IMAGE STEGNOGRAPHY USING LSB TECHNIQUE

Submitted by:

1. **Altaf Kalappatt (17BCE2235)**
2. **Mohammed Farhaanuddin(17BCI0165)**
3. **Pranav Narayan (17BCE2213)**

Prepared For

IMAGE PROCESSING (CSE4019) – PROJECT COMPONENT

Dr. Nagaraja Rao A
Senior Professor

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING (SCOPE)

Abstract

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other forms of data. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the Internet. For hiding secret information in images, there exists a large variety of stenographic techniques some are more complex than others and all of them have respective strong and weak points. Different applications have different requirements of the steganography technique used. For example, some applications may require absolute invisibility of the secret information, while others require a larger secret message to be hidden. This project intends to give an overview of image steganography, its uses and techniques. For a more secure approach, the project embeds the message using secret key and then sends it to the receiver. The receiver then decrypts the message to get the original one.

1 Introduction to the topic

1.1 Image steganography in a broad scale

Steganography derives from the Greek word 'steganos',meaning covered or secret, and graph (writing or drawing).On the simplest level, steganography is hidden writing, whether it consists of invisible ink on paper. Where cryptography scrambles a message into a code to obscure its meaning, steganography hides the message entirely. These two secret communication technologies can be used separately or together—for example, by first embedding a message, then hiding it in another file for transmission. As the world becomes more anxious about the use of any secret communication, and as regulations are created by governments to limit uses of embed , steganography's role is gaining prominence. What Steganography essentially does is exploit human perception, human senses are not trained to look for files that have information hidden inside of them, although there are programs available that can do what is called 'Steganalysis' (Detecting use of Steganography.) Steganography hide the secrete message within the host data set and presence imperceptible and is to be reliably communicated to a receiver. The host data set is purposely corrupted, but in a covert way, designed to be invisible to an information analysis.

1.2 Where is the hidden data present?

With graphic images, you can remove redundant bits of color from the image and still produce a picture that looks intact to human eye and is difficult to discern from its original. It is in those bits that the program hides its data. A program uses an algorithm, to embed data in an image file, and a password scheme to allow you to retrieve information.
It is also worth noting that the image noises observed are Gaussian Noises as each pixel is being affected independently in its Least Significant Bit (LSB) to its core binary value.

1.3 What is Gaussian Noise?

Gaussian Noise in digital images arise due to acquisition. A typical model of image noise is Gaussian, additive, independent at each pixel, and independent of the signal intensity, caused primarily by Johnson–Nyquist noise (thermal noise), including that which comes from the reset noise of capacitors.

We can observe that at each pixel the value of the RGB is changed in the slightest by changing the LSB making the image that has the embedded image have Gaussian Noise.

1.4 Our aim in this project

1.4.1) Hiding the text message in an image file

1.4.2) The decoding of the message, decryption an source message retrieval are also supported

2 Objective

The goal of steganography is covert communication. So, a fundamental requirement of this steganography system is that the hider message carried by stego-media should not be sensible to human beings.
The other goal of steganography is to avoid drawing suspicion to the existence of a hidden message. This approach of information hiding technique has recently become important in a number of application area
This project has following objectives:

- To product security tool based on steganography techniques.
- To explore techniques of hiding data using embed module of this project
- To extract techniques of getting secret data using decryption module.

Steganography sometimes is used when embed is not permitted. Or, more commonly, steganography is used to supplement embed . An embedded file may still hide information using steganography, so even if the embedded file is deciphered, the hidden message is not seen.

---

3 Module Description

(i) Home Module

This page will have the home display of the software and it will have the link to other modules also. Once the module is open it can then link to the database and retrieve the settings from the database. This module is also responsible to apply the settings on the other modules. This module will have only database reading capabilities and not writing.

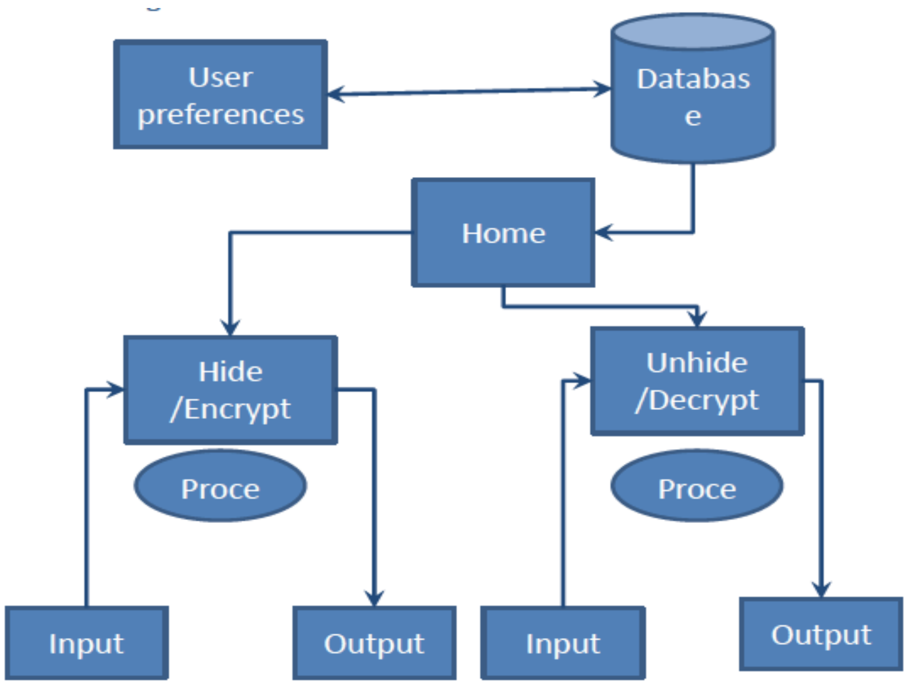
(ii) Hide and Embed data

CREATE A FILE USING TEXT EDITOR-: This module will be used to create a file that will contain the Plaint text with no formatting. Also the file created under this module will not be saved in the system and can and will only be saved as an Embedded (Visual embed) or will be concealed under an image file.

IMPORT FILE-: In this module we will be able to import a file from the computer and that will be concealed under an image file.

(iii) Unhide and Decrypt data

IMPORT FILE-: This module will have the ability to import file(s). It will be able to decrypt, Unhide the file from the image file and it will also be able to retrieve the message from the image files generated by the embed module using the Visual Cryptography.



Method of Data flow

5 Software Support and Applications Required

Operating System Required-:

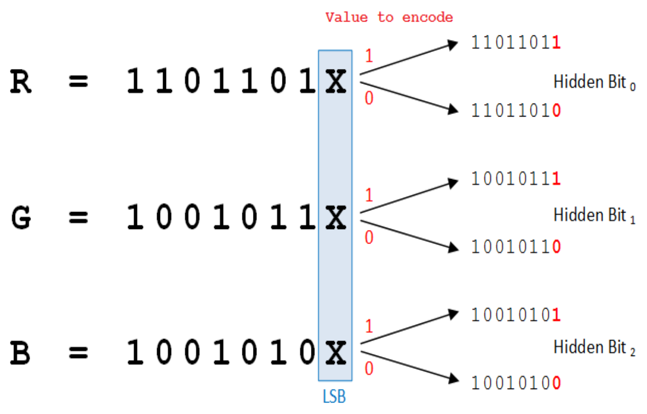
Windows or any Linux platform

JDK(1.7)-:The Java Development Kit (JDK) is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation

Swing-: Swing, which is an extension library to the AWT, includes new & improved components that enhance the look and functionality of GUIs. It employs model/view design architecture. Swing is more portable and more flexible than AWT. Swing is built on top of AWT and is entirely written in Java, using AWT's lightweight component support.

6 LSB algorithm used in the implementation of IS

Structure of image files is that an image is created from pixels that any pixel created from three colors (Red, green and blue said RGB) each color of a pixel is one byte information that shows the density of that color. Merging these three color makes every color that we see in these pictures. We know that every byte in computer science is created from 8 bit that first bit is Most-Significant-Bit (MSB) and last bit Least-Significant-Bit (LSB), the idea of using Steganography science is in this place; we use LSB bit for writing our security information inside pictures. So if we just use last layer (8st layer) of information, we should change the last bit of pixels, in other hands we have 3 bits in each pixel so we have 3*height*width bits memory to write our information. But before writing our data we must write name of data(file), size of name of data & size of data. We can do this by assigning some first bits of memory (8st layer).



7 Implementation Of Image Steganography

7.1)Technical details of Implementation

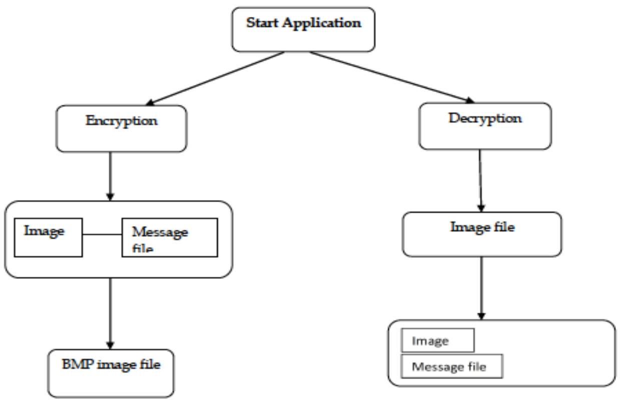
o Using java.awt.Image , ImageIO o The package contains all the necessary classes and methods along with interfaces that are necessary for the manipulation of the images.

7.2) Basic Encoding Algorithm

The steganography technique used is LSB coding.
- The offset of the image is retrieved from its header.
- That offset is left as it is to preserve the integrity of the header, and from the next byte, we start our encoding process.
- For encoding, we first take the input carrier file i.e. an image file and then direct the user to the selection of the text file.
- Creation of User Space:
    - o User Space is created for preserving the original file, so that all the modifications are done in the user space.
    - o In the object of Buffered Image, using ImageIO.readmethod we take the original image.
    - o Using createGraphics and drawRenderedImage method of Graphics class, we create our user space in BufferedImage object.
- The text file is taken as input and separated in stream of bytes.
- Now, each bit of these bytes is encoded in the LSB of each next pixel.
- And, finally we get the final image that contains the encoded message and it is saved, at the specified path given by user, in PNG format using ImageIO.write method.

7.3) Basic Decoding algorithm

The offset of the image is retrieved from its header.
- Create the user space using the same process as in the Encoding.
- Using getRaster() and getDataBuffer() methods of Writable Raster and DataBufferByte classes. The data of image is taken into byte array.
- Using above byte array, the bit stream of original text file, is retrieved into the another byte array.
- And above byte array is written into the decoded text file, which leads to the original message.



Diagramatic Representation of the process

}

7.4)Important functions of Code used in the program

(i)Embedding the Image-:

```
private void embedMessage(BufferedImage img, String mess) {
int messageLength = mess.length();

int imageWidth = img.getWidth(), imageHeight = img.getHeight(),
imageSize = imageWidth * imageHeight;
if(messageLength * 8 + 32 > imageSize) {
JOptionPane.showMessageDialog(this, "Message is too long for the chosen image",
"Message too long!", JOptionPane.ERROR_MESSAGE);
return;
}
embedInteger(img, messageLength, 0, 0);

byte b[] = mess.getBytes();
for(int i=0; i<b.length; i++)
embedByte(img, b[i], i*8+32, 0);
}

private void embedInteger(BufferedImage img, int n, int start, int storageBit) {
int maxX = img.getWidth(), maxY = img.getHeight(),
startX = start/maxY, startY = start - startX*maxY, count=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(n, count);
rgb = setBitValue(rgb, storageBit, bit);
img.setRGB(i, j, rgb);
count++;
}
}
}
```

```
private void embedByte(BufferedImage img, byte b, int start, int storageBit) {
int maxX = img.getWidth(), maxY = img.getHeight(),
startX = start/maxY, startY = start - startX*maxY, count=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(b, count);
rgb = setBitValue(rgb, storageBit, bit);
img.setRGB(i, j, rgb);
count++;
}
}
}
private int getBitValue(int n, int location) {
   int v = n & (int) Math.round(Math.pow(2, location));
   return v==0?0:1;
   }

 private int setBitValue(int n, int location, int bit) {
   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;
  }
public static void main(String arg[]) {
   new EmbedMessage();
   }
 }
```

(ii)Decoding Code

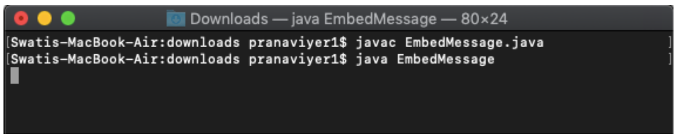
```
private void decodeMessage() {
   int len = extractInteger(image, 0, 0);
   byte b[] = new byte[len];
   for(int i=0; i<len; i++)
      b[i] = extractByte(image, i*8+32, 0);
   message.setText(new String(b));
   }

 private int extractInteger(BufferedImage img, int start, int storageBit) {
   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++;
         }
      }
   return length;
   }

 private byte extractByte(BufferedImage img, int start, int storageBit) {
   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++;
        }
      }
    return b;
    }
  private int getBitValue(int n, int location) {
    int v = n & (int) Math.round(Math.pow(2, location));
    return v==0?0:1;
    }

  private int setBitValue(int n, int location, int bit) {
    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;
    }

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

7.5)How to RUN the java program and how to embed and decrypt the image

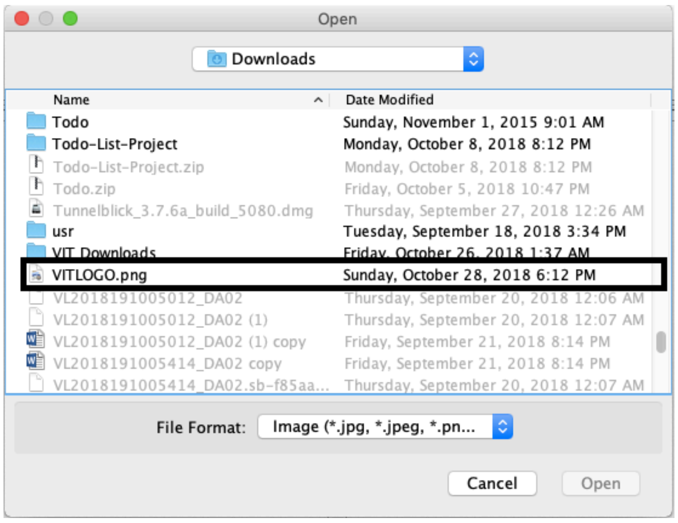
(1) Embed

Step 1-:Run the Java file to embed from terminal



```
Swatis-MacBook-Air:downloads pranaviyer1$ javac EmbedMessage.java
Swatis-MacBook-Air:downloads pranaviyer1$ java EmbedMessage
```

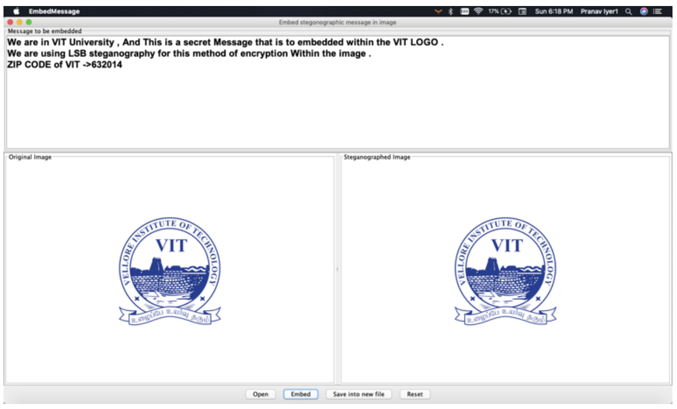
Step 2-:Wait for the Java UI window to open , Click on open to choose the image on which you want to embed the hidden message



Name	Date Modified
Todo	Sunday, November 1, 2015 9:01 AM
Todo-List-Project	Monday, October 8, 2018 8:12 PM
Todo-List-Project.zip	Monday, October 8, 2018 8:12 PM
Todo.zip	Friday, October 5, 2018 10:47 PM
Tunnelblick_3.7.6a_build_5080.dmg	Thursday, September 27, 2018 12:26 AM
usr	Tuesday, September 18, 2018 3:34 PM
VIT Downloads	Friday, October 26, 2018 1:37 AM
VITLOGO.png	Sunday, October 28, 2018 6:12 PM
VL2018191005012_DA02	Thursday, September 20, 2018 12:06 AM
VL2018191005012_DA02 (1)	Thursday, September 20, 2018 12:07 AM
VL2018191005012_DA02 (1) copy	Friday, September 21, 2018 8:14 PM
VL2018191005414_DA02 copy	Friday, September 21, 2018 8:14 PM
VL2018191005414_DA02.sb-f85aa...	Thursday, September 20, 2018 12:07 AM

File Format: Image (*.jpg, *.jpeg, *.pn...

Step 3-:Type the message in the message box or load text file content onto the file , and click on Embed Message .

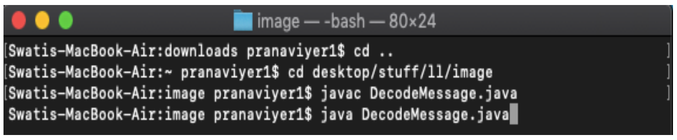


We are in VIT University , And This is a secret Message that is to embedded within the VIT LOGO .
We are using LSB steganography for this method of encryption Within the image .
ZIP CODE of VIT ->632014

Step 4-:Save the Steganographed image onto the directory

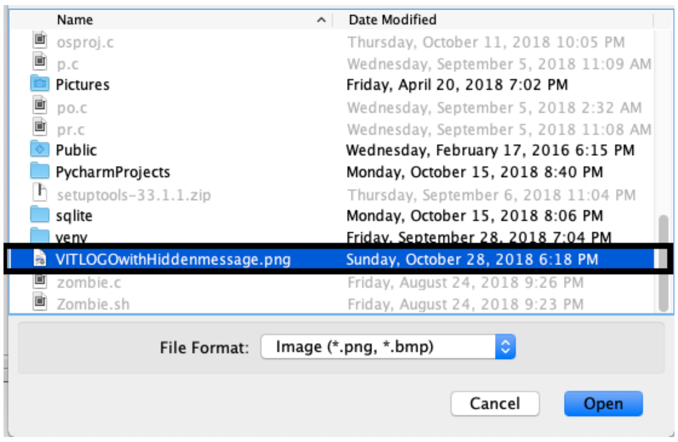
(II)Decryption of the image

Step 1-:Start the java file for De-embed in terminal



```
Swatis-MacBook-Air:downloads pranaviyer1$ cd ..
Swatis-MacBook-Air:~ pranaviyer1$ cd desktop/stuff/ll/image
Swatis-MacBook-Air:image pranaviyer1$ javac DecodeMessage.java
Swatis-MacBook-Air:image pranaviyer1$ java DecodeMessage.java
```

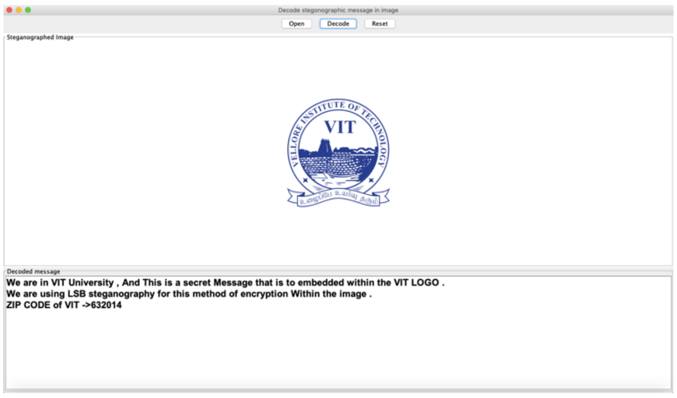
Step 2-:Open the File saved with the hidden message in the embed part of the program



Name	Date Modified
osproj.c	Thursday, October 11, 2018 10:05 PM
p.c	Wednesday, September 5, 2018 11:09 AM
Pictures	Friday, April 20, 2018 7:02 PM
po.c	Wednesday, September 5, 2018 2:32 AM
pr.c	Wednesday, September 5, 2018 11:08 AM
Public	Wednesday, February 17, 2016 6:15 PM
PycharmProjects	Monday, October 15, 2018 8:40 PM
setuptools-33.1.1.zip	Thursday, September 6, 2018 11:04 PM
sqlite	Monday, October 15, 2018 8:06 PM
venv	Friday, September 28, 2018 7:04 PM
VITLOGOwithHiddenmessage.png	Sunday, October 28, 2018 6:18 PM
zombie.c	Friday, August 24, 2018 9:26 PM
Zombie.sh	Friday, August 24, 2018 9:23 PM

File Format: Image (*.png, *.bmp)

Step 3-:Click on Decode to retrieve the hidden message within the Image



Decode stegonographic message in image
Open
Decode
Reset
Steganographed Image
VIT
VELLORE INSTITUTE OF TECHNOLOGY
Decoded message

```
We are in VIT University , And This is a secret Message that is to embedded within the VIT LOGO .
We are using LSB steganography for this method of encryption Within the image .
ZIP CODE of VIT ->632014
```

8 Conclusions

Steganography can be used for hidden communication. We have explored the limits of steganography theory and practice. We printed out the enhancement of the image steganography system using LSB approach to provide a means of secure communication. A stego-key has been applied to the system during embedment of the message into the cover image. This steganography application software provided for the purpose to how to use any type of image formats to hiding any type of files inside them. The master work of this application is in supporting any type of pictures without need to convert to bitmap, and lower limitation on file size to hide, because of using maximum memory space in pictures to hide the file. Steganography, like cryptography, will play an increasing role in the future of secure communication in the “digital world”.

9 References

[1]http://ijact.org/volume3issue4/IJ0340004.pdf

[2]http://shodhganga.inflibnet.ac.in/bitstream/10603/41637/10/10_chapter%203.pdf

[3]https://ieeexplore.ieee.org/document/1259249/

[4]https://ieeexplore.ieee.org/document/1259249

[5]https://ieeexplore.ieee.org/document/6949808