GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Digital Image Steganography

A Project Report of Capstone Project 2

Submitted by

DIVYANSH SINGH
(1613101263/ 16SCSE101346)

in partial fulfillment for the award of

the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SCIENCE AND ENGINEERING

Under the Supervision of

Mr. Dinesh Kumar Baghel (M. Tech)
Assistant Professor

APRIL / MAY- 2020

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "**DIGITAL IMAGE STEGANOGRAPHY**" is the bonafide work of "**DIVYANSH SINGH (1613101263)**" who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. Munish Shabarwal, PhD (Management),
PhD (CS)
Professor and Dean
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Mr. Dinesh Kumar Baghel,
M. Tech
Assistant Professor
School of Computing Science & Engineering

ABSTRACT

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other digital media. 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 steganography techniques, some are more complex than others and all of them have respective strong and weak points. Different applications may require absolute invisibility of the secret information, while others require a large secret message to be hidden. This project intends to give an overview of image steganography, its uses, and techniques. It also attempts to identify the requirements of a good steganography algorithm and briefly reflects on which steganography techniques are more suitable for which applications. One of the reasons that intruders can be successful is most of the information they acquire from a system is in a form that they can read and comprehend. Intruders may reveal the information to others, modify it to misrepresent an individual or organization or use it to launch an attack. One solution to this problem is, through the use of steganography. In contrast to cryptography, it is not to keep others from knowing the hidden information, but it is to keep others from thinking that the information even exists. Steganography includes an array of secret communication methods that hide the message from being seen or discovered.

LIST OF FIGURES

TABLE OF CONTENTS

1. Introduction

Due to advances in ICT, most of the information is kept electronically. Consequently, the security of information has become a fundamental issue. Besides cryptography, steganography can be employed to secure information. In cryptography, the message or encrypted message is embedded in a digital host before passing it through the network, thus the existence of the message is unknown. Besides hiding data for confidentiality, this approach of information hiding can be extended to copyright protection for digital media: audio, video, and images.

The growing possibilities of modern communications need special means of security, especially on computer network. Network security is becoming more important as the data being exchanged on the internet increases. Therefore, confidentiality and data integrity are required to protect against unauthorized access and use. This has resulted in an explosive growth in the field of information hiding.

1.1. Overall Description

Information hiding is an emerging research area, which encompasses applications such as copyright protection for digital media, watermarking, fingerprinting, and steganography. In watermarking applications, the message contains information such as owner identification and a digital time stamp, which usually applied for copyright protection. Fingerprint, the owner of the data set embeds a serial number that uniquely identifies the user of the data set. This adds to copyright information to makes it possible to trace any unauthorized use of the data set back to the user. Steganography hides 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 covertly, designed to be invisible to information analysis.

Software Requirements:

- .NET Framework 4.5 and above

Hardware Requirements:

- Processor: Preferably 1.0 GHz or Greater
- RAM: 512 MB or Greater

1.2. Purpose

Steganography is the practice of hiding private or sensitive information within something that appears to be nothing out to the usual. This technique is chosen because this system includes not only imperceptibility but also undetectability by any steganalysis tool. The art of detecting Steganography is referred to as Steganalysis.

Steganography is often confused with cryptography because the two are similar in the way that they both are used to protect important information. The difference between the two is that steganography involves hiding information, so it appears that no information is hidden at all. If a person views the object that the information is hidden inside of he or she will have no idea that there is any hidden information, therefore the person will not attempt to decrypt the information.

What steganography essentially does is exploit human perception, human senses are not trained to look for files that have information inside of them, although this software is available that can do what is called Steganography. The most common use of steganography is to hide a file inside another file.

1.3. Motivations and Scope

This project is developed for hiding information in any image file. The scope of the project is the implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save Image and extruded file.

User needs to run the application. The user has two-tab options- encrypt and decrypt. If the user select encrypt, the application gives the screen to select the image file, information file, and option to save the image file. If the user select decrypt, the application gives the screen to select only the image file and ask the path where the user wants to save the secret file.

This project has two methods – Encrypt and Decrypt.

In encryption, the secret information is hiding in with any type of image file. Decryption is getting the secret information from the image file.

2. Literature Review

The goal of steganography is to avoid drawing suspicion to the transmission of a hidden message. The recent growth in computational power and technology has propelled it to the forefront of today's security techniques of contemporary steganography techniques for an image in spatial and transform domains and steganalysis techniques for the detection of secret message in the image, authors explore the steganography, its history, features, tools and various techniques like LSB, masking, filtering and other transformations used for hiding messages in an image. It describes various methods to hide the secret or confidential message in an original file so that it is unintelligible to an interceptor, addressed the concept of embedding the secret message into an image using LSB technique, and then applying the AES algorithm to provide better security. In a paper published, a user enters a username, password, and a key. A key is taken from an automatic

key generator device which generates a unique key after some specific time. After this, the secret message and key are encrypted, and encrypted message is embedded into the cover image and stego-image is produced. In another paper, the secret message is first compressed then the message is hashed and encrypted using an encryption key. This method results in a robust model and achieves two important principles of security i.e. privacy and authenticity. A. Joseph Raphael introduces basic terminologies of cryptography and steganography and ensures that the combination of both gives multiple layers of security and will achieve requirements like capacity, security, and robustness. In another research, the user enters a username, password to login to the system. After a successful login, the user can embed a secret message into an image using a key and produces a stego image. The same key is used at the receiver site for retrieving the hidden data. Here the secret message is transferred into a text file first. Then the text file is compressed into a zip file. Then the zip text file is used for converting it into binary codes. Zipping the text file is more secured and is hard to detect. In another research, it presents a method for encrypting and decrypting a secret file that embeds into an image file using the random LSB insertion method in which bits of the secret message is spread into image bits randomly. These random numbers are generated by using a key.

3. Problem Statement

The former consists of linguistic or language forms of hidden writing. The later, such as invisible ink, try to hide messages physically. One disadvantage of linguistic steganography is that users must equip themselves to have a good knowledge of linguistry. In recent years, everything is trending toward digitization. And with the development of internet technology, digital media can be transmitted conveniently over the network. Therefore, messages can be secretly carried by digital media by using steganography techniques and then be transmitted through the internet rapidly. So, steganography is used to make the information hiding simpler and user friendly.

4. Proposed System

The word steganography comes from the Greek “Steganos”, which means covered or secret and– “graphy” means writing or drawing. Therefore, steganography means, literally, covered writing. It is the art and science of hiding information such its presence cannot be detected, and communication is happening. Secret information is encoded in a manner such that the very existence of the information is concealed. Paired with existing communication methods, steganography can be used to carry out hidden exchanges. There has been a rapid growth of interest in steganography for two reasons-

The publishing and broadcasting industries have become interested in techniques for hiding encrypted copyright marks and serial numbers in digital films, audio recordings, books, and multimedia products. Moves by various governments to restrict the availability of encryption services have motivated people to study methods by which private messages can be embedded in seemingly innocuous cover messages.

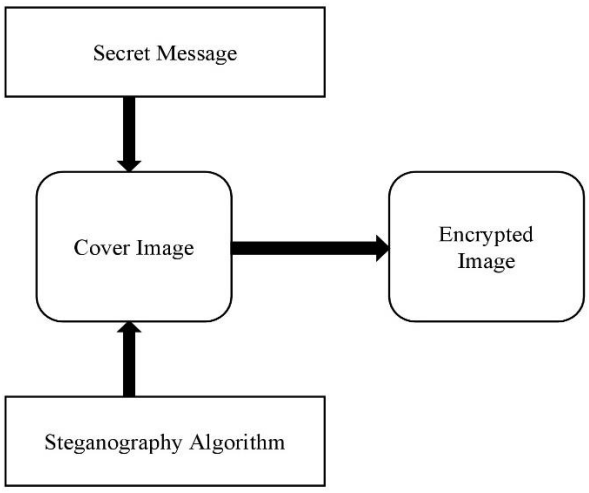

Figure 4.1 Principle of Steganography

The main goal of this project is to communicate securely in a completely undetectable manner and to avoid drawing suspicion to the transmission of a hider data. The basic model of steganography consists of a carrier, message, or password. Carrier is also known as cover-object, in which the message is embedded and serves to hide the presence of the message.

The message is the data that the sender wishes to remain confidential. It can be plain text, ciphertext, another image, or anything that can be embedded in a bitstream such as a copyright mark, a covert communication, or a serial number. The password is known as stego-key, which ensures that only the recipient who knows the corresponding decoding key will be able to extract the message from a cover-object. The cover-object with the secretly embedded message is then called the Stego-object.

Recovering the message from a stego-object requires the cover-object itself and a corresponding decoding key if a stego-key was used during the encoding process. The original image may or may not be required in most applications to extract the message.

There are several suitable carriers below to be the cover-object:

- Network protocols such as TCP, IP, and UDP.
- Audio that using digital audio formats such as WAV, MIDI, AVI, MPEG, MPI, and VOC.
- File and Disk that can hide and append files by using the slack space.
- Text such as null characters, just like Morse code including HTML and Java.
- Images file such as BMP, GIF, and JPG, where they can be both color and gray-scale.

In general, the information hiding process extracts redundant bits from the cover-object. The process consists of two steps-

- Identification of redundant bits in a cover-object. Redundant bits are those bits that can be modified without corrupting the quality or destroying the integrity of the cover-object.

- Embedding process then selects the subset of the redundant bits to be replaced with data from a secret message. The stego-object is created by replacing the selected redundant bits with message bits.

5. Steganography Techniques

Digital images are the most preferred and widely used cover objects for steganography, due to their availability and wide-ranging application. The steganography algorithm varies for different file formats.

An image is a collection of bytes, known as a pixel. Pixel, also known as pel, or picture element is the smallest element of an image. Each pixel represents any one value. The number of distinct colors that can be represented by a pixel depends on the number of bits per pixel (bpp). A 1 bpp image uses 1-bit for each pixel, so each pixel can be either on or off. Each additional bit doubles the number of colors available. So, a 2 bpp image can have 4 colors, and a 3 bpp image can have 8 colors:

1 bpp, 2^1 = 2 colors (Monochrome)

2 bpp, 2^2 = 4 colors

3 bpp, 2^3 = 8 colors

…

8 bpp, 2^8 = 256 colors

16 bpp, 2^{16} = 65,536 colors

24 bpp, 2^{24} = 16,777,216 colors

8 bpp and 24 bpp are the most typical carrier image formats. Both have their advantages and disadvantages. 8 bpp images are used due to their relatively small size but only 256 color combinations are available in total, which can prove to be a potential problem during encoding.

24 bpp images provide a lot more flexibility when used for steganography. More amount of secret data can be embedded into 24 bpp images in comparison to the 8 bpp images. The limitation of 24 bpp images is their high file size. More the number of colors (beyond 16-million), more it makes difficult for the human visual system (HVS) to detect the difference between the original image and the encoded image. Determining the type of message and the type of image, different algorithms are used.

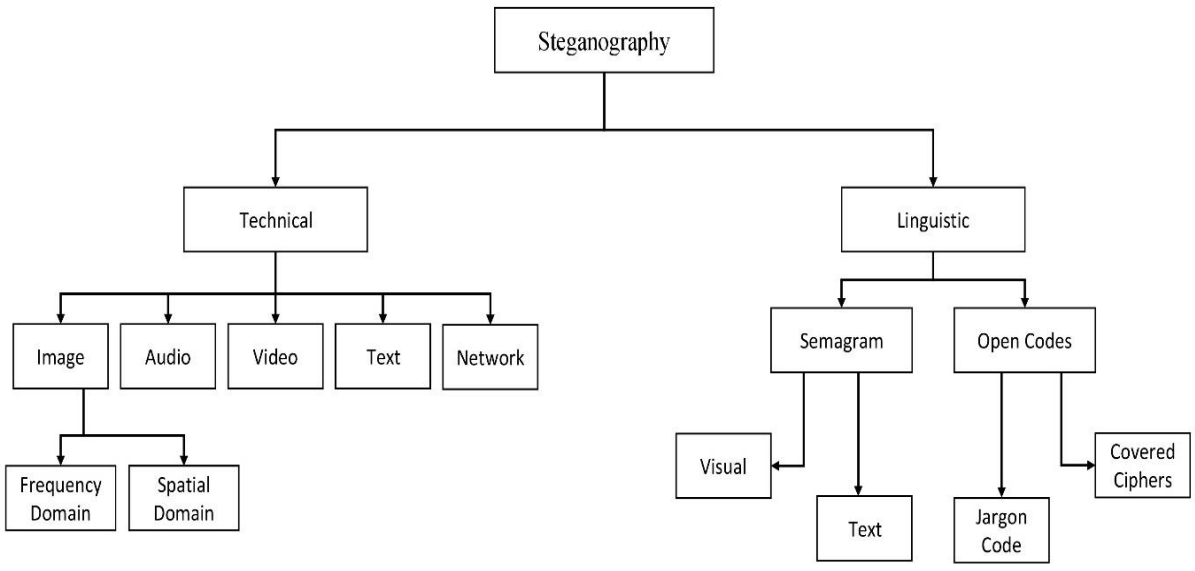

Figure 5.1 Types of Steganography

5.1. Least Significant Bit Substitution

Least Significant Bit (LSB) substitution is one of the oldest and widely known steganography algorithms used for images. LSB replacement is the simplest technique in LSB steganography. As the name suggests, it involves the tempering of the LSB data value of the pixels. The message is stored in the LSB value of the pixels, which does not show any kind of distortion in the encrypted image.

Consider an 8 bpp grayscale bitmap image where an individual pixel is stored as a byte representing a grayscale value. Suppose the first eight pixels of the original image has the following grayscale values:

00010010	11111000
11010101	10100111
01100001	11000100
01011000	01111011

To hide the letter ‘D’ whose binary value is 01000100, we would replace the LSB’s of the original image with the value we need to hide. The new grayscale values are:

0001001**0**	1111100**1**
1101010**0**	1010011**0**
0110000**0**	1100010**1**
0101100**0**	0111101**0**

The value of LSB changes but this change is so minimal that it is not visible to the human eye. However, one of its major disadvantages is the small size of data which can be embedded by using only the least significant bits. Moreover, LSB’s can be vulnerable to attacks. LSB

techniques applied to 24 bpp images are tough to detect and more data can be embedded contrary to 8 bpp file format.

5.2. Masking and Filtering

This technique works somewhat similar to the digital watermarking technique. Instead of hiding the data behind the image, it is appended on such a space which is safe and secure from attackers. The information is hidden in a manner like watermarks on an actual paper. Masking the image changes the image. Therefore, to make sure the changes cannot be detected, it is recommended to make changes in multiple small portions of the image.

The masking and filtering technique is more robust than the LSB technique. Image compression is performed because data is embedded on a secured surface and visible to everyone. If the image is copied, then the information is also carried in the copy. Watermarks are integrated into the image; therefore, this technique can be applied without the fear of image destruction. This technique is mostly used for 24 bpp and grayscale images.

5.3. Redundant Pattern Encoding

Redundant Pattern Encoding (RPE) technique to a certain degree is similar to the spread spectrum technique. The message is dispersed throughout the cover image with the help of an algorithm. However, using this technique image cropping and rotation cannot be achieved.

Multiple small cover images with redundancy increase the chance of recovering even when the stego-image is altered.

5.4. Distortion Technique

In this technique, the secret message is embedded by distorting the cover image. The receiver then decodes the encrypted image using an algorithm. A sequence of alterations is performed in the cover image. Then this sequence is applied to compare the encrypted message with the forwarded message. The decoder computes the differences between the original and the encrypted image to detect the sequence of alterations to recover the encrypted message. The data is encrypted behind random pixels.

For the distortion technique, it is advised that a cover image should only be used once, or else it would be easy for intruders to attack and access the encrypted data.

5.5. Statistical Technique

Using the statistical technique, the message is encrypted by modifying the statistical properties of the cover image. To send multiple bits, the cover image is split into blocks, each corresponding to a single bit of the message. However, these images are vulnerable to scaling, cropping, and rotating attacks. To tackle these issues, the blocks should be selected based on picture elements, for instance, the faces in a crowd.

5.6. Transform Domain Technique

This technique is more complex to hide secret data in an image as compared to the other techniques. It can be defined as a domain of encrypting techniques for which several algorithms have been suggested. In this technique, the secret data is encrypted into the frequency domain of the cover image, which is much preferable than compared to the technique of embedding the data in the time domain.

The benefit of this technique is that there is no need for data compression and the images are safe from attackers. It is worth saying that most of the strong steganographic systems today operate within the transform domain. Different methods of transforms used in steganography are–

1. Fourier Transform (FT)
    a. Discrete Fourier Transform (DFT)
    b. Short Term Fourier Transform
2. Discrete Cosine Transform (DCT)
3. Wavelet Transform
    a. Continuous Wavelet Transform
    b. Discrete Wavelet Transform (DWT)
4. Graph Wavelet Transform

6. Implementation

The algorithm used for Encryption and Decryption in this application provides using several layers lieu of using only the LSB layer of the image. Writing data starts from the last layer (8th or LSB layer); because the significance of this layer is least, and every upper layer has double significance from its below layer. So, every step we go to upper-layer image quality decreases and image retouching transpires.

The encrypt module is used to hide information into the image; no one can see that information or file. This module requires any type of image and message and gives the only one image file in the destination.

The decrypting module is used to get the hidden information in an image file. It takes the image file as output and gives two files at the destination folder, one is the same image file, and another is the hidden message file.

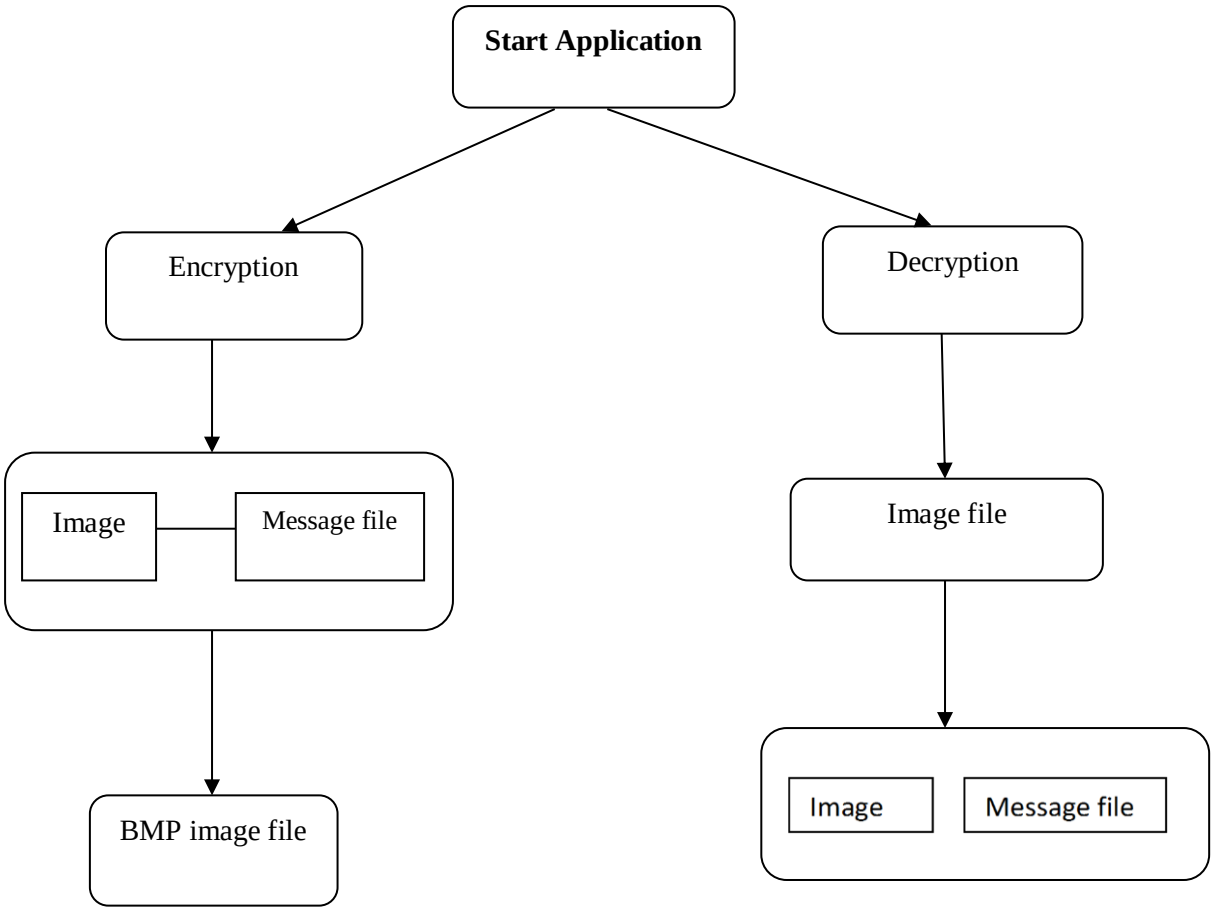

Figure 6.1 Graphical Representation of LSB technique

Encryption Process:

+

Image File

Secret Message

Stego-Image

Decryption Process:

Stego-Image

Secret Message

Image File

6.1. Code Analysis

```
using System;

using System.Drawing;

using System.Windows.Forms;

using System.IO;

namespace Text2Image

{

        public partial class FrmSteganography : Form

        {

                public FrmSteganography()

                {

                        InitializeComponent();

                }

// public values:

                string loadedTrueImagePath, loadedFilePath,

                saveToImage,DLoadImagePath,DSaveFilePath;

                int height, width;
```

```
long fileSize, fileNameSize;

Image loadedTrueImage, DecryptedImage ,AfterEncryption;

Bitmap loadedTrueBitmap, DecryptedBitmap;

Rectangle previewImage = new Rectangle(20,160,490,470);

bool canPaint = false, EncriptionDone = false;

byte[] fileContainer;

private void EnImageBrowse_btn_Click(object sender, EventArgs e)

{

        if (openFileDialog1.ShowDialog() == DialogResult.OK)

        {

                loadedTrueImagePath = openFileDialog1.FileName;

                EnImage_tbx.Text = loadedTrueImagePath;

                loadedTrueImage = Image.FromFile(loadedTrueImagePath);

                height = loadedTrueImage.Height;

                width = loadedTrueImage.Width;

                loadedTrueBitmap = new Bitmap(loadedTrueImage);

                FileInfo imginf = new FileInfo(loadedTrueImagePath);
```

```
                float fs = (float)imginf.Length / 1024;

                ImageSize_lbl.Text = smalldecimal(fs.ToString(), 2) + " KB";

                ImageHeight_lbl.Text = loadedTrueImage.Height.ToString() + "
                Pixel";

                ImageWidth_lbl.Text = loadedTrueImage.Width.ToString() + "
                Pixel";

                double cansave = (8.0 * ((height * (width / 3) * 3) / 3 - 1)) / 1024;

                CanSave_lbl.Text = smalldecimal(cansave.ToString(), 2) + "
                KB";

                canPaint = true;

                this.Invalidate();

            }

}

private string smalldecimal(string inp, int dec)

{

            int i;

            for (i = inp.Length - 1; i > 0; i--)

            if (inp[i] == '.')
```

```
            break;
            try
            {
                return inp.Substring(0, i + dec + 1);
            }
            catch
            {
                return inp;
            }
        }
        private void EnFileBrowse_btn_Click(object sender, EventArgs e)
        {
            if (openFileDialog2.ShowDialog() == DialogResult.OK)
                {
                    loadedFilePath = openFileDialog2.FileName;
                    EnFile_tbx.Text = loadedFilePath;
                    FileInfo finfo = new FileInfo(loadedFilePath);
```

```
                fileSize = finfo.Length;

                fileNameSize = justFName(loadedFilePath).Length;

            }

}

private void Encrypt_btn_Click(object sender, EventArgs e)

{

    if (saveFileDialog1.ShowDialog() == DialogResult.OK)

    {

        saveToImage = saveFileDialog1.FileName;

    }

    else

    return;

    if (EnImage_tbx.Text == String.Empty || EnFile_tbx.Text ==
    String.Empty)

    {

        MessageBox.Show("Encrypton information is
        incomplete!\nPlease complete them first.", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        }

        if (8*((height * (width/3)*3)/3 - 1) < fileSize + fileNameSize)

        {

            MessageBox.Show("File size is too large!\nPlease use a larger
            image to hide this file.", "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error);

            return;

        }

        fileContainer = File.ReadAllBytes(loadedFilePath);

        EncryptLayer();

}

private void EncryptLayer()

{

        toolStripStatusLabel1.Text ="Encrypting... Please wait";

        Application.DoEvents();

        long FSize = fileSize;

        Bitmap changedBitmap = EncryptLayer(8, loadedTrueBitmap, 0, (height
        * (width/3)*3) / 3 - fileNameSize - 1, true);
```

```
 FSize -= (height * (width / 3) * 3) / 3 - fileNameSize - 1;

if(FSize > 0)

{

        for (int i = 7; i >= 0 && FSize > 0; i--)

        {

                changedBitmap = EncryptLayer(i, changedBitmap, (((8 -

                i) * height * (width / 3) * 3) / 3 - fileNameSize - (8 - i)),

                (((9 - i) * height * (width / 3) * 3) / 3 - fileNameSize - (9 -

                i)), false);

                FSize -= (height * (width / 3) * 3) / 3 - 1;

        }

}

changedBitmap.Save(saveToImage);

toolStripStatusLabel1.Text = "The image has been encrypted

successfully.";

EncriptionDone = true;

AfterEncryption = Image.FromFile(saveToImage);

this.Invalidate();
```

```
}

private Bitmap EncryptLayer(int layer, Bitmap inputBitmap, long startPosition,
long endPosition, bool writeFileName)

{

        Bitmap outputBitmap = inputBitmap;

        layer--;

        int i = 0, j = 0;

        long FNSize = 0;

        bool[] t = new bool[8];

        bool[] rb = new bool[8];

        bool[] gb = new bool[8];

        bool[] bb = new bool[8];

        Color pixel = new Color();

        byte r, g, b;

        if (writeFileName)

        {

                FNSize = fileNameSize;
```

```
                string fileName = justFName(loadedFilePath);

// write filename:

                for (i = 0; i < height && i * (height / 3) < fileNameSize; i++)

                for (j = 0; j < (width / 3) * 3 && i * (height / 3) + (j / 3) <
                fileNameSize; j++)

                {

                        byte2bool((byte)fileName[i * (height / 3) + j / 3], ref t);

                        pixel = inputBitmap.GetPixel(j, i);

                        r = pixel.R;

                        g = pixel.G;

                        b = pixel.B;

                        byte2bool(r, ref rb);

                        byte2bool(g, ref gb);

                        byte2bool(b, ref bb);

                        if (j % 3 == 0)

                        {

                                rb[7] = t[0];
```

```
            gb[7] = t[1];

            bb[7] = t[2];

        }

        else if (j % 3 == 1)

        {

            rb[7] = t[3];

            gb[7] = t[4];

            bb[7] = t[5];

        }

        else

        {

            rb[7] = t[6];

            gb[7] = t[7];

        }

        Color result = Color.FromArgb((int)bool2byte(rb),

        (int)bool2byte(gb), (int)bool2byte(bb));

        outputBitmap.SetPixel(j, i, result);
```

```
                }

                i--;

            }

// write file (after file name):

            int tempj = j;

            for (; i < height && i * (height / 3) < endPosition - startPosition + FNSize
            && startPosition + i * (height / 3) < fileSize + FNSize; i++)

            for (j = 0; j < (width / 3) * 3 && i * (height / 3) + (j / 3) < endPosition -
            startPosition + FNSize && startPosition + i * (height / 3) + (j / 3) <
            fileSize + FNSize; j++)

            {

                if (tempj != 0)

                {

                    j = tempj;

                    tempj = 0;

                }

                byte2bool((byte)fileContainer[startPosition + i * (height / 3) + j /
                3 - FNSize], ref t);
```

```
pixel = inputBitmap.GetPixel(j, i);

r = pixel.R;

g = pixel.G;

b = pixel.B;

byte2bool(r, ref rb);

byte2bool(g, ref gb);

byte2bool(b, ref bb);

if (j % 3 == 0)

{

        rb[layer] = t[0];

        gb[layer] = t[1];

        bb[layer] = t[2];

}

else if (j % 3 == 1)

{

        rb[layer] = t[3];

        gb[layer] = t[4];
```

```
                bb[layer] = t[5];
            }
            else
            {
                rb[layer] = t[6];
                gb[layer] = t[7];
            }
            Color result = Color.FromArgb((int)bool2byte(rb),
            (int)bool2byte(gb), (int)bool2byte(bb));
            outputBitmap.SetPixel(j, i, result);
        }
        long tempFS = fileSize, tempFNS = fileNameSize;
        r = (byte)(tempFS % 100);
        tempFS /= 100;
        g = (byte)(tempFS % 100);
        tempFS /= 100;
        b = (byte)(tempFS % 100);
```

```
            Color flenColor = Color.FromArgb(r,g,b);

            outputBitmap.SetPixel(width - 1, height - 1, flenColor);

            r = (byte)(tempFNS % 100);

            tempFNS /= 100;

            g = (byte)(tempFNS % 100);

            tempFNS /= 100;

            b = (byte)(tempFNS % 100);

            Color fnlenColor = Color.FromArgb(r,g,b);

            outputBitmap.SetPixel(width - 2, height - 1, fnlenColor);

            return outputBitmap;

}

private void DecryptLayer()

{

            toolStripStatusLabel1.Text = "Decrypting... Please wait";

            Application.DoEvents();

            int i, j = 0;

            bool[] t = new bool[8];
```

```
                    bool[] rb = new bool[8];

                    bool[] gb = new bool[8];

                    bool[] bb = new bool[8];

                    Color pixel = new Color();

                    byte r, g, b;

                    pixel = DecryptedBitmap.GetPixel(width - 1, height - 1);

                    long fSize = pixel.R + pixel.G * 100 + pixel.B * 10000;

                    pixel = DecryptedBitmap.GetPixel(width - 2, height - 1);

                    long fNameSize = pixel.R + pixel.G * 100 + pixel.B * 10000;

                    byte[] res = new byte[fSize];

                    string resFName = "";

                    byte temp;

// read file name:

                    for (i = 0; i < height && i * (height / 3) < fNameSize; i++)

                    for (j = 0; j < (width / 3) * 3 && i * (height / 3) + (j / 3) < fNameSize;
                    j++)

                    {
```

```
pixel = DecryptedBitmap.GetPixel(j, i);

r = pixel.R;

g = pixel.G;

b = pixel.B;

byte2bool(r, ref rb);

byte2bool(g, ref gb);

byte2bool(b, ref bb);

if (j % 3 == 0)

{

        t[0] = rb[7];

        t[1] = gb[7];

        t[2] = bb[7];

}

else if (j % 3 == 1)

{

        t[3] = rb[7];

        t[4] = gb[7];
```

```
                t[5] = bb[7];

            }

            else

            {

                t[6] = rb[7];

                t[7] = gb[7];

                temp = bool2byte(t);

                resFName += (char)temp;

            }

        }

// read file on layer 8 (after file name):

        int tempj = j;

        i--;

        for (; i < height && i * (height / 3) < fSize + fNameSize; i++)

        for (j = 0; j < (width / 3) * 3 && i * (height / 3) + (j / 3) < (height *

        (width / 3) * 3) / 3 - 1 && i * (height / 3) + (j / 3) < fSize + fNameSize;

        j++)

        {
```

```
if (tempj != 0)

{

        j = tempj;

        tempj = 0;

}

pixel = DecryptedBitmap.GetPixel(j, i);

r = pixel.R;

g = pixel.G;

b = pixel.B;

byte2bool(r, ref rb);

byte2bool(g, ref gb);

byte2bool(b, ref bb);

if (j % 3 == 0)

{

        t[0] = rb[7];

        t[1] = gb[7];

        t[2] = bb[7];
```

```
                }
                else if (j % 3 == 1)
                {
                    t[3] = rb[7];
                    t[4] = gb[7];
                    t[5] = bb[7];
                }
                else
                {
                    t[6] = rb[7];
                    t[7] = gb[7];
                    temp = bool2byte(t);
                    res[i * (height / 3) + j / 3 - fNameSize] = temp;
                }
            }
// read file on other layers:
            long readedOnL8 = (height * (width/3)*3) /3 - fNameSize - 1;
```

```
for (int layer = 6; layer >= 0 && readedOnL8 + (6 - layer) * ((height *
(width / 3) * 3) / 3 - 1) < fSize; layer--)

for (i = 0; i < height && i * (height / 3) + readedOnL8 + (6 - layer) *
((height * (width / 3) * 3) / 3 - 1) < fSize; i++)

for (j = 0; j < (width / 3) * 3 && i * (height / 3) + (j / 3) + readedOnL8 +
(6 - layer) * ((height * (width / 3) * 3) / 3 - 1) < fSize; j++)

{

        pixel = DecryptedBitmap.GetPixel(j, i);

        r = pixel.R;

        g = pixel.G;

        b = pixel.B;

        byte2bool(r, ref rb);

        byte2bool(g, ref gb);

        byte2bool(b, ref bb);

        if (j % 3 == 0)

        {

                t[0] = rb[layer];

                t[1] = gb[layer];
```

```
                t[2] = bb[layer];

            }

            else if (j % 3 == 1)

            {

                t[3] = rb[layer];

                t[4] = gb[layer];

                t[5] = bb[layer];

            }

            else

            {

                t[6] = rb[layer];

                t[7] = gb[layer];

                temp = bool2byte(t);

                res[i * (height / 3) + j / 3 + (6 - layer) * ((height * (width /
                3) * 3) / 3 - 1) + readedOnL8] = temp;

            }

        }
```

```
        if (File.Exists(DSaveFilePath + "\\" + resFName))
        {
            MessageBox.Show("File \"" + resFName + "\" already exist
            please choose another path to save file",
            "Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
            return;
        }
        else
        File.WriteAllBytes(DSaveFilePath + "\\" + resFName, res);
        toolStripStatusLabel1.Text = "Decrypted file has been saved
        successfully.";
        Application.DoEvents();
}
private void byte2bool(byte inp, ref bool[] outp)
{
        if(inp>=0 && inp<=255)
        for (short i = 7; i >= 0; i--)
        {
```

```
                if (inp % 2 == 1)
                outp[i] = true;
                else
                outp[i] = false;
                inp /= 2;
            }
            else
            throw new Exception("Input number is illegal.");
}
private byte bool2byte(bool[] inp)
{
            byte outp = 0;
            for (short i = 7; i >= 0; i--)
            {
                if (inp[i])
                outp += (byte)Math.Pow(2.0, (double)(7-i));
            }
```

```
        return outp;
}
private void Decrypt_btn_Click(object sender, EventArgs e)
{
        if (DeSaveFile_tbx.Text == String.Empty || DeLoadImage_tbx.Text ==
        String.Empty)
        {
                MessageBox.Show("Text boxes must not be empty!", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
        }
        if (System.IO.File.Exists(DeLoadImage_tbx.Text) == false)
        {
                MessageBox.Show("Select image file.", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                DeLoadImage_tbx.Focus();
                return;
        }
```

```
        DecryptLayer();
}
private void DeLoadImageBrowse_btn_Click(object sender, EventArgs e)
{
        if (openFileDialog3.ShowDialog() == DialogResult.OK)
        {
                DLoadImagePath = openFileDialog3.FileName;
                DeLoadImage_tbx.Text = DLoadImagePath;
                DecryptedImage = Image.FromFile(DLoadImagePath);
                height = DecryptedImage.Height;
                width = DecryptedImage.Width;
                DecryptedBitmap = new Bitmap(DecryptedImage);
                FileInfo imginf = new FileInfo(DLoadImagePath);
                float fs = (float)imginf.Length / 1024;
                ImageSize_lbl.Text = smalldecimal(fs.ToString(), 2) + " KB";
                ImageHeight_lbl.Text = DecryptedImage.Height.ToString() + "
                Pixel";
```

```
                ImageWidth_lbl.Text = DecryptedImage.Width.ToString() + "
                Pixel";

                double cansave = (8.0 * ((height * (width / 3) * 3) / 3 - 1)) / 1024;

                CanSave_lbl.Text = smalldecimal(cansave.ToString(), 2) + "
                KB";

                canPaint = true;

                this.Invalidate();

            }

        }

        private void DeSaveFileBrowse_btn_Click(object sender, EventArgs e)

        {

            if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

            {

                DSaveFilePath = folderBrowserDialog1.SelectedPath;

                DeSaveFile_tbx.Text = DSaveFilePath;

            }

        }

        private void Form1_Paint(object sender, PaintEventArgs e)
```

```
{
        if(canPaint)
        try
        {
                if (!EncriptionDone)
                e.Graphics.DrawImage(loadedTrueImage, previewImage);
                else
                e.Graphics.DrawImage(AfterEncryption, previewImage);
        }
        catch
        {
                e.Graphics.DrawImage(DecryptedImage, previewImage);
        }
}
private string justFName(string path)
{
        string output;
```

```
        int i;

        if (path.Length == 3)   // i.e: "C:\\"

        return path.Substring(0, 1);

        for (i = path.Length - 1; i > 0; i--)

        if (path[i] == '\\')

        break;

        output = path.Substring(i + 1);

        return output;

}

private string justEx(string fName)

{

        string output;

        int i;

        for (i = fName.Length - 1; i > 0; i--)

        if (fName[i] == '.')

        break;

        output = fName.Substring(i + 1);
```

```
            return output;

        }

        private void Close_btn_Click(object sender, EventArgs e)

        {

            this.Close();

        }

    }

}
```

```
namespace Text2Image
{
    partial class FrmSteganography
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
        otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
```

```
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.label5 = new System.Windows.Forms.Label();
            this.Decrypt_btn = new System.Windows.Forms.Button();
            this.DeLoadImage_tbx = new System.Windows.Forms.TextBox();
            this.DeSaveFile_tbx = new System.Windows.Forms.TextBox();
            this.DeSaveFileBrowse_btn = new System.Windows.Forms.Button();
```

```
this.label6 = new System.Windows.Forms.Label();

this.DeLoadImageBrowse_btn = new System.Windows.Forms.Button();

this.groupBox2 = new System.Windows.Forms.GroupBox();

this.Encrypt_btn = new System.Windows.Forms.Button();

this.EnFileBrowse_btn = new System.Windows.Forms.Button();

this.EnImageBrowse_btn = new System.Windows.Forms.Button();

this.label2 = new System.Windows.Forms.Label();

this.EnFile_tbx = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.EnImage_tbx = new System.Windows.Forms.TextBox();

this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();

this.openFileDialog2 = new System.Windows.Forms.OpenFileDialog();

this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();

this.openFileDialog3 = new System.Windows.Forms.OpenFileDialog();

this.label3 = new System.Windows.Forms.Label();

this.groupBox3 = new System.Windows.Forms.GroupBox();

this.ByteCapacity_lbl = new System.Windows.Forms.Label();
```

```
this.CanSave_lbl = new System.Windows.Forms.Label();

this.ImageWidth_lbl = new System.Windows.Forms.Label();

this.ImageHeight_lbl = new System.Windows.Forms.Label();

this.ImageSize_lbl = new System.Windows.Forms.Label();

this.label9 = new System.Windows.Forms.Label();

this.label10 = new System.Windows.Forms.Label();

this.label8 = new System.Windows.Forms.Label();

this.label7 = new System.Windows.Forms.Label();

this.folderBrowserDialog1 = new
System.Windows.Forms.FolderBrowserDialog();

this.Close_btn = new System.Windows.Forms.Button();

this.statusStrip1 = new System.Windows.Forms.StatusStrip();

this.toolStripStatusLabel1 = new
System.Windows.Forms.ToolStripStatusLabel();

this.tabControl1 = new System.Windows.Forms.TabControl();

this.tabPage1 = new System.Windows.Forms.TabPage();

this.tabPage2 = new System.Windows.Forms.TabPage();

this.linkLabel1 = new System.Windows.Forms.LinkLabel();
```

```
            this.groupBox1.SuspendLayout();

            this.groupBox2.SuspendLayout();

            this.groupBox3.SuspendLayout();

            this.statusStrip1.SuspendLayout();

            this.tabControl1.SuspendLayout();

            this.tabPage1.SuspendLayout();

            this.tabPage2.SuspendLayout();

            this.SuspendLayout();

// groupBox1

            this.groupBox1.Controls.Add(this.label5);

            this.groupBox1.Controls.Add(this.Decrypt_btn);

            this.groupBox1.Controls.Add(this.DeLoadImage_tbx);

            this.groupBox1.Controls.Add(this.DeSaveFile_tbx);

            this.groupBox1.Controls.Add(this.DeSaveFileBrowse_btn);

            this.groupBox1.Controls.Add(this.label6);

            this.groupBox1.Controls.Add(this.DeLoadImageBrowse_btn);

            this.groupBox1.Font = new System.Drawing.Font("Microsoft Sans
```

```
                    Serif", 8.25F, System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.groupBox1.Location = new System.Drawing.Point(60, 0);
                    this.groupBox1.Name = "groupBox1";
                    this.groupBox1.Size = new System.Drawing.Size(320, 111);
                    this.groupBox1.TabIndex = 0;
                    this.groupBox1.TabStop = false;
// label5
                    this.label5.AutoSize = true;
                    this.label5.Font = new System.Drawing.Font("Arial", 8.25F,
                    System.Drawing.FontStyle.Regular,
                    System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                    this.label5.Location = new System.Drawing.Point(7, 22);
                    this.label5.Name = "label5";
                    this.label5.Size = new System.Drawing.Size(65, 14);
                    this.label5.TabIndex = 1;
                    this.label5.Text = "Load image:";
// Decrypt_btn
```

```
this.Decrypt_btn.Font = new System.Drawing.Font("Arial", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.Decrypt_btn.Location = new System.Drawing.Point(123, 82);

this.Decrypt_btn.Name = "Decrypt_btn";

this.Decrypt_btn.Size = new System.Drawing.Size(75, 23);

this.Decrypt_btn.TabIndex = 3;

this.Decrypt_btn.Text = "Decrypt";

this.Decrypt_btn.UseVisualStyleBackColor = true;

this.Decrypt_btn.Click += new
System.EventHandler(this.Decrypt_btn_Click);

// DeLoadImage_tbx

this.DeLoadImage_tbx.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.Anc
horStyles.Top | System.Windows.Forms.AnchorStyles.Left) |
System.Windows.Forms.AnchorStyles.Right)));

this.DeLoadImage_tbx.Location = new System.Drawing.Point(78, 19);

this.DeLoadImage_tbx.Name = "DeLoadImage_tbx";

this.DeLoadImage_tbx.Size = new System.Drawing.Size(155, 20);
```

```
                this.DeLoadImage_tbx.TabIndex = 0;

// DeSaveFile_tbx

                this.DeSaveFile_tbx.Anchor =
                ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.Anc
                horStyles.Top | System.Windows.Forms.AnchorStyles.Left) |
                System.Windows.Forms.AnchorStyles.Right)));

                this.DeSaveFile_tbx.Location = new System.Drawing.Point(78, 54);

                this.DeSaveFile_tbx.Name = "DeSaveFile_tbx";

                this.DeSaveFile_tbx.Size = new System.Drawing.Size(155, 20);

                this.DeSaveFile_tbx.TabIndex = 2;

// DeSaveFileBrowse_btn

                this.DeSaveFileBrowse_btn.Anchor =
                ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
                orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));

                this.DeSaveFileBrowse_btn.Font = new System.Drawing.Font("Arial",
                8.25F, System.Drawing.FontStyle.Regular,
                System.Drawing.GraphicsUnit.Point, ((byte)(0)));

                this.DeSaveFileBrowse_btn.Location = new System.Drawing.Point(239,
                52);

                this.DeSaveFileBrowse_btn.Name = "DeSaveFileBrowse_btn";
```

```
            this.DeSaveFileBrowse_btn.Size = new System.Drawing.Size(75, 23);

            this.DeSaveFileBrowse_btn.TabIndex = 3;

            this.DeSaveFileBrowse_btn.Text = "Browse";

            this.DeSaveFileBrowse_btn.UseVisualStyleBackColor = true;

            this.DeSaveFileBrowse_btn.Click += new
            System.EventHandler(this.DeSaveFileBrowse_btn_Click);

// label6

            this.label6.AutoSize = true;

            this.label6.Font = new System.Drawing.Font("Arial", 8.25F,
            System.Drawing.FontStyle.Regular,
            System.Drawing.GraphicsUnit.Point, ((byte)(0)));

            this.label6.Location = new System.Drawing.Point(7, 57);

            this.label6.Name = "label6";

            this.label6.Size = new System.Drawing.Size(64, 14);

            this.label6.TabIndex = 1;

            this.label6.Text = "Save file to:";

// DeLoadImageBrowse_btn

            this.DeLoadImageBrowse_btn.Anchor =
```

```
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));

this.DeLoadImageBrowse_btn.Font = new System.Drawing.Font("Arial",
8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.DeLoadImageBrowse_btn.Location = new
System.Drawing.Point(239, 17);

this.DeLoadImageBrowse_btn.Name = "DeLoadImageBrowse_btn";

this.DeLoadImageBrowse_btn.Size = new System.Drawing.Size(75, 23);

this.DeLoadImageBrowse_btn.TabIndex = 1;

this.DeLoadImageBrowse_btn.Text = "Browse";

this.DeLoadImageBrowse_btn.UseVisualStyleBackColor = true;

this.DeLoadImageBrowse_btn.Click += new
System.EventHandler(this.DeLoadImageBrowse_btn_Click);

// groupBox2

this.groupBox2.Controls.Add(this.Encrypt_btn);

this.groupBox2.Controls.Add(this.EnFileBrowse_btn);

this.groupBox2.Controls.Add(this.EnImageBrowse_btn);

this.groupBox2.Controls.Add(this.label2);
```

```
this.groupBox2.Controls.Add(this.EnFile_tbx);

this.groupBox2.Controls.Add(this.label1);

this.groupBox2.Controls.Add(this.EnImage_tbx);

this.groupBox2.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.groupBox2.Location = new System.Drawing.Point(60, 0);

this.groupBox2.Name = "groupBox2";

this.groupBox2.Size = new System.Drawing.Size(320, 111);

this.groupBox2.TabIndex = 0;

this.groupBox2.TabStop = false;

// Encrypt_btn

this.Encrypt_btn.Font = new System.Drawing.Font("Arial", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.Encrypt_btn.Location = new System.Drawing.Point(123, 83);

this.Encrypt_btn.Name = "Encrypt_btn";

this.Encrypt_btn.Size = new System.Drawing.Size(75, 23);
```

```
            this.Encrypt_btn.TabIndex = 3;

            this.Encrypt_btn.Text = "Encrypt";

            this.Encrypt_btn.UseVisualStyleBackColor = true;

            this.Encrypt_btn.Click += new
            System.EventHandler(this.Encrypt_btn_Click);

// EnFileBrowse_btn

            this.EnFileBrowse_btn.Font = new System.Drawing.Font("Arial", 8.25F,
            System.Drawing.FontStyle.Regular,
            System.Drawing.GraphicsUnit.Point, ((byte)(0)));

            this.EnFileBrowse_btn.Location = new System.Drawing.Point(239, 52);

            this.EnFileBrowse_btn.Name = "EnFileBrowse_btn";

            this.EnFileBrowse_btn.Size = new System.Drawing.Size(75, 23);

            this.EnFileBrowse_btn.TabIndex = 3;

            this.EnFileBrowse_btn.Text = "Browse";

            this.EnFileBrowse_btn.UseVisualStyleBackColor = true;

            this.EnFileBrowse_btn.Click += new
            System.EventHandler(this.EnFileBrowse_btn_Click);

// EnImageBrowse_btn
```

```
this.EnImageBrowse_btn.Font = new System.Drawing.Font("Arial",
8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.EnImageBrowse_btn.Location = new System.Drawing.Point(239,
17);

this.EnImageBrowse_btn.Name = "EnImageBrowse_btn";

this.EnImageBrowse_btn.Size = new System.Drawing.Size(75, 23);

this.EnImageBrowse_btn.TabIndex = 1;

this.EnImageBrowse_btn.Text = "Browse";

this.EnImageBrowse_btn.UseVisualStyleBackColor = true;

this.EnImageBrowse_btn.Click += new
System.EventHandler(this.EnImageBrowse_btn_Click);

// label2

this.label2.AutoSize = true;

this.label2.Font = new System.Drawing.Font("Arial", 8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label2.Location = new System.Drawing.Point(6, 57);

this.label2.Name = "label2";
```

```
            this.label2.Size = new System.Drawing.Size(51, 14);

            this.label2.TabIndex = 1;

            this.label2.Text = "Load file:";

// EnFile_tbx

            this.EnFile_tbx.Location = new System.Drawing.Point(77, 54);

            this.EnFile_tbx.Name = "EnFile_tbx";

            this.EnFile_tbx.Size = new System.Drawing.Size(156, 20);

            this.EnFile_tbx.TabIndex = 2;

// label1

            this.label1.AutoSize = true;

            this.label1.Font = new System.Drawing.Font("Arial", 8.25F,
            System.Drawing.FontStyle.Regular,
            System.Drawing.GraphicsUnit.Point, ((byte)(0)));

            this.label1.Location = new System.Drawing.Point(6, 22);

            this.label1.Name = "label1";

            this.label1.Size = new System.Drawing.Size(65, 14);

            this.label1.TabIndex = 1;

            this.label1.Text = "Load image:";
```

```
// EnImage_tbx

                this.EnImage_tbx.Location = new System.Drawing.Point(77, 20);

                this.EnImage_tbx.Name = "EnImage_tbx";

                this.EnImage_tbx.Size = new System.Drawing.Size(156, 20);

                this.EnImage_tbx.TabIndex = 0;

// openFileDialog1

                this.openFileDialog1.Filter = "Bitmap Files (*.bmp)|*.bmp|All
                files(*.*)|*.*";

// openFileDialog2

                this.openFileDialog2.Filter = "All files (*.*)|*.*";

// saveFileDialog1

                this.saveFileDialog1.Filter = "Bitmap Files (*.bmp)|*.bmp";

// openFileDialog3

                this.openFileDialog3.Filter = "Bitmap Files (*.bmp)|*.bmp";

// label3

                this.label3.AutoSize = true;

                this.label3.Font = new System.Drawing.Font("Arial", 8.25F,
                ((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold |
```

```
                System.Drawing.FontStyle.Underline))),
                System.Drawing.GraphicsUnit.Point, ((byte)(0)));
                this.label3.ForeColor = System.Drawing.Color.Black;
                this.label3.Location = new System.Drawing.Point(8, 145);
                this.label3.Name = "label3";
                this.label3.Size = new System.Drawing.Size(92, 14);
                this.label3.TabIndex = 1;
                this.label3.Text = "Image preview:";
// groupBox3
                this.groupBox3.BackColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(233)))),
                ((int)(((byte)(240)))), ((int)(((byte)(221)))));
                this.groupBox3.Controls.Add(this.ByteCapacity_lbl);
                this.groupBox3.Controls.Add(this.CanSave_lbl);
                this.groupBox3.Controls.Add(this.ImageWidth_lbl);
                this.groupBox3.Controls.Add(this.ImageHeight_lbl);
                this.groupBox3.Controls.Add(this.ImageSize_lbl);
                this.groupBox3.Controls.Add(this.label9);
```

```
                this.groupBox3.Controls.Add(this.label10);

                this.groupBox3.Controls.Add(this.label8);

                this.groupBox3.Controls.Add(this.label7);

                this.groupBox3.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

                this.groupBox3.Font = new System.Drawing.Font("Arial", 8.25F,
                System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
                ((byte)(0)));

                this.groupBox3.ForeColor = System.Drawing.Color.Black;

                this.groupBox3.Location = new System.Drawing.Point(461, 22);

                this.groupBox3.Name = "groupBox3";

                this.groupBox3.Size = new System.Drawing.Size(144, 122);

                this.groupBox3.TabIndex = 2;

                this.groupBox3.TabStop = false;

                this.groupBox3.Text = "Image information";

// ByteCapacity_lbl

                this.ByteCapacity_lbl.Anchor =
                ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
                orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
```

```
            this.ByteCapacity_lbl.AutoSize = true;

            this.ByteCapacity_lbl.Location = new System.Drawing.Point(88, 47);

            this.ByteCapacity_lbl.Name = "ByteCapacity_lbl";

            this.ByteCapacity_lbl.Size = new System.Drawing.Size(0, 14);

            this.ByteCapacity_lbl.TabIndex = 2;

// CanSave_lbl

            this.CanSave_lbl.Anchor =
            ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
            orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));

            this.CanSave_lbl.AutoSize = true;

            this.CanSave_lbl.Location = new System.Drawing.Point(77, 97);

            this.CanSave_lbl.Name = "CanSave_lbl";

            this.CanSave_lbl.Size = new System.Drawing.Size(35, 14);

            this.CanSave_lbl.TabIndex = 1;

            this.CanSave_lbl.Text = "none";

// ImageWidth_lbl

            this.ImageWidth_lbl.Anchor =
            ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
```

```
                orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
                this.ImageWidth_lbl.AutoSize = true;
                this.ImageWidth_lbl.Location = new System.Drawing.Point(77, 72);
                this.ImageWidth_lbl.Name = "ImageWidth_lbl";
                this.ImageWidth_lbl.Size = new System.Drawing.Size(35, 14);
                this.ImageWidth_lbl.TabIndex = 1;
                this.ImageWidth_lbl.Text = "none";
// ImageHeight_lbl
                this.ImageHeight_lbl.Anchor =
                ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
                orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
                this.ImageHeight_lbl.AutoSize = true;
                this.ImageHeight_lbl.Location = new System.Drawing.Point(77, 47);
                this.ImageHeight_lbl.Name = "ImageHeight_lbl";
                this.ImageHeight_lbl.Size = new System.Drawing.Size(35, 14);
                this.ImageHeight_lbl.TabIndex = 1;
                this.ImageHeight_lbl.Text = "none";
// ImageSize_lbl
```

```
            this.ImageSize_lbl.Anchor =
            ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
            orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
            this.ImageSize_lbl.AutoSize = true;
            this.ImageSize_lbl.Location = new System.Drawing.Point(77, 22);
            this.ImageSize_lbl.Name = "ImageSize_lbl";
            this.ImageSize_lbl.Size = new System.Drawing.Size(35, 14);
            this.ImageSize_lbl.TabIndex = 1;
            this.ImageSize_lbl.Text = "none";
// label9
            this.label9.Anchor =
            ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
            orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(13, 97);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(63, 14);
            this.label9.TabIndex = 0;
```

```
            this.label9.Text = "Can save: ";
// label10
            this.label10.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(13, 72);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(44, 14);
            this.label10.TabIndex = 0;
            this.label10.Text = "Width: ";
// label8
            this.label8.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(13, 47);
            this.label8.Name = "label8";
```

```
            this.label8.Size = new System.Drawing.Size(48, 14);

            this.label8.TabIndex = 0;

            this.label8.Text = "Height: ";

// label7

            this.label7.Anchor =
            ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
            orStyles.Top | System.Windows.Forms.AnchorStyles.Right)));

            this.label7.AutoSize = true;

            this.label7.Location = new System.Drawing.Point(13, 22);

            this.label7.Name = "label7";

            this.label7.Size = new System.Drawing.Size(36, 14);

            this.label7.TabIndex = 0;

            this.label7.Text = "Size: ";

// Close_btn

            this.Close_btn.Anchor =
            ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
            orStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));

            this.Close_btn.DialogResult =
            System.Windows.Forms.DialogResult.Cancel;
```

```
this.Close_btn.Location = new System.Drawing.Point(527, 604);

this.Close_btn.Name = "Close_btn";

this.Close_btn.Size = new System.Drawing.Size(75, 23);

this.Close_btn.TabIndex = 3;

this.Close_btn.Text = "Close";

this.Close_btn.UseVisualStyleBackColor = true;

this.Close_btn.Visible = false;

this.Close_btn.Click += new
System.EventHandler(this.Close_btn_Click);
```

// statusStrip1

```
this.statusStrip1.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(221)))),
((int)(((byte)(240)))), ((int)(((byte)(192)))));

this.statusStrip1.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {

this.toolStripStatusLabel1});

this.statusStrip1.Location = new System.Drawing.Point(0, 630);

this.statusStrip1.Name = "statusStrip1";
```

```
            this.statusStrip1.Size = new System.Drawing.Size(614, 22);

            this.statusStrip1.SizingGrip = false;

            this.statusStrip1.TabIndex = 4;

            this.statusStrip1.Text = "statusStrip1";

// toolStripStatusLabel1

            this.toolStripStatusLabel1.Name = "toolStripStatusLabel1";

            this.toolStripStatusLabel1.Size = new System.Drawing.Size(38, 17);

            this.toolStripStatusLabel1.Text = "Ready";

// tabControl1

            this.tabControl1.Controls.Add(this.tabPage1);

            this.tabControl1.Controls.Add(this.tabPage2);

            this.tabControl1.Font = new System.Drawing.Font("Arial", 8.25F,
            System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
            ((byte)(0)));

            this.tabControl1.ImeMode =
            System.Windows.Forms.ImeMode.NoControl;

            this.tabControl1.Location = new System.Drawing.Point(5, 0);

            this.tabControl1.Multiline = true;
```

```
this.tabControl1.Name = "tabControl1";

this.tabControl1.SelectedIndex = 0;

this.tabControl1.Size = new System.Drawing.Size(449, 145);

this.tabControl1.TabIndex = 5;

// tabPage1

this.tabPage1.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(233)))),
((int)(((byte)(240)))), ((int)(((byte)(221)))));

this.tabPage1.Controls.Add(this.groupBox2);

this.tabPage1.Location = new System.Drawing.Point(4, 23);

this.tabPage1.Name = "tabPage1";

this.tabPage1.Padding = new System.Windows.Forms.Padding(3);

this.tabPage1.Size = new System.Drawing.Size(441, 118);

this.tabPage1.TabIndex = 0;

this.tabPage1.Text = "Encrypt Image";

// tabPage2

this.tabPage2.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(233)))),
```

```
((int)(((byte)(240)))), ((int)(((byte)(221)))));

this.tabPage2.Controls.Add(this.groupBox1);

this.tabPage2.Location = new System.Drawing.Point(4, 23);

this.tabPage2.Name = "tabPage2";

this.tabPage2.Padding = new System.Windows.Forms.Padding(3);

this.tabPage2.Size = new System.Drawing.Size(441, 118);

this.tabPage2.TabIndex = 1;

this.tabPage2.Text = "Decrypt Image";

// linkLabel1

this.linkLabel1.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.Anch
orStyles.Bottom | System.Windows.Forms.AnchorStyles.Right)));

this.linkLabel1.AutoSize = true;

this.linkLabel1.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(221)))),
((int)(((byte)(240)))), ((int)(((byte)(192)))));

this.linkLabel1.Location = new System.Drawing.Point(446, 634);

this.linkLabel1.Name = "linkLabel1";
```

```
            this.linkLabel1.Size = new System.Drawing.Size(168, 13);

            this.linkLabel1.TabIndex = 6;

            this.linkLabel1.TabStop = true;

            this.linkLabel1.TextAlign =
            System.Drawing.ContentAlignment.BottomRight;

// FrmSteganography

            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

            this.BackColor =
            System.Drawing.Color.FromArgb(((int)(((byte)(210)))),
            ((int)(((byte)(218)))), ((int)(((byte)(196)))));

            this.CancelButton = this.Close_btn;

            this.ClientSize = new System.Drawing.Size(614, 652);

            this.Controls.Add(this.linkLabel1);

            this.Controls.Add(this.tabControl1);

            this.Controls.Add(this.statusStrip1);

            this.Controls.Add(this.Close_btn);

            this.Controls.Add(this.groupBox3);
```

```
this.Controls.Add(this.label3);

this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.Name = "FrmSteganography";

this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Digital Steganography";

this.Paint += new
System.Windows.Forms.PaintEventHandler(this.Form1_Paint);

this.groupBox1.ResumeLayout(false);

this.groupBox1.PerformLayout();

this.groupBox2.ResumeLayout(false);

this.groupBox2.PerformLayout();

this.groupBox3.ResumeLayout(false);

this.groupBox3.PerformLayout();

this.statusStrip1.ResumeLayout(false);

this.statusStrip1.PerformLayout();
```

```
            this.tabControl1.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.tabPage2.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.Button EnImageBrowse_btn;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox EnImage_tbx;
        private System.Windows.Forms.Button EnFileBrowse_btn;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox EnFile_tbx;
        private System.Windows.Forms.Button Encrypt_btn;
        private System.Windows.Forms.OpenFileDialog openFileDialog1;
```

```
private System.Windows.Forms.OpenFileDialog openFileDialog2;

private System.Windows.Forms.SaveFileDialog saveFileDialog1;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Button Decrypt_btn;

private System.Windows.Forms.TextBox DeLoadImage_tbx;

private System.Windows.Forms.TextBox DeSaveFile_tbx;

private System.Windows.Forms.Button DeSaveFileBrowse_btn;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Button DeLoadImageBrowse_btn;

private System.Windows.Forms.OpenFileDialog openFileDialog3;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.GroupBox groupBox3;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label9;

private System.Windows.Forms.Label ByteCapacity_lbl;

private System.Windows.Forms.Label ImageSize_lbl;

private System.Windows.Forms.FolderBrowserDialog folderBrowserDialog1;
```

```
        private System.Windows.Forms.Label CanSave_lbl;

        private System.Windows.Forms.Label ImageWidth_lbl;

        private System.Windows.Forms.Label ImageHeight_lbl;

        private System.Windows.Forms.Label label10;

        private System.Windows.Forms.Label label8;

        private System.Windows.Forms.Button Close_btn;

        private System.Windows.Forms.StatusStrip statusStrip1;

        private System.Windows.Forms.ToolStripStatusLabel toolStripStatusLabel1;

        private System.Windows.Forms.TabControl tabControl1;

        private System.Windows.Forms.TabPage tabPage1;

        private System.Windows.Forms.TabPage tabPage2;

        private System.Windows.Forms.LinkLabel linkLabel1;

    }

}
```

7. Result/ Screenshot

Encryption Process:

1. Select “Encrypt Image” tab for encryption process.

Figure 7.1 Encrypt Image- Startup page

2. Load carrier image using “Browse” tab in “Load Image” field.

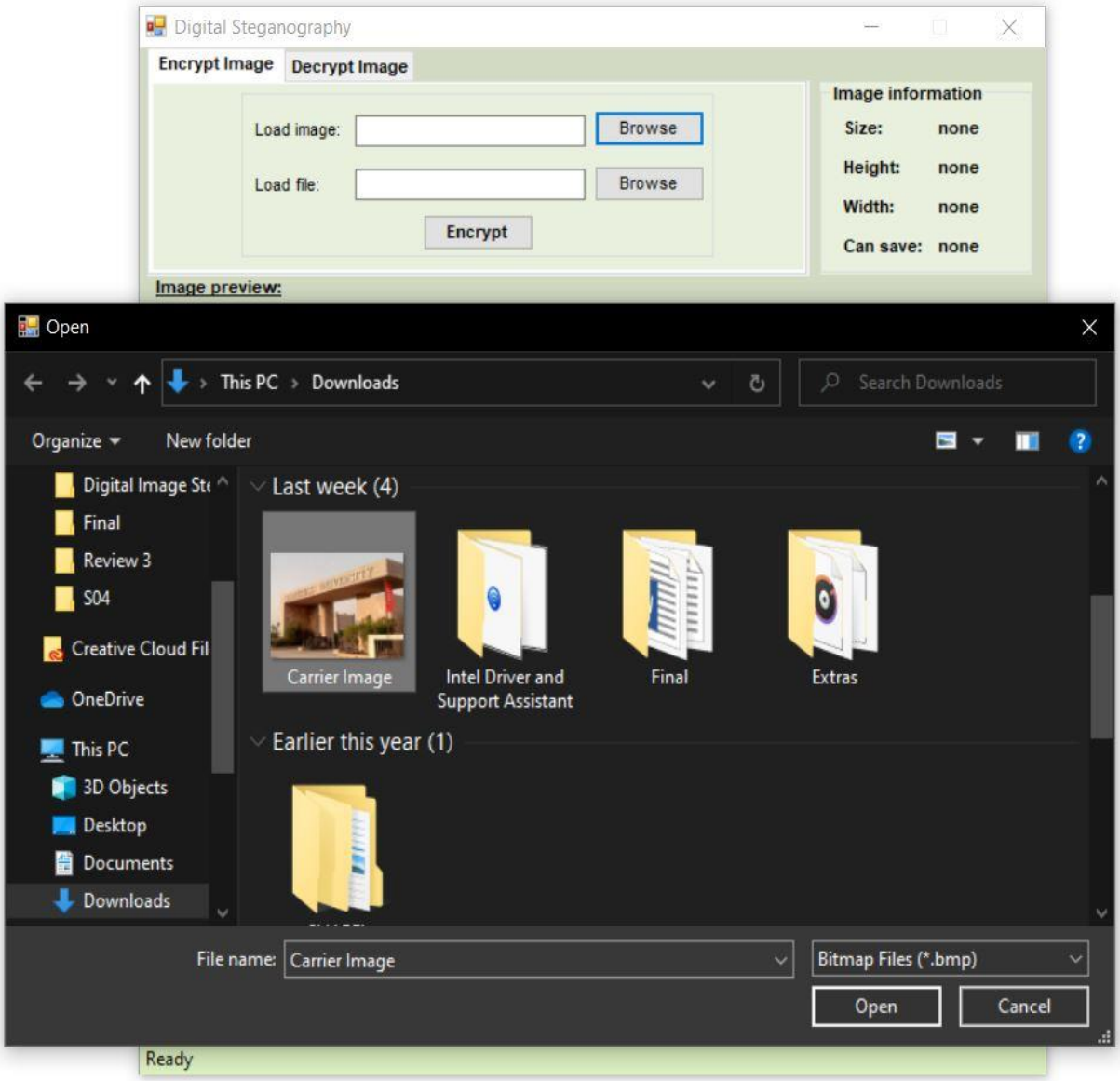

Figure 7.2 Loading Carrier image

3. Load secret message using “Browse” tab in the “Load File” field.

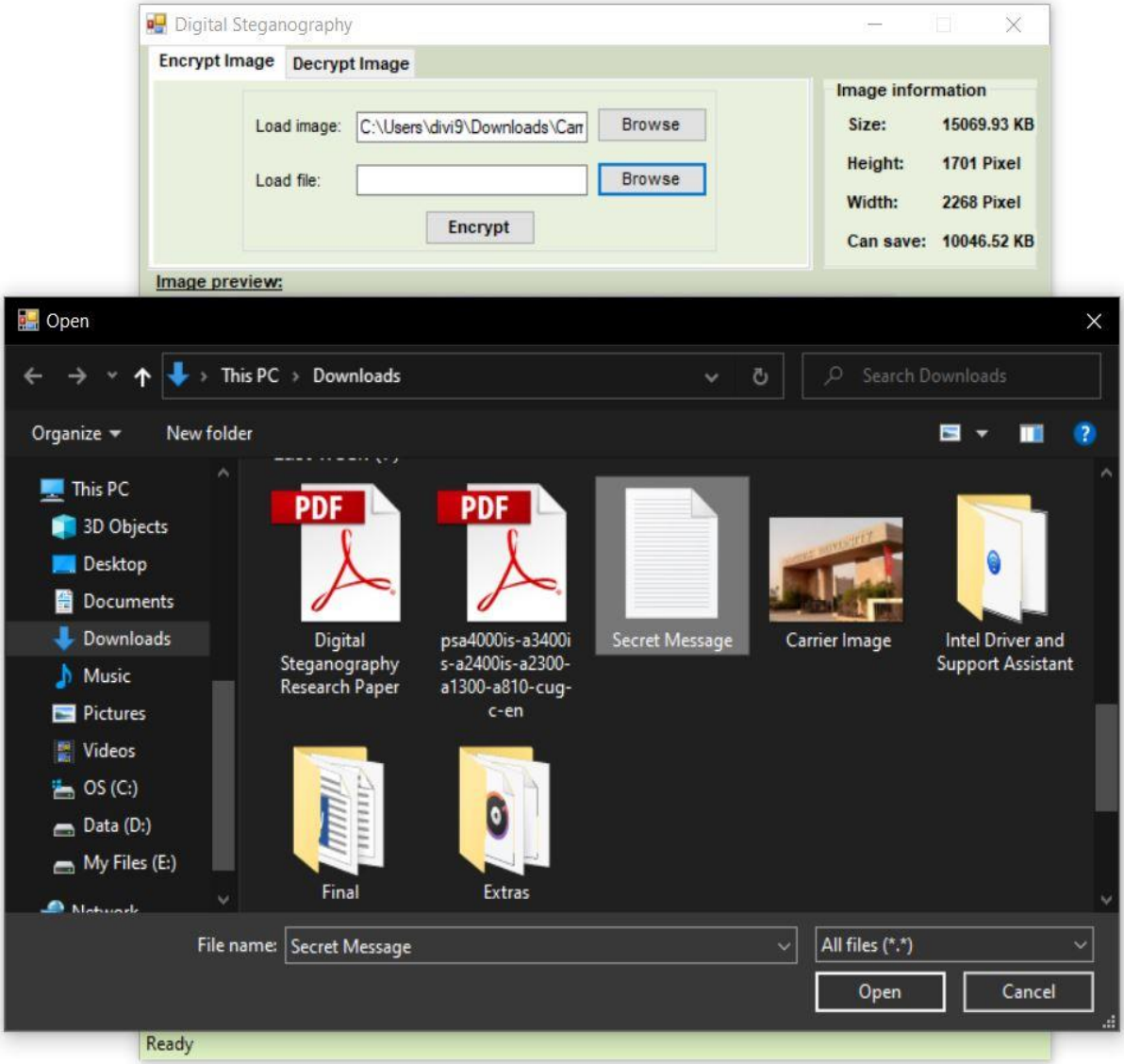

Figure 7.3 Loading secret message

4. Click on “Encrypt” button, type the name by which it has to be saved in the destination folder. Click on “Save” button.

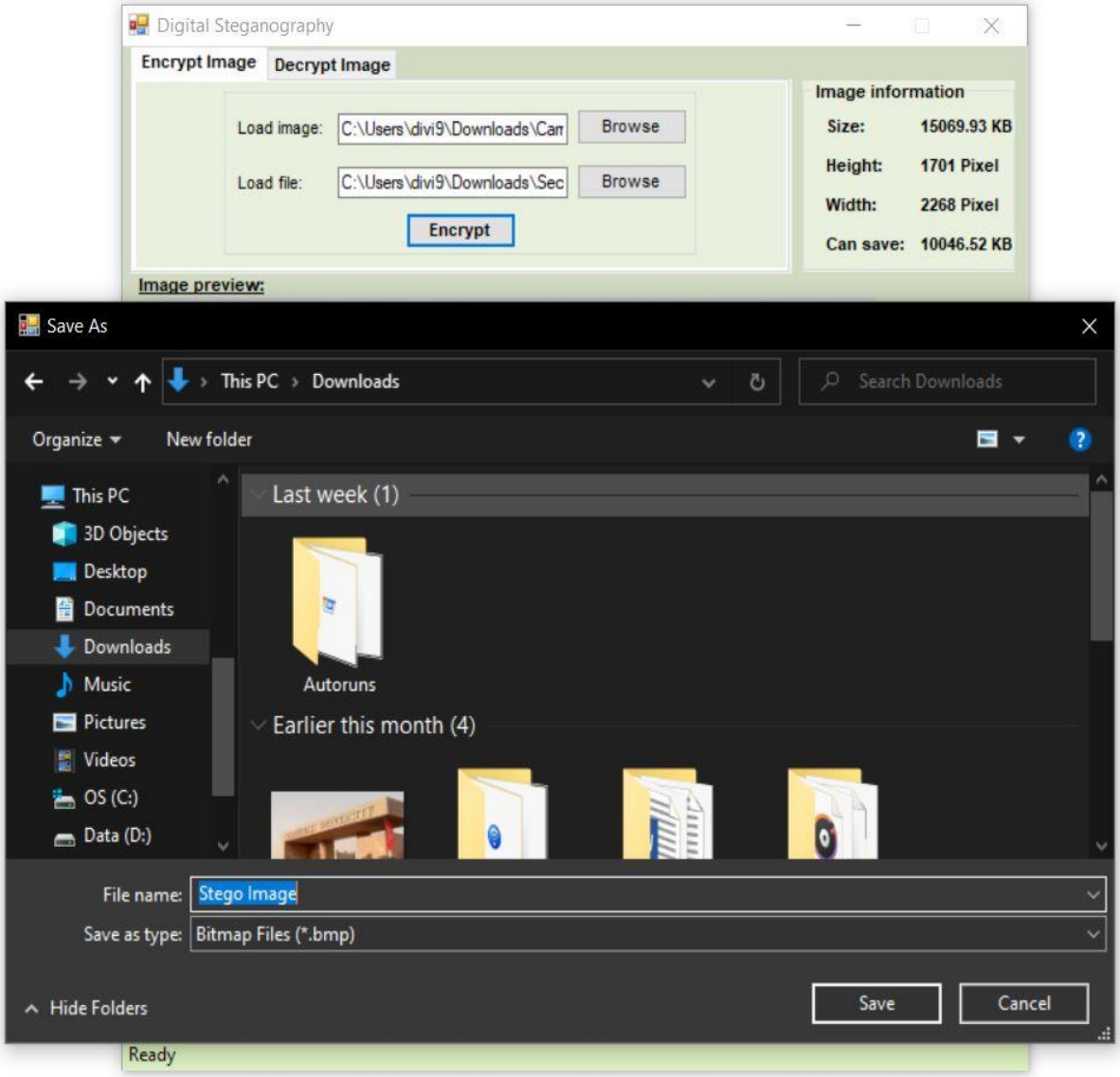

Figure 7.4 Saving the encrypted image

5. Encrypted image is saved in the destination folder.

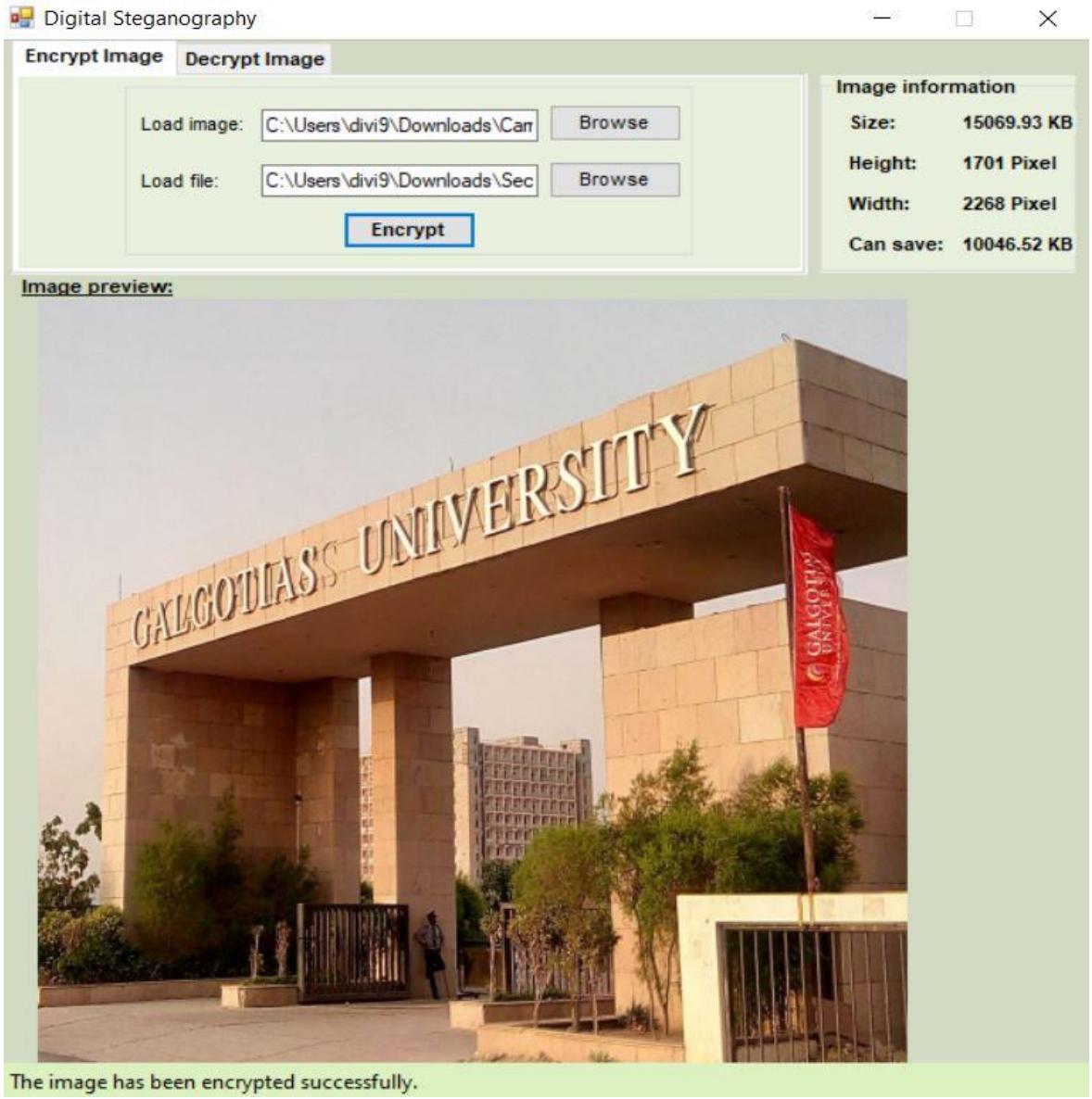

Figure 7.5 Encryption successful

Decryption Process:

1. Select the “Decrypt Image” tab at the top.

Figure 7.6 Decrypt Image- Startup page

2. Load encrypted image using the “Browse” tab in the “Load Image” field.

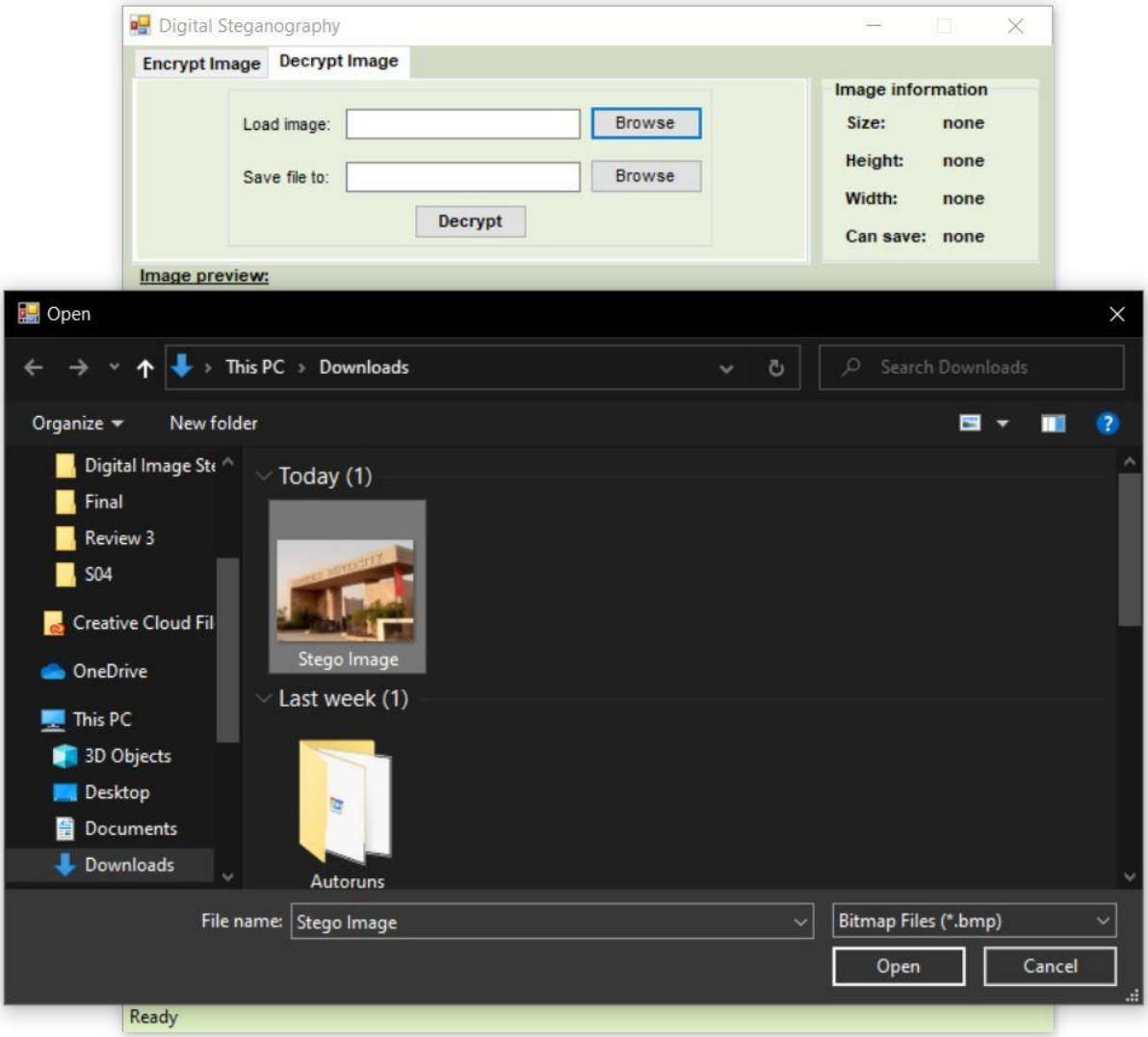

Figure 7.7 Loading encrypted image

3. Select path or folder where hidden message is to be saved, using the “Browse” tab in the “Save File To” field.

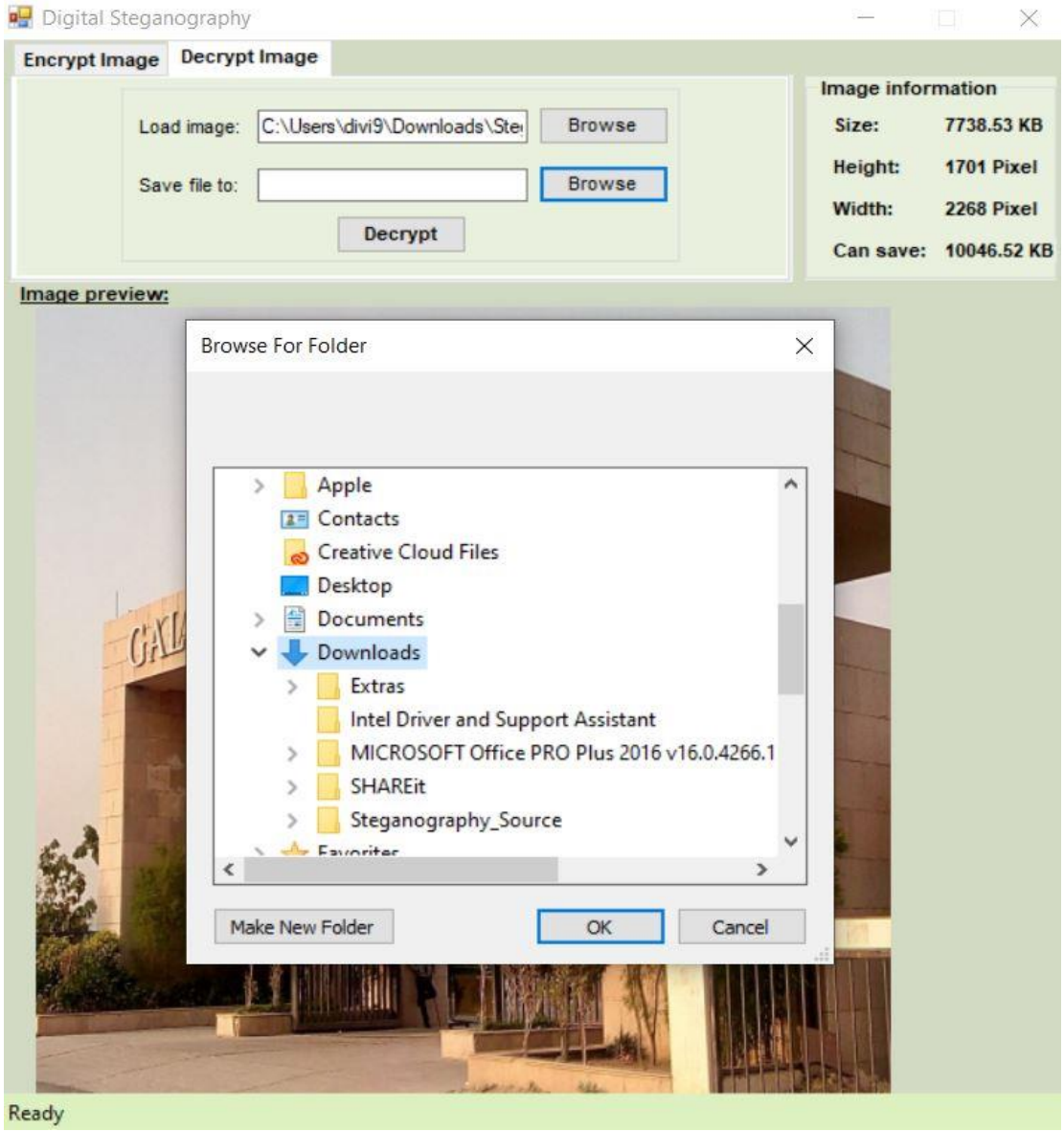

Figure 7.8 Setting path for decrypted message

4. Click on “Decrypt” button. The message will be saved in the selected folder.

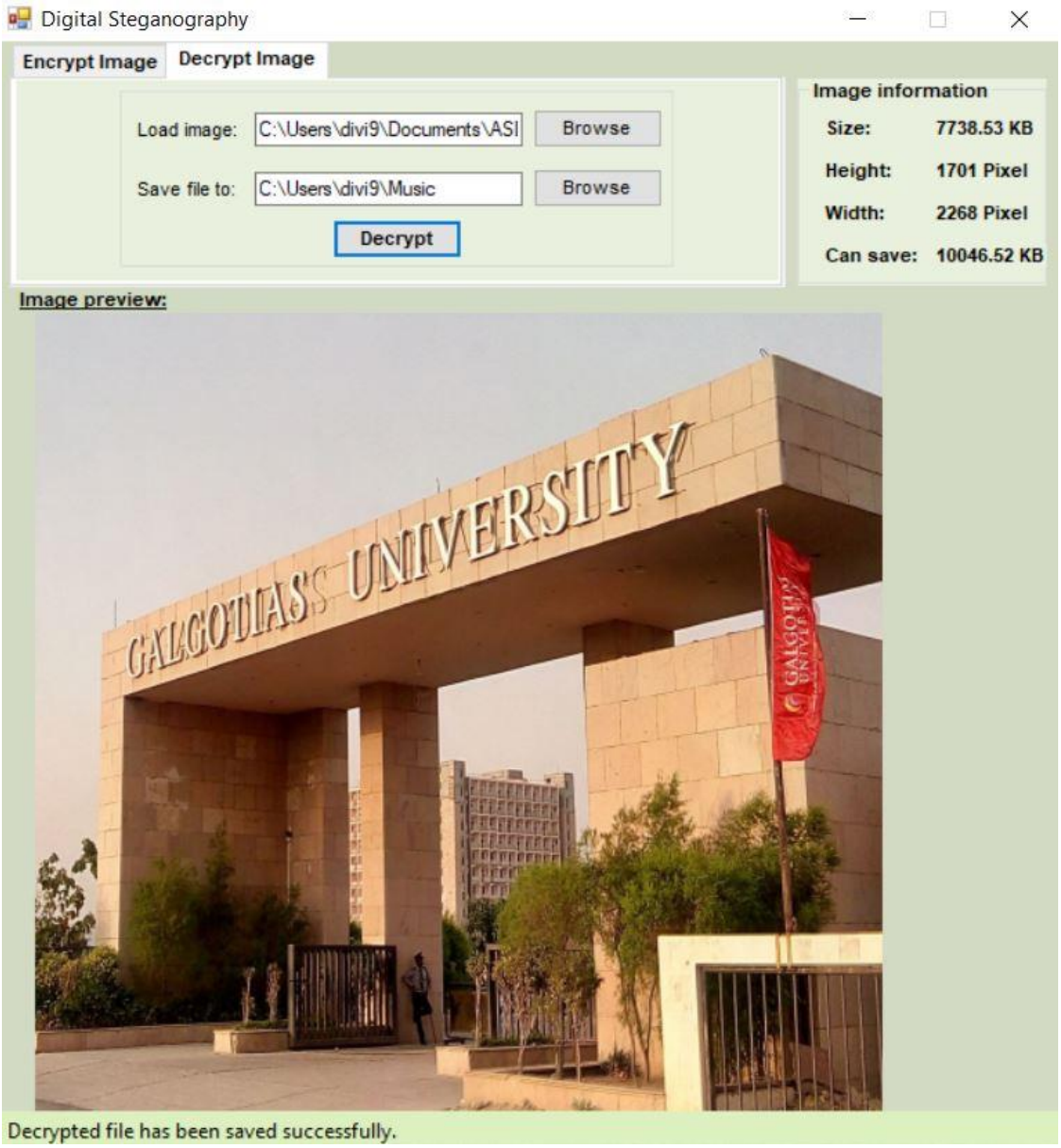

Figure 7.9 Decryption successful

Difference between Carrier Image and Encrypted Image

Figure 7.10 Carrier image

Figure 7.11 Encrypted image

8. Conclusion

Steganography is a really interesting subject and outside of the mainstream cryptography and system administration that most of us deal with day after day. It 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 the LSB approach to provide a means of secure communication. A stego-key has been applied to the system during the embedment of the message into the cover image.

This steganography application software provided for how to use any type of image format to hide any type of files inside them. The masterwork of this application is in supporting any type of pictures without the 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 is the art of sharing encrypted information by concealing the fact that communication is even taking place. There are many steganography techniques already available and currently in use. Some of these techniques like LSB substitution is one of the oldest techniques but it is still preferred due to its simplicity. On the other hand, techniques like Transform Domain are complex but robust.

We also learn that the message can also be embedded by using more than one technique together. For instance, the LSB substitution method can be modified with the RPE method to fill the gaps. Also, we can set a unique code to the already encrypted image so that if somehow someone gets to the encrypted image and decodes the message, then also that unique code would be needed to reveal the message. The most suitable technique depends on the user and the purpose it is required for.

References

[1] Srishti Rajvanshi, Shrikrishna Sawant, Vedant Tiwari, Anurag Waghmare, Manjiri Gogate, “Image Steganography” International Journal for Research in Applied Science & Engineering Technology (IJRASET)- Volume 7, Issue XI, Nov 2019

[2] Ramadhan Mstafa, Christian Bach, “Information Hiding in Images Using Steganography Techniques” (2013) ASEE Northeast Section Conference

[3] Mustafa Cem Kasapbaşi, Wisam Elmasry, “New LSB-Based Colour Image Steganography Method to enhance the Efficiency in Payload Capacity, Security and Integrity Check” Sādhanā volume 43, Article number: 68 (2018)

[4] Aldi Wiliar Wira Permana, Silvester Dian Handy Permana, Yaddarabullah, “Modification of Least Significant Bit Method with Redundant Pattern Encoding for Protection of Message Integration from Image Modification” International Journal of Scientific & Technology Research, Volume 9, Issue 04, April 2020

[5] Mr. Jayesh Surana, Anirudh Sonsale, Bhavesh Joshi, Deepesh Sharma, Nilesh Choudhary, “Steganography Techniques” (2017) IJEDR, Volume 5, Issue 2, ISSN: 2321-9939

[6] Ashadeep Kaur, Rakesh Kumar, Kamaljeet Kainth, “Review Paper on Image Steganography” International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 6, June 2016

[7] Ruchi, Umesh Ghanekar, “A Brief Review on Image Steganography Techniques” Proceedings of ICAEEC-2019, IIIT Allahabad India, 31st May - 1st June 2019

[8] The Types and Techniques Of Steganography Computer Science Essay https://www.ukessays.com/essays/computer-science/the-types-and-techniques-of-steganography-computer-science-essay.php

[9] Abbas Cheddad, Joan Condell, Kevin Curran, Paul Mc Kevitt, “Digital Image Steganography: Survey and Analysis of Current Methods” (2009) Elsevier B.V

[10] Stuti Goel, Arun Rana, Manpreet Kaur, “A Review of Comparison Techniques of Image Steganography” Global Journal of Computer Science and Technology Graphics & Vision, Volume 13, Issue 4, Version 1.0, Year 2013

[11] Bin Li, Junhui He, Jiwu Huang, Yun Qing Shi, “A Survey on Image Steganography and Steganalysis” Journal of Information Hiding and Multimedia Signal Processing, Volume 2, Number 2, April 2011