Project Documentation

Project Topic – Image Steganography

(A new technique to hide information within image file)

Table of Contents -

I. Abstract –

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. 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 report 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 steganographic techniques are more suitable for which applications.

II. INTRODUCTION -

One of the reasons that intruders can be successful is the 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. Steganography is a technique of hiding information in digital media. 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.

Secret communication is very important because if you're message is important and if you don't want others to know about your message then you can use different kind of technique to hide your message from third person and steganography is one such technique. However criminals & terrorist organization are using this for their own purpose. However to talk about Steganography we should the term 'Cryptography' which is the science of writing in secret codes. Basically Cryptography makes message meaningless to the casual reader by encrypting a data using set of values which are unknown to both sender and receiver only the intended receiver with decryption key can extract the actual message. Thus the when attackers discovers the message it is still difficult to him to get the secret message. If cryptography is a strong way to encrypt and secure a communication then why do we need a new technique.

Answer is very simple when we are using any cryptography technique when it is send a secret key and the third person can easily judge that secret kind of communication is going on. In simple terms, Cryptography does not try to hide the fact that secret message is been send. **This is how** Steganography comes into the picture. The main reason of using steganography is that you're hiding a secret message behind a ordinary file that no one can suspect that fact that a communication or some sort of secret message is been send. People will generally think it is an ordinary file and your secret message will go without any suspicion. So, if I want to define Steganography is the art of and science of embedding secret messages in cover message in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message.

III. Synopsis –

- **What is Steganography?**

Steganography is the practice of hiding private or sensitive information within something that appears to be nothing out to the usual. Steganography is often confused with cryptology because the two are similar in the way that they both are used to protect important information. The difference between two is that steganography involves hiding information so it appears that no information is hidden at all. If a person or persons 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.

- **History of Steganography -**

Some examples of use of Steganography is past times are:

1. During World War 2 invisible ink was used to write information on pieces of paper so that the paper appeared to the average person as just being blank pieces of paper. Liquids such as milk, vinegar and fruit juices were used, because when each one of these substances are heated they darken and become visible to the human eye.
2. In Ancient Greece they used to select messengers and shave their head, they would then write a message on their head. Once the message had been written the hair was allowed to grow back.

- **Types of Steganography –**

Depending on the nature of the cover object (actual object in which secret data is embedded), steganography can be divided into five types:

1. Text Steganography
2. Image Steganography
3. Video Steganography
4. Audio Steganography
5. Network Steganography
6. Email Steganography

1. Text Steganography -

Text Steganography is hiding information inside the text files. It involves things like changing the format of existing text, changing words within a text, generating random character sequences or using context-free grammars to generate readable texts. Various techniques used to hide the data in the text are:

- Format Based Method
- Random and Statistical Generation
- Linguistic Method

2. Image Steganography -

Hiding the data by taking the cover object as the image is known as image steganography. In digital steganography, images are widely used cover source because there are a huge number of bits present in the digital representation of an image. There are a lot of ways to hide information inside an image. Common approaches include:

- Least Significant Bit Insertion
- Masking and Filtering
- Redundant Pattern Encoding
- Encrypt and Scatter

3. Audio Steganography -

In audio steganography, the secret message is embedded into an audio signal which alters the binary sequence of the corresponding audio file. Hiding secret messages in digital sound is a much more difficult process when compared to others, such as Image Steganography. Different methods of audio steganography include:

- Least Significant Bit Encoding
- Parity Encoding
- Phase Coding
- Spread Spectrum

This method hides the data in WAV, AU, and even MP3 sound files.

4. Video Steganography -

In Video Steganography you can hide kind of data into digital video format. The advantage of this type is a large amount of data can be hidden inside and the fact that it is a moving stream of images and sounds. You can think of this as the combination of Image Steganography and Audio Steganography. Two main classes of Video Steganography include:

- Embedding data in uncompressed raw video and compressing it later
- Embedding data directly into the compressed data stream

5. Network Steganography (Protocol Steganography) –

Network Steganography is a technique that uses common network protocols (the header field, the payload field or both) to hide a secret message. This technique implements a storage based network steganography that uses the timestamp option which is used for debugging and measurement over the networks.

6. Email Steganography –

Email-based steganography is one of the most popular techniques for the text steganography. This technique hides the secret message within the email body and email addresses. Email addresses are used to hide the secret data, regardless they are valid addresses or not.

- **Why This Steganography?**

This technique is chosen, because this system includes not only imperceptibility but also un-delectability by any steganolysis tool.

- Steganography techniques –

The word Steganography is derived from Steganos means “Concealed or Hidden” and Graphen means “Drawing or writing”.

IV. Project Scope -

This project is developed for hiding information in any image file. The scope of the project is 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.

V. Methodology -

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

This project has two methods – Encrypt and Decrypt.

In encryption the secrete information is hiding in with any type of image file.

Decryption is getting the secrete information from image file.

VI. Requirements –

- **Software Requirements:**
  - **.NET Framework 3.5**

- **Hardware Requirements:**

  Processor: Preferably 1.0 GHz or Greater.

  RAM : 512 MB or Greater.

VII. Objective -

The goal of steganography is to hide information within image file.

This approach of information hiding technique has recently became 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 encryption module of this project
- To extract techniques of getting secret data using decryption module.

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

VIII. Overview -

Steganography mean, literally, covered writing. It is the art and science of hiding information such its presence cannot be detected and a communication is happening. A secret information is encoding 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.

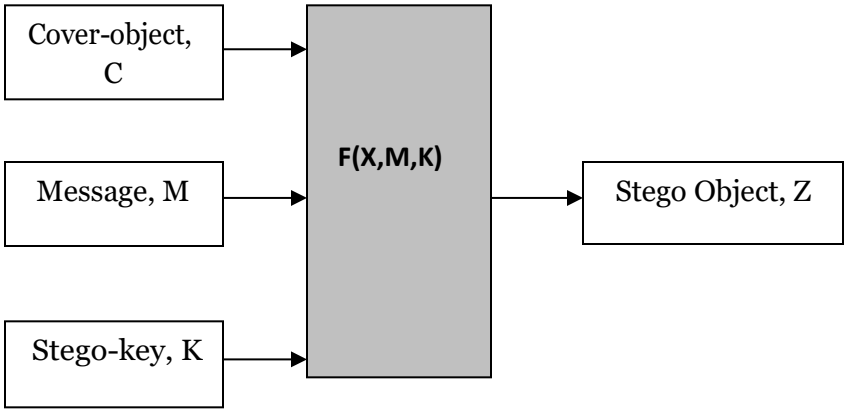
The main goal of this projects it to communicate securely in a completely undetectable manner and to avoid drawing suspicion to the transmission of a hider data in image form. There has been a rapid growth of interest in steganography for two reasons:

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.

The basic model of steganography consists of Carrier, Message and password. Carrier is also known as cover-object, which the message is embedded and serves to hide the presence of the message.

Basically, the model for steganography is shown on following figure:



Message is the data that the sender wishes to remain it confidential. It can be plain text, ciphertext, other image, or anything that can be embedded in a bit stream such as a copyright mark, a covert communication, or a serial number. Password is known as *stego- key*, which ensures that only recipient who know 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 message from a *stego-object* requires the *cover-object* itselt 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 hides and append files by using the slack space
- Text such as null characters, just alike 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 *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

IX. Steganography vs Cryptography -

Basically, the purpose of cryptography and steganography is to provide secret communication. However, steganography is not the same as cryptography. Cryptography hides the contents of a secrete message from a malicious people, whereas steganography even conceal the existence of the message. In cryptography, the system is broken when the attacker can read the secret message. Breaking a steganography system need the attacker to detect that steganography has been used.

It is possible to combine the techniques by encrypting message using cryptography and then hiding the encrypted message using steganography. The resulting stego-image can be transmitted without revealing that secret information is being exchanged.

Let's explore in detailed about Image Steganography -

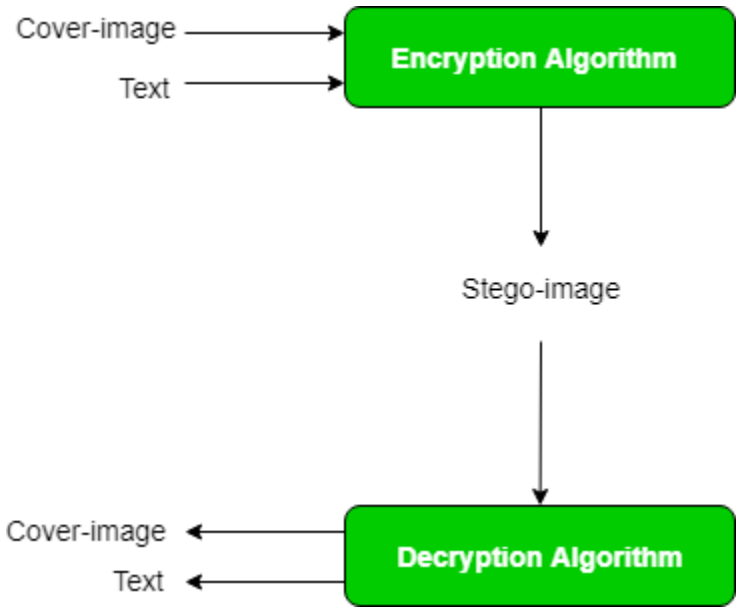
X. Image Steganography –

As the name suggests, Image Steganography refers to the process of hiding data within an image file. The image selected for this purpose is called the cover-image and the image obtained after steganography is called the stego-image.

- **How is it done?**

An image is represented as an N*M (in case of greyscale images) or N*M*3 (in case of colour images) matrix in memory, with each entry representing the intensity value of a pixel. In image steganography, a message is embedded into an image by altering the values of some pixels, which are chosen by an encryption algorithm.

The recipient of the image must be aware of the same algorithm in order to known which pixels he or she must select to extract the message.



Detection of the message within the cover-image is done by the process of steganalysis . This can be done through comparison with the cover image, histogram plotting, or by noise detection.

While efforts are being invested in developing new algorithms with a greater degree of immunity against such attacks, efforts are also being devoted towards improving existing algorithms for steganalysis, to detect exchange of secret information between terrorists or criminal element.

- **Image Steganography and bitmap pictures -**

Using bitmap pictures for hiding secret information is one of most popular choices for Steganography. Many types of software built for this purpose, some of these software use password protection to encrypting information on picture. To use these software you must have a 'BMP' format of a pictures to use it, but using other type of pictures like "JPEG", "GIF" or any other types is rather or never used, because of algorithm of "BMP" pictures for Steganography is simple. Also we know that in the web most popular of image types are "JPEG" and other types not "BPM", so we should have a solution for this problem.

This software provide the solution of this problem, it can accept any type of image to hide information file, but finally it give the only "BMP" image as an output that has hidden file inside it.

Bitmap Steganography -

Bitmap type is the simplest type of picture because that it doesn't have any technology for decreasing file size. Structure of these files is that a bitmap image 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 BMP pictures. So if we just use last layer (8st layar) of information, we should change the last bit of pixels, in other hands we have 3 bits in each pixel so we have 3*hight*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).

(00101101 00011101 11011100)

(10100110 11000101 00001100)

(11010010 10101100 01100011)

Using each 3 pixel of picture to save a byte of data

XI. System Analysis & Design -

Steganography system requires any type of image file and the information or message that is to be hidden. It has two modules encrypt and decrypt.

Microsoft .Net framework prepares a huge amount of tool and options for programmers that they simples programming. One of .Net tools for pictures and images is auto-converting most types of pictures to BMP format. I used this tool in this software called "Steganography" that is written in C#.Net language and you can use this software to hide your information in any type of pictures without any converting its format to BMP (software converts inside it).

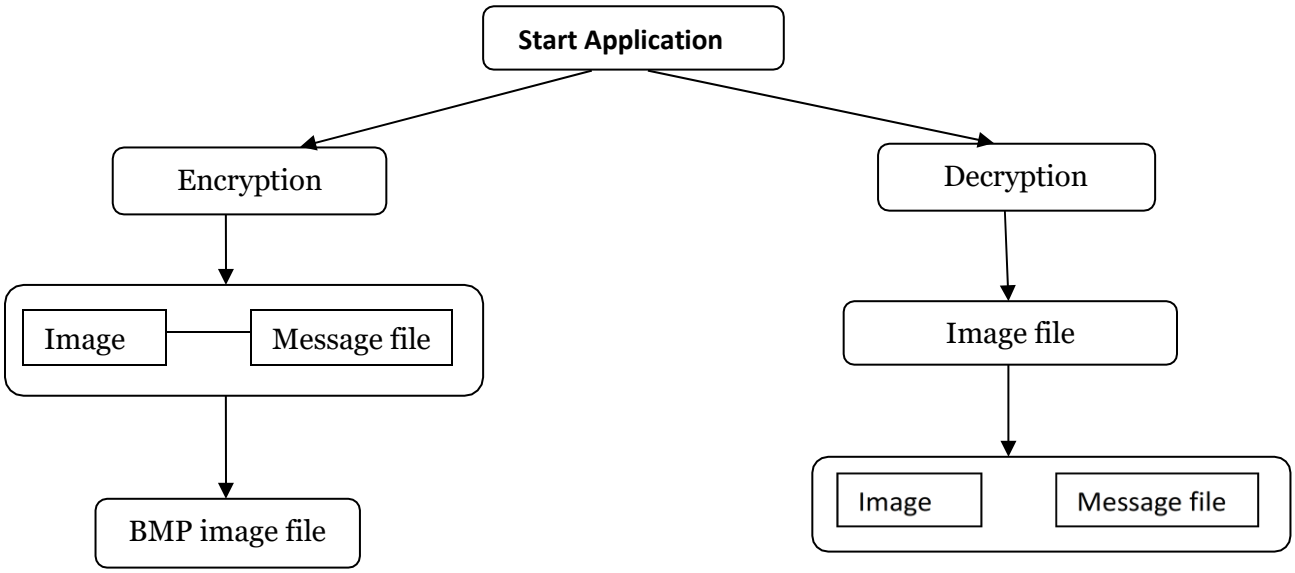
The algorithm used for Encryption and Decryption in this application provides using several layers lieu of using only LSB layer of image. Writing data starts from last layer (8st or LSB layer); because significant of this layer is least and every upper layer has doubled significant from its down layer. 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 destination.

The decrypt module is used to get the hidden information in an image file. It take the image file as an output, and give two file at destination folder, one is the same image file and another is the message file that is hidden it that.

Before encrypting file inside image we must save name and size of file in a definite place of image. We could save file name before file information in LSB layer and save file size and file name size in most right-down pixels of image. Writing this information is needed to retrieve file from encrypted image in decryption state.

The graphical representation of this system is as follows:


Start Application
Encryption
Decryption
Image
Message file
Image file
BMP image file
Image
Message file

XII. 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 frist.", "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 = "Encrypted image has been
successfully saved.";
            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))
            {
```

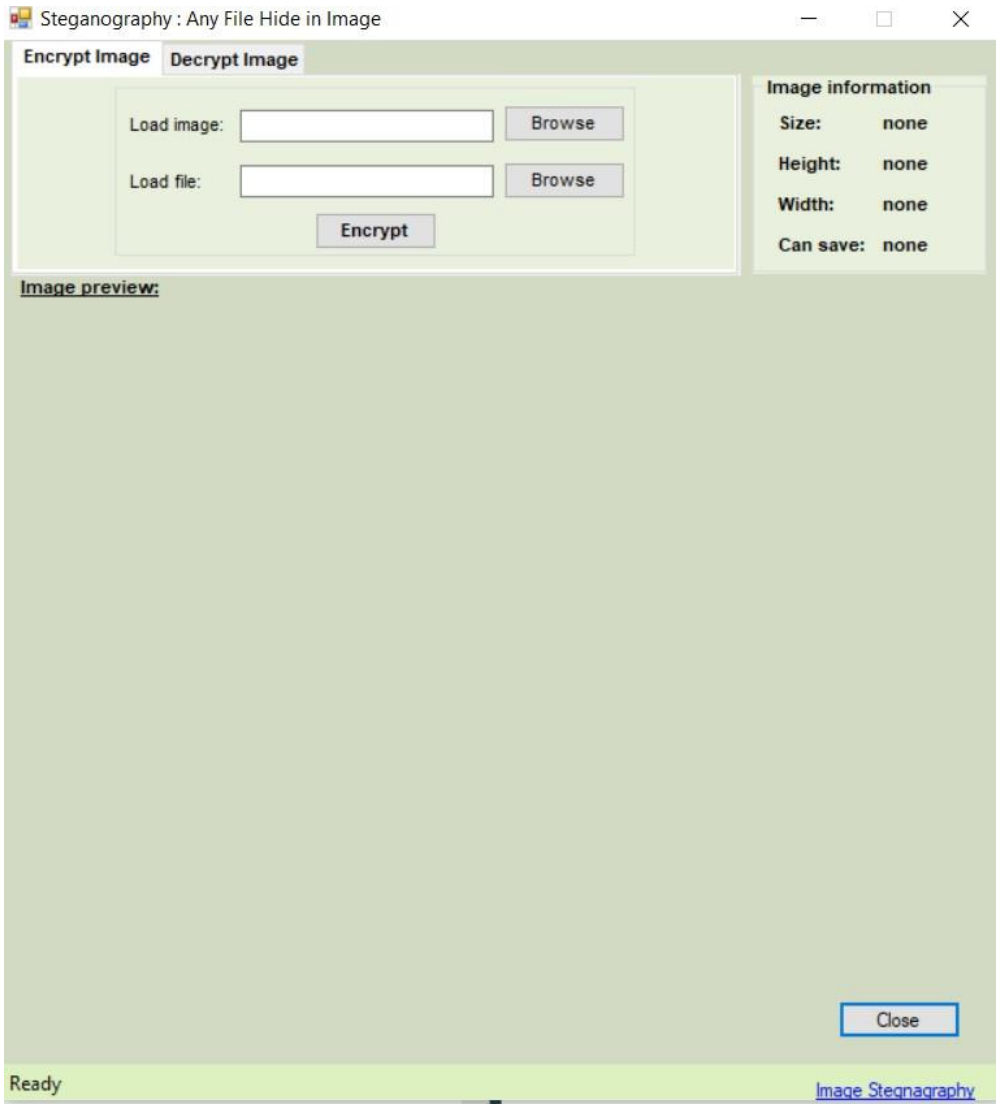
```
                MessageBox.Show("File \"" + resFName + "\" already exist
please choose another path to save file",
"Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
                return;
            }
            else
                File.WriteAllBytes(DSaveFilePath + "\\" + resFName, res);
            toolStripStatusLabel1.Text = "Decrypted file has been
successfully saved.";
            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();
        }

        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            System.Diagnostics.Process.Start("");

        }

    }
}
```

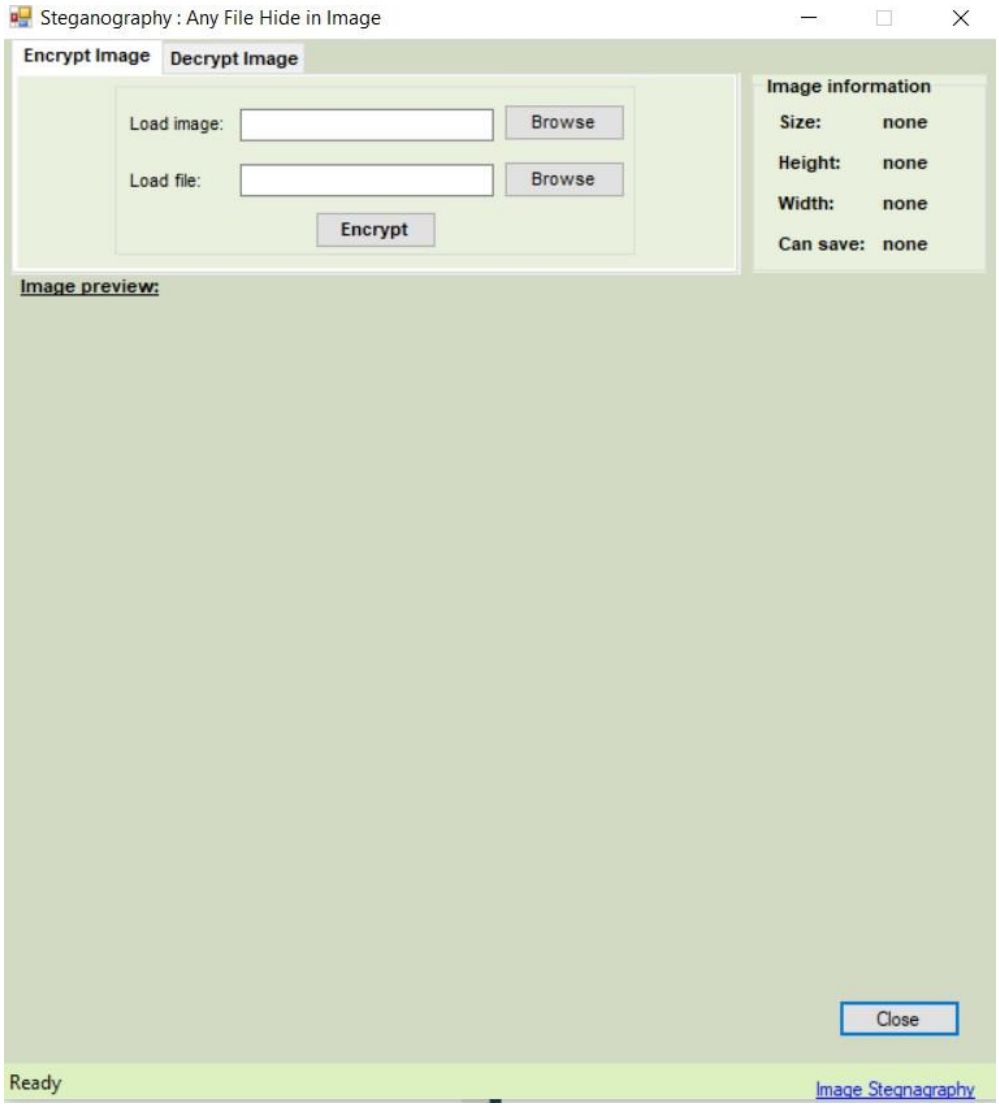
XIII. User Manual –

- This is the first screen which has two tab options – one is Encrypt Image for encryption and another is Decrypt image for decryption. In right – top panel is displays the information about the image such as size, height and width.

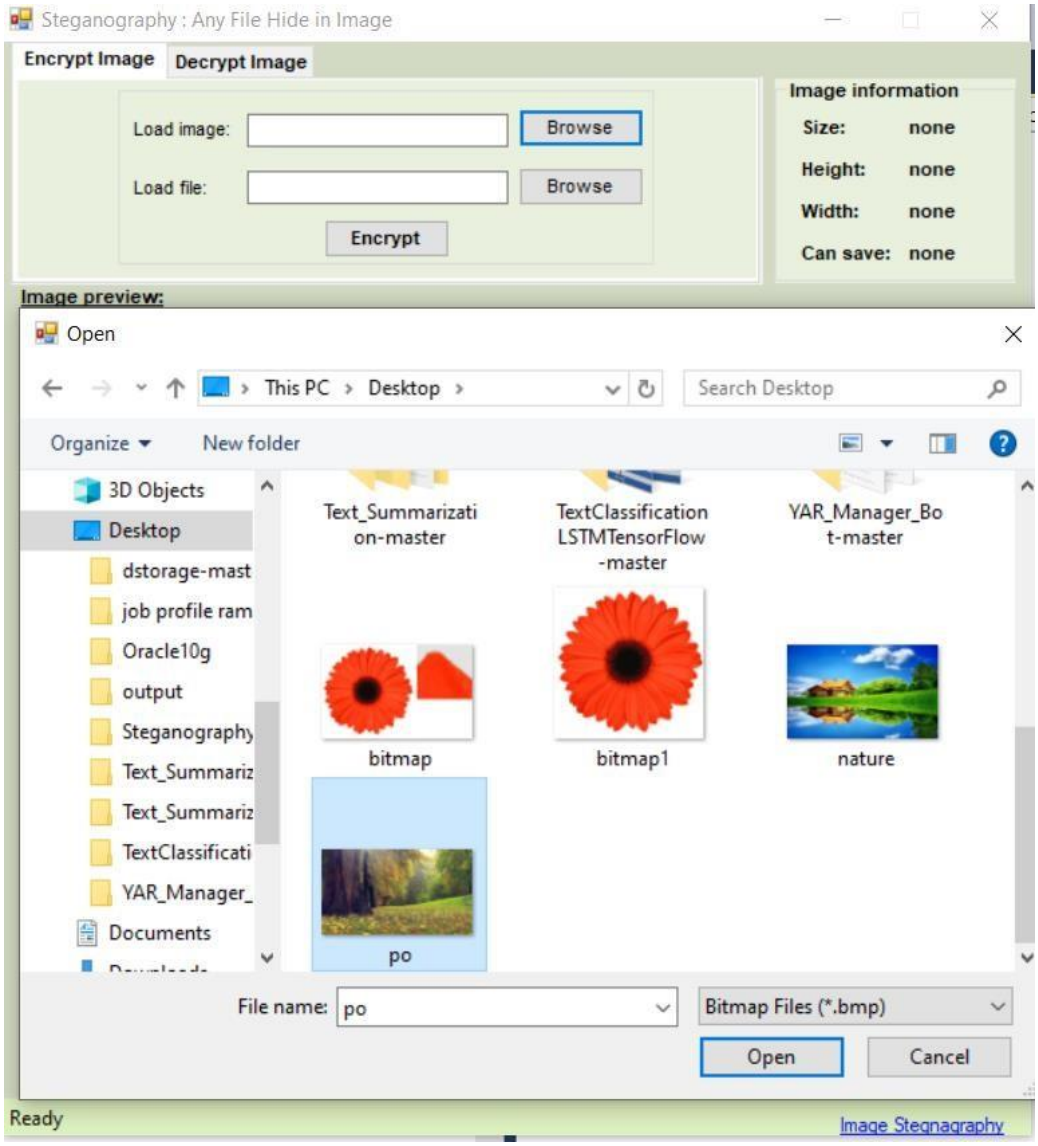


- **Encryption -**

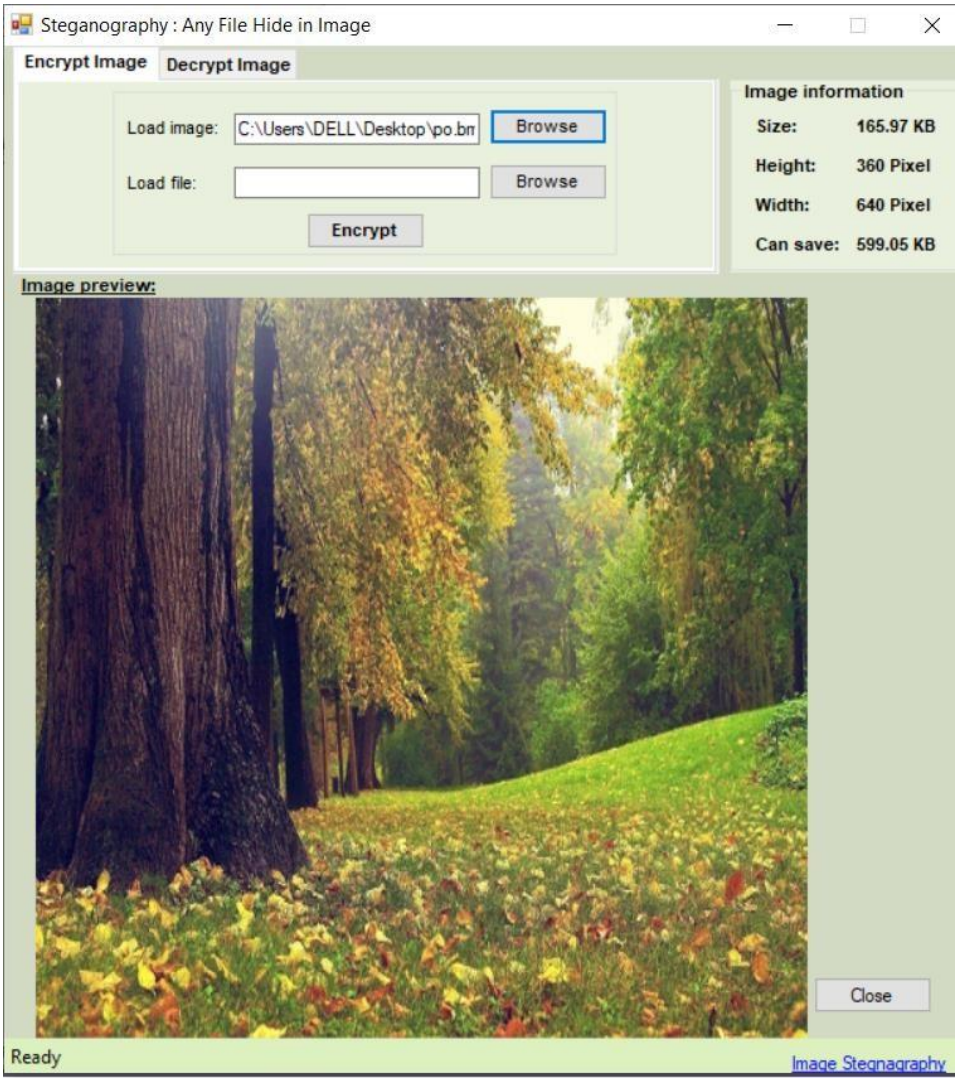
1. For Encryption select Encrypt Image tab option.



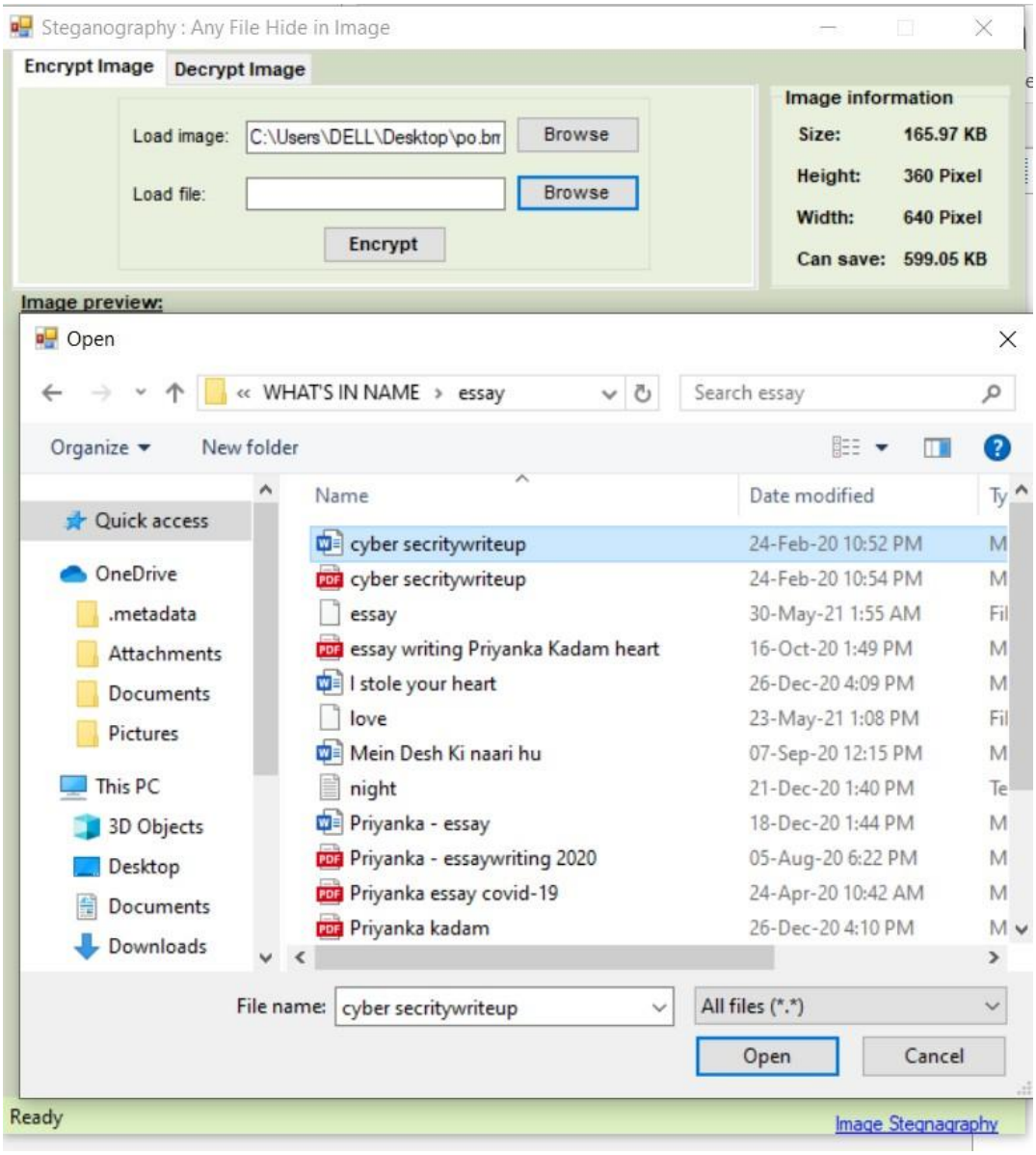
2. For load image click on button “Browse” that is next to the Load Image textbox. The file open dialog box will displays as follows, select the Image file, which you want to use hide information and click on Open button.



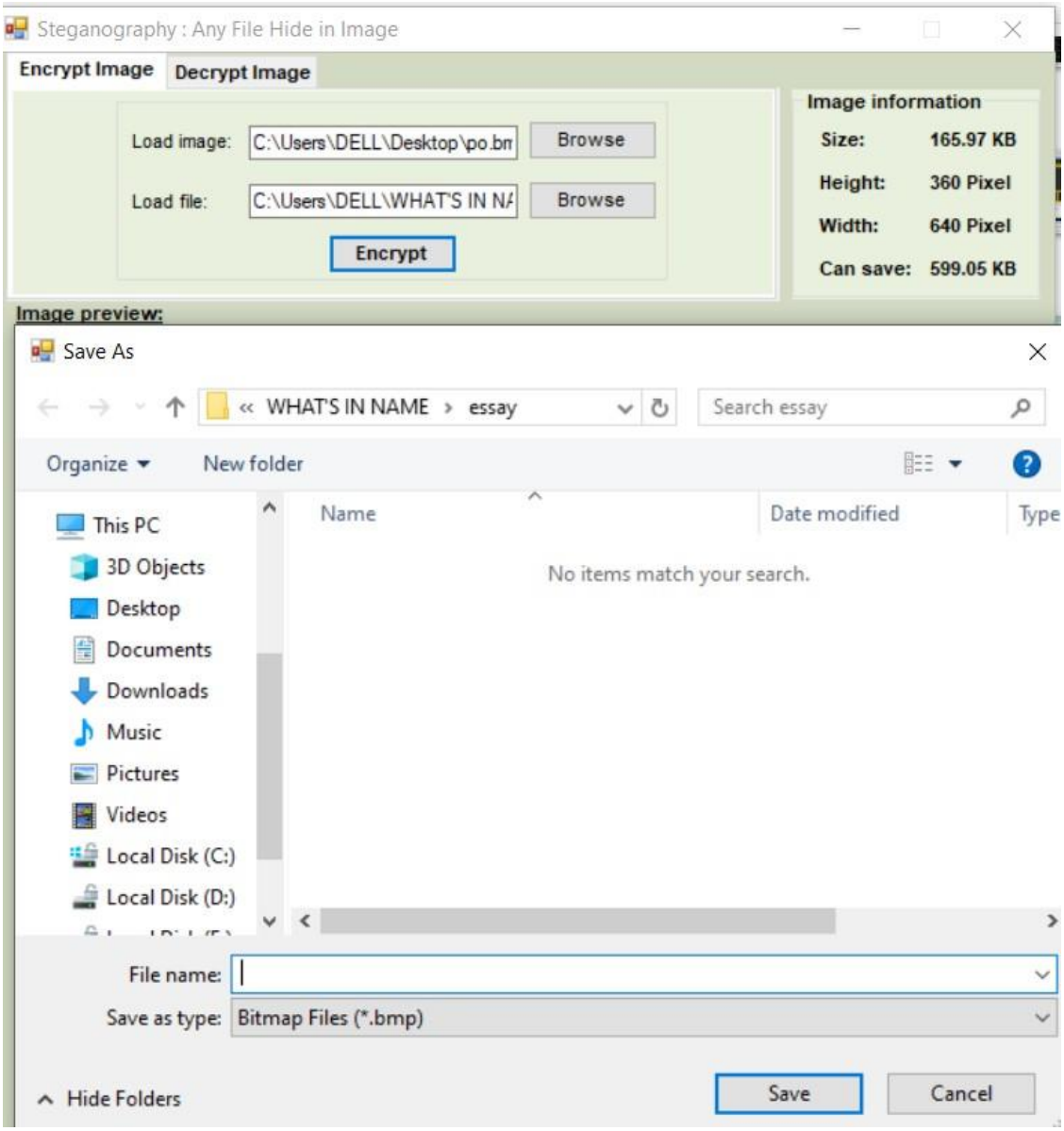
3. The image file will opened and is displays as follows. Next, click on “Browse” button that is next to the Load File textbox.

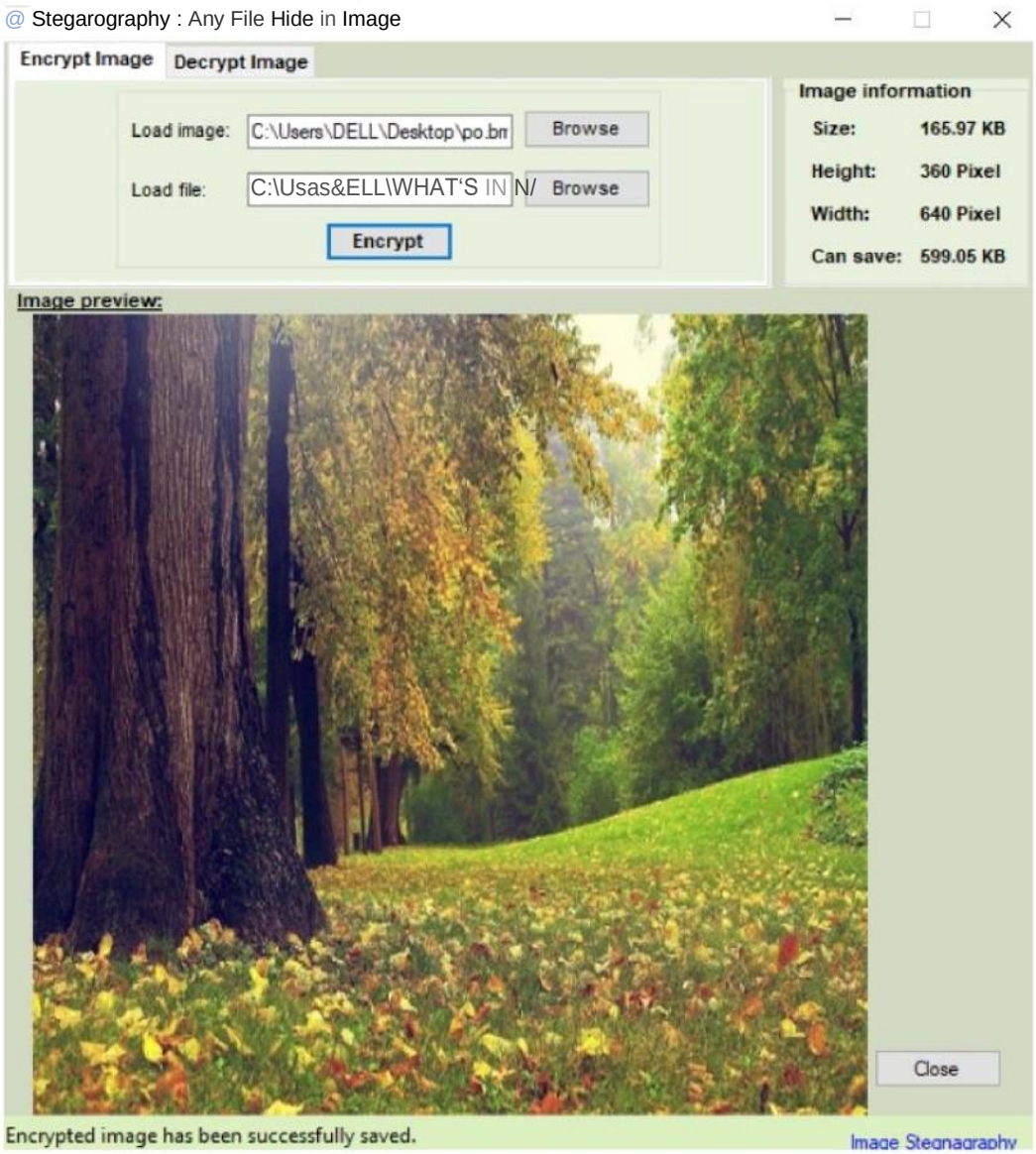


4. Again the file open dialog box will appear, select any type of file whatever you want to hide with the image and click on ok button.



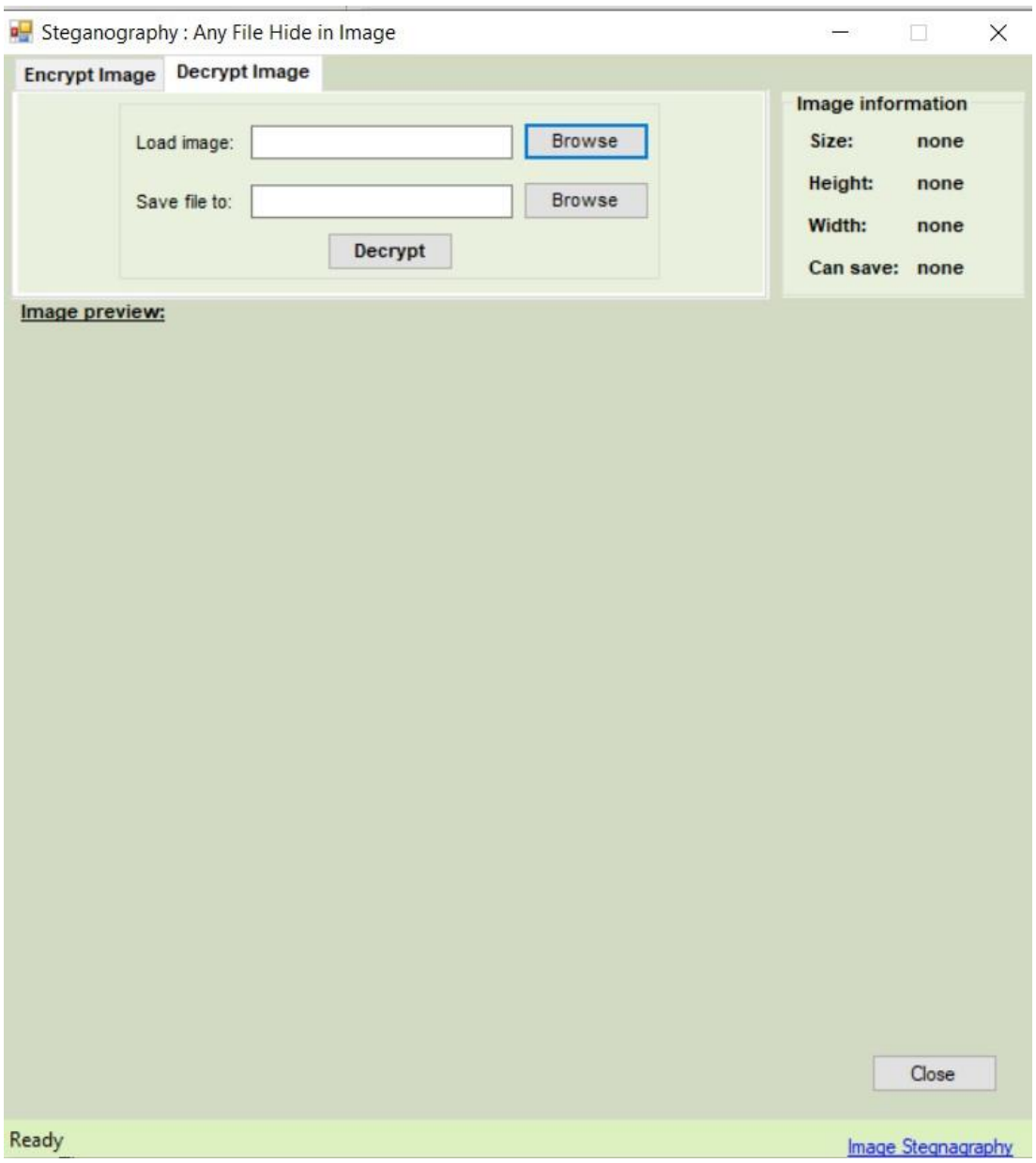
5. The next step is to encrypt the file. Now click on “Encrypt” button, it will open the save dialog box which ask you to select the path to save the New image file and the Image file name. The default format of image file is BMP.



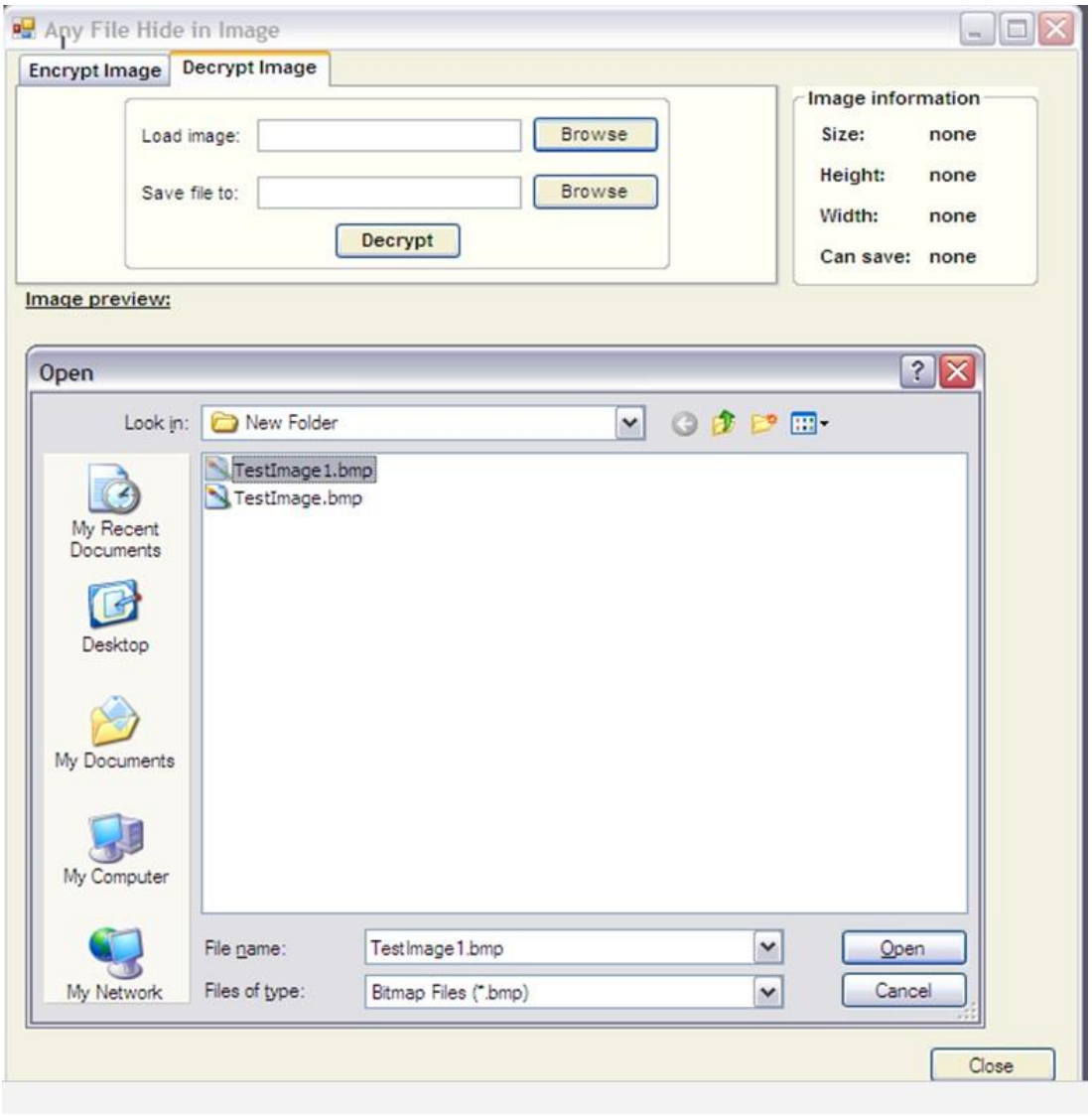

@ Stegarography : Any File Hide in Image
Encrypt Image
Decrypt Image
Load image:
C:\Users\DELL\Desktop\po.bn
Browse
Load file:
C:\Usas&ELL\WHAT'S IN N/
Browse
Encrypt
Image information
Size: 165.97 KB
Height: 360 Pixel
Width: 640 Pixel
Can save: 599.05 KB
Image preview:
Close
Encrypted image has been successfully saved.
Image Steganography

- ## Decryption

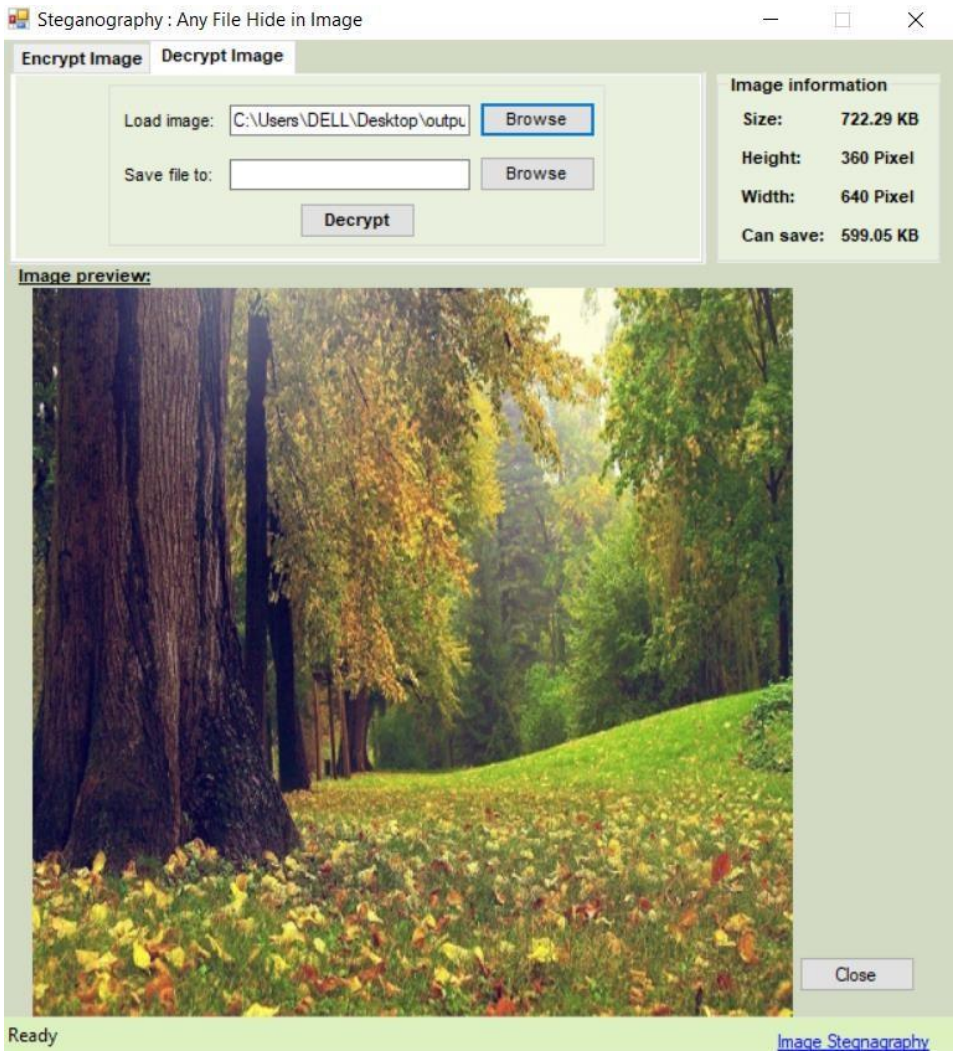
1. Select the Decryption Image tab option.



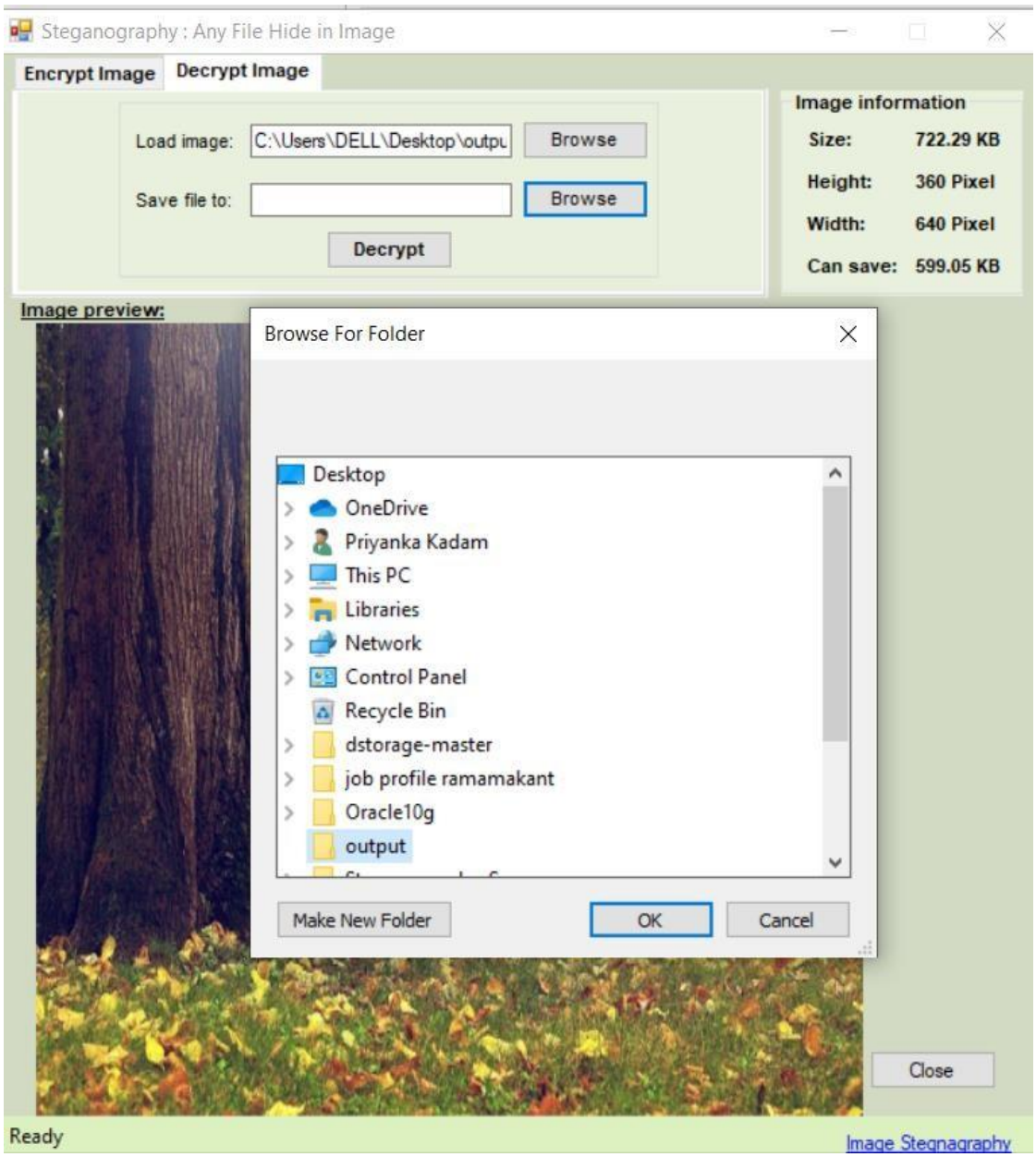
2. Next click on the “Browse” button, which open the Open file dialog box, here you have to select the image which is Encrypted and has hidden information file. Select the image file and click on Open button.



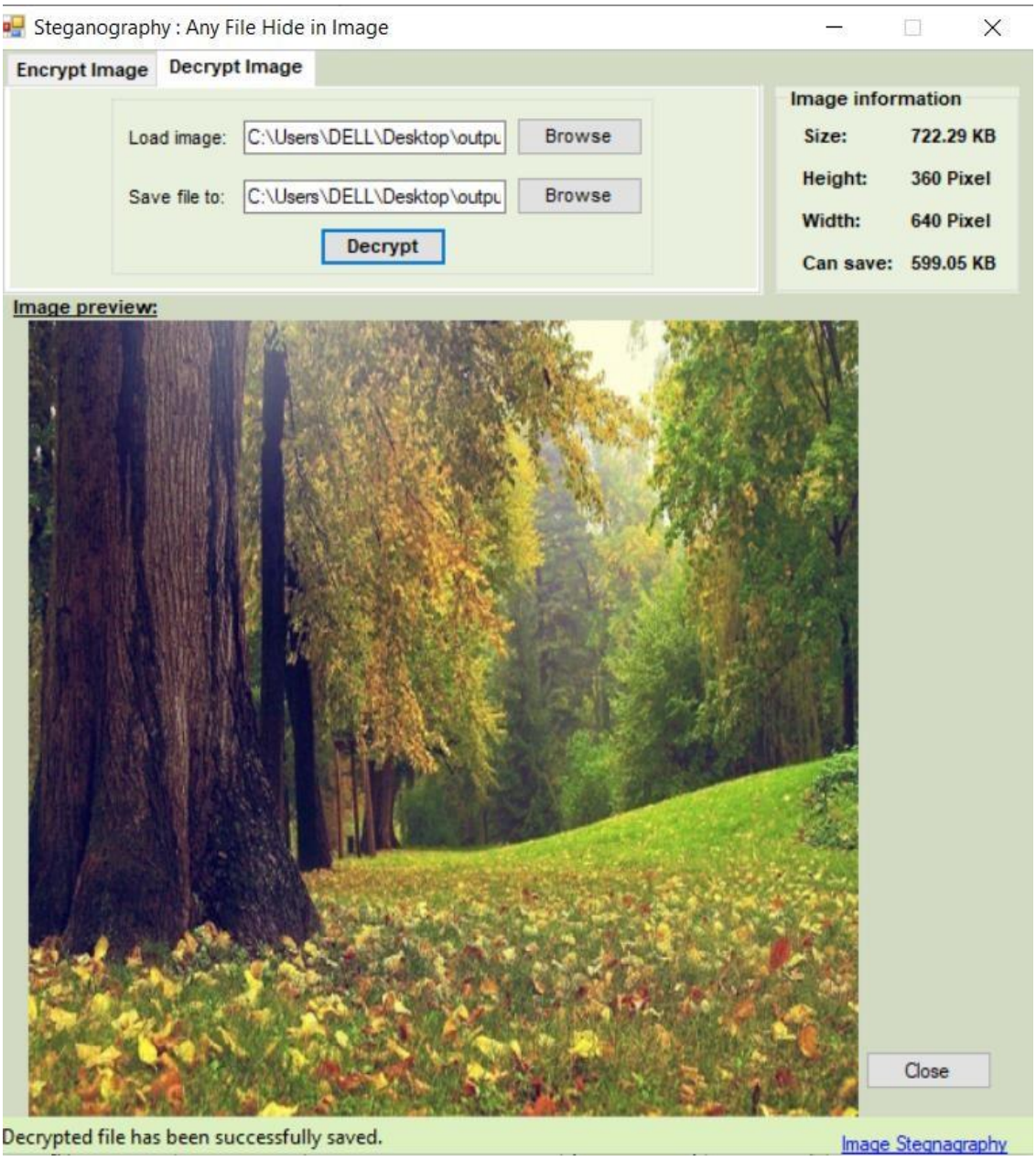
3. The image file displayed as follows –



4. Now click on “Browse” button which is next to “Save file to” textbox. It will open a dialog box that is “Browse for folder”. It ask you to select the path or folder, where you want to extract the hidden file. Select the folder and click on Ok button.



5. Now click on Decrypt button, it will decrypt the image, the hidden file and image file is saved into selected folder. The message for successful decryption is displayed on the status bar which is places at bottom of the screen.



XIV. 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.

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 their. 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.

Since ancient times, man has found a desire in the ability to communicate covertly. The recent explosion of research in watermarking to protect intellectual property is evidence that steganography is not just limited to military or espionage applications. Steganography, like cryptography, will play an increasing role in the future of secure communication in the “digital world”.

XV. Limitations of the Software -

This project has an assumption that is both the sender and receiver must have shared some secret information before imprisonment. Pure steganography means that there is none prior information shared by two communication parties.

XVI. Bibliography -

- http://www.google.com
- http://www.microsoft.com
- http://www.codeproject.com
- http://www.asp.net
- http://www.asp123.com
- http://www.wikipedia.org

XVIII. Future Enhancements -

To make it pure steganography application. May be there is still room for a lot of improvements, but will leave that for the future work on this topic.