PROJECT REPORT ON

IMPROVED ENCRYPTED IMAGE STEGANOGRAPHY ON CLIENT-SERVER

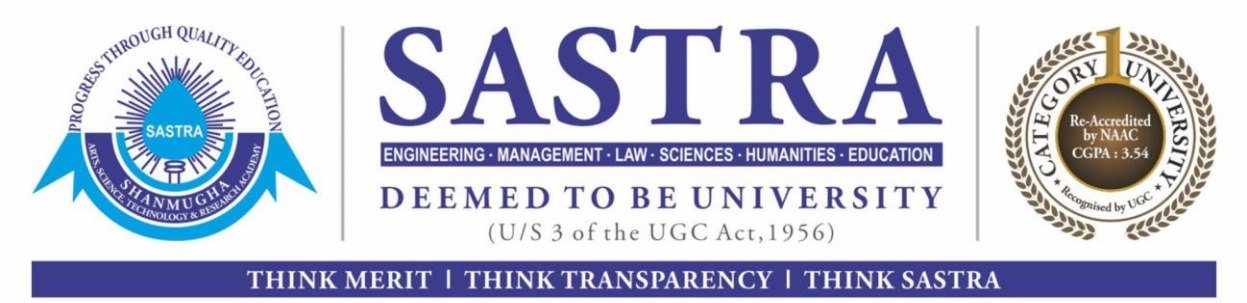
Report submitted to the SASTRA Deemed to be University as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by Peshwar Rajesh
Reg. No.: 124157042

Programme: B.Tech Computer Science and Engineering(Cybersecurity and blockchain Technology)

DECEMBER 2022



DEPARTMENT OF SOC/Main Campus
THANJAVUR, TAMILNADU, INDIA- 613401

DEPARTMENT OF SOC/Main Campus
THANJAVUR, TAMILNADU, INDIA- 613401

Bonafide Certificate

This is to certify that the report titled "IMPROVED ENCRYPTED IMAGE STEGANOGRAPHY ON CLIENT-SERVER" submitted as a requirement for the course, CSE302: COMPUTER NETWORKS for B.Tech. is a bonafide record of the work done by Mr. PESHWAR RAJESH (Reg. No.124157042,CSE CS&BT G) during the academic year 2022, in the department of CSE CS&BT.

Project Based Work Viva Voce held on _____

Examiner 1 Examiner 2

Table of Contents

List Of Figures

Abbreviations

OSI Open Systems Interconnections

LAN Local Area Networks

WAN Wide Area Networks

MSB Most Significant Bits

LSB Least Significant Bits

TCP Transmission Control Protocols

IP Internet Protocols

OTP One Time Password

ABSTRACT

The modern computing world revolves around the word “DATA”, but just what is so intriguing about it? In today’s world, data is the power and business start realizing it because data can predict customer trends potentially , get increased sales, and help the organization to achieve newer heights. The technology has become so advanced and our topmost priority is to secure data. Nowadays data sharing is increasing rapidly as there are thousands of messages and large number of data transmission taking place on the internet every day from one place to another. The secured data transmission is the primary concern of the sender, which is to protect data and it is really important to transmit our message in a secret way that only the receiver can able to understand.Steganography literally means hiding data in plainsight.

In this project, we will understand what is image steganography and how can we implement it using a chat application in java.

KEYWORDS: Steganography, Data security, java

INTRODUCTION- Steganography

The process of hiding a secret message within a larger one so that the contents or presence of the hidden message could not be known to anyone and this process is known as steganography. Steganography serves the main purpose which is to provide secret communication between two groups. Cryptography which can conceals only the contents of a secret message but steganography can able to conceal the fact that a message is communicated. Though there are some differences between steganography and cryptography, there are many analogies between them and some authors categorize steganography as a type of cryptography because hidden communication is a type of secret message.

We can perform steganography on different transmission media like images, video, text, or audio.

TYPES OF STEGANOGRAPHY

Based on the way of transmission steganography can be classified into Text steganography, Image steganography, Video steganography, Network steganography, E-mail steganography, Audio steganography

1. Types of steganography

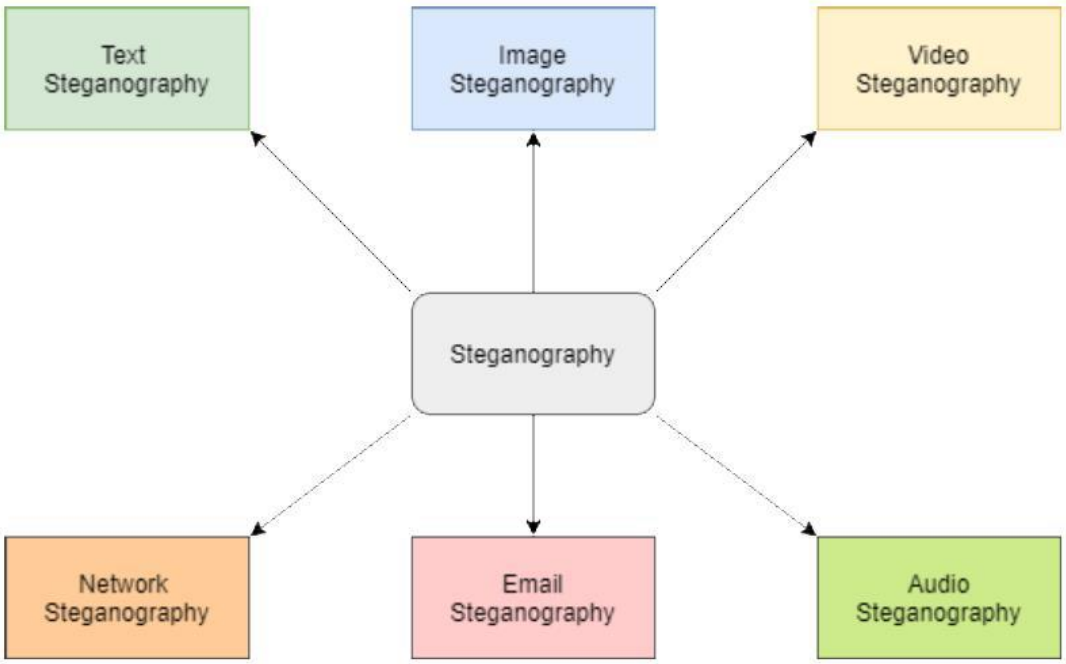


Figure1.Types of Steganography

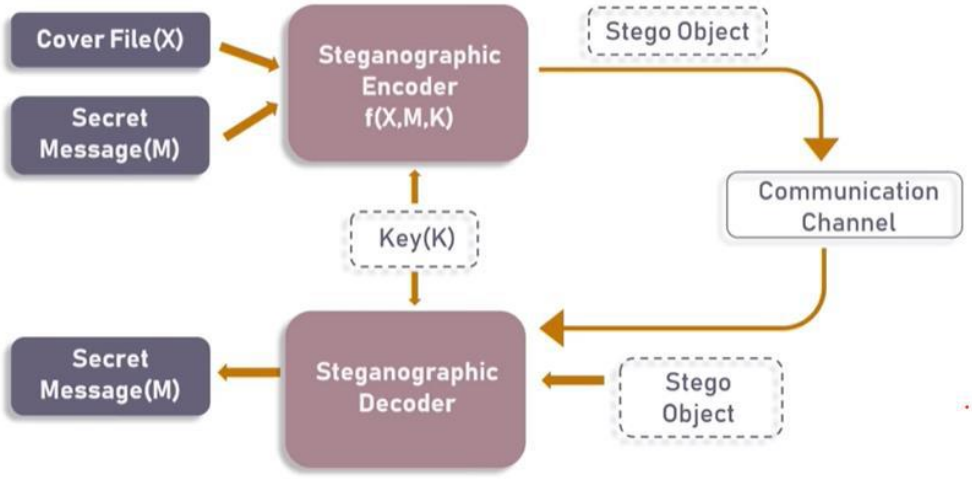
STEPS TO BE FOLLOWED TO PERFORM IMAGE

STEGANOGRAPHY Obtain the secret information which is to be shared between sender and receiver Select an image in which you are going to encode the secret message Encode the secret message to the selected image The sender shares the message to receiver In receiver end, while decoding the secret information is extracted from the image So, in others point of view it is just an image but to the sender and receiver the secret message is successfully shared. My project is about network security (secured data transmission) by image steganography using Python.

BASIC STEGANOGRAPHIC MODEL

In this basic model of steganography, a cover file(image) and secret message is given to steganographic encoder and the stego object is communicated from sender to receiver and steganographic decoder decodes and only the secret message is obtained.

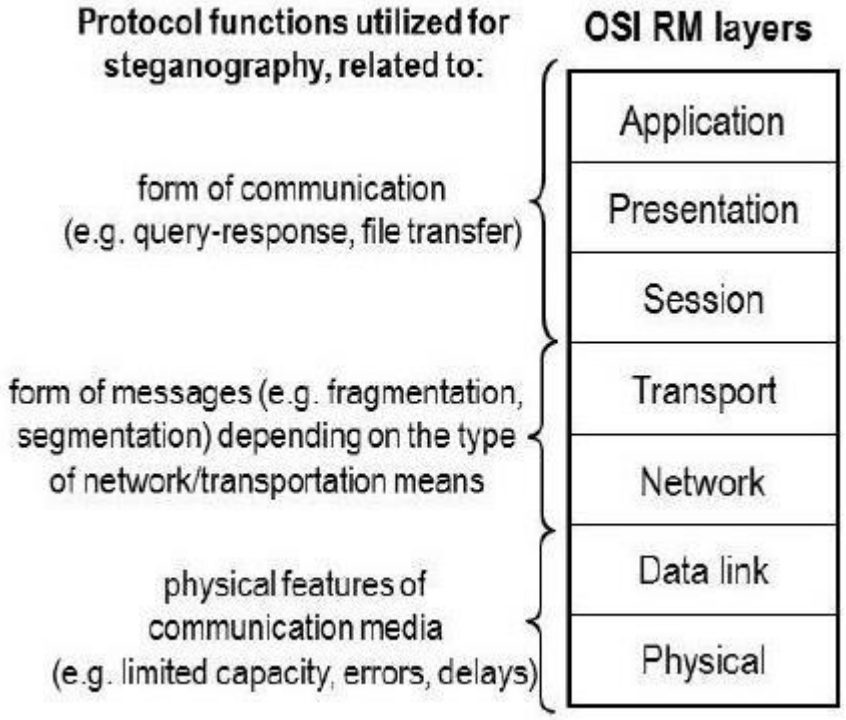
Figure2. Stego model



PROTOCOL FUNCTION UTILIZED BY STEGANOGRAPHY

The various OSI RM layers that are related to steganography since it is mainly a data transmission between two parties

Figure 3. Protocol function utilized for steganography

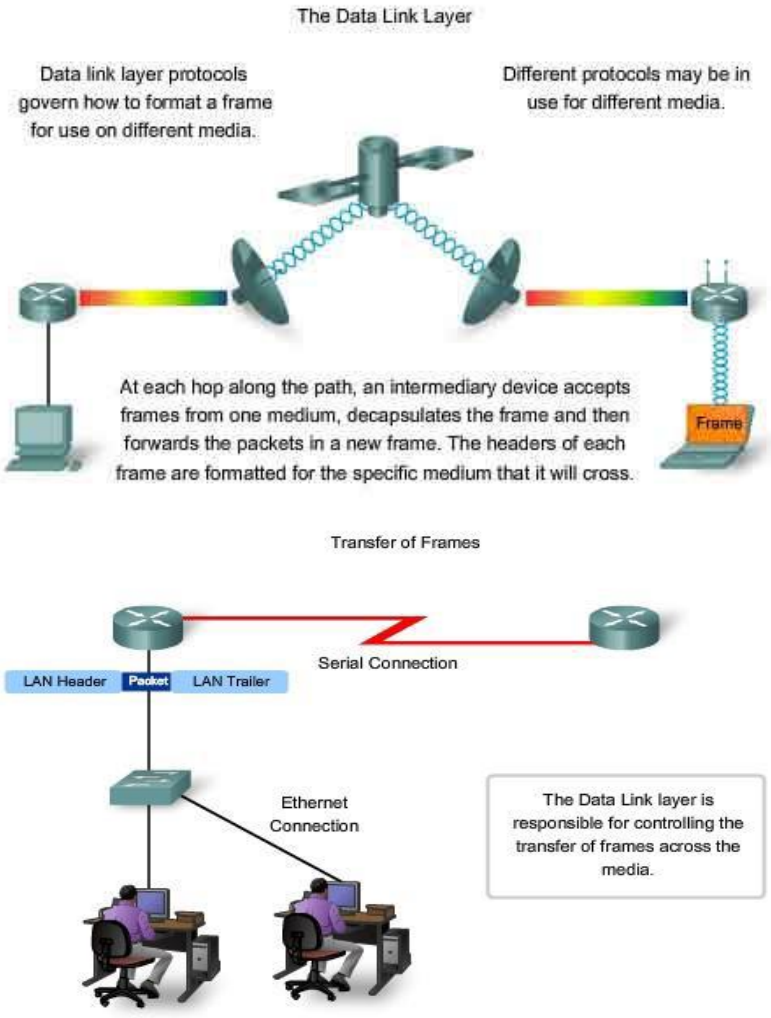


OSI RM LAYERS THAT TAKE PART IN IMAGE STEGANOGRAPHY

Datalink layer

In the Open Systems Interconnection (OSI) architecture model for a set of telecommunication protocols, layer 2 is the datalink layer. In this layer, data bits were encoded, decoded and organized before they are transported. The modified data bits are transported as frames between two adjacent nodes on the same WAN or LAN.

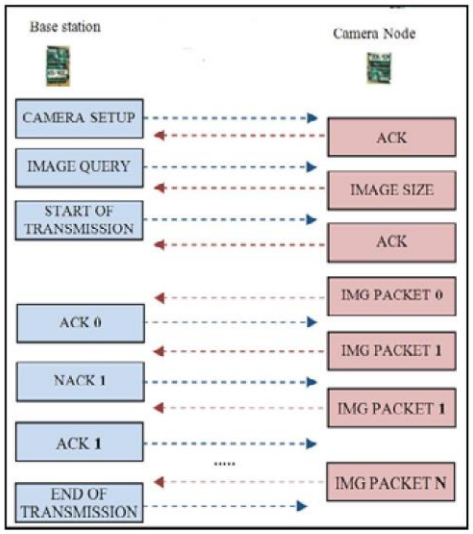
Figure 4. Datalink layer and transfer of frames



Application layer

The layer which specifies the shared communications protocols and the host's interface methods used in a communications network is known as application layer. It is otherwise known as abstraction layer. Both the standard models of computer networking, the Internet Protocol Suite (TCP/IP) and the OSI model uses the application layer abstraction.

Figure 5. Application layer



TOOL USED:

JAVA(SERVER AND CLIENT WITH GUI AND OTP)FINAL CODE:

- JAVA is a very versatile programming language.We can use so many libraries,swing and so on for the GUI,OTP,Encoding,Decoding and so on!

BASIC CONCEPT OF LEAST SIGNIFICANT BIT TECHNIQUE

• We use the least significant bit technique to implement our steganography in the image. The data is converted to bits and modification in LSB doesn't affect the data much while MSB does. So, while encoding our secret message changes are made in LSB.

Figure 6.LSB technique

STEPS TO BE FOLLOWED IN RUNNING THE CODE

The JAVA code: -

- Run Server java file
- Immediately run Client java file
- GUI Opens up and connection is established
- Streams will be set, and OTP will have to be used
- Then follow-on screen functions!

MERITS OF NETWORK SECURITY BY IMAGE STEGANOGRAPHY

• It provides secured data transmission.

• Up to now, cryptography plays a vital role in protecting the secret communication between the sender and the intended receiver. However, nowadays people use steganography techniques besides cryptography because it adds more protective layers to the hidden data.

• In comparison to cryptography alone, steganography uses a method that does not draw attention to the intended message.

• It is difficult to detect only the intended receiver can be able to detect as it is sneaky enough to slip through.

• If we use large number of software's, it can be done faster.

DEMERITS OF NETWORK SECURITY BY IMAGE STEGANOGRAPHY

• Steganography is popular among cyber criminals. Recent attacks shows that the attackers used Steganography to embed the malicious code inside the file, which is found by security researchers that attackers use new malware campaign which uses image to hide the data

• Image may become distorted. While compressing image such as JPEG, secret information may be lost. When steganography is implemented properly, it can be difficult to detect, but not impossible.

• Large number of data and large file size which is considered to be a disadvantage.

IMPROVED CODE AFTER SUGGESTIONS USING JAVA

This code is built using Java that enables a Client to connect to a Server for Chatting using Image Steganography.Server generates an OTP on connecting with the Client for authorization.Normal chat will take place but embedding the message within images using Encode.To read any message the Client will select a message and opens the Decoder and will be able to decode the image using the provided OTP

Serv.java

```
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.InvocationTargetException;
import java.net.ServerSocket;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.Box;
import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JCheckBox;
import javax.swing.JOptionPane;


public class Serv extends JFrame{
    private JLabel statuslabel,chatlabel,instruction,instruction1;
    private JTextArea statusarea;
    private JPanel chatbox;
    private JScrollPane statusscroll,scroll;
    private JButton Embed,Send,Decode;
    private ServerSocket server;
    private Socket connection;
```

```
private ObjectInputStream input;
private ObjectOutputStream output;
private JCheckBox []checks;
private BufferedImage[] images;
private ButtonGroup grp;
public BufferedImage Bufim;
public Serv sser;
public JLabel setlabel;
private int JC;
private String security=null;

public Serv(){
   super("Server");
   Bufim=null;
   sser=this;
   JC=-1;
   this.setVisible(true);
   this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   this.setBounds(10,20,700,700);
   this.setResizable(false);
   this.setLayout(null);
   this.getContentPane().setBackground(Color.BLUE);

   this.addWindowListener(new WindowAdapter(){

      @Override
      public void windowClosing(WindowEvent e) {
         closeConnection();
      }
   });

   initcomponents();
}
private void initcomponents(){
         statuslabel=new JLabel("STATUS :");
      statuslabel.setForeground(Color.WHITE);
      statuslabel.setBounds(50,5,70,50);
      add(statuslabel);

      statusarea=new JTextArea();
      statusscroll=new JScrollPane(statusarea);
      statusscroll.setBounds(115,5,300,100);
      add(statusscroll);
```

```
   instruction=new JLabel("Please 'EMBED' to prepare message !");
instruction.setForeground(Color.WHITE);
instruction.setBounds(420,5,250,50);
add(instruction);

   instruction1=new JLabel("Please select a message to 'READ' !");
instruction1.setForeground(Color.WHITE);
instruction1.setBounds(420,60,270,50);
add(instruction1);

chatlabel=new JLabel("CHAT WINDOW :");
chatlabel.setForeground(Color.WHITE);
chatlabel.setBounds(50,110,100,50);
add(chatlabel);

checks=new JCheckBox[100];
grp=new ButtonGroup();
images=new BufferedImage[100];

Embed=new JButton("EMBED");
Embed.setBounds(50,610,100,40);
Embed.addActionListener(new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
      new EmbedMessage(sser,security);
   }
});
add(Embed);

   setlabel=new JLabel("No image is set!");
setlabel.setForeground(Color.WHITE);
setlabel.setBounds(150,610,150,40);
add(setlabel);

Decode=new JButton("READ");
Decode.setBounds(550,610,100,40);
Decode.setEnabled(false);
Decode.addActionListener(new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
      BufferedImage BB=null;
      for(int i=0;i<=JC;i++){
         if(checks[i].isSelected()){
            BB=images[i];
```

```
                    break;
                }
            }
            new DecodeMessage(BB);
        }
    });
    add(Decode);

    chatbox=new JPanel();
    BoxLayout layout=new BoxLayout(chatbox,BoxLayout.Y_AXIS);
    chatbox.setBackground(Color.CYAN);
    chatbox.setBorder(BorderFactory.createLineBorder(Color.CYAN));
    chatbox.setLayout(layout);
    scroll = new JScrollPane(chatbox);
    scroll.setBounds(50,150,600,450);
    add(scroll);

    Send=new JButton("SEND");
    Send.setBounds(300,610,100,40);
    Send.setEnabled(false);
    Send.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            SwingUtilities.invokeLater(new Runnable(){
                @Override
                public void run() {
                    BufferedImage IM=Bufim;
                    if(IM==null){
                        return;
                    }
                    Decode.setEnabled(true);
                    JPanel Panel=new JPanel();
                    Panel.setLayout(new FlowLayout(FlowLayout.RIGHT));
                    Panel.setBackground(Color.CYAN);

                    JCheckBox chk=new JCheckBox("SERVER");
                    chk.setSize(100, 50);
                    checks[++JC]=chk;
                    grp.add(checks[JC]);

                    JLabel lb=new JLabel();
                    lb.setSize(150,150);
                    lb.setIcon(new ImageIcon(IM.getScaledInstance(lb.getWidth(), lb.getHeight(),
Image.SCALE_SMOOTH)));
```

```
                    images[JC]=IM;

                    Panel.add(checks[JC]);
                    Panel.add(lb);
                    chatbox.add(Panel);
                    chatbox.add(Box.createVerticalStrut(20));
                    sser.validate();
                    sser.repaint();
                    sendMessage(IM);

                    }
                });
            }
        });
        add(Send);

        sser.validate();
        sser.repaint();

    }
    public static void main(String[] args) throws InterruptedException, InvocationTargetException
{
        Serv ser=new Serv();
        ser.startRunning();

    }

    public void startRunning(){//////////////////////////////
        try{
            server = new ServerSocket(7777,100);
                while(true){
                    try{
                        waitForConnection();
                        setupStreams();
                        whileChatting();
                            }catch(Exception eofException){
                                showMessage("\nServer ended the connection!");
                            } finally{
                                closeConnection();
                            }
                        }
                } catch (IOException ioException){
                        ioException.printStackTrace();
```

```
                }
        }
        private void waitForConnection() throws IOException{////////////////////////////////////
                showMessage("\nWaiting for someone to connect...");
                connection = server.accept();
                showMessage("\nNow connected to " +
connection.getInetAddress().getHostName());
        }
        private void setupStreams() throws IOException{////////////////////////////////////////////////
                output = new ObjectOutputStream(connection.getOutputStream());
                output.flush();

                input = new ObjectInputStream(connection.getInputStream());

                showMessage("\nStreams are now setup");
        }
        private void whileChatting() throws IOException{////////////////////////////////////
        chatbox.removeAll();
        this.repaint();
        this.validate();
           /*String s=JOptionPane.showInputDialog(this,"Hey Server please set the OTP for
Client!");
           if(s.equals("")){
              s="12345";
           }*/
           double dd;
           while((dd=Math.random())<0.1){}
           Integer xx=(int)(dd*100000);
           String s=xx.toString();
           security=s;
                String message = " You are now connected!\n Your OTP : "+s;
                sendMessage(message);
        try {
           String ss=(String)input.readObject();
           if(!ss.equals(security)){
              sendMessage("Incorrect OTP !");
              return;
           }
           else sendMessage("Verification Successfull !");
        } catch (ClassNotFoundException ex) {
           Logger.getLogger(Clin.class.getName()).log(Level.SEVERE, null, ex);
        }
                ableToSend(true);
           Object mess=null;
```

```
        do{
                try{
                        mess =input.readObject();
                if(mess instanceof String){
                    if(!((String)mess).equals("CLOSE"))
                        showMessage("\n" + mess);
                }
                else {
                    ByteArrayInputStream bin=new ByteArrayInputStream((byte[]) mess);
                    BufferedImage ms=ImageIO.read(bin);
                    displayMessage(ms);
                    bin.close();
                }
                  }catch(Exception Ex){
                        showMessage("\nThe user has sent an unknown object!");
                  }
        }while(mess instanceof byte[]||(mess!=null&&!((String)mess).equals("CLOSE")));
}

public void closeConnection(){////////////////////////////////////////////////////////
    sendMessage("CLOSE");
        showMessage("\nClosing Connections...");
        ableToSend(false);
        try{
                output.close();
                input.close();
                connection.close();
        }catch(IOException ioException){
                showMessage("\nError in closing the streams!");
        }
}

private void sendMessage(Object message){//////////////////////////////////////////
    ByteArrayOutputStream bos;
    if(message==null){
        return;
    }
        try{
        if(message instanceof String){
            output.writeObject(message);
            output.flush();
        }else{
            BufferedImage bm=(BufferedImage)message;
            bos=new ByteArrayOutputStream();
```

```
                ImageIO.write(bm, "png", bos );
                byte[] imageInByte = bos.toByteArray();
                bos.flush();
                        output.writeObject(imageInByte);
                output.flush();
                bos.close();
            }
            setlabel.setText("No image is set!");
            Bufim=null;
              }catch(IOException ioException){
                        statusarea.append("\n Error : Cannot send message!, PLEASE RETRY");
              }
        }

    private void displayMessage(BufferedImage mes){
            SwingUtilities.invokeLater(new Runnable(){
                @Override
                public void run() {
                  if(mes==null){
                     return;
                  }
                  Decode.setEnabled(true);
                  BufferedImage IM=mes;
                  JPanel Panel=new JPanel();
                  Panel.setLayout(new FlowLayout(FlowLayout.LEFT));
                  Panel.setBackground(Color.CYAN);

                  JCheckBox chk=new JCheckBox("CLIENT");
                  chk.setSize(100, 50);
                  checks[++JC]=chk;
                  grp.add(checks[JC]);

                  JLabel lb=new JLabel();
                  lb.setSize(150,150);
                  lb.setIcon(new ImageIcon(IM.getScaledInstance(lb.getWidth(), lb.getHeight(),
Image.SCALE_SMOOTH)));

                  images[JC]=IM;

                  Panel.add(lb);
                  Panel.add(checks[JC]);

                  chatbox.add(Panel);
                  chatbox.add(Box.createVerticalStrut(20));
```

```
                sser.repaint();
                sser.validate();
            }
        });
    }

    private void showMessage(final String text){//////////////////////////////////////
        SwingUtilities.invokeLater(
            new Runnable(){
            @Override
                public void run(){
                    statusarea.append(text);
                }
            }
        );
    }

    private void ableToSend(final boolean tof){//////////////////////////////////////
        SwingUtilities.invokeLater(
            new Runnable(){
            @Override
                public void run(){
                 Send.setEnabled(tof);
                }
            }
        );
    }
}
```

Clin.java

```
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
```

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.InvocationTargetException;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.Box;
import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JCheckBox;
import javax.swing.JOptionPane;

public class Clin extends JFrame{
    private JLabel statuslabel,chatlabel,instruction,instruction1;
    private JTextArea statusarea;
    private JPanel chatbox;
    private JScrollPane statusscroll;
    private JButton Embed,Send,Decode;
    private Socket connection;
    private ObjectInputStream input;
    private ObjectOutputStream output;
    private JCheckBox []checks;
    private BufferedImage[] images;
    private ButtonGroup grp;
    private String ServerIP;
    public BufferedImage Bufim;
    public Clin cli;
    public JLabel setlabel;
    private int JC;
    private String security=null;
```

```
public Clin(String host){
    super("Client");
    Bufim=null;
    cli=this;
    JC=-1;
    ServerIP=host;
    this.setVisible(true);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setBounds(720,20,700,700);
    this.setResizable(false);
    this.setLayout(null);
    this.getContentPane().setBackground(Color.BLUE);

    this.addWindowListener(new WindowAdapter(){

        @Override
        public void windowClosing(WindowEvent e) {
            closeConnection();
        }
    });

    initcomponents();
}
private void initcomponents(){
            statuslabel=new JLabel("STATUS :");
        statuslabel.setForeground(Color.WHITE);
        statuslabel.setBounds(50,5,70,50);
        add(statuslabel);

        statusarea=new JTextArea();
        statusscroll=new JScrollPane(statusarea);
        statusscroll.setBounds(115,5,300,100);
        add(statusscroll);

            instruction=new JLabel("Please 'EMBED' to prepare message !");
        instruction.setForeground(Color.WHITE);
        instruction.setBounds(420,5,250,50);
        add(instruction);

            instruction1=new JLabel("Please select a message to 'READ' !");
        instruction1.setForeground(Color.WHITE);
        instruction1.setBounds(420,60,270,50);
        add(instruction1);
```

```
chatlabel=new JLabel("CHAT WINDOW :");
chatlabel.setForeground(Color.WHITE);
chatlabel.setBounds(50,110,100,50);
add(chatlabel);

checks=new JCheckBox[100];
grp=new ButtonGroup();
images=new BufferedImage[100];

Embed=new JButton("EMBED");
Embed.setBounds(50,610,100,40);
Embed.addActionListener(new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
      new EmbedMessage(cli,security);
   }
});
add(Embed);

   setlabel=new JLabel("No image is set!");
setlabel.setForeground(Color.WHITE);
setlabel.setBounds(150,610,150,40);
add(setlabel);

Decode=new JButton("READ");
Decode.setBounds(550,610,100,40);
Decode.setEnabled(false);
Decode.addActionListener(new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
      BufferedImage BB=null;
      for(int i=0;i<=JC;i++){
         if(checks[i].isSelected()){
            BB=images[i];
            break;
         }
      }
      new DecodeMessage(BB);
   }
});
add(Decode);

chatbox=new JPanel();
```

```
        BoxLayout layout=new BoxLayout(chatbox,BoxLayout.Y_AXIS);
        chatbox.setBackground(Color.CYAN);
        chatbox.setBorder(BorderFactory.createLineBorder(Color.CYAN));
        chatbox.setLayout(layout);
        JScrollPane scroll = new JScrollPane(chatbox);
        scroll.setBounds(50,150,600,450);
        add(scroll);

        grp=new ButtonGroup();

        Send=new JButton("SEND");
        Send.setEnabled(false);
        Send.setBounds(300,610,100,40);
        Send.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e) {
                SwingUtilities.invokeLater(new Runnable(){
                    @Override
                    public void run() {
                        BufferedImage IM=Bufim;
                        if(IM==null){
                            return;
                        }
                        Decode.setEnabled(true);
                        JPanel Panel=new JPanel();
                        Panel.setLayout(new FlowLayout(FlowLayout.RIGHT));
                        Panel.setBackground(Color.CYAN);

                        JCheckBox chk=new JCheckBox("CLIENT");
                        chk.setSize(100, 50);
                        checks[++JC]=chk;
                        grp.add(checks[JC]);

                        JLabel lb=new JLabel();
                        lb.setSize(150,150);
                        lb.setIcon(new ImageIcon(IM.getScaledInstance(lb.getWidth(), lb.getHeight(),
Image.SCALE_SMOOTH)));

                        images[JC]=IM;

                        Panel.add(checks[JC]);
                        Panel.add(lb);
                        chatbox.add(Panel);
                        chatbox.add(Box.createVerticalStrut(20));
```

```
                    cli.validate();
                    cli.repaint();
                    sendMessage(IM);
                }
            });
        }
    });
    add(Send);

    cli.validate();
    cli.repaint();

}
public static void main(String[] args) throws InterruptedException, InvocationTargetException
{
    Clin cli=new Clin("localhost");
    cli.startRunning();

}

public void startRunning(){//////////////////////////////
        try{
                connectToServer();
                setupStreams();
                whileChatting();
        }catch(EOFException eofException){
                showMessage("\nClient terminated the connection");
        }catch(IOException ioException){
                ioException.printStackTrace();
        }finally{
                closeConnection();
        }
    }
    private void connectToServer() throws IOException{
        showMessage("\nAttempting connection...");
        connection = new Socket(InetAddress.getByName(ServerIP), 7777);
        showMessage("\nConnection Established! Connected to: " +
connection.getInetAddress().getHostName());
    }
    private void setupStreams() throws IOException{//////////////////////////////////////////////////
        output = new ObjectOutputStream(connection.getOutputStream());
        output.flush();

        input = new ObjectInputStream(connection.getInputStream());
```

```
                showMessage("\nStreams are now setup");
        }
        private void whileChatting() throws IOException{////////////////////////////////////
        ableToSend(true);
        Object mess=null;
        try {
            showMessage("\n" + (String)input.readObject());
            security=JOptionPane.showInputDialog(this,"Hey Client please provide the OTP you
just recieved!");
            if(security==null){
                security="-";
            }
            sendMessage(security);
            String rs=(String)input.readObject();
            showMessage("\n" + ((rs.equals("CLOSE"))?"":rs));
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Clin.class.getName()).log(Level.SEVERE, null, ex);
        }
                do{
                        try{
                                mess =input.readObject();
                    if(mess instanceof String){
                        if(!((String)mess).equals("CLOSE"))
                            showMessage("\n" + mess);
                    }
                    else {
                        ByteArrayInputStream bin=new ByteArrayInputStream((byte[]) mess);
                        BufferedImage ms=ImageIO.read(bin);
                        displayMessage(ms);
                        bin.close();
                    }
                       }catch(Exception Ex){
                                showMessage("\nThe server has sent an unknown object!");
                       }
                }while(mess instanceof byte[]||(mess!=null&&!((String)mess).equals("CLOSE")));
        }

        public void closeConnection(){////////////////////////////////////////////////////////
            sendMessage("CLOSE");
                showMessage("\nClosing Connections...");
                ableToSend(false);
                try{
                        output.close();
```

```
                input.close();
            connection.close();
            }catch(IOException ioEx){
                showMessage("\nError in closing the streams!");
            }
    }

    private void sendMessage(Object message){////////////////////////////////////////
    ByteArrayOutputStream bos;
    /*if(message==null){
        return;
    }*/
            try{
          if(message==null||message instanceof String){
            output.writeObject(message);
            output.flush();
          }else{
            BufferedImage bm=(BufferedImage)message;
            bos=new ByteArrayOutputStream();
            ImageIO.write(bm, "png", bos );
            byte[] imageInByte = bos.toByteArray();
            bos.flush();
                output.writeObject(imageInByte);
            output.flush();
            bos.close();
          }
          setlabel.setText("No image is set!");
          Bufim=null;
            }catch(IOException ioEx){
                statusarea.append("\n Error : Cannot send message!, PLEASE RETRY");
            }
    }

private void displayMessage(BufferedImage mes){
        SwingUtilities.invokeLater(new Runnable(){
            @Override
            public void run() {
                if(mes==null){
                    return;
                }
                Decode.setEnabled(true);
                BufferedImage IM=mes;
                JPanel Panel=new JPanel();
                Panel.setLayout(new FlowLayout(FlowLayout.LEFT));
```

```
                    Panel.setBackground(Color.CYAN);

                    JCheckBox chk=new JCheckBox("SERVER");
                    chk.setSize(100, 50);
                    checks[++JC]=chk;
                    grp.add(checks[JC]);

                    JLabel lb=new JLabel();
                    lb.setSize(150,150);
                    lb.setIcon(new ImageIcon(IM.getScaledInstance(lb.getWidth(), lb.getHeight(),
Image.SCALE_SMOOTH)));

                    images[JC]=IM;

                    Panel.add(lb);
                    Panel.add(checks[JC]);
                    chatbox.add(Panel);
                    chatbox.add(Box.createVerticalStrut(20));
                    cli.repaint();
                    cli.validate();
                }
            });
    }

    private void showMessage(final String text){//////////////////////////////////////
            SwingUtilities.invokeLater(
                    new Runnable(){
                @Override
                        public void run(){
                                statusarea.append(text);
                        }
                    }
            );
    }

    private void ableToSend(final boolean tof){//////////////////////////////////////
            SwingUtilities.invokeLater(
                    new Runnable(){
                @Override
                        public void run(){
                     Send.setEnabled(tof);
                        }
                    }
            );
```

```
        }
}
```

EmbedMessage.java

```
//EmbedMessage.java
 import java.awt.image.*;
 import javax.swing.*;
 import java.awt.*;
 import java.awt.event.*;
 import javax.imageio.*;

 public class EmbedMessage extends JFrame implements ActionListener
 {
 JButton open = new JButton("Open"), embed = new JButton("Embed"),
    set = new JButton("Done"), reset = new JButton("Reset");
 JTextArea message = new JTextArea(10,3);
 BufferedImage sourceImage = null, embeddedImage = null;
 JSplitPane sp = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
 JScrollPane originalPane = new JScrollPane(),
    embeddedPane = new JScrollPane();
 Serv SER=null;
 Clin CLI=null;
 String secur=null;

 public EmbedMessage(Object iim,String sec) {

    super("Embed stegonographic message in image");
    secur=sec;
    if(iim instanceof Serv){
        SER=(Serv)iim;
    }
    else{
        CLI=(Clin)iim;
    }
    assembleInterface();

    this.setBounds(500,100,500, 500);
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    this.setVisible(true);
    sp.setDividerLocation(0.5);
    this.validate();
    }
```

```
private void assembleInterface() {
   JPanel p = new JPanel(new FlowLayout());
   p.add(open);
   p.add(embed);
   p.add(set);
   p.add(reset);
   this.getContentPane().add(p, BorderLayout.SOUTH);
   open.addActionListener(this);
   embed.addActionListener(this);
   set.addActionListener(this);
   reset.addActionListener(this);
   open.setMnemonic('O');
   embed.setMnemonic('E');
   set.setMnemonic('S');
   reset.setMnemonic('R');

   p = new JPanel(new GridLayout(1,1));
   p.add(new JScrollPane(message));
   message.setFont(new Font("Arial",Font.BOLD,20));
   p.setBorder(BorderFactory.createTitledBorder("Message to be embedded"));
   this.getContentPane().add(p, BorderLayout.NORTH);

   sp.setLeftComponent(originalPane);
   sp.setRightComponent(embeddedPane);
   originalPane.setBorder(BorderFactory.createTitledBorder("Original Image"));
   embeddedPane.setBorder(BorderFactory.createTitledBorder("Steganographed Image"));
   this.getContentPane().add(sp, BorderLayout.CENTER);
   }

@Override
public void actionPerformed(ActionEvent ae) {
   Object o = ae.getSource();
   if(o == open)
      openImage();
   else if(o == embed)
      embedMessage();
   else if(o == set){
      setImage();
      this.dispose();
   }
   else if(o == reset)
      resetInterface();
   }
```

```
private java.io.File showFileDialog(final boolean open) {
  JFileChooser fc = new JFileChooser("Open an image");
  javax.swing.filechooser.FileFilter ff = new javax.swing.filechooser.FileFilter() {
    @Override
    public boolean accept(java.io.File f) {
      String name = f.getName().toLowerCase();
      if(open)
        return f.isDirectory() || name.endsWith(".jpg") || name.endsWith(".jpeg") ||
          name.endsWith(".png") || name.endsWith(".gif") || name.endsWith(".tiff") ||
          name.endsWith(".bmp") || name.endsWith(".dib");
      return f.isDirectory() || name.endsWith(".png") ||    name.endsWith(".bmp");
      }
    @Override
    public String getDescription() {
      if(open)
        return "Image (*.jpg, *.jpeg, *.png, *.gif, *.tiff, *.bmp, *.dib)";
      return "Image (*.png, *.bmp)";
      }
    };
  fc.setAcceptAllFileFilterUsed(false);
  fc.addChoosableFileFilter(ff);

  java.io.File f = null;
 if(open && fc.showOpenDialog(this) == fc.APPROVE_OPTION)
    f = fc.getSelectedFile();
  else if(!open && fc.showSaveDialog(this) == fc.APPROVE_OPTION)
    f = fc.getSelectedFile();
  return f;
  }

private void openImage() {
  java.io.File f = showFileDialog(true);
  try {
    sourceImage = ImageIO.read(f);
    JLabel l = new JLabel(new ImageIcon(sourceImage));
    originalPane.getViewport().add(l);
    this.validate();
    } catch(Exception ex) { ex.printStackTrace(); }
  }

private void embedMessage() {
   if(sourceImage==null){
      JOptionPane.showMessageDialog(this, "Please choose an image !");
      return;
```

```
    }
   String mess = message.getText();
   embeddedImage = sourceImage;
   if(encode(embeddedImage, mess)){
       JLabel l = new JLabel(new ImageIcon(embeddedImage));
       embeddedPane.getViewport().add(l);
       this.validate();
   }
   else{
       embeddedImage=null;
   }
   }

private void setImage() {
   if(embeddedImage == null) {
      JOptionPane.showMessageDialog(this, "No message has been embedded!",
        "Nothing to send", JOptionPane.ERROR_MESSAGE);
      return;
     }
   if(SER!=null){
       SER.Bufim=embeddedImage;
       SER.setlabel.setText("Image is ready to be sent!");
   }
   else{
       CLI.Bufim=embeddedImage;
       CLI.setlabel.setText("Image is ready to be sent!");
   }


  /*java.io.File f = showFileDialog(false);
   String name = f.getName();
   String ext = name.substring(name.lastIndexOf(".")+1).toLowerCase();
   if(!ext.equals("png") && !ext.equals("bmp") &&   !ext.equals("dib")) {
         ext = "png";
         f = new java.io.File(f.getAbsolutePath()+".png");
         }
   try {
      if(f.exists()) f.delete();
      ImageIO.write(embeddedImage, ext.toUpperCase(), f);
      } catch(Exception ex) { ex.printStackTrace(); }*/
   }

private void resetInterface() {
   message.setText("");
```

```
originalPane.getViewport().removeAll();
embeddedPane.getViewport().removeAll();
sourceImage = null;
embeddedImage = null;
sp.setDividerLocation(0.5);
this.validate();
}
        public boolean encode(BufferedImage img, String message)
        {
                return add_text(img,message);
        }
        private boolean add_text(BufferedImage image, String text)
        {
            String s=secur;
            text+=s;
            char ln=(char)s.length();
            text+=ln;
                byte img[]  = get_byte_data(image);
                byte msg[] = text.getBytes();
                byte len[]   = bit_conversion(msg.length);
                try
                {
                        encode_text(img, len,  0);
                        encode_text(img, msg, 32);
                }
                catch(Exception e)
                {
                        JOptionPane.showMessageDialog(null, "Target File cannot hold
message!", "Error",JOptionPane.ERROR_MESSAGE);
                 return false;
                }
                return true;
        }

        private byte[] get_byte_data(BufferedImage image)
        {
                WritableRaster raster   = image.getRaster();
                DataBufferByte buffer = (DataBufferByte)raster.getDataBuffer();
                return buffer.getData();
        }

        private byte[] bit_conversion(int i)
        {
                byte byte3 = (byte)((i & 0xFF000000) >>> 24);
```

```
            byte byte2 = (byte)((i & 0x00FF0000) >>> 16);
            byte byte1 = (byte)((i & 0x0000FF00) >>> 8 );
            byte byte0 = (byte)((i & 0x000000FF)       );
            return(new byte[]{byte3,byte2,byte1,byte0});
        }

        private byte[] encode_text(byte[] image, byte[] addition, int offset)
        {
            if(addition.length + offset > image.length)
            {
                throw new IllegalArgumentException("File not long enough!");
            }
            for(int i=0; i<addition.length; ++i)
            {
                int add = addition[i];
                for(int bit=7; bit>=0; --bit, ++offset)
                {
                    int b = (add >>> bit) & 1;
                    image[offset] = (byte)((image[offset] & 0xFE) | b );
                }
            }
            return image;
        }

 /*public static void main(String arg[]) {
    new EmbedMessage();
    }*/
}
```

DecodeMessage.java

```
/DecodeMessage.java
 import java.awt.image.*;
 import javax.swing.*;
 import java.awt.*;
```

```
import java.awt.event.*;
import javax.imageio.*;

public class DecodeMessage extends JFrame implements ActionListener
{
JButton open = new JButton("Open"), decode = new JButton("Decode"),
   reset = new JButton("Reset");
JTextArea message = new JTextArea(10,3);
BufferedImage image = null;
JScrollPane imagePane = new JScrollPane();

public DecodeMessage(BufferedImage BM) {
   super("Decode stegonographic message in image");
   image=BM;
   assembleInterface();

this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
   this.setBounds(500,100,500,500);
   this.setVisible(true);
   }

private void assembleInterface() {
   JPanel p = new JPanel(new FlowLayout());
   p.add(open);
   p.add(decode);
   p.add(reset);
   this.getContentPane().add(p, BorderLayout.NORTH);
   open.addActionListener(this);
   decode.addActionListener(this);
   decode.setEnabled(false);
   reset.addActionListener(this);
   open.setMnemonic('O');
   decode.setMnemonic('D');
   reset.setMnemonic('R');

   p = new JPanel(new GridLayout(1,1));
   p.add(new JScrollPane(message));
   message.setFont(new Font("Arial",Font.BOLD,20));
   p.setBorder(BorderFactory.createTitledBorder("Decoded message"));
   message.setEditable(false);
   this.getContentPane().add(p, BorderLayout.SOUTH);

   imagePane.setBorder(BorderFactory.createTitledBorder("Steganographed Image"));
  this.getContentPane().add(imagePane, BorderLayout.CENTER);
```

```
  }
 @Override
 public void actionPerformed(ActionEvent ae) {
   Object o = ae.getSource();
   if(o == open)
     openImage();
   else if(o == decode)
     decodeMessage();
   else if(o == reset)
     resetInterface();
   }

 /*private java.io.File showFileDialog(boolean open) {
   JFileChooser fc = new JFileChooser("Open an image");
   javax.swing.filechooser.FileFilter ff = new javax.swing.filechooser.FileFilter() {
     @Override
     public boolean accept(java.io.File f) {
       String name = f.getName().toLowerCase();
       return f.isDirectory() ||   name.endsWith(".png") || name.endsWith(".bmp")||
name.endsWith(".jpg")|| name.endsWith(".jpeg");
       }
     @Override
     public String getDescription() {
       return "Image (*.png, *.bmp, *.jpg, *.jpeg)";
       }
     };
   fc.setAcceptAllFileFilterUsed(false);
   fc.addChoosableFileFilter(ff);

   java.io.File f = null;
   if(open && fc.showOpenDialog(this) == fc.APPROVE_OPTION)
     f = fc.getSelectedFile();
   else if(!open && fc.showSaveDialog(this) == fc.APPROVE_OPTION)
     f = fc.getSelectedFile();
   return f;
  }*/

 private void openImage() {
   try {
      if(image==null){
         JOptionPane.showMessageDialog(this, "No message has been selected !");
         return;
      }
     JLabel l = new JLabel(new ImageIcon(image));
```

```
    imagePane.getViewport().add(l);
    decode.setEnabled(true);
     this.validate();
     } catch(Exception ex) { ex.printStackTrace(); }
  }

private void decodeMessage() {
  message.setText(decode(image));
  }


private void resetInterface() {
  message.setText("");
  imagePane.getViewport().removeAll();
  image = null;
  this.validate();
  }
        public String decode(BufferedImage img)
        {
        class OTEx extends Exception{
            String display(){
                return "Incorrect OTP!";
            }
        }
                byte[] decod;
                try
                {
                        decod = decode_text(get_byte_data(img));
                String dc=new String(decod);
                String otp="";
                int Ch=(int)dc.charAt(dc.length()-1);
                for(int i=dc.length()-2;i>=dc.length()-1-Ch;i--){
                    otp+=dc.charAt(i);
                }
                //otp recieved is reversed so i must reverse it again
                int ci=otp.length()-1;
                char[] ott=otp.toCharArray();
                for(int i=0;i<otp.length();i++){
                    if(i<ci){
                        char ch=ott[i];
                        ott[i]=ott[ci];
                        ott[ci]=ch;
                        ci--;
                    }
```

```
            else break;
        }
        dc=dc.substring(0, dc.length()-otp.length()-1);
        String oTp=JOptionPane.showInputDialog("Please enter the OTP!");
        boolean match=true;
        if(oTp.length()!=otp.length()){
            match=false;
        }
        if(match){
            for(int i=0;i<otp.length();i++){
                if(oTp.charAt(i)!=ott[i]){
                    match=false;
                    break;
                }
            }
        }
        if(!match){
            throw new OTEx();

        }
                return(dc);
        }
    catch(OTEx ex){
        JOptionPane.showMessageDialog(null,
                        ex.display(),"Error",
                        JOptionPane.ERROR_MESSAGE);
        return "";
    }
        catch(Exception e)
        {
                JOptionPane.showMessageDialog(null,
                        "Image not found !","Error",
                        JOptionPane.ERROR_MESSAGE);
                return "";
        }
}

private byte[] get_byte_data(BufferedImage image)
{
        WritableRaster raster   = image.getRaster();
        DataBufferByte buffer = (DataBufferByte)raster.getDataBuffer();
        return buffer.getData();
}
```

```
        private byte[] decode_text(byte[] image)
        {
                int length = 0;
                int offset  = 32;
                for(int i=0; i<32; ++i)
                {
                        length = (length << 1) | (image[i] & 1);
                }

                byte[] result = new byte[length];

                for(int b=0; b<result.length; ++b )
                {
                        for(int i=0; i<8; ++i, ++offset)
                        {
                                result[b] = (byte)((result[b] << 1) | (image[offset] & 1));
                        }
                }
                return result;
        }

/*public static void main(String arg[]) {
   new DecodeMessage();
  }*/
}
```

O/P screenshots

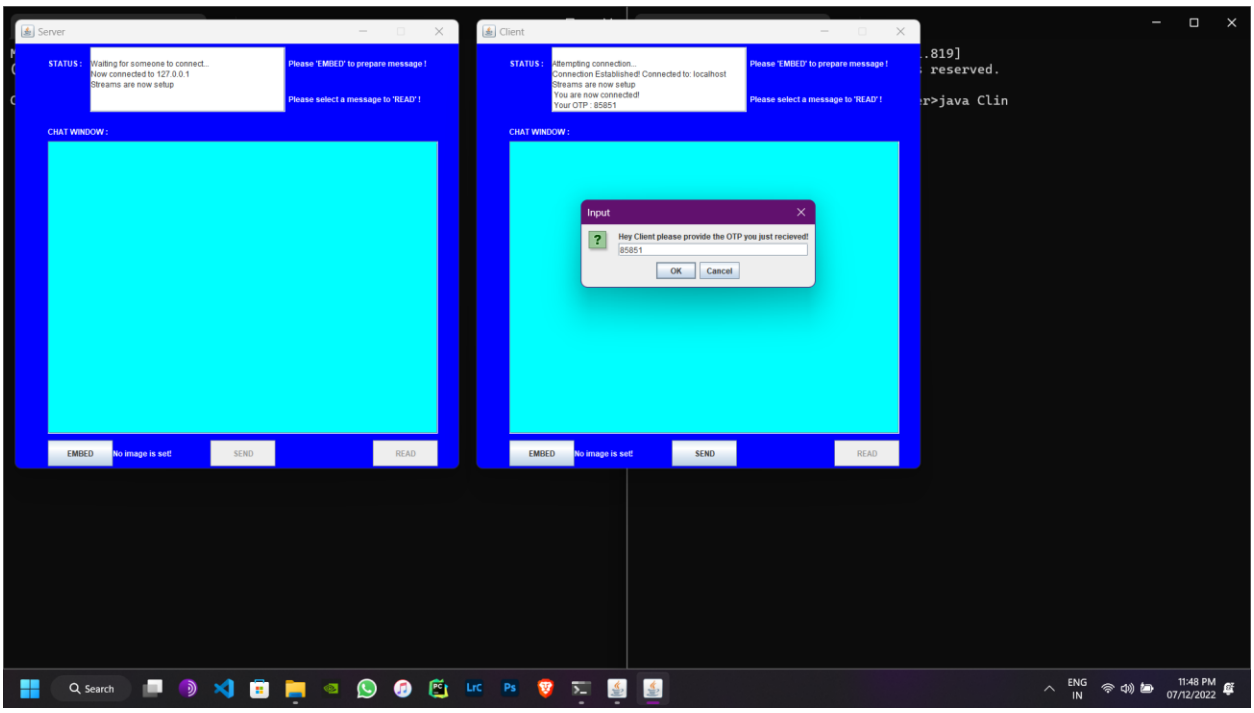
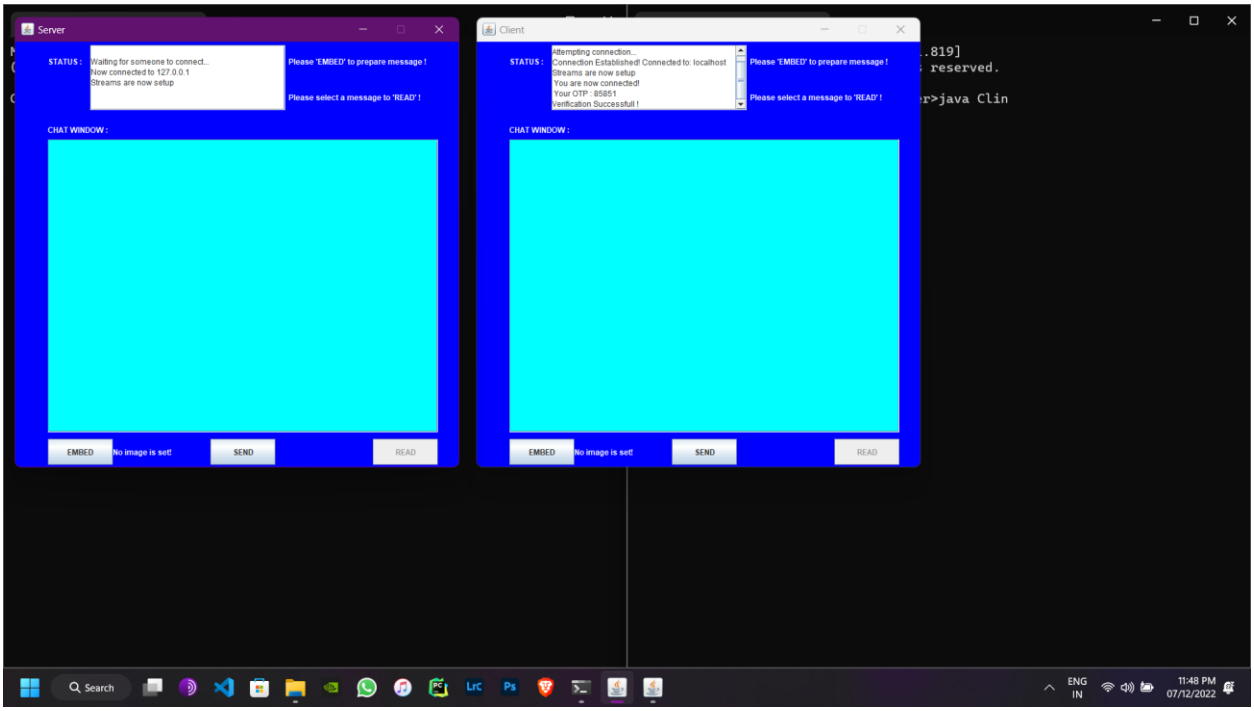
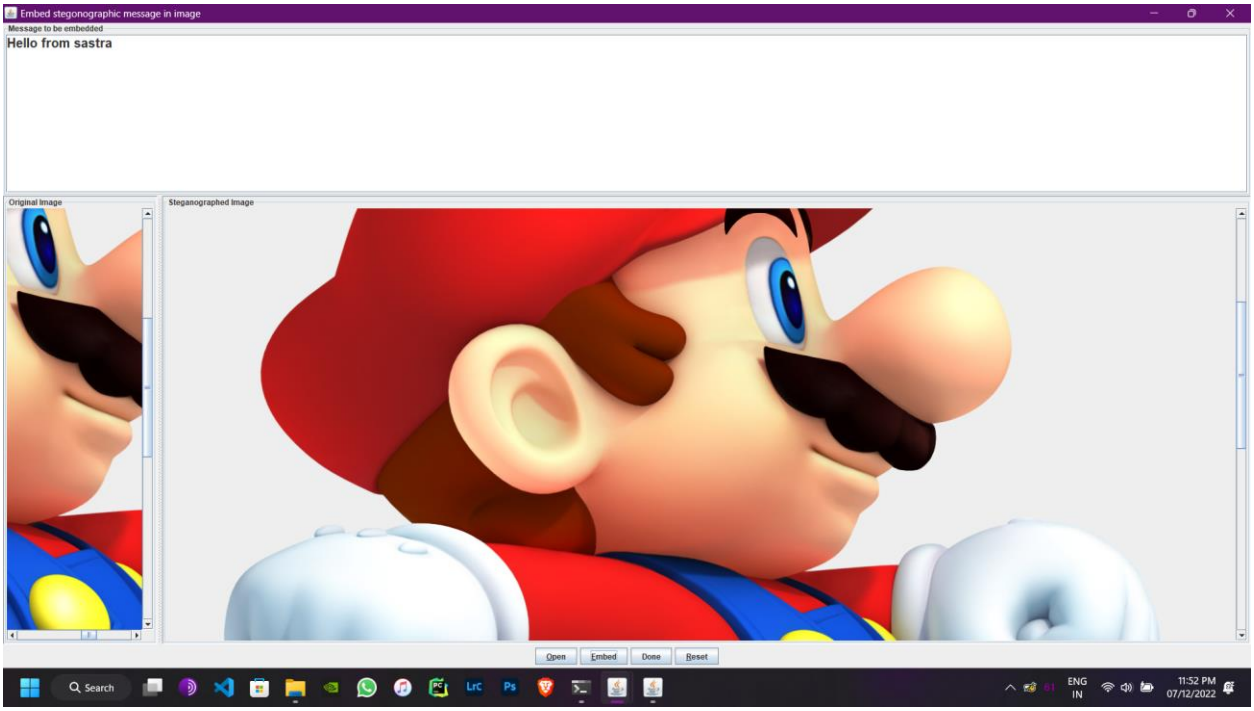
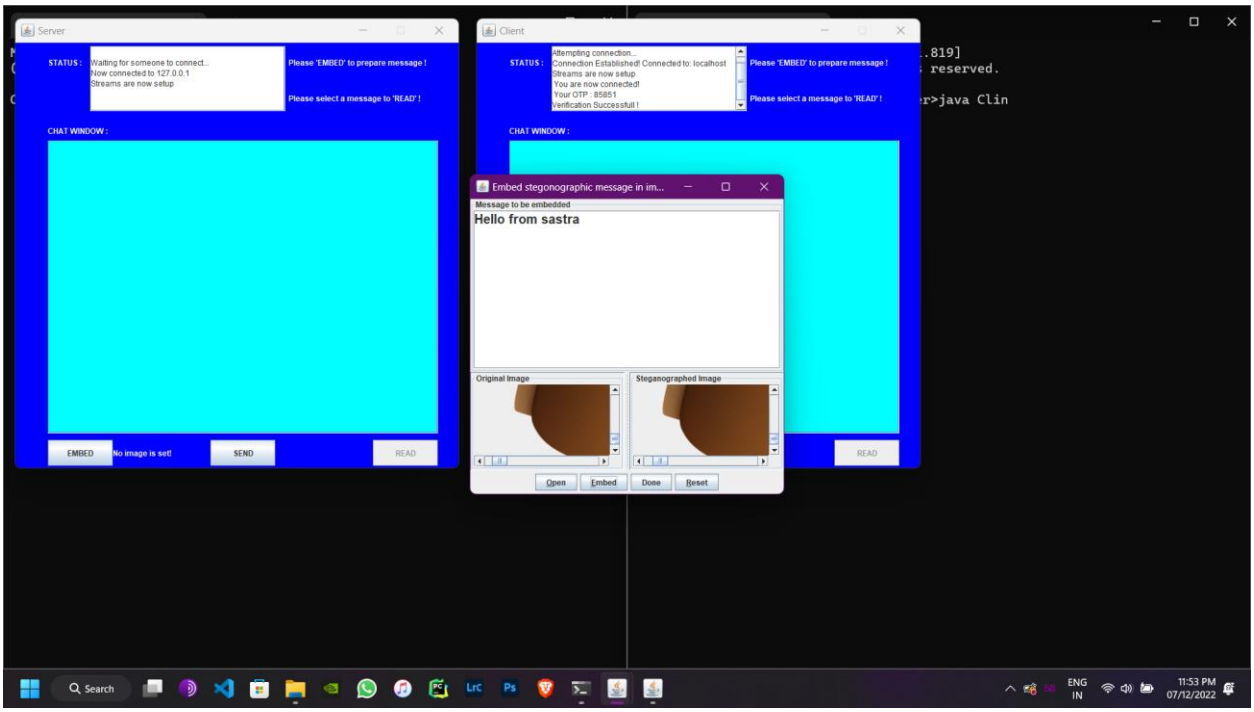


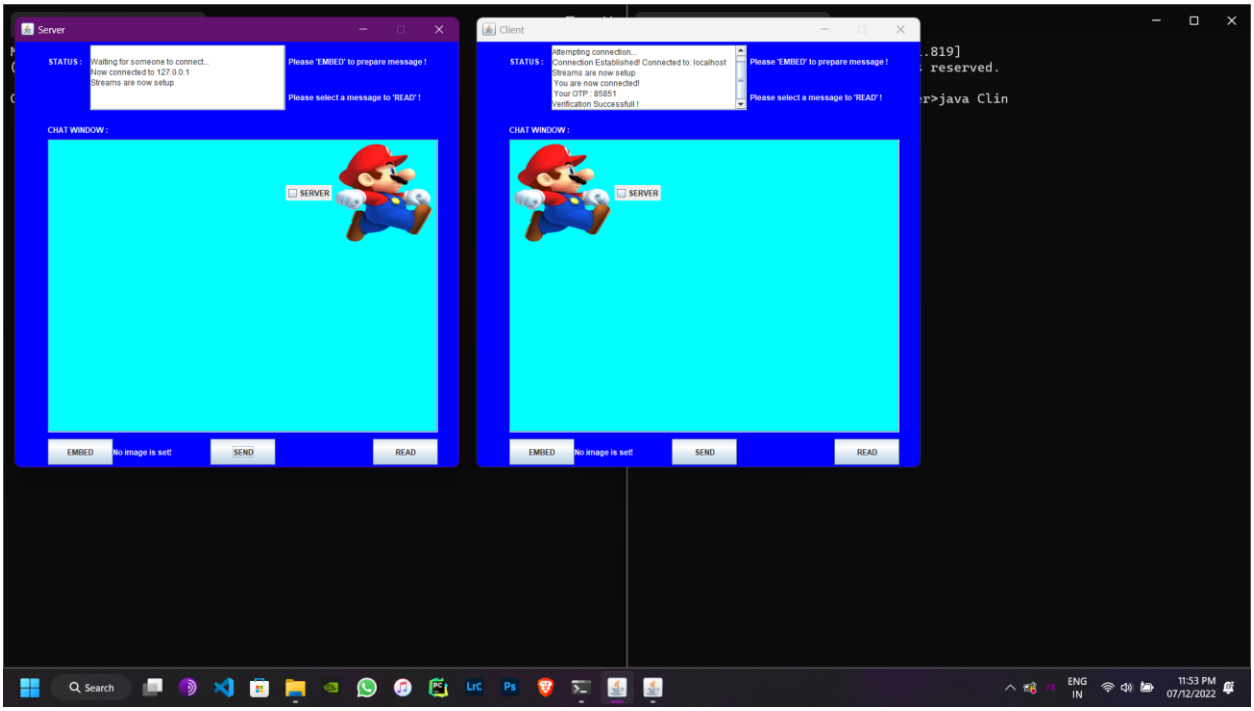
Figure 7:Connecting GUI

Figure8.1-8.5: SERVER SIDE ENCODE AND DECODE




Embed stegonographic message in image
Message to be embedded
Hello from sastra
Original Image
Steganographed Image
Open
Embed
Done
Reset
Search
ENG
IN
11:52 PM
07/12/2022


Server
STATUS :
Waiting for someone to connect...
Now connected to 127.0.0.1
Streams are now setup
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
EMBED
No image is set!
SEND
READ
Client
STATUS :
Attempting connection...
Connection Established! Connected to: localhost
Streams are now setup
You are now connected!
Your OTP : 85851
Verification Successfull !
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
READ
Embed stegonographic message in im...
Message to be embedded
Hello from sastra
Original Image
Steganographed Image
Open
Embed
Done
Reset
.819]
reserved.
>java Clin
Search
ENG
IN
11:53 PM
07/12/2022


Server
STATUS :
Waiting for someone to connect...
Now connected to 127.0.0.1
Streams are now setup
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
SERVER
EMBED
No image is set!
SEND
READ
Client
STATUS :
Attempting connection...
Connection Established! Connected to: localhost
Streams are now setup
You are now connected!
Your OTP : 85851
Verification Successfull !
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
SERVER
EMBED
No image is set!
SEND
READ
.819]
reserved.
>java Clin
Search
ENG
IN
11:53 PM
07/12/2022

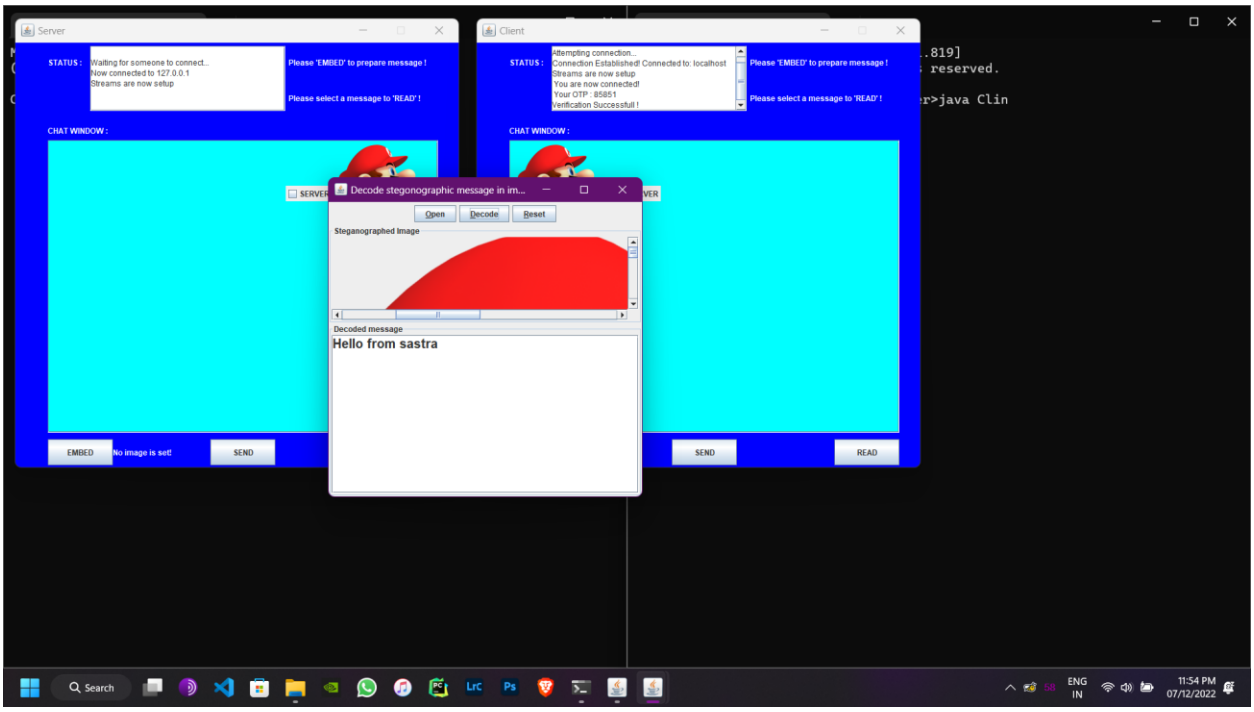
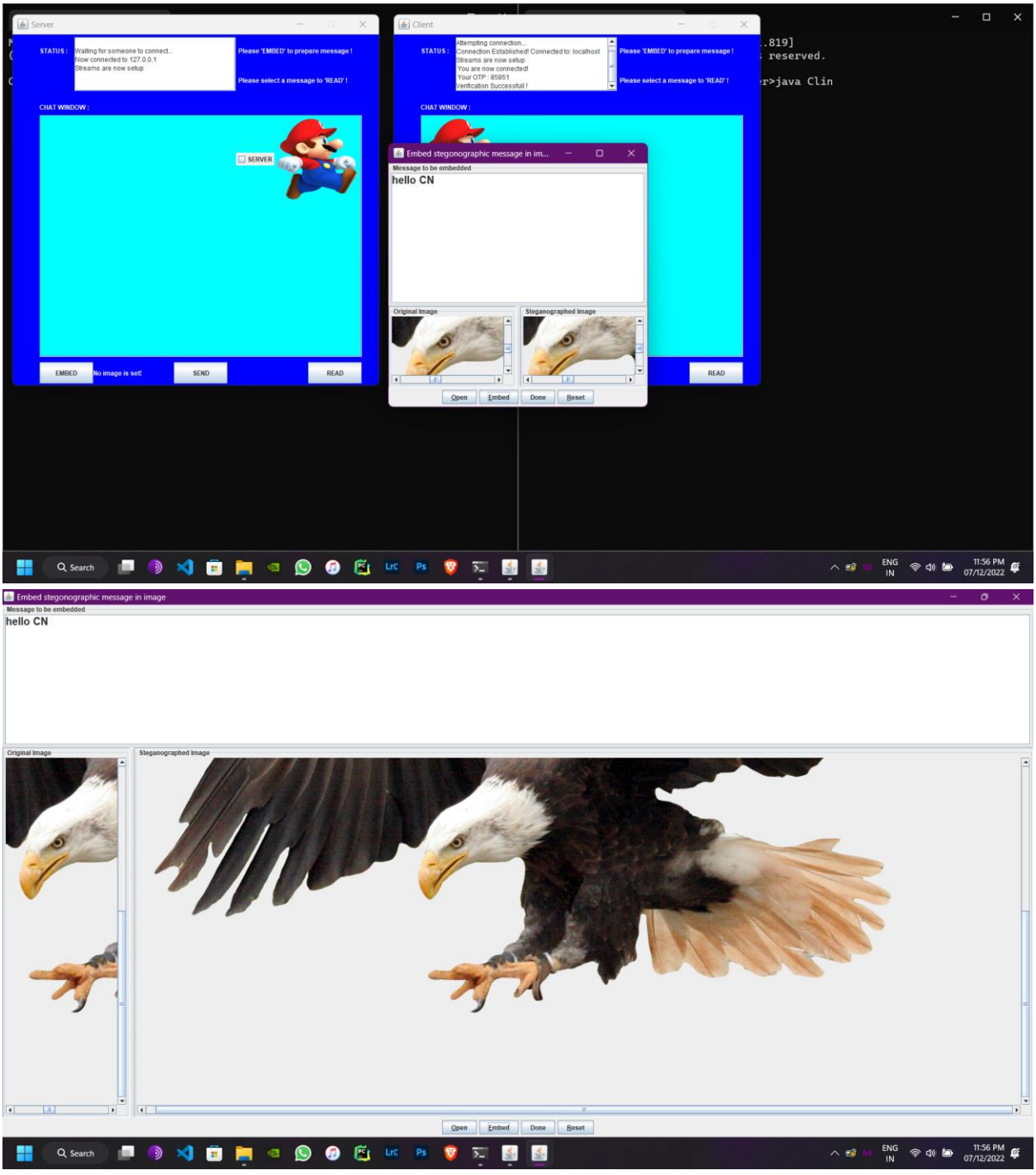
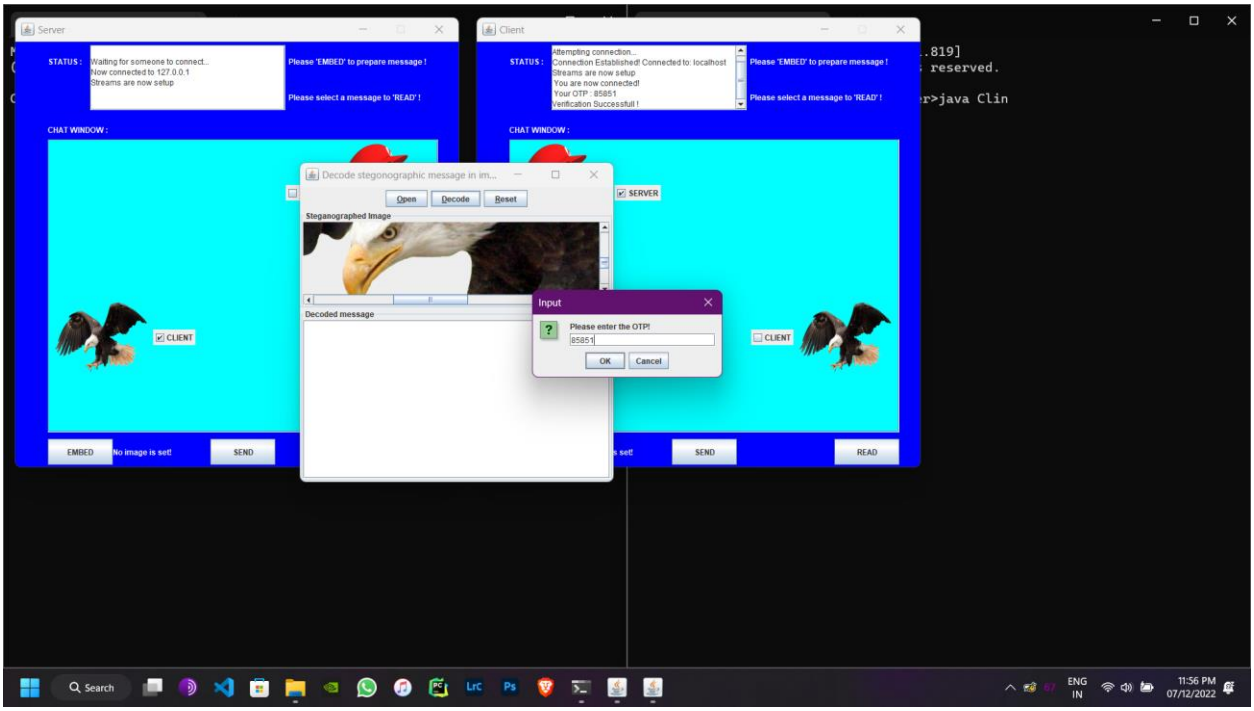

Server
STATUS :
Waiting for someone to connect...
Now connected to 127.0.0.1
Streams are now setup
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
EMBED
No image is set!
SEND
Client
STATUS :
Attempting connection...
Connection Established! Connected to: localhost
Streams are now setup
You are now connected!
Your OTP : 85851
Verification Successfull !
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
SEND
READ
Decode stegonographic message in im...
Open
Decode
Reset
Steganographed Image
Decoded message
Hello from sastra
.819]
reserved.
>java Clin
Search
ENG
IN
11:54 PM
07/12/2022

Figure 9.1-9.4: CLIENT SIDE ENCODE AND DECODE




Server
STATUS :
Waiting for someone to connect...
Now connected to 127.0.0.1
Streams are now setup
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
CLIENT
EMBED
No image is set!
SEND
Client
STATUS :
Attempting connection...
Connection Established! Connected to: localhost
Streams are now setup
You are now connected!
Your OTP : 85851
Verification Successfull !
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
SERVER
CLIENT
SEND
READ
Decode stegonographic message in im...
Open
Decode
Reset
Steganographed Image
Decoded message
Input
Please enter the OTP!
85851
OK
Cancel
.819]
reserved.
>java Clin
Search
ENG
IN
11:56 PM
07/12/2022

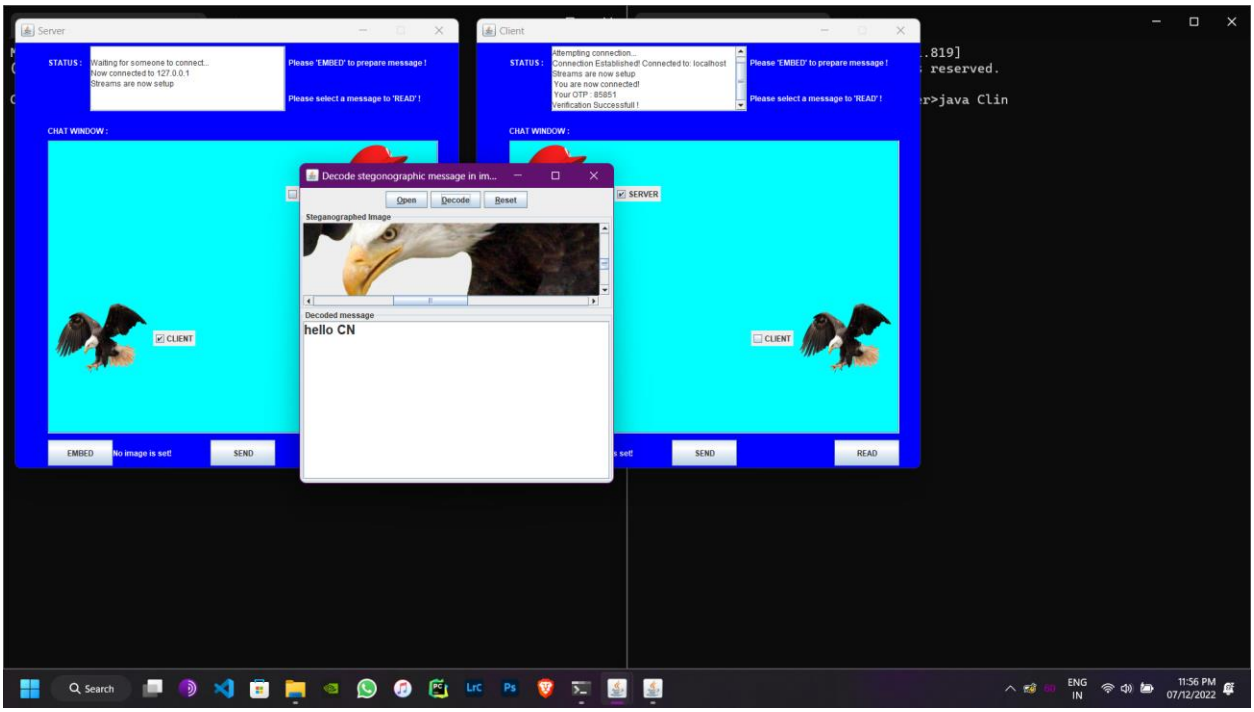
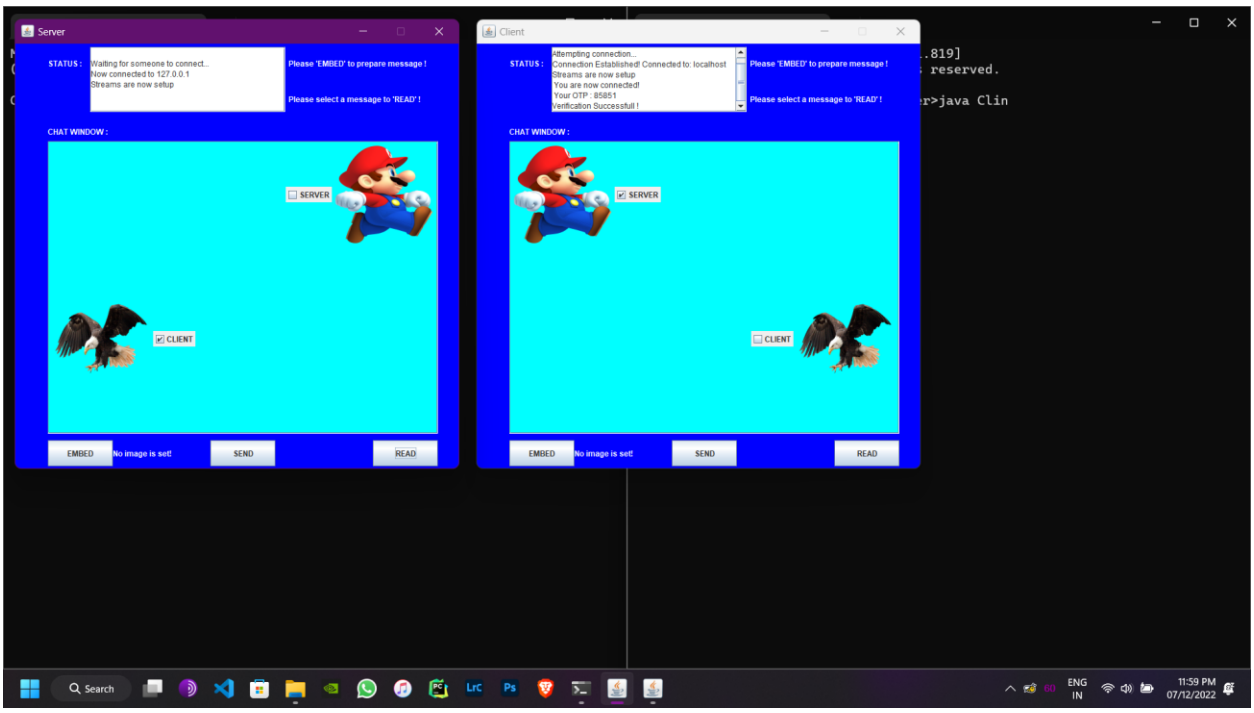

Server
STATUS :
Waiting for someone to connect...
Now connected to 127.0.0.1
Streams are now setup
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
CLIENT
EMBED
No image is set!
SEND
Client
STATUS :
Attempting connection...
Connection Established! Connected to: localhost
Streams are now setup
You are now connected!
Your OTP : 85851
Verification Successfull !
Please 'EMBED' to prepare message !
Please select a message to 'READ' !
CHAT WINDOW :
SERVER
CLIENT
SEND
READ
Decode stegonographic message in im...
Open
Decode
Reset
Steganographed Image
Decoded message
hello CN
.819]
reserved.
>java Clin
Search
ENG
IN
11:56 PM
07/12/2022

Figure 10: BOTH SERVER AND CLIENT



CONCLUSION AND FUTURE PLANS

Steganography is the nascent stage of development. This project provides new method of image steganography i.e., using java. Our method is general and extended to other LSB based steganographic methods

Identifying the maximum capacity of information that can be hidden in a message using a particular steganographic tool has to be modeled. Positively speaking, steganography has the real potential for the safe transmission of data in this hacker's world when utilized in a proper manner.

This type of Steganography can be gradually tested for very secure info transmission and then rolled out for the public
like in Chat applications as seen in our code for very secure data sharing as the image looks very unsuspecting for the average person.

REFERENCES

• Gowda,S.N and Sulakhe, s., 2016,April. Block based least significant bit algorithm for image steganography. Annual Int'l conference on intelligent computing, computer science and information systems (ICCSIS-16).

• http://en.wikipedia.org/wiki/steganography

• Article by AshwinGoel for image-based steganography using python, improved by PratikBhattacharjee2 and AshwinGoel.

• https://www.geeksforgeeks.org/image-steganography-in-cryptography/