

# Secure File Sharing System Using Image Steganography And Cryptography Techniques

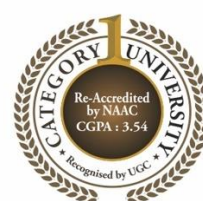
*Report submitted to the SASTRA Deemed to be University  
as the requirement for the course*

## CSE302: COMPUTER NETWORKS

*Submitted by*

**B.V.SAI GEETHANJALI**  
**124157076,CSE(CSBC)**

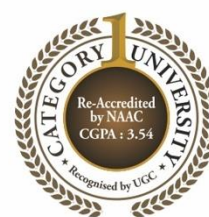
**DECEMBER-2022**



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

**SCHOOL OF COMPUTING**

**THANJAVUR, TAMIL NADU, INDIA – 613 401**



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

**SCHOOL OF COMPUTING**  
**THANJAVUR – 613 401**

**Bonafide Certificate**

This is to certify that the report titled “**Secure File sharing System Using Image Steganography And Cryptography Techniques**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **B.V.Sai Geethanjali (Reg. No.124157076,CSE(CSBC))** during the academic year 2020-21, in the School of Computing

Project Based Work *Viva voce* held on

**Examiner 1**

**Examiner 2**

## **ACKNOWLEDGEMENTS**

First, I want to express my gratitude to the Almighty for his unending favours. I want to thank my parents and everyone else who encouraged me to become interested in the project and helped me stay motivated to finish it before the deadline with minimal effort.

I would like to express my sincere gratitude to Dr S. Vaidya Subramaniam, Vice-Chancellor for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and I would like to thank Dr A. Uma Makeswri, Dean, School of Computing and R. Chandramouli, Registrar for their overwhelming support provided during my course span in SASTRA Deemed University.

I would specially thank and express my gratitude to Dr. Shankar Sriram, Associate Dean, School of Computing for providing me an opportunity to do this project and for his guidance and support to successfully complete the project and academic help extended for the past two years of my life in School of computing. I would specially thank and express my gratitude to Dr. Kavitha R, School of Computing for her guidance and support to successfully complete the project.

I also thank all the Teaching and Non-teaching faculty, and all other people who have directly or indirectly help me through their support, encouragement and all other assistance extended for completion of my project and for successful completion of all courses during my academic life at SASTRA Deemed University.

Finally, I thank my parents and all others who help me acquire this interest in project and aided me in completing it within the deadline without much struggle.

## ***Abstract***

Information security is one of today's most difficult issues. One of the factors contributing to invaders' success is the fact that the majority of the data they obtain from a system is in a format they can read and understand. Intruders may divulge the information to third parties, alter it to falsely represent a person or group, or use it to carry out an attack. Various techniques have been used to protect the transfer of sensitive data via the internet.

This project suggests a system that combines cryptography with image steganography. User chooses the file and enters a key and a picture. Using this key, the user data (file data) is encrypted, and after that, the cypher text is encoded in the image. There are two layers of security: one that hides information using steganography, and the other that encrypts it to keep it safe.

### **KEYWORDS**

Steganography, Encryption, Decryption, Java

## Table Of Contents

S.No	TITLE	Page No
	Bonafide Certificate	ii
	Abstract	iv
1	Introduction	1
2	Motivation	3
3	Steps	4
4	Algorithm Used	5
5	Source Code	7
6	Results/Snapshots	33
7	Advantages	37
8	Conclusion and references	37

# **1.INTRODUCTION**

This project proposes a system that combines visual steganography and cryptography. A key and a picture are entered once the user selects the file. This key is used to encrypt the user data (file data), which is then used to encode the cypher text in the image. There are two levels of security: one that uses steganography to conceal information, and the other that encrypts it to protect it. A user can input the image of his choice into which the user text information is encoded in the encoding area. Before encoding the text into the image, the text is additionally encrypted. The user can then download the encoded image. The decrypted text information is displayed to the user in a new file once the user enters the decoded picture in the decoding area.

Steganography is the practise of concealing information in digital media by embedding secret messages in a way that only the sender and the intended recipient(s) can recognise their presence.

The ability to access and understand the majority of the system's information is one of the factors that contribute to the attackers' success in their intrusion. The information may be shared with others, used improperly or modified, misrepresented to a person or organisation, or used to prepare even more serious attacks by intruders. Utilizing steganography and cryptography is one approach of solving this issue.

## STEGANOGRAPHY TYPES

Steganography can be divided into several categories depending on how it is transmitted, including text, image, video, network, email, and audio steganography.



## STEPS TO TAKE TO COMPLETE IMAGE STEGANOGRAPHY

Obtain the confidential information that the sender and receiver are to exchange

Choose the image that you want to use to encode the secret message.

Add the secret message's encryption to the chosen picture.

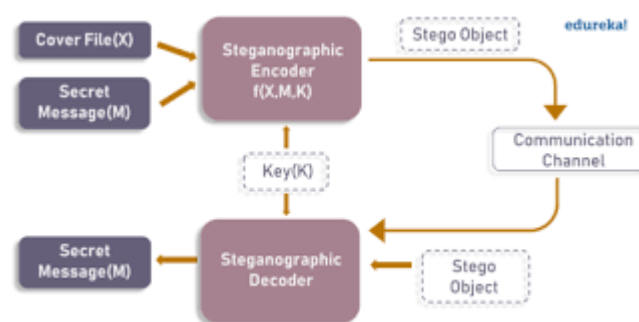
Sender transmits message to recipient

While decoding at the receiver end, the image's secret information is recovered.

In other people's eyes, it is merely an image, but to the sender and recipient, it successfully communicates a secret message.

## BASICS OF STEGANOGRAPHY

The cover file (picture) and secret message are provided to the steganographic encoder in this fundamental steganographic paradigm. The stego object is then communicated from sender to receiver, and steganographic decoder then decodes and only the secret message is acquired.



## 2.MOTIVATION

Information may be easily transferred over vast distances thanks to the internet. Given that both the receiver and the anonymous intruders can see our information, this is both a blessing and a curse. The most common method for protecting data has been encryption, however with enough computer power, this security can be bypassed. A different method of encrypting data is to conceal it within a picture so that the user only sees the image and not the data underneath. These two levels of security guarantee secure data flow.



### **3.STEPS**

A straightforward user interface is constructed, with options for selecting a file, an image, and file data. File data is presented, encrypted file data appears, and the selected image is also shown. The encrypted image can then be downloaded into the laptop and transmitted to the recipient. The data is displayed in the window when the decrypted image is selected on the receiving side. The information can be saved into the new file.

Sender's Side:

- 1.First the user selects the file which he wants to store in image.
- 2.Then the user will select the image.
- 3.Now the user will enter the key with the encryption algorithm using the key now the data will be encoded into the cipher text.
- 4.Then the user selects the encode button in which the cipher text is encoded Into the image and the image is given to the receiver.

Receiver's Side:

1. The cypher text will be shown to the receiver after he selects the steganography image and the decode button in this section.
2. He wants to know the key that the sender used to encrypt the content in order to view the plain text.
- 3.The recipient will be able to view the original text and Save it in a new file after entering the key.

## 4.ALGORITHM USED

### Encryption Algorithm:

```
String key, message;
    String cipher="";
    for(i=0;i<message.length();i++)
    {
        if(j>key.length()-1){
            j=0;
        }
        int temp=message.charAt(i) ^ key.charAt(j);
        cipher=cipher+String.format("%02x", (byte)temp);
        j++;
    }
    System.out.println(cipher);
```

### Decryption Algorithm:

```
String key,cipher;
String hexToDeci = "";
for (int i = 0; i < cipher.length(); i+=2) {
    // splitting hex into a pair of two
    String output = cipher.substring(i, (i+2));
    int decimal = Integer.parseInt(output, 16);
    hexToDeci += (char)decimal;
}
String decrypText = "";
int keyItr = 0;
for (int i = 0; i < hexToDeci.length(); i++) {
    // XOR Operation
    int temp = hexToDeci.charAt(i)^key.charAt(keyItr);
    decrypText += (char)temp;
    keyItr++;
    if(keyItr >= key.length()){keyItr = 0; }
}
```

### **Least Significant Bit Algorithm:**

- 1.Convert the image into pixels.
- 2.Convert the image pixels into bytes.
- 3.Store the length of the text in image pixels.
- 4.Convert the text into Ascii Numbers and into bytes.
- 5.Traverse through each pixel of the array and do
- 6.Convert the pixel into binary
- 7.Get the next bit to be embedded form the text
- 8.Select the last bit and replace with text
- 9.Repeat until the text gets over.

## **Example Of LSB Method**

- A grid for 3 pixels of a 24-bit image can be as follows:

```
(00101101 00011100 11011100)
(10100110 11000100 00001100)
(11010010 10101101 01100011)
```

- When the number 200, which binary representation is 11001000, is embedded into the least significant bits of this part of the image, the resulting grid is as follows:

```
(00101101 00011101 11011100)
(10100110 11000101 00001100)
(11010010 10101100 01100011)
```

## 5.Source code

### home1.java:

```
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;

public class home1 implements ActionListener{
    public static JButton button1,button2;
    public static JFrame frame;
    public static void main(String []args){

        frame=new JFrame();
        frame.setTitle("Steganography");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(700,500);
        frame.setLayout(null);

        ImageIcon icon1=new
ImageIcon("C://Users//HP//Desktop//assignments//2-1/a.jpeg");
        ImageIcon icon2=new
ImageIcon("C://Users//HP//Desktop//assignments//2-1/a.jpeg");

        button1=new JButton();
        button1.setBounds(200, 100, 250, 50);
        button1.setText("Encode");
        button1.setFocusable(false);
        //    button1.setIcon(icon1);
        button1.addActionListener(new home1());
```

```

button1.setFont(new Font("Comic Sans",Font.BOLD,25));
button1.setForeground(Color.cyan);
button1.setBackground(Color.lightGray);
button1.setBorder(BorderFactory.createEtchedBorder());

button2=new JButton();
button2.setBounds(200, 200, 250, 50);
button2.setText("Decode");
button2.setFocusable(false);
button2.setFont(new Font("Comic Sans",Font.BOLD,25));
// button2.setIcon(icon2);
button2.setForeground(Color.cyan);
button2.setBackground(Color.lightGray);
button2.setBorder(BorderFactory.createEtchedBorder());
button2.addActionListener(new home1());

frame.add(button1);
frame.add(button2);
frame.setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==button1){
        frame.dispose();
        System.out.println("Encode button is cliclcked");
        encode mywindow=new encode();
    }

    else if(e.getSource()==button2){
        frame.dispose();
        System.out.println("Decode button is cliclcked");
        decode window2=new decode();
    }
}
}

```

## **encode.java:**

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 *
 *
 * button 3-Select File
```

```

* button 4-Select Image
* button 5-Encrypt Image
* button 6-Save into new image file
* button 7-Encrypt
* button 8-Decrypt
*
*/
public class encode implements ActionListener{

    JFrame frame=new JFrame();
    JLabel jlabelImage,jlabelimage2;
    JButton button3,button4,button5,button6,button7,button8;
    JTextArea text1,text2;
    JScrollPane pane1,pane2,pane3,pane4;
    JTextField j1;
    BufferedImage sourceimage,embeddedimage;
    JButton button9,button10,button11,button12;

    encode(){

        j1=new JTextField();
        j1.setFont(new Font("arial",Font.BOLD,20));
        j1.setBounds(1000, 50, 150, 50);

        button7=new JButton();
        button7.setText("Encrypt");
        button7.setFocusable(false);
        button7.setBounds(1180, 50, 100, 50);

        button8=new JButton();
        button8.setText("Decrypt");
        button8.setFocusable(false);
        button8.setBounds(1300,50,100,50);
    }
}

```

```

button11=new JButton();
button11.setText("File Data");
button11.setFocusable(false);
button11.setBounds(300, 110, 150, 35);

button12=new JButton();
button12.setText("Encrypted File Data");
button12.setFocusable(false);
button12.setBounds(900, 110, 150, 35);

jlabelImage=new JLabel();
jlabelImage.setBounds(50,400,600,400);
jlabelImage.setBackground(Color.red);

jlabelimage2=new JLabel();
jlabelimage2.setBounds(50, 650, 600, 400);

button3=new JButton();
button4=new JButton();
button5=new JButton();
button6=new JButton();

button3.setText("Select File");
button3.setFocusable(false);
button3.setBounds(50, 50, 150, 50);
button4.setText("Select Image");
button4.setFocusable(false);
button4.setBounds(250, 50, 150, 50);
button5.setText("Encrypt Image");
button5.setFocusable(false);
button5.setBounds(450, 50, 150, 50);
button6.setText("Save Into New File");
button6.setBounds(650, 50, 150, 50);
button6.setFocusable(false);
// label.setBounds(0,0,100,50);
// frame.add(label);

```



```

text1=new JTextArea();
text2=new JTextArea();
text1.setFont(new Font("arial",Font.BOLD,20));
text2.setFont(new Font("arial",Font.BOLD,20));
text1.setBounds(50,150, 600, 300);
//    text1.setText("Raghavendra");
text1.setEditable(false);
text1.setForeground(Color.red);
text1.setBackground(Color.white);


text1.setLineWrap(true);
text1.setWrapStyleWord(true);
text1.setBorder(BorderFactory.createBevelBorder(1));


pane1=new JScrollPane(text1);
pane1.setBounds(50,150,600,300);

pane1.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB
AR_ALWAYS);


//    text2.setText("Geethanjali");
text2.setForeground(Color.red);
text2.setBackground(Color.white);
text2.setEditable(false);
text2.setLineWrap(true);
text2.setWrapStyleWord(true);
text2.setBorder(BorderFactory.createBevelBorder(1));


pane2=new JScrollPane(text2);
pane2.setBounds(50,150,600,300);

```

```
pane2.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB  
AR_ALWAYS);
```

```
pane2.setBounds(750,150,600,300);
```

```
pane3=new JScrollPane(jlabelImage);
```

```
pane3.setBounds(50,500,600,300);
```

```
pane3.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB  
AR_ALWAYS);
```

```
pane3.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCR  
OLLBAR_ALWAYS);
```

```
pane4=new JScrollPane(jlabelimage2);
```

```
pane4.setBounds(750, 500, 600, 300);
```

```
pane4.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCR  
OLLBAR_ALWAYS);
```

```
pane4.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB  
AR_ALWAYS);
```

```
button9=new JButton();
```

```
button10=new JButton();
```

```
button9.setText("Normal Image");
```

```
button9.setFocusable(false);
```

```
button9.setBounds(300, 460, 150, 35);
```

```
button10.setText("Steganographed Image");
```

```
button10.setFocusable(false);
```

```
button10.setBounds(900, 460, 250, 35);
```

```
button3.addActionListener(this);
```

```
button4.addActionListener(this);
```

```

button5.addActionListener(this);
button6.addActionListener(this);
button7.addActionListener(this);
button8.addActionListener(this);
frame.add(button3);
frame.add(button4);
frame.add(button5);
frame.add(button6);
frame.add(button7);
frame.add(button8);
frame.add(button9);
frame.add(button10);
frame.add(button11);
frame.add(button12);
frame.add(j1);
frame.add(pane1);
frame.add(pane2);
frame.add(pane3);
frame.add(pane4);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(500,500);
frame.setLayout(null);
frame.setVisible(true);
}

```

@Override

```

public void actionPerformed(ActionEvent e) {
    if(e.getSource()==button3){

```

```

// System.out.println("Button 1 is clicked");

```

```

    FileReader reader=null;
    try {
        JFileChooser chooser=new JFileChooser();
        chooser.showOpenDialog(null);
        File f=chooser.getSelectedFile();

```

```

String filename=f.getAbsolutePath();
reader = new FileReader(filename);
BufferedReader br=new BufferedReader(reader);
try {
    text1.read(br,null);
    //    text2.read(br,null);
    // System.out.println(text1);
} catch (IOException ex) {
    Logger.getLogger(encode.class.getName()).log(Level.SEVERE,
null, ex);
}
try {
    br.close();
} catch (IOException ex) {
    Logger.getLogger(encode.class.getName()).log(Level.SEVERE,
null, ex);
}
text1.requestFocus();
//    text2.requestFocus();
} catch (FileNotFoundException ex) {
    Logger.getLogger(encode.class.getName()).log(Level.SEVERE, null,
ex);
} finally {
    try {
        reader.close();
    } catch (IOException ex) {
        Logger.getLogger(encode.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
}

```

```

else if(e.getSource()==button4){

```

```

    JFileChooser chooser=new JFileChooser();

```

```

        FileNameExtensionFilter fnef=new
FileNameExtensionFilter("png","jpeg");
        chooser.addChoosableFileFilter(fnef);
        chooser.showOpenDialog(null);

        File selectedimage=chooser.getSelectedFile();
        try {
            sourceimage=ImageIO.read(selectedimage);
        } catch (IOException ex) {
            Logger.getLogger(encode.class.getName()).log(Level.SEVERE, null,
ex);
        }
        String selectedpath=selectedimage.getAbsolutePath();
        ImageIcon i1=new ImageIcon(selectedpath);
        Image image=i1.getImage();

        jlabelImage.setIcon(new ImageIcon(image));

        // jlabelimage2.setIcon(new ImageIcon(image));
    }

    else if(e.getSource()==button7){
        String key=j1.getText();
        String message=text1.getText();
        // System.out.println(message);
        // System.out.println(key.length());
        //encryption
        String cipher="";
        int j=0;
        for(int i=0;i<message.length();i++){
            if(j>key.length()-1){
                j=0;
            }
            int temp=message.charAt(i) ^ key.charAt(j);
            cipher=cipher+String.format("%02x", (byte)temp);
            j++;
        }
    }

```

```

    }
    text2.setText(cipher);
}

else if(e.getSource()==button8){

    String key=j1.getText();
    String cipher=text2.getText();

    String hexToDeci = "";
    for (int i = 0; i < cipher.length()-1; i+=2) {
        // splitting hex into a pair of two
        String output = cipher.substring(i, (i+2));
        int decimal = Integer.parseInt(output, 16);
        hexToDeci += (char)decimal;
    }

    // decryption
    String decrypText = "";
    int keyItr = 0;
    for (int i = 0; i < hexToDeci.length(); i++) {
        // XOR Operation
        int temp = hexToDeci.charAt(i) ^ key.charAt(keyItr);

        decrypText += (char)temp;
        keyItr++;
        if(keyItr >= key.length()){
            // once all of key's letters are used, repeat the key
            keyItr = 0;
        }
    }

    text2.setText(decrypText);
}

```

```

        else if(e.getSource()==button5){
            String mess=text2.getText();
            //    System.out.println(f);
            //    ImageIcon image=(ImageIcon) jLabelImage.getIcon();
            //    System.out.println(image);
            //    System.out.println(image.getIconWidth());

embeddedimage=sourceimage.getSubimage(0,0,sourceimage.getWidth(),source
image.getHeight());
            embedmessage(embeddedimage,mess);
            jLabelimage2.setIcon(new ImageIcon(embeddedimage));

        }

        else if(e.getSource()==button6){
            //            if(embeddedimage == null) {
            //            JOptionPane.showMessageDialog(frame, "No message has been
            embedded!",
            //            "Nothing to save", JOptionPane.ERROR_MESSAGE);
            //            return;
            //            }
            //
            //            try {
            //            //JFileChooser f=new JFileChooser();
            //            ImageIO.write(embeddedimage, "PNG", new
            File("filename1.png"));
            //            } catch (IOException ex) {
            //
            //            Logger.getLogger(encode.class.getName()).log(Level.SEVERE, null, ex);
            //            }
            //

            if(embeddedimage == null) {
                JOptionPane.showMessageDialog(frame, "No message has been
                embedded!",

```

```

        "Nothing to save", JOptionPane.ERROR_MESSAGE);
    return;
}
    File f = showFileDialog(false);
    String name = f.getName();
    String ext = name.substring(name.lastIndexOf(".") + 1).toLowerCase();
    // System.out.println(ext);
    if(!ext.equals("png") && !ext.equals("jpeg") && !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(); }
}

}

public static void main(String []args){
    encode encode=new encode();
}

private void embedmessage(BufferedImage img, String mess) {
    int messagelength=mess.length();
    int imagewidth=img.getWidth();
    int imageheight=img.getHeight();
    int imagesize=imagewidth*imageheight;
    if(messagelength*8+32>imagesize){
        JOptionPane.showMessageDialog(frame, "Message is too long for the
chosen image", "Message too long!", JOptionPane.ERROR_MESSAGE);
        return;
    }
    embedinteger(img,messagelength,0,0);
    // System.out.println(imagesize);
    byte b[]=mess.getBytes();

```



```

//  System.out.println(b);
//for(int i=0;i<b.length;i++){
//  System.out.println(b[i]);

    for(int i=0;i<b.length;i++){
        embedbyte(img,b[i],i*8+32,0);
    }
}

private File showFileDialog(boolean open) {
//  return null;
    JFileChooser fc = new JFileChooser("Open an image");
    FileFilter ff = new FileFilter() {
        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(".jpeg") ;
        }
        public String getDescription() {
            if(open)
                return "Image (*.jpg, *.jpeg, *.png, *.gif, *.tiff, *.bmp, *.dib)";
            return "Image (*.png, *.bmp)";
        }
    };

    fc.setAcceptAllFileFilterUsed(false);
    fc.addChoosableFileFilter(ff);
    File f=null;
    if(open && fc.showOpenDialog(frame) ==
JFileChooser.APPROVE_OPTION)

```

```

        f = fc.getSelectedFile();
    else if(!open && fc.showSaveDialog(frame) ==
JFileChooser.APPROVE_OPTION)
        f = fc.getSelectedFile();
    return f;
}

private void embedinteger(BufferedImage img, int n, int start, int storagebit)
{
    // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
    int x=img.getWidth(),y=img.getHeight();
    int startx=start/x,starty=start-startx*y,count=0;
    for(int i=startx; i<x && count<32; i++) {
    for(int j=starty; j<y && count<32; j++) {
        int rgb = img.getRGB(i, j);
        int bit = getBitValue(n, count);
        rgb = setBitValue(rgb, storagebit, bit);
        img.setRGB(i, j, rgb);
        count++;
        if(j==x-1) {
            starty = 0;
        }
    }
}
}
}
}

```

```

private void embedbyte(BufferedImage img, int n, int start, int storagebit) {
    // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
    int x=img.getWidth(),y=img.getHeight();
    int startx=start/x,starty=start-startx*y,count=0;
    for(int i=startx; i<x && count<32; i++) {
        // System.out.println(i+" "+count);
    for(int j=starty; j<y && count<32; j++) {
        int rgb = img.getRGB(i, j);

```

```

int bit = getBitValue(n, count);

int a=(rgb>>24) & 0xff;
int r = (rgb >> 16) & 0xff;
int g = (rgb >> 8) & 0xff;
int b = (rgb >> 0) & 0xff;
System.out.println("Normal "+a+" "+" "+r+" "+g+" "+b+" bit: "+bit+"
count: "+count);

```

```

rgb = setBitValue(rgb, storagebit, bit);

a=(rgb>>24)&0xff;
r = (rgb >> 16) & 0xff;
g = (rgb >> 8) & 0xff;
b = (rgb >> 0) & 0xff;
System.out.println("stego "+a+" "+" "+r+" "+g+" "+b+" bit: "+bit+"
count: "+count);
img.setRGB(i, j, rgb);
count++;
if(j==x-1) {
    starty = 0;
}
}
}
}
}

```

```

private int getBitValue(int n, int location) {
    // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
    int v = n & (int) Math.round(Math.pow(2, location));
    return v==0?0:1;
}

```

```

private int setBitValue(int n, int location, int bit) {
    // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.

```

```

        int toggle = (int) Math.pow(2, location),
            bv = getBitValue(n, location);
        if(bv == bit)
            return n;
        if(bv == 0 && bit == 1)
            n |= toggle;
        else if(bv == 1 && bit == 0)
            n ^= toggle;
        return n;
    }
}

```

### **decode.java:**

```

import java.awt.Color;
import java.awt.Font;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

```

```

import javax.swing.ScrollPaneConstants;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.io.File;
import java.io.FileWriter;

public class decode implements ActionListener {
    JFrame frame=new JFrame();
    JLabel jlabelImage,jlabelimage2;
    JButton button3,button4,button5,button6,button7,button8;
    JTextArea text1,text2;
    JScrollPane pane1,pane2,pane3,pane4;
    JTextField j1;
    BufferedImage sourceimage,embeddedimage;
    JButton button9,button10,button11,button12;
    BufferedImage image;

    public decode(){

        j1=new JTextField();
        j1.setFont(new Font("arial",Font.BOLD,20));
        j1.setBounds(1000, 50, 150, 50);

        button7=new JButton();
        button7.setText("Decrypt");
        button7.setFocusable(false);
        button7.setBounds(1180, 50, 100, 50);

        //    button8=new JButton();
        //    button8.setText("Decrypt");
        //    button8.setFocusable(false);
        //    button8.setBounds(1300,50,100,50);

        button11 =new JButton();

```

```

button11.setText("Decrypted File Data");
button11.setFocusable(false);
button11.setBounds(300, 110, 150, 35);

button12 =new JButton();
button12.setText("Encrypted File Data");
button12.setFocusable(false);
button12.setBounds(900, 110, 150, 35);

jlabelImage=new JLabel();
jlabelImage.setBounds(50,400,600,400);
jlabelImage.setBackground(Color.red);

jlabelimage2=new JLabel();
jlabelimage2.setBounds(50, 650, 600, 400);

button3=new JButton();
button4=new JButton();
button5=new JButton();
button6=new JButton();

button3.setText("Select File");
button3.setFocusable(false);
button3.setBounds(50, 50, 150, 50);
button4.setText("Select Image");
button4.setFocusable(false);
button4.setBounds(250, 50, 150, 50);
button5.setText("Decrypt Image");
button5.setFocusable(false);
button5.setBounds(450, 50, 150, 50);
button6.setText("Save Data Into New File");
button6.setBounds(650, 50, 150, 50);
button6.setFocusable(false);
// label.setBounds(0,0,100,50);
// frame.add(label);

```

```

text1=new JTextArea();
text2=new JTextArea();
text1.setFont(new Font("arial",Font.BOLD,20));
text2.setFont(new Font("arial",Font.BOLD,20));
text1.setBounds(50,150, 600, 300);
//    text1.setText("Raghavendra");
text1.setEditable(false);
text1.setForeground(Color.red);
text1.setBackground(Color.white);


text1.setLineWrap(true);
text1.setWrapStyleWord(true);
text1.setBorder(BorderFactory.createBevelBorder(1));


pane1=new JScrollPane(text1);
pane1.setBounds(50,150,600,300);

pane1.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB
AR_ALWAYS);


//    text2.setText("Geethanjali");
text2.setForeground(Color.red);
text2.setBackground(Color.white);
text2.setEditable(false);
text2.setLineWrap(true);
text2.setWrapStyleWord(true);
text2.setBorder(BorderFactory.createBevelBorder(1));


pane2=new JScrollPane(text2);
pane2.setBounds(50,150,600,300);

```

```
pane2.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB  
AR_ALWAYS);
```

```
pane2.setBounds(750,150,600,300);
```

```
pane3=new JScrollPane(jlabelImage);
```

```
pane3.setBounds(50,500,600,300);
```

```
pane3.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB  
AR_ALWAYS);
```

```
pane3.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCR  
OLLBAR_ALWAYS);
```

```
pane4=new JScrollPane(jlabelimage2);
```

```
pane4.setBounds(50, 500, 1200, 300);
```

```
pane4.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCR  
OLLBAR_ALWAYS);
```

```
pane4.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLB  
AR_ALWAYS);
```

```
button9=new JButton();
```

```
button10=new JButton();
```

```
button9.setText("Decrypted Image");
```

```
button9.setFocusable(false);
```

```
button9.setBounds(300, 460, 150, 35);
```

```
button10.setText("Steganographed Image");
```

```
button10.setFocusable(false);
```

```
button10.setBounds(500, 460, 250, 35);
```

```
button3.addActionListener(this);
```

```
button4.addActionListener(this);
```



```

        button5.addActionListener(this);
        button6.addActionListener(this);
        button7.addActionListener(this);
//    button8.addActionListener(this);
        frame.add(button3);
        frame.add(button4);
        frame.add(button5);
        frame.add(button6);
        frame.add(button7);
//    frame.add(button8);
//    frame.add(button9);
        frame.add(button10);
        frame.add(button11);
        frame.add(button12);
        frame.add(j1);
        frame.add(pane1);
        frame.add(pane2);
//    frame.add(pane3);
        frame.add(pane4);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500,500);
        frame.setLayout(null);
        frame.setVisible(true);
    }
    public static void main(String []args){
        decode n1=new decode();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==button4){
            JFileChooser chooser=new JFileChooser();
            FileNameExtensionFilter fnef=new
FileNamesExtensionFilter("png","jpg","jpeg");
            chooser.addChoosableFileFilter(fnef);
            chooser.showOpenDialog(null);

```

```

        File selectedimage=chooser.getSelectedFile();
        try {
            sourceimage=ImageIO.read(selectedimage);
        } catch (IOException ex) {
            Logger.getLogger(encode.class.getName()).log(Level.SEVERE, null,
ex);
        }
        String selectedpath=selectedimage.getAbsolutePath();
        ImageIcon i1=new ImageIcon(selectedpath);

        Image image2=i1.getImage();
        jlabelimage2.setIcon(new ImageIcon(image2));
    }

    else if(e.getSource()==button7){
        // System.out.println("button is clicked");

        String key=j1.getText();
        String cipher=text2.getText();

        String hexToDeci = "";
        for (int i = 0; i < cipher.length()-1; i+=2) {
            // splitting hex into a pair of two
            String output = cipher.substring(i, (i+2));
            int decimal = Integer.parseInt(output, 16);
            hexToDeci += (char)decimal;
        }

        // decryption
        String decrypText = "";
        int keyItr = 0;
        for (int i = 0; i < hexToDeci.length(); i++) {
            // XOR Operation
            int temp = hexToDeci.charAt(i) ^ key.charAt(keyItr);

```

```

        decryptText += (char)temp;
        keyItr++;
        if(keyItr >= key.length()){
            // once all of key's letters are used, repeat the key
            keyItr = 0;
        }

    }

    text1.setText(decryptText);
}
else if(e.getSource()==button5){
    // System.out.println("button is clicked");
    int len = extractInteger(sourceimage, 0, 0);
    byte b[] = new byte[len];
    for(int i=0; i<len; i++)
        b[i] = extractByte( sourceimage, i*8+32, 0);
    text2.setText(new String(b));

}
else if(e.getSource()==button6){
    // System.out.println("button is pe");
    File output=new File("C://Users//HP//Desktop//2-2//output.txt");
    try{
        FileWriter filewriter=new FileWriter("C://Users//HP//Desktop//2-
2//output.txt");
        String out=text1.getText();
        filewriter.write(out);
        filewriter.close();
        System.out.println(text1.getText());
    }catch(IOException e1){
        System.out.println(e1);
    }
}
}
}

```

```

private int extractInteger(BufferedImage img, int start, int storageBit) {
    // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
    int maxX = img.getWidth(), maxY = img.getHeight(),
        startX = start/maxY, startY = start - startX*maxY, count=0;
    int length = 0;
    for(int i=startX; i<maxX && count<32; i++) {
        for(int j=startY; j<maxY && count<32; j++) {
            int rgb = img.getRGB(i, j), bit = getBitValue(rgb, storageBit);
            length = setBitValue(length, count, bit);
            count++;
            if(j==maxY-1) {
                startY = 0;
            }
        }
    }
    return length;
}

```

```

private byte extractByte(BufferedImage img, int start, int storageBit) {
    // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
    int maxX = img.getWidth(), maxY = img.getHeight(),
        startX = start/maxY, startY = start - startX*maxY, count=0;
    byte b = 0;
    for(int i=startX; i<maxX && count<8; i++) {
        for(int j=startY; j<maxY && count<8; j++) {
            int rgb = img.getRGB(i, j), bit = getBitValue(rgb, storageBit);
            b = (byte)setBitValue(b, count, bit);
            count++;
            if(j==maxY-1) {
                startY = 0;
            }
        }
    }
    return b;
}

```

```

    }

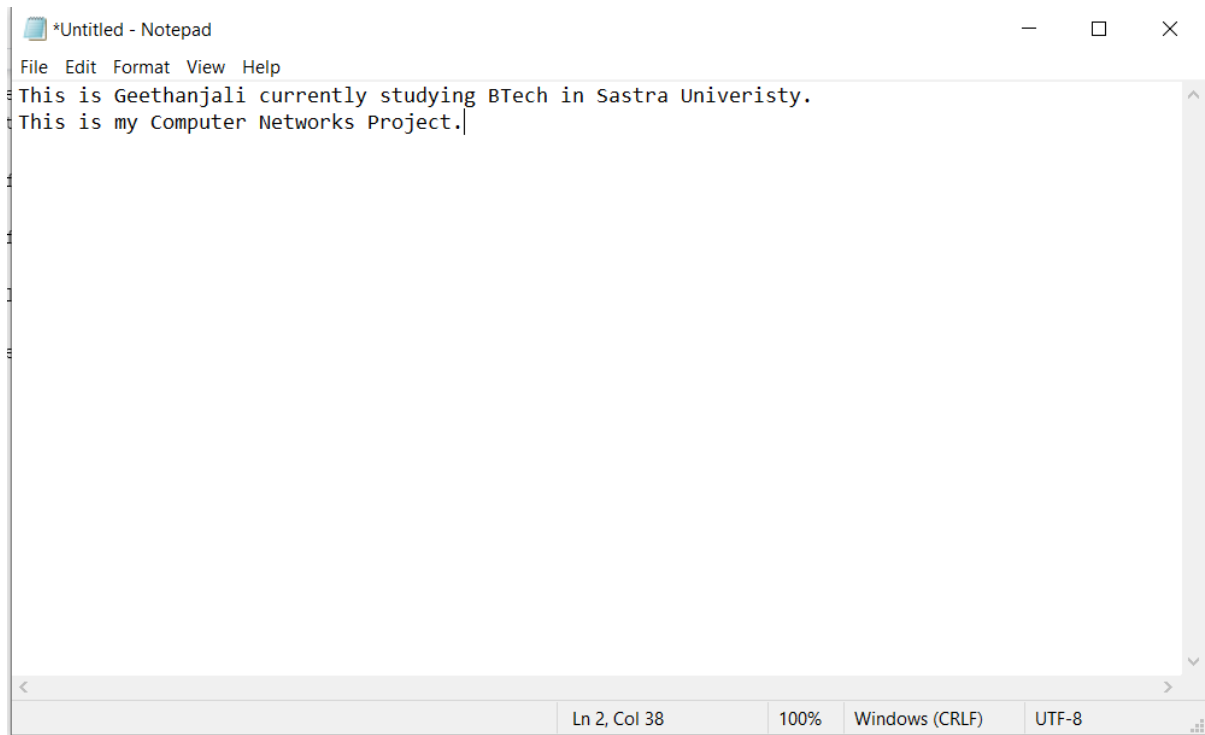
    private int getBitValue(int n, int location) {
        // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
        int v = n & (int) Math.round(Math.pow(2, location));
        return v==0?0:1;
    }

    private int setBitValue(int n, int location, int bit) {
        // throw new UnsupportedOperationException("Not supported yet."); //To
change body of generated methods, choose Tools | Templates.
        int toggle = (int) Math.pow(2, location),
            bv = getBitValue(n, location);
        if(bv == bit)
            return n;
        if(bv == 0 && bit == 1)
            n |= toggle;
        else if(bv == 1 && bit == 0)
            n ^= toggle;
        return n;
    }
}

```

## 6.Results / Snapshots

Original File:

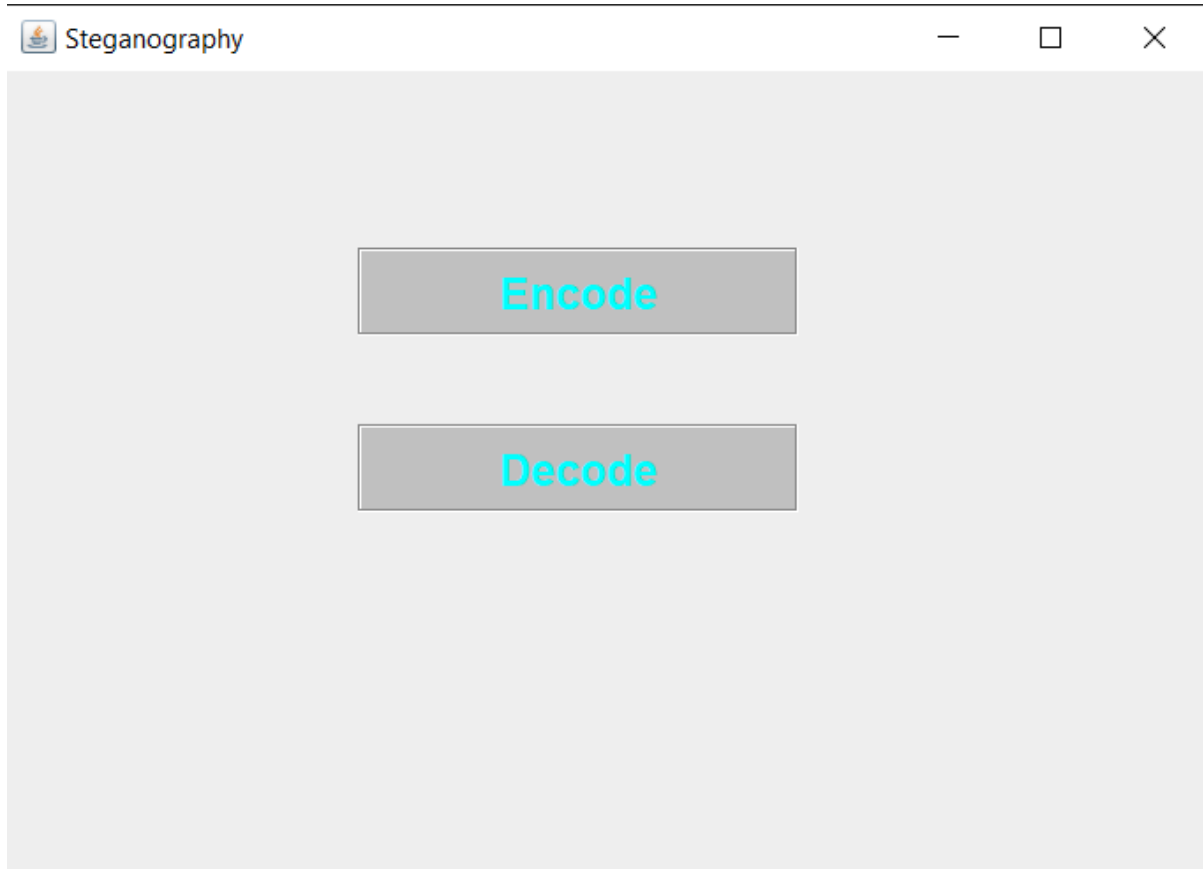


A screenshot of a Notepad window titled '\*Untitled - Notepad'. The window has a menu bar with 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains two lines: 'This is Geethanjali currently studying BTech in Sastra Univeristy.' and 'This is my Computer Networks Project.' The status bar at the bottom shows 'Ln 2, Col 38', '100%', 'Windows (CRLF)', and 'UTF-8'.

```
*Untitled - Notepad
File Edit Format View Help
This is Geethanjali currently studying BTech in Sastra Univeristy.
This is my Computer Networks Project.
```

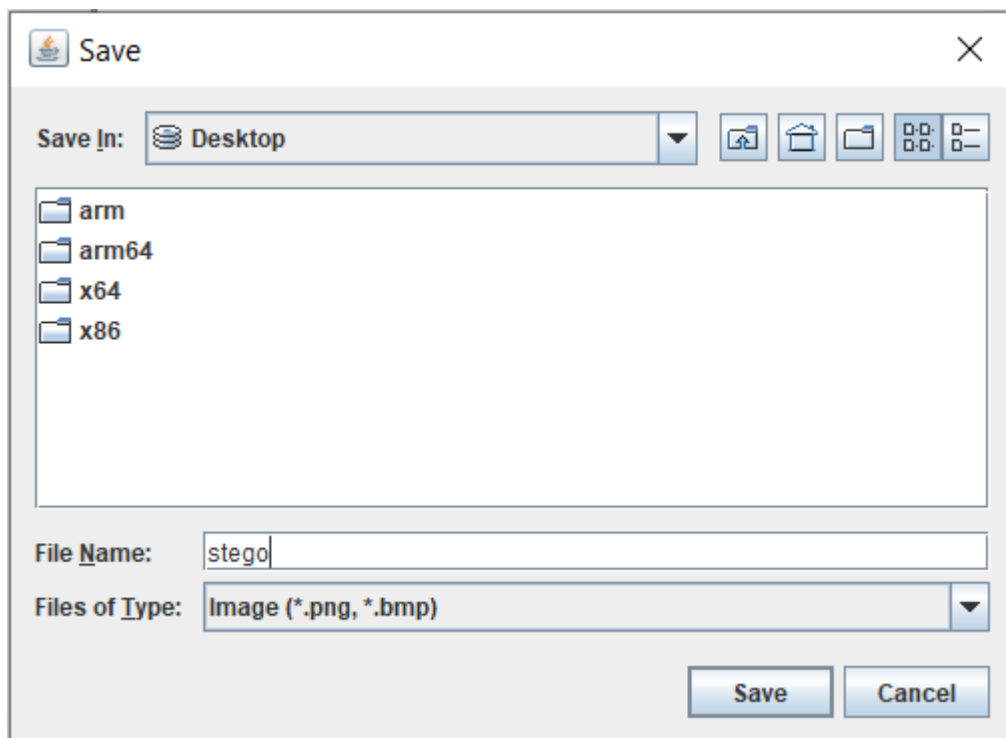
Ln 2, Col 38 100% Windows (CRLF) UTF-8

## Home Page:

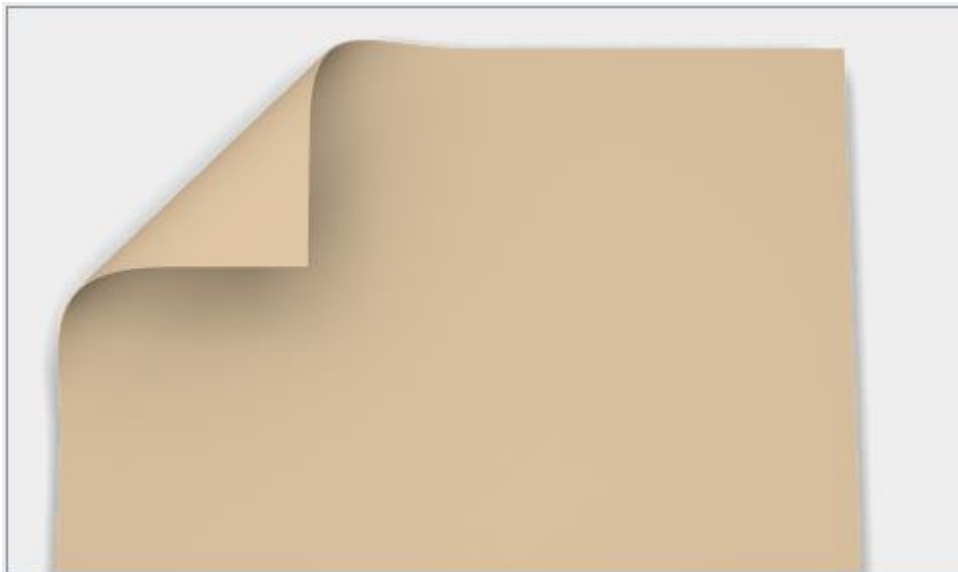


## Encode Page:



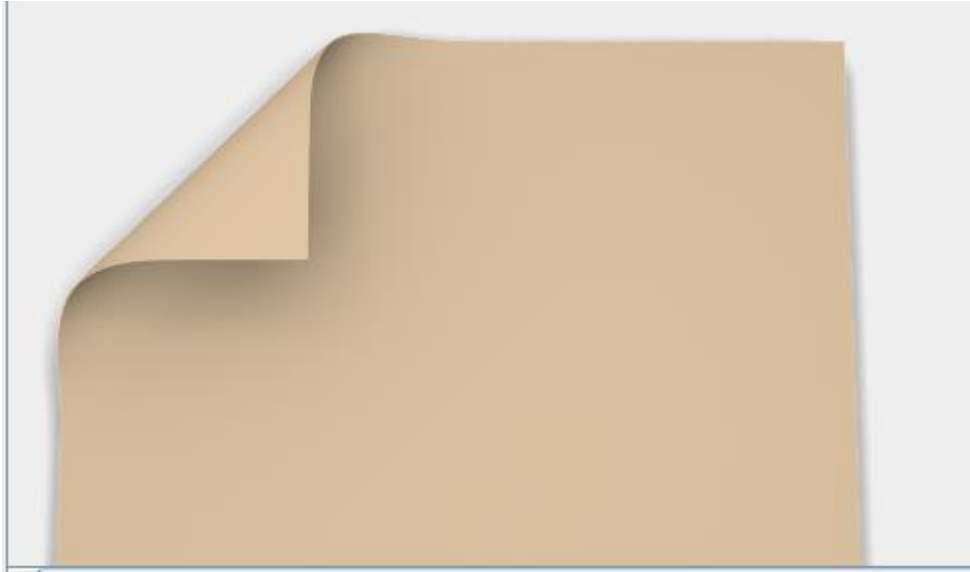


Normal Image:





Encrypted Image:



Decode Page:

Software interface for decoding a steganographed image. The interface includes a top toolbar with buttons: "Select File", "Select Image", "Decrypt Image", "Save Data Into No...", "saif" (text input), and "Decrypt". Below the toolbar are two text areas: "Decrypted File Data" and "Encrypted File Data". The "Decrypted File Data" area contains the text: "This is Geethanjali currently studying BTech in Sastra Univeristy. This is my Computer Networks Project." The "Encrypted File Data" area contains a long hexadecimal string: "27090000410000412e16041d1b00071900051a410a06131b160f1d1f184900151c1718001d064931350c1009491a0f4920001a0713085334071a170c01081a0718477935011a12491a12491e1849300e0403141d1613493d041d040e1b18124923130619040a074f". Below these text areas is a "Steganographed Image" button and a preview window showing a blank, aged, cream-colored page with a folded top-left corner, identical to the one above.

## **7.ADVANTAGES**

1. Files are shared over the network in a secure manner without the data being visible.
2. Two layers of security that improve the system's performance.
3. It is more effective due to the use of cryptography and least significant bit algorithms.
4. Our laptops and computers can use this protection of the files contained in the photographs.

## **8.CONCLUSION AND REFERENCES**

We have successfully created a desktop application for image steganography combined with cryptography techniques, in which the user can decode the information from the image using the same algorithm and insert encrypted file data into the image using cryptography and the least significant bit algorithm. Even though the implementation is exclusive to picture files, the system functions as a general-purpose system that supports media types including audio and video. However, the appropriate steganographic techniques must be used.

- <https://blog.ipswitch.com/the-importance-of-file-transfer-encryption>
- <https://en.wikipedia.org/wiki/Cryptography>
- <http://tutorials.jenkov.com/java-cryptography/index.html>
- <https://en.wikipedia.org/wiki/Steganography>