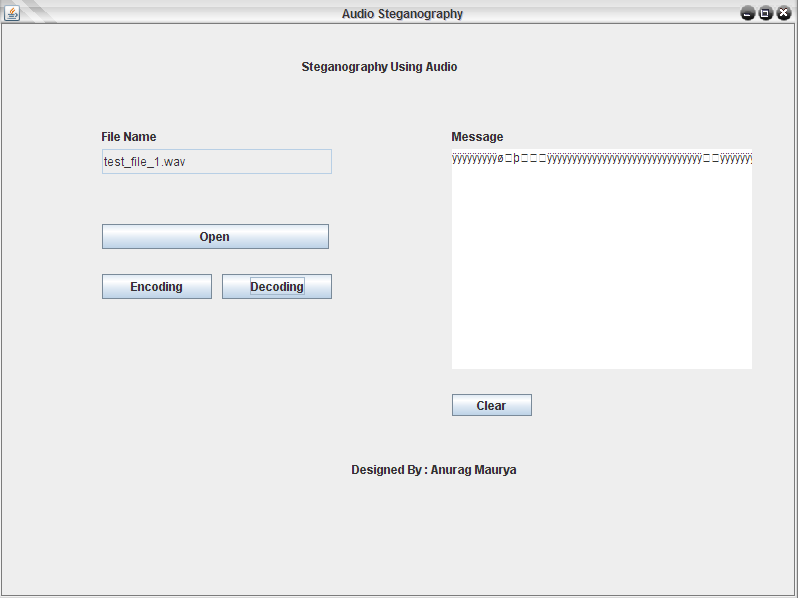
**FIGURE 3.11**: *ASKING FOR KEY TO DECRYPT*



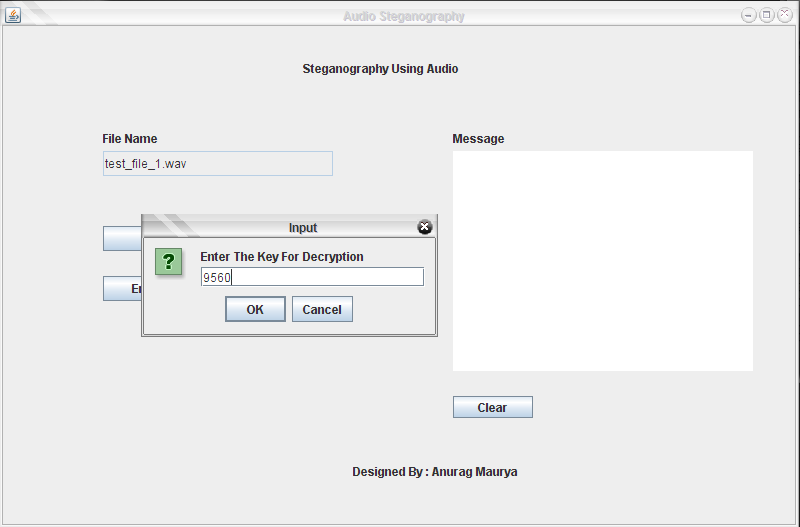
**FIGURE 3.12**: *ERROR MESSAGE ON* *ENTERING AN EMPTY KEY*



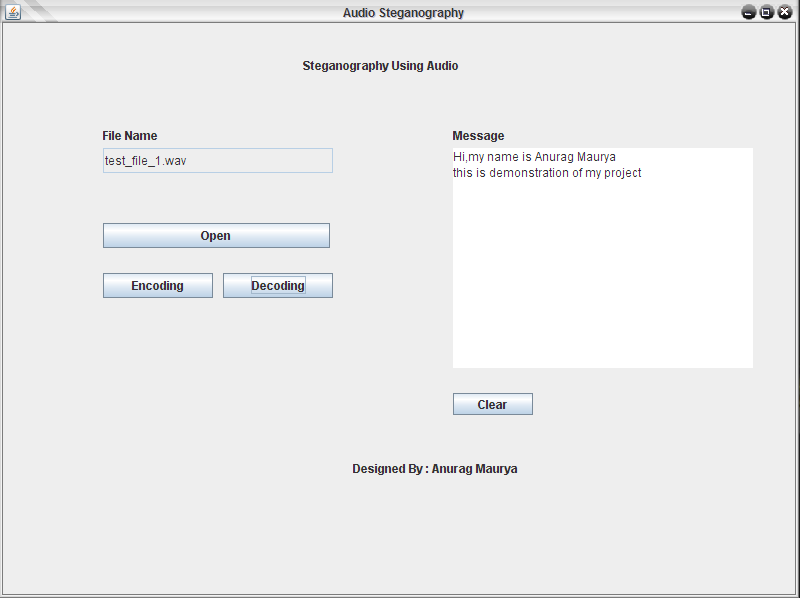
**FIGURE 3.13**: *ENTERING WRONG KEY*



**FIGURE 3.14**: *SCRIBBLED MESSAGE ON ENTERING WRONG KEY*



**FIGURE 3.15**: *ENTERING RIGHT KEY*



**FIGURE 3.16**: *SECRET MESSAGE REVEALED CORRECTLY!!*

***References***

* [*www.wikipedia.org*](http://www.wikipedia.org)
* The Handbook of Applied Cryptography by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone
* Applied Cryptography by Bruce Schneier
* A project report on Audio Steganography by

Brian Jacokes, Yuliya Kodysh, Andrew Lisy

***Appendix***

**Source code:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.net.\*;

import java.io.\*;

import java.util.\*;

import sun.audio.\*;

import java.applet.\*;

import java.math.\*;

import java.awt.image.\*;

class mainframe extends JFrame implements ActionListener,Runnable

{

JLabel Lfilename,Lmessage,Ldesign,Ltitle;

JButton Bplay,Bopen,Bsave,Bencrypt,Bdecrypt,Bclear;

JTextArea Amessage;

JTextField Tfilename;

String Ekey,Dkey,address,name;

JFileChooser filechooser;

File Ofilename,Sfilename,tempfilename;

int Copened,Cencrypt,Cdecrypt,Csave;

InputStream ins;

AudioStream as;

Thread t;

public mainframe()throws Exception

{

// frame

super("Audio Steganography");

Container con=getContentPane();

con.setLayout(null);

// Basic

Copened=0;

Cencrypt=0;

Cdecrypt=0;

Csave=0;

t=new Thread(this);

t.start();

// file chooser

filechooser=new JFileChooser();

filechooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

// comp

Ltitle=new JLabel("Steganography Using Audio");

Ldesign=new JLabel("Designed By : Anurag Maurya");

Lfilename=new JLabel("File Name ");

Lmessage=new JLabel("Message ");

Bopen=new JButton("Open");

Bsave=new JButton("Save");

Bclear=new JButton("Clear");

Bencrypt=new JButton("Encoding");

Bdecrypt=new JButton("Decoding");

Amessage=new JTextArea();

Tfilename=new JTextField();

// tool tips

Tfilename.setToolTipText("Opened filename");

Bopen.setToolTipText("open");

Bsave.setToolTipText("save");

Tfilename.setEditable(false);

// Bounds

Ltitle.setBounds(300,30,250,25);

Lfilename.setBounds(100,100,100,25);

Tfilename.setBounds(100,125,230,25);

Lmessage.setBounds(450,100,100,25);

Amessage.setBounds(450,125,300,220);

Bclear.setBounds(450,370,80,22);

Bopen.setBounds(100,200,110,25);

Bsave.setBounds(220,200,110,25);

Bencrypt.setBounds(100,250,110,25);

Bdecrypt.setBounds(220,250,110,25);

Ldesign.setBounds(350,420,400,50);

// add

con.add(Ltitle);

con.add(Ldesign);

con.add(Lfilename);

con.add(Tfilename);

con.add(Lmessage);

con.add(Amessage);

con.add(Bclear);

con.add(Bopen);

con.add(Bsave);

con.add(Bencrypt);

con.add(Bdecrypt);

// actionListener

Bclear.addActionListener(this);

Bopen.addActionListener(this);

Bsave.addActionListener(this);

Bencrypt.addActionListener(this);

Bdecrypt.addActionListener(this);

} // constr of mainframe

public void run()

{

try

{

recv r=new recv();

}

catch(Exception e)

{

System.out.println(e);

}

}

public void Audioencrypt(String message,File file,int key) throws Exception

{

byte b[]=new byte[1];

BigInteger Abi,Mbi;

int k,k1;

InputStream ins=new FileInputStream(file);

OutputStream outs=new FileOutputStream(new File("d://encoded.wav"));

for(int c=0;c<key;c++)

{

int ch=ins.read();

outs.write(ch);

}

int len=message.length();

byte mess[]=new byte[1];

char chmess[]=new char[len+1];

k=k1=0;

for(int i=0;i<=len;i++)

{

message.getChars(0,len,chmess,0);

if(i==0)

{

BigDecimal bd=new BigDecimal(len);

BigInteger Blen=bd.toBigInteger();

String Slen=Blen.toString(2);

char Clen[]=new char[Blen.bitLength()];

Slen.getChars(0,Blen.bitLength(),Clen,0);

for(int j=0;j<=7;j++)

{

if(j==0)

{

for(k=0;k<8-Blen.bitLength();k++)

{

int n=ins.read(b);

Abi=new BigInteger(b);

String Aby=Abi.toString(2);

int Alen=Abi.bitLength();

if(b[0]<0)

Alen++;

char Ach[]=new char[Alen+1];

Aby.getChars(0,Alen,Ach,0);

if(b[0]==0)

{

}

else

{

if(Ach[Alen-1]=='1')

{

if(Alen==Abi.bitLength())

{

BigInteger bi=new BigInteger("11111110",2);

BigInteger big=Abi.and(bi);

b=big.toByteArray();

}

else

{

BigInteger bi=new BigInteger("-1",2);

BigInteger big=Abi.subtract(bi);

b=big.toByteArray();

}

}

outs.write(b);

}

} //for loop k

j=j+k-1;

} // if of j

else

{

int n=ins.read(b);

Abi=new BigInteger(b);

String Aby=Abi.toString(2);

int Alen=Abi.bitLength();

if(b[0]<0)

Alen++;

char Ach[]=new char[Alen+1];

Aby.getChars(0,Alen,Ach,0);

if(b[0]==0)

{

Alen=1;

}

if(Clen[j-k]=='0' && Ach[Alen-1]=='1')

{

if(Alen==Abi.bitLength())

{

BigInteger bi=new BigInteger("11111110",2);

BigInteger big=Abi.and(bi);

b=big.toByteArray();

}

else

{

BigInteger bi=new BigInteger("-1",2);

BigInteger big=Abi.subtract(bi);

b=big.toByteArray();

}

}

else if(Clen[j-k]=='1' && Ach[Alen-1]=='0')

{

if(Alen==Abi.bitLength())

{

BigInteger bi=new BigInteger("1",2);

BigInteger big=Abi.add(bi);

b=big.toByteArray();

}

else

{

BigInteger bi=new BigInteger("-1",2);

BigInteger big=Abi.add(bi);

b=big.toByteArray();

}

}

outs.write(b);

} // end else

} // for loop j

} // end of if

else

{

String slen=String.valueOf(chmess[i-1]);

byte blen[]=slen.getBytes();

BigInteger Blen=new BigInteger(blen);

String Slen=Blen.toString(2);

char Clen[]=new char[Blen.bitLength()];

Slen.getChars(0,Blen.bitLength(),Clen,0);

for(int j=0;j<=7;j++)

{

if(j==0)

{

for(k1=0;k1<8-Blen.bitLength();k1++)

{

int n=ins.read(b);

Abi=new BigInteger(b);

String Aby=Abi.toString(2);

int Alen=Abi.bitLength();

if(b[0]<0)

Alen++;

char Ach[]=new char[Alen+1];

Aby.getChars(0,Alen,Ach,0);

if(b[0]==0)

{

}

else

{

if(Ach[Alen-1]=='1')

{

if(Alen==Abi.bitLength())

{

BigInteger bi=new BigInteger("11111110",2);

BigInteger big=Abi.and(bi);

b=big.toByteArray();

}

else

{

BigInteger bi=new BigInteger("-1",2);

BigInteger big=Abi.subtract(bi);

b=big.toByteArray();

}

}

}

outs.write(b);

} //for loop k

j=j+k1-1;

} // if of j

else

{

int n=ins.read(b);

Abi=new BigInteger(b);

String Aby=Abi.toString(2);

int Alen=Abi.bitLength();

if(b[0]<0)

Alen++;

char Ach[]=new char[Alen+1];

Aby.getChars(0,Alen,Ach,0);

if(b[0]==0)

{

Alen=1;

}

if(Clen[j-k1]=='0' && Ach[Alen-1]=='1')

{

if(Alen==Abi.bitLength())

{

BigInteger bi=new BigInteger("11111110",2);

BigInteger big=Abi.and(bi);

b=big.toByteArray();

}

else

{

BigInteger bi=new BigInteger("-1",2);

BigInteger big=Abi.subtract(bi);

b=big.toByteArray();

}

}

else if(Clen[j-k1]=='1' && Ach[Alen-1]=='0')

{

if(Alen==Abi.bitLength())

{

BigInteger bi=new BigInteger("1",2);

BigInteger big=Abi.add(bi);

b=big.toByteArray();

}

else

{

BigInteger bi=new BigInteger("-1",2);

BigInteger big=Abi.add(bi);

b=big.toByteArray();

}

}

outs.write(b);

} // end else

} // for loop j

} // end of else

} // for loop i

while(true)

{

int i=ins.read();

if(i==-1) break;

outs.write(i);

}

ins.close();

outs.close();

}

public void Audiodecrypt(File filename,int key)throws Exception

{

InputStream ins=new FileInputStream(filename);

byte b[]=new byte[1];

BigInteger bb1;

char mess[]=new char[8];

int c=0;

for(int i=0;i<key;i++)

{

int n=ins.read();

}

for(int i=0;i<8;i++)

{

ins.read(b);

bb1=new BigInteger(b);

String str=bb1.toString(2);

int len=bb1.bitLength();

if(b[0]<0)

len++;

char ch[]=new char[len+1];

str.getChars(0,len,ch,0);

if(b[0]==0)

mess[i]='0';

else

mess[i]=ch[len-1];

}

String dd=new String(mess);

BigInteger bb=new BigInteger(dd,2);

String s=bb.toString(2);

int l=bb.intValue();

char me[]=new char[l];

int count=0;

for(int m=0;m<l;m++)

{

for(int i=0;i<8;i++)

{

ins.read(b);

bb1=new BigInteger(b);

String str=bb1.toString(2);

int len=bb1.bitLength();

if(b[0]<0)

len++;

char ch[]=new char[len+1];

str.getChars(0,len,ch,0);

if(b[0]==0)

mess[i]='0';

else

mess[i]=ch[len-1];

}

String dd1=new String(mess);

BigInteger bb2=new BigInteger(dd1,2);

String s1=bb2.toString(2);

int l1=bb2.intValue();

me[count]=(char)l1;

count++;

}

String message=new String(me);

Amessage.setText(message);

ins.close();

}

public void actionPerformed(ActionEvent ae)

{

try

{

// Action for encryption button

if(ae.getSource()==Bencrypt)

{

if(Copened==1)

{

Ekey=JOptionPane.showInputDialog("Enter The Key For Encryption");

//String type

if(Ekey.trim().equals(""))

JOptionPane.showMessageDialog(this,"Enter the Key","Error",JOptionPane.ERROR\_MESSAGE);

else

{

// encrypt the message

int key=Integer.parseInt(Ekey);

Audioencrypt(Amessage.getText(),Ofilename,key);

Cencrypt=1;

}

}

else

{

JOptionPane.showMessageDialog(this,"File Not Opened","Error",JOptionPane.ERROR\_MESSAGE);

}

} // end of Bencrypt

// Action for Clear button

else if(ae.getSource()==Bclear)

{

Amessage.setText("");

} // end of clear button

// Action for Decrypt button

else if(ae.getSource()==Bdecrypt)

{

if(Copened==1)

{

Dkey=JOptionPane.showInputDialog("Enter The Key For Decryption");

//String type

if(Dkey.trim().equals(""))

JOptionPane.showMessageDialog(this,"Enter the Key","Error",JOptionPane.ERROR\_MESSAGE);

else

{

// decrypt message

int key=Integer.parseInt(Dkey);

Audiodecrypt(Ofilename,key);

Cdecrypt=1;

}

}

else

JOptionPane.showMessageDialog(this,"File Not Opened","Error",JOptionPane.ERROR\_MESSAGE);

} // end of Decrypt button

// Action for Play button

else if(ae.getSource()==Bplay)

{

if(Copened==1)

{

ins=new FileInputStream(Ofilename);

as=new AudioStream(ins);

AudioPlayer.player.start(as);

} // start playing

else

JOptionPane.showMessageDialog(this,"File Not Opened","Error",JOptionPane.ERROR\_MESSAGE);

} // end of play button

// Action for open Button

else if(ae.getSource()==Bopen)

{

int r=filechooser.showOpenDialog(this);

tempfilename=filechooser.getSelectedFile(); //File type

if(r==JFileChooser.CANCEL\_OPTION)

JOptionPane.showMessageDialog(this,"File Not Selected","Error",JOptionPane.ERROR\_MESSAGE);

else

{

name=tempfilename.getName();

if(!(name.endsWith(".wav")))

JOptionPane.showMessageDialog(this,"Select Only Wav","Error",JOptionPane.ERROR\_MESSAGE);

else

{

Copened=1;

Ofilename=tempfilename;

Tfilename.setEditable(true);

Tfilename.setText(name);

Tfilename.setEditable(false);

}

}

} // end of Open button

// Action for Save Button

else if(ae.getSource()==Bsave)

{

if(Copened==1 && Cencrypt==1 || Cdecrypt==1)

{

int r=filechooser.showSaveDialog(this);

Sfilename=filechooser.getSelectedFile(); //File type

InputStream in=new FileInputStream("d:\\encoded.wav");

OutputStream out=new FileOutputStream(Sfilename);

Ofilename=Sfilename;

name=Sfilename.getName();

Tfilename.setEditable(true);

Tfilename.setText(name);

Tfilename.setEditable(false);

while(true)

{

int i=in.read();

if(i==-1) break;

out.write(i);

}

in.close();

out.close();

}

else

{

String s;

if(Copened==0)

s="File not Opened";

else if(Cencrypt==0)

s="Not Encrypted";

else

s="Not Decrypted";

JOptionPane.showMessageDialog(this,s,"Error",JOptionPane.ERROR\_MESSAGE);

}

} // end of save button

} // end try

catch(Exception e)

{

//

JOptionPane.showMessageDialog(this,e,"Error",JOptionPane.ERROR\_MESSAGE);

}

} // end of actionperformed

}//end of class

class recv extends JFrame implements Runnable

{

JFileChooser fc;

ServerSocket ss;

Socket s;

InputStream ins;

OutputStream out;

byte b[];

int len;

public recv() throws Exception

{

b=new byte[100];

fc=new JFileChooser();

fc.setFileSelectionMode(JFileChooser.FILES\_ONLY);

ss=new ServerSocket(6000);

torun();

}

public void torun() throws Exception

{

while(true)

{

s=ss.accept();

ins=s.getInputStream();

String str="Your have Receive An AudioFile.Save them";

JOptionPane.showMessageDialog(this,str,"Information",JOptionPane.INFORMATION\_MESSAGE);

int r=fc.showSaveDialog(this);

File file=fc.getSelectedFile();

out=new FileOutputStream(file);

Thread t=new Thread(this);

t.start();

}

}

public void run()

{

try

{

while(true)

{

int n=ins.read();

if(n==-1) break;

out.write(n);

}

// s.close();

ins.close();

out.close();

}

catch(Exception e)

{

System.out.println(e);

}

} // end of run

} //end of class

public class steghide

{

public static void main(String arg[])throws Exception

{

mainframe frame=new mainframe();

frame.setSize(800,600);

frame.setVisible(true);

frame.addWindowListener( new WindowAdapter()

{

public void windowClosing(WindowEvent we)

{

System.exit(0);

}

});

} // end of main

} // end of class