Fall 2015

Steganography

Naresh Emmadi
Governors State University

Varun Kumar Kattekolu
Governors State University

Sudhakar Shamul
Governors State University

ABSTRACT

Multi Layer Security (MLS) 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 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 encryption, its uses and techniques. It also attempts to identify the requirements of a good algorithm and briefly reflects on which techniques are more suitable for applications.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Introduction to the project

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 our application MLS. MLS works on the of 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.

Due to advances in ICT, most of information is kept electronically. Consequently, the security of information has become a fundamental issue. Besides cryptography, our application 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.

1.2 Purpose of the project:

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

Steganography hide the secrete message within the host data set and presence imperceptible and is to be reliably communicated to a receiver. The host data set is purposely corrupted, but in a covert way, designed to be invisible to an information analysis.

1.2.1 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.

1.2.2 History of Steganography:

Through out history Steganography has been used to secretly communicate information between people.

Some examples of use of Steganography are 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 is 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. After the hair grew back the messenger was sent to deliver the message, the recipient would shave off the messengers hair to see the secrete message.

1.2.3 Why This Steganography?

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

1.2.4 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.

2.ANALYSIS

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

2.1 Technical Feasibility

The Technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

2.2 Operational Feasibility

2.2.1 Reliability

The package wills pick-up current transactions on line. Regarding the old transactions,

2.2.2 Security

The application has a dedicated username and password.

2.2.3 Portability

The application will be developed using C# .Net language etc these software will work both on Windows and Linux o/s. Hence portability problems will not arise.

2.2.4 Availability

This software will be available always.

2.2.5 Maintainability

The system called the ewheelz uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses My-SQL, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

2.3 ECONOMIC FEASIBILITY

The application takes care of the present existing system's data flow and procedures completely. It should be built as a stand alone application. This is required as the encryption/decryption can be done at any where possible.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1. INTRODUCTION

The project entitled "Multi Layer Security" has been developed using Java, Oracle, and Jdbc as backend technologies.

3.1.1 Purpose

The purpose of this Software Requirement Specification (SRS) is to help the project. It is provided with some requirements which are used in MLS. All parts; design, coding and testing will be prepared with helping of SRS. The purpose of this document is to detail the requirements used in the MLS and how the components of the system are to work with each other with external systems.

This document will be checked by group member's supervisor and it will corrected by members if supervisor orders.

3.2. FUNCTIONAL REQUIREMENTS:

The **Functional Requirement** document (also called Functional Specifications or Functional Requirement Specifications), defines the capabilities and functions that system must be able to perform successfully.

Functional Requirements should include:

- Descriptions of data to be entered into the system
- Descriptions of operations performed by each screen
- Descriptions of work-flows performed by the system
- Descriptions of system reports or other outputs
- Who can enter the data into the system.
- How the system meets applicable regulatory requirements

The functional specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

3.3. NON- FUNCTIONAL REQUIREMENTS:

A Non-Functional Requirement is usually some form of constraint or restriction that must be considered when designing the solution.

For the most part when people are talking about Constraints, they are referring to Non-Functional Requirements. Non-Functional Requirements have the same following characteristics:

- uses simple language
- not ambiguous
- contains only one point
- specific to one type of user
- is qualified
- describes what and not how

Non-Functional requirements tend to identify "user" constraints and "system" constraints. Business requirements should be kept pure and not reflect any solution thinking.

A system constraint is a constraint imposed by the system and not dictated by a Business Need. Since system constraints are part of a "solution", they should be documented in the System Specifications document.

These are the mainly following:

- 24 X 7 availability
- Better component design to get better performance at peak time
- Flexible service based architecture will be highly desirable for future extension

3.4. PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements.

It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

  The existing system is completely dependent on the user to perform all the duties

3.5 HARDWARE REQUIREMENTS:

Hardware	:	Pentium IV systems or above
RAM	:	512MB (minimum)
Hard Drive space	:	8GB

3.6 SOFTWARE REQUIREMENTS:

Operating System	:	Windows xp
Technology	:	C# .Net

4. SOFTWARE DEVELOPMENT ENVIRONMENT

4.1 C# .Net

C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure.

.NET is built on the Windows Server System to take major advantage of the OS and which comes with a host of different servers which allows for building, deploying, managing and maintaining Web-based solutions. The Windows Server System is designed with performance as priority and it provides scalability, reliability, and manageability for the global, Web-enabled enterprise. The Windows Server System integrated software products are built for interoperability using open Web standards such as XML and SOAP.

.NET is a "Software Platform". It is a language-neutral environment for developing rich .NET experiences and building applications that can easily and securely operate within it. When developed applications are deployed, those applications will target .NET and will execute wherever .NET is implemented instead of targeting a particular Hardware/OS combination. The components that make up the .NET platform are collectively called the .NET Framework.

The .NET Framework is a managed, type-safe environment for developing and executing applications. The .NET Framework manages all aspects of program execution, like, allocation of memory for the storage of data and instructions, granting and denying permissions to the application, managing execution of the application and reallocation of memory for resources that are not needed.

The .NET Framework is designed for cross-language compatibility. Cross-language compatibility means, an application written in Visual Basic .NET may reference a DLL file written in C# (C-Sharp). A Visual Basic .NET class might be derived from a C# class or vice versa.

The .NET Framework consists of two main components:

- Common Language Runtime (CLR)
- Class Libraries

The advantage of C# includes

1. Powerful Windows-based Applications
2. Building Web-based Applications
3. Simplified Deployment
    - Powerful, Flexible, Simplified Data Access
    - Improved Coding
    - Direct Access to the Platform
    - Full Object-Oriented Constructs
    - XML Web Services
    - COM Interoperability

The ECMA standard lists these design goals for C#:

- C# language is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Source code portability is very important, as is programmer portability, especially for those programmers already familiar with C and C++.
- Support for internationalization is very important.

- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.
- Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

4.2 Features:

- It has no global variables or functions. All methods and members must be declared within classes. Static members of public classes can substitute for global variables and functions.
- Local variables cannot shadow variables of the enclosing block, unlike C and C++. Variable shadowing is often considered confusing by C++ texts.
- C# supports a strict Boolean data type, bool. Statements that take conditions, such as while and if, require an expression of a type that implements the true operator, such as the boolean type. While C++ also has a boolean type, it can be freely converted to and from integers, and expressions such as if(a) require only that a is convertible to bool, allowing a to be an int, or a pointer. C# disallows this "integer meaning true or false" approach, on the grounds that forcing programmers to use expressions that return exactly bool can prevent certain types of common programming mistakes in C or C++ such as if (a = b) (use of assignment = instead of equality ==).
- In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined null value; it is impossible to obtain a reference to a "dead" object (one that has been garbage collected), or to a random block of memory. An unsafe pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack. Code that is not marked as unsafe can still store and manipulate pointers through the System.IntPtr type, but it cannot dereference them.

- Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem of memory leaks by freeing the programmer of responsibility for releasing memory that is no longer needed.
- In addition to the try...catch construct to handle exceptions, C# has a try...finally construct to guarantee execution of the code in the finally block.
- Multiple inheritance is not supported, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication and simplify architectural requirements throughout CLI.
- C#, like C++, but unlike Java, supports operator overloading.
- C# is more type safe than C++. The only implicit conversions by default are those that are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime. No implicit conversions occur between booleans and integers, nor between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type). Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default. Starting with version 4.0, C# supports a "dynamic" data type that enforces type checking at runtime only.
- Enumeration members are placed in their own scope.
- C# provides properties as syntactic sugar for a common pattern in which a pair of methods, accessor (getter) and mutator (setter) encapsulate operations on a single attribute of a class.
- Checked exceptions are not present in C# (in contrast to Java). This has been a conscious decision based on the issues of scalability and versionability.[28]
- Though primarily an imperative language, since C# 3.0 it supports functional programming techniques through first-class function objects and lambda expressions.

5. DESIGN

Microsoft .Net framework prepares a huge amount of tool and options for programmers that they simples programming. 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).

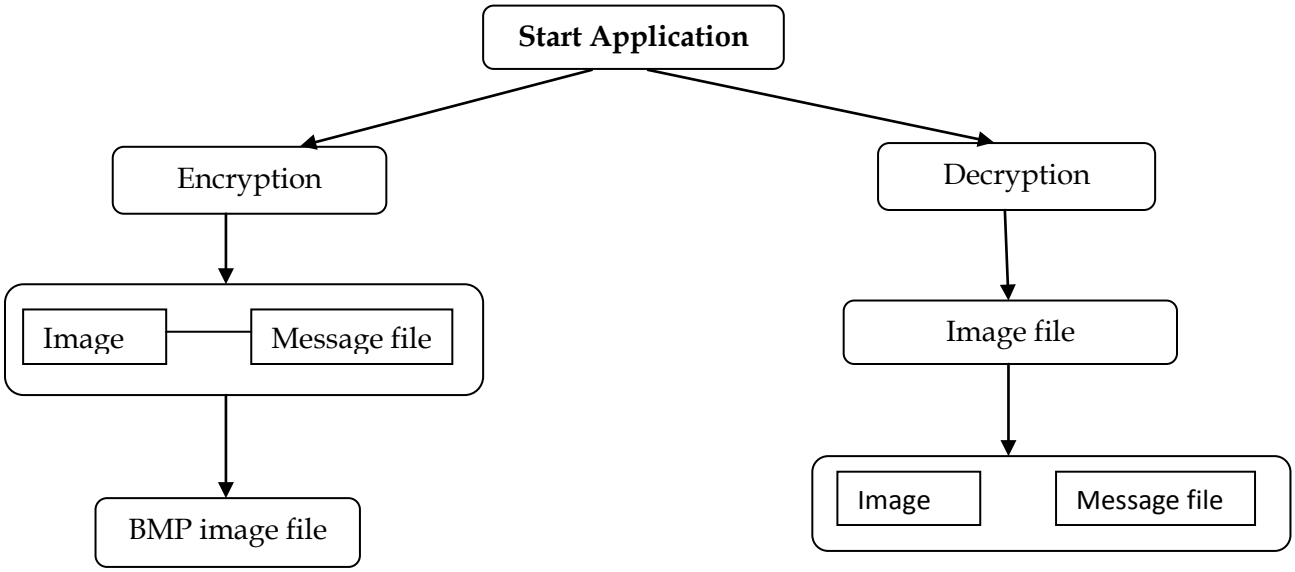
The algorithm used for Encryption and Decryption in this application provides using several layers lieu of using only LSB layer of image. Writing data starts from last layer (8st or LSB layer); because significant of this layer is least and every upper layer has doubled significant from its down layer.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



5.1 Detecting Steganography:

The art of detecting Steganography is referred to as **Steganalysis.**

To put is simply Steganalysis involves detecting the use of Steganography inside of a file. Steganalysis does not deal with trying to decrypt the hidden information inside of a file, just discovering it.

There are many methods that can be used to detect Steganography such as:

“Viewing the file and comparing it to another copy of the file found on the Internet (Picture file). There are usually multiple copies of images on the internet, so you may want to look for several of them and try and compare the suspect file to them. For example if you download a JPEG and your suspect file is also a JPEG and the two files look almost identical apart from the fact that one is larger than the other, it is most probable you suspect file has hidden information inside of it.

5.2 Problem Statement*:*

The former consists of linguistic or language forms of hidden writing. The later, such as invisible ink, try of 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 the internet technology, digital media can be transmitted conveniently over the network. Therefore, messages can be secretly carried by digital media by using the steganography techniques, and then be transmitted through the internet rapidly

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.

So we prepare this application, to make the information hiding more simple and user friendly.

5.3 Objective

The goal of steganography is covert communication. So, a fundamental requirement of this steganography system is that the hider message carried by stego-media should not be sensible to human beings.

The other goad of steganography is to avoid drawing suspicion to the existence of a hidden message. 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.

5.4 Overview

The word steganography comes from the Greek “Seganos”, which mean covered or secret and – “graphy” mean writing or drawing. Therefore, 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 secrete 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.

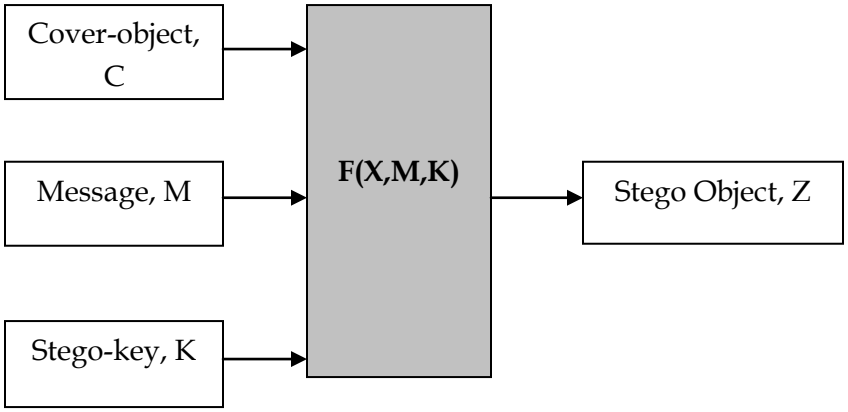
The main goal of this projects it to communicate securely in a completely undetectable manner and to avoid drawing suspicion to the transmission of a hider data. There has been a rapid growth of interest in steganography for two reasons:

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

5.5 Steganography vs Cryptography:

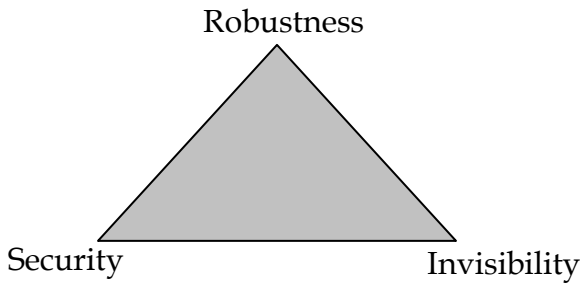
Basically, the purpose of cryptography and steganography is to provide secret communication. However, steganography is not the same as cryptography. Cryptography hides the contents of a secrete message from a malicious people, whereas steganography even conceal the existence of the message. In cryptography, 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.

5.6 Steganography vs Watermarking:

Steganography pay attention to the degree of Invisibility while watermarking pay most of its attribute to the robustness of the message and its ability to withstand attacks of removal, such as image operations(rotation, cropping, filtering), audio operations(rerecording, filtering)in the case of images and audio files being watermarked respectively.

It is a non-questionable fact that delectability of a vessel with an introduced data (steganographic message or a watermark) is a function of the changeability function of the algorithm over the vessel.



That is the way the algorithm changes the vessel and the severity of such an operation determines with no doubt the delectability of the message, since delectability is a function of file characteristics deviation from the norm, embedding operation attitude and change severity of such change decides vessel file delectability.

A typical triangle of conflict is message Invisibility, Robustness, and Security. Invisibility is a measure of the in notability of the contents of the message within the vessel.

Security is sinominous to the cryptographic idea to message security, meaning inability of reconstruction of the message without the proper secret key material shared.

Robustness refers to the endurance capability of the message to survive distortion or removal attacks intact. It is often used in the watermarking field since watermarking seeks the persistence of the watermark over attacks, steganographic messages on the other hand tend to be of high sensitivity to such attacks. The more invisible the message is the less secure it is (cryptography needs space) and the less robust it is (no error checking/recovery introduced).The more robust the message is embedded the more size it requires and the more visible it is.

5.7 Steganography Techniques:

Over the past few years, numerous steganography techniques that embed hidden messages in multimedia objects have been proposed. There have been many techniques for hiding information or messages in images in such a manner that alteration made to the image is perceptually indiscernible. Commonly approaches are include LSB, Masking and filtering and Transform techniques.

Least significant bit (LSB) insertion is a simple approach to embedding information in image file. The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover-image in a

deterministic sequence. Modulating the least significant bit does not result in human perceptible difference because the amplitude of the change is small. In this technique, the embedding capacity can be increased by using two or more least significant bits. At the same time, not only the risk of making the embedded message statistically detectable increase but also the image fidelity degrades. Hence a variable size LSB embedding schema is presented, in which the number of LSBs used for message embedding/extracting depends on the local characteristics of the pixel. The advantage of LSB-based method is easy to implement and high message pay-load.

Although LSB hides the message in such way that the humans do not perceive it, it is still possible for the opponent to retrieve the message due to the simplicity of the technique. Therefore, malicious people can easily try to extract the message from the beginning of the image if they are suspicious that there exists secret information that was embedded in the image.

Therefore, a system named Secure Information Hiding System (SIHS) is proposed to improve the LSB scheme. It overcomes the sequence-mapping problem by embedding the massage into a set of random pixels, which are scattered on the cover-image.

Masking and filtering techniques, usually restricted to 24 bits and gray scale image, hide information by marking an image, in a manner similar to paper watermarks. The technique perform analysis of the image, thus embed the information in significant areas so that the hidden message is more integral to cover image than just hiding it in the noise level.

Transform techniques embed the message by modulating coefficient in a transform domain, such as the Discrete Fourier Transform, or Wavelet Transform. These methods hide messages in significant

areas of the cover image, which make them more robust to attack. Transformations can be applied over the entire image, to block throughout the image, or other variant.

5.8 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.

5.9 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)

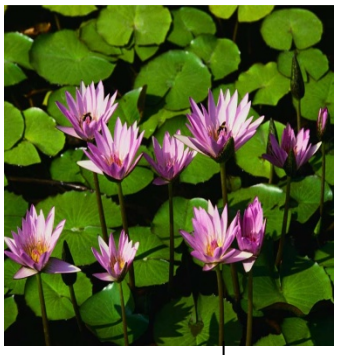
(10100110 11000101 00001100)

(11010010 10101100 01100011)

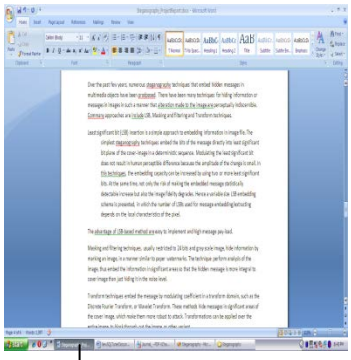
6 Encryption & Decryption Process

6.1 Encryption Process :

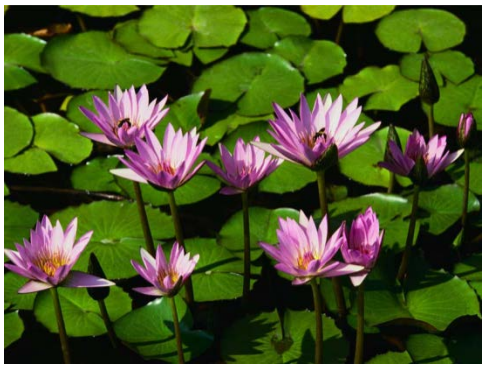
Image File



Information File

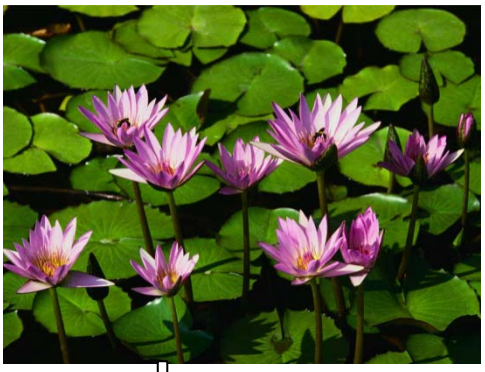


Encrypted Image File



6.2 Decryption Process

Encrypted Image File



Information File

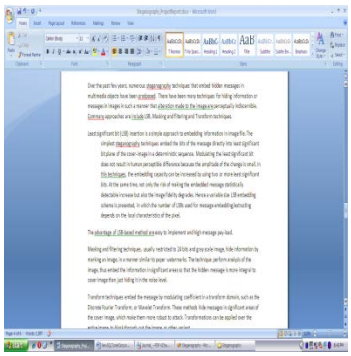
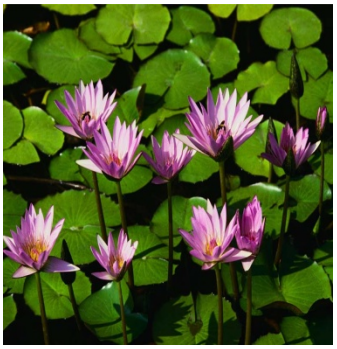


Image File



7 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("http:\\\\www.programmer2programmer.net");
        }
    }
}
```

8. SYSTEM TESTING

8.1 Introduction to Testing:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. It is the major quality measure employed during software development.

Testing is the exposure of the system to trial input to see whether it produces correct output. It is a process, which reveals errors in the program. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

8.2 Testing Phases:

Software testing phases include the following:

1. Test activities are determined and test data selected.
2. The test is conducted and test results are compared with the expected results.

There are various types of Testing:

8.2.1 Unit Testing:

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program.

8.2.2 Integration Testing:

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules.

8.2.3 System Testing:

It is mainly used if the software meets its requirements. The reference document for this process is the requirement document.

8.2.4 Acceptance Testing:

It is performed with realistic data of the client to demonstrate that the software is working satisfactorily.

8.3 Testing Methods :

Testing is a process of executing a program to find out errors. If testing is conducted successfully, it will uncover all the errors in the software. Any testing can be done basing on two ways:

8.3.1 White Box Testing:

It is a test case design method that uses the control structures of the procedural design to derive test cases. In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

Using this testing a Software Engineer can derive the following test cases:

Exercise all the logical decisions on either true or false sides. Execute all loops at their boundaries and within their operational boundaries. Exercise the internal data structures to assure their validity.

White Box testing attempts to find errors in the following categories:

- Guarantee that all independent paths have been executed.
- Execute all logical decisions on their true and false Sides.
- Execute all loops at their boundaries and within their operational bounds
- Execute internal data structures to ensure their validity.

8.3.2 Black Box Testing:

It is a test case design method used on the functional requirements of the software. In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program It will help a software engineer to derive sets of input conditions that will exercise all the functional requirements of the program.

Black Box testing attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access

- Performance errors
- Initialization and termination errors

8.4 TESTING APPROACH:

Testing can be done in two ways:

- Bottom up approach
- Top down approach

8.4.1 Bottom up Approach:

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

8.4.2 Top down approach:

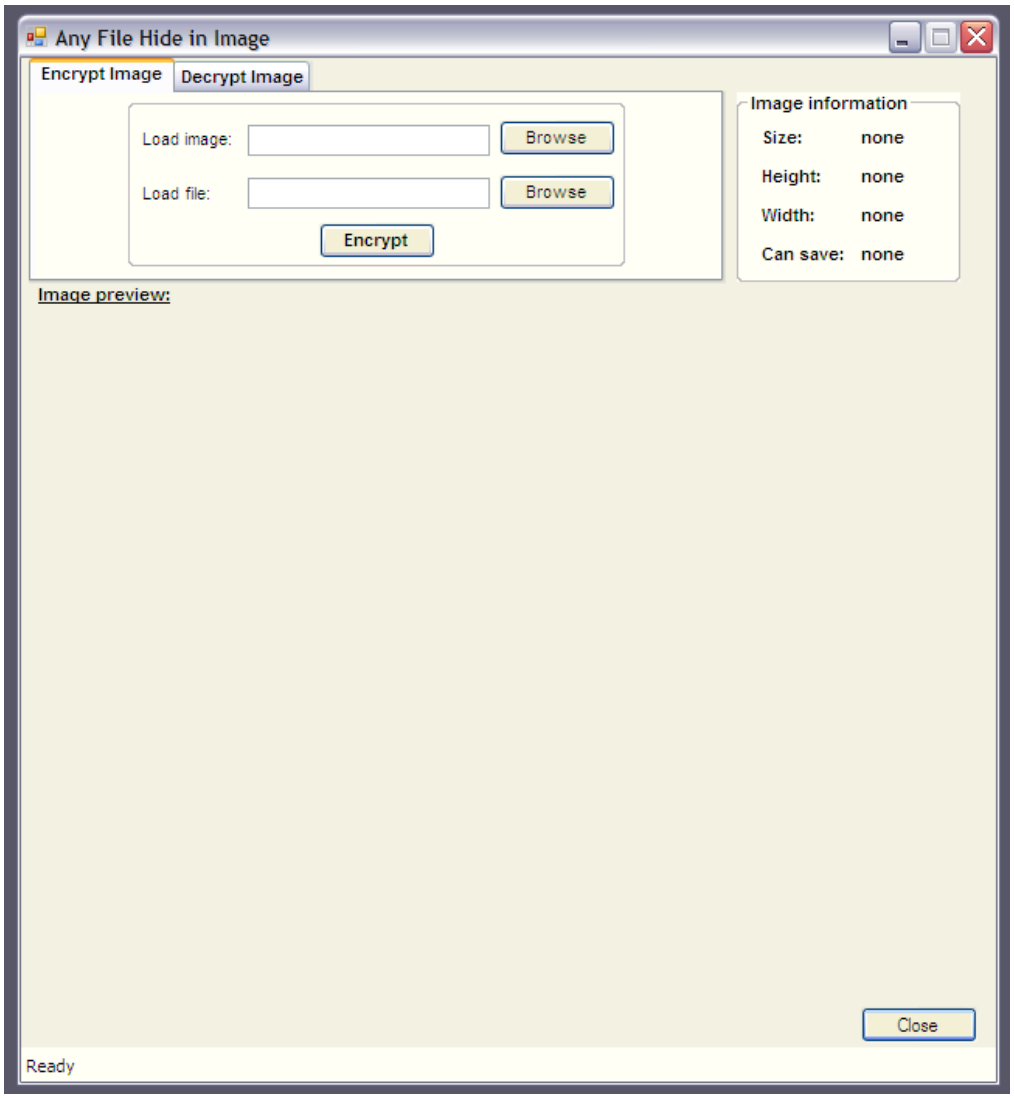
This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module

8.4.3 Validation:

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

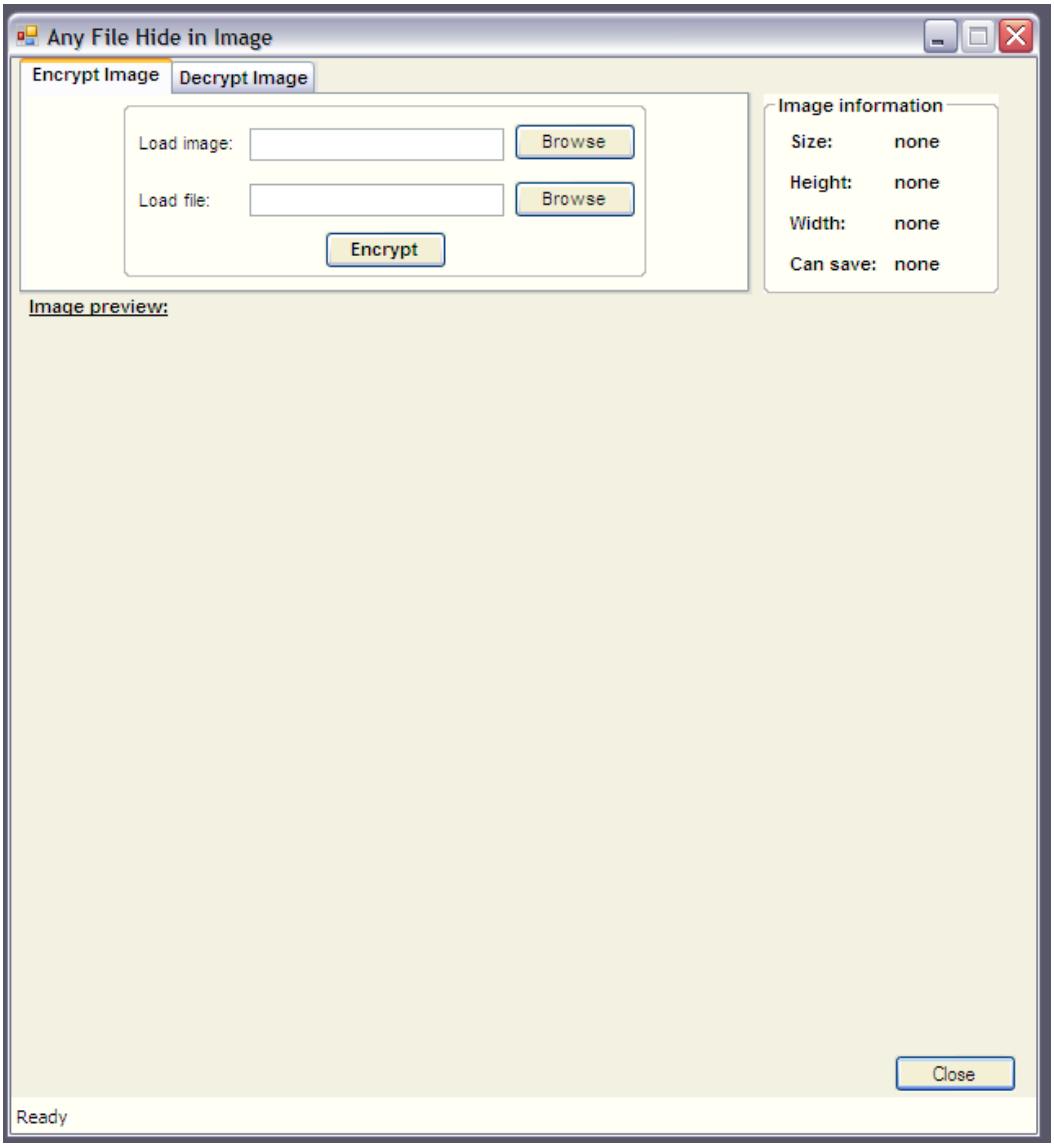
9. Inputs & Output Screens

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.

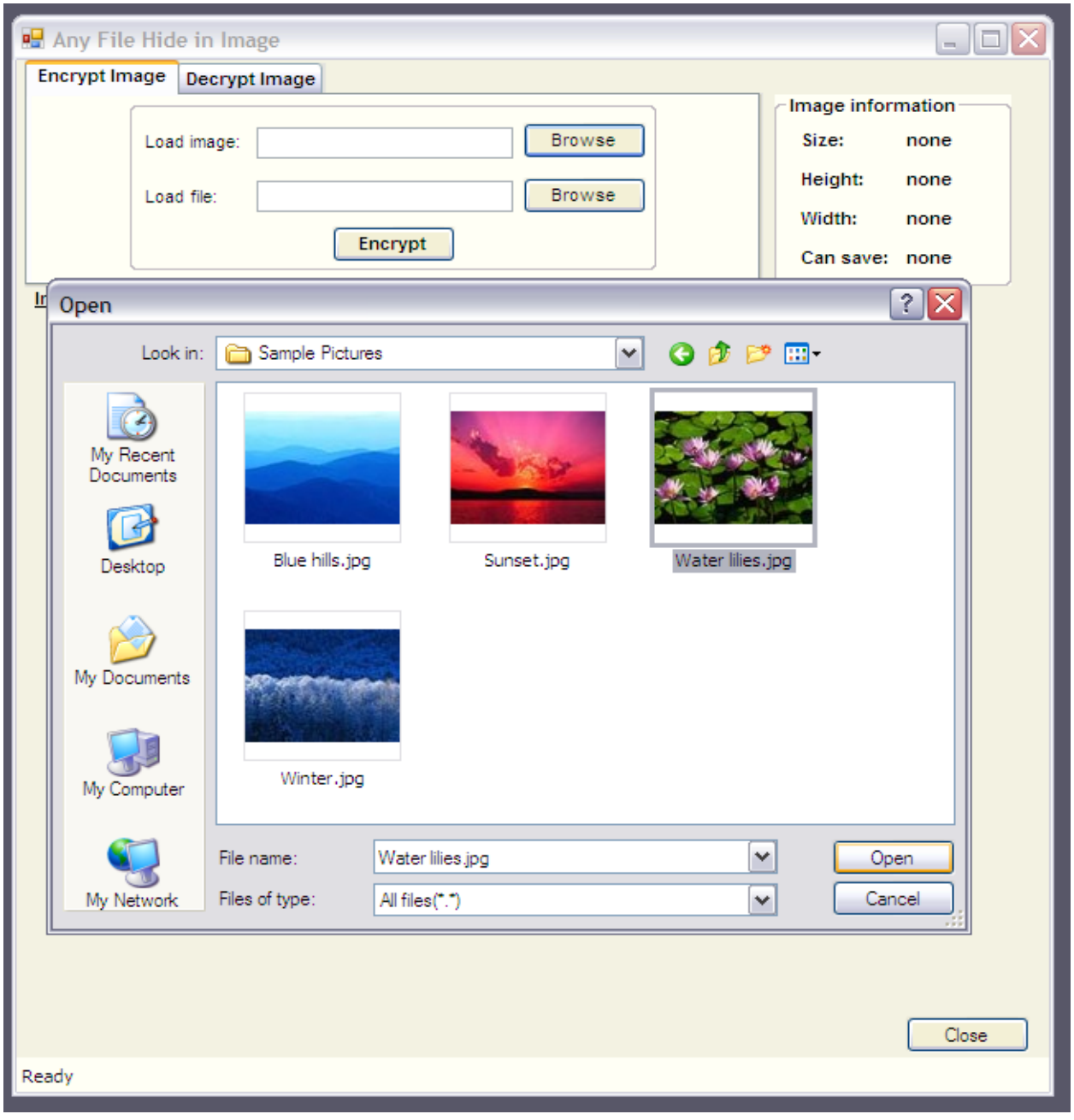


9.1 Encryption

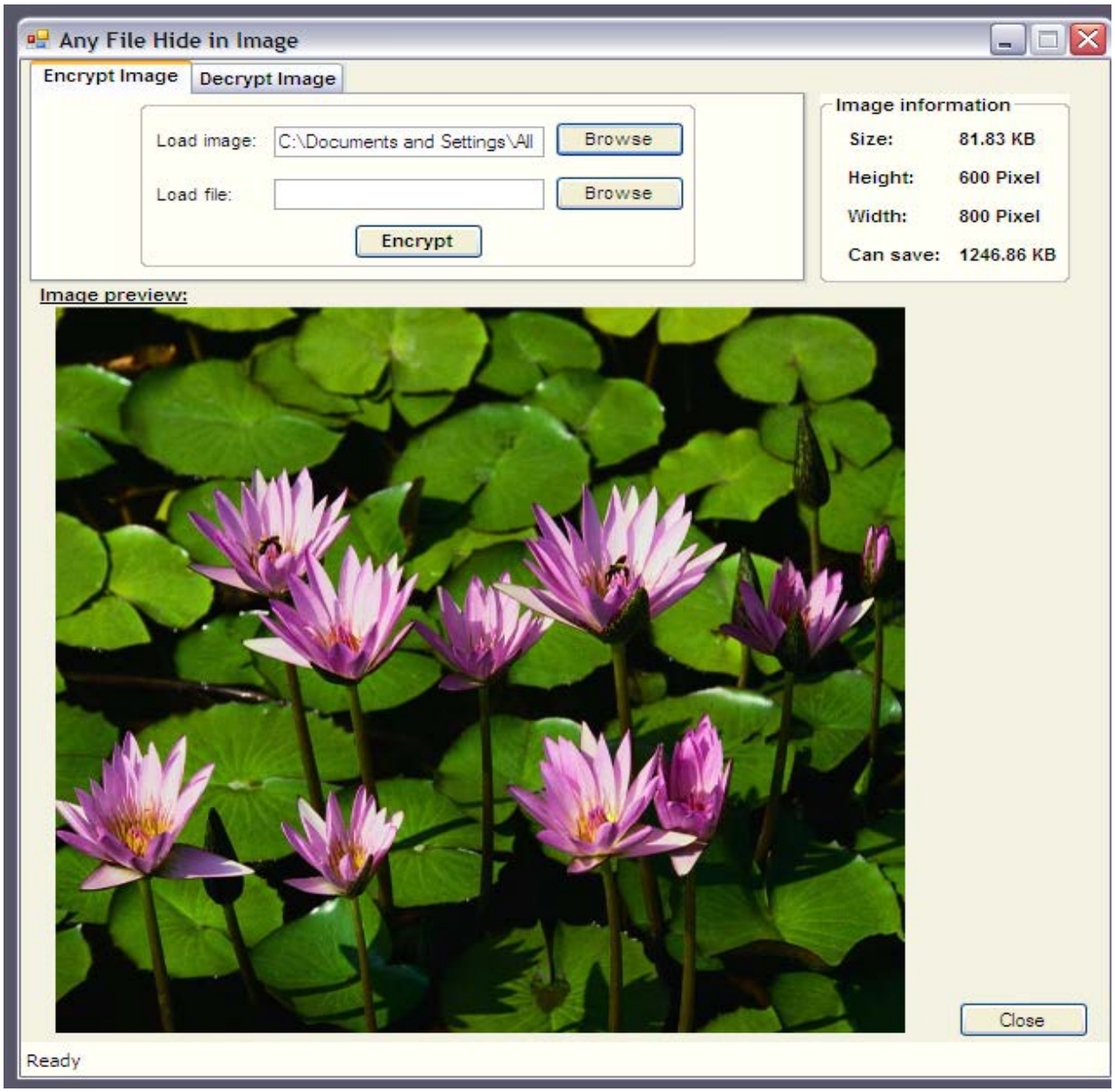
1. For Encryption select Encrypt Image tab option.



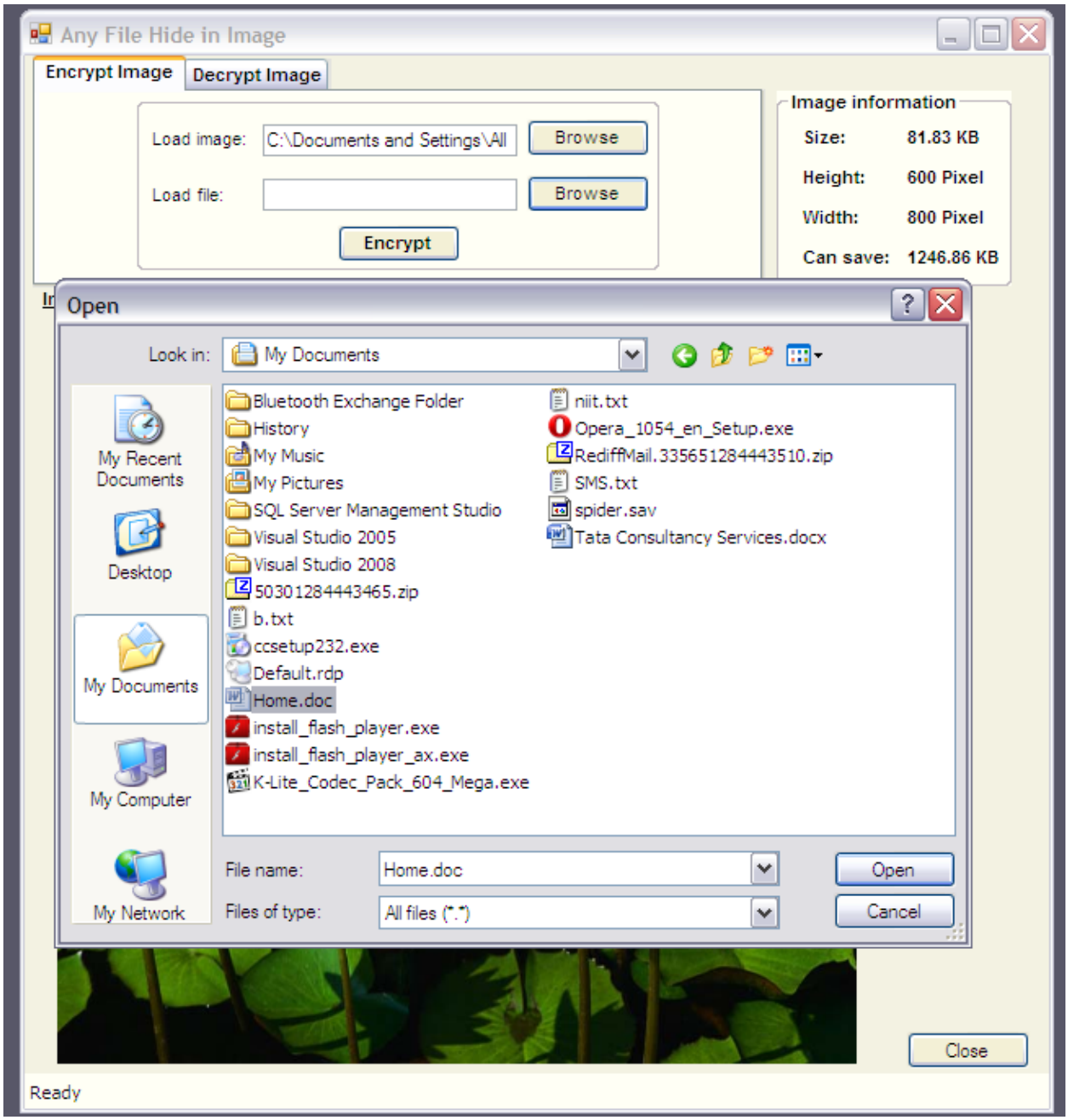
2. For load image click on button “Browse” that is next to the Load Image textbox. The file open dialog box will displays as follows, select the Image file, which you want to use hide information and click on Open button.



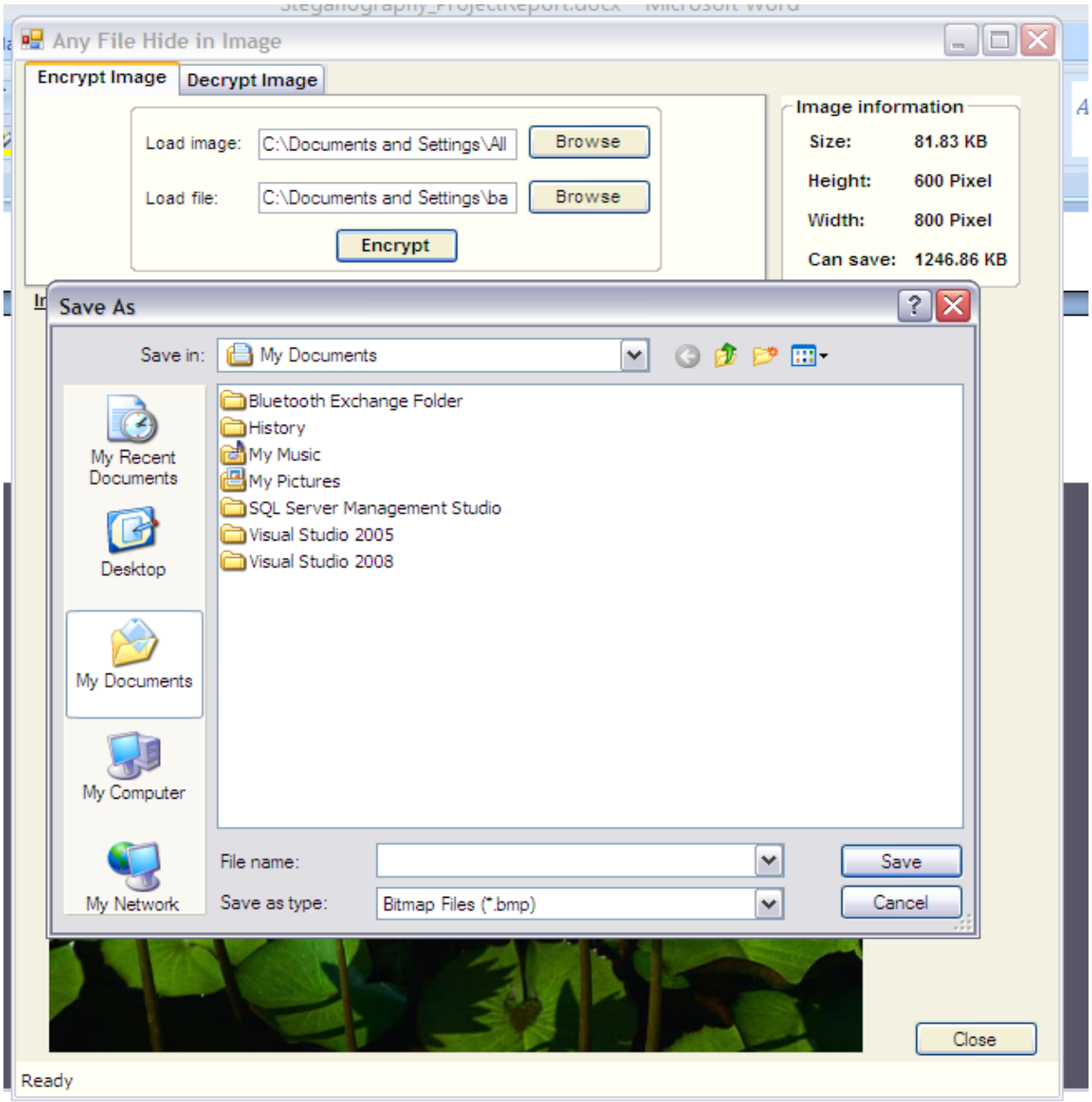
1. The image file will opened and is displays as follows. Next, click on “Browse” button that is next to the Load File textbox.

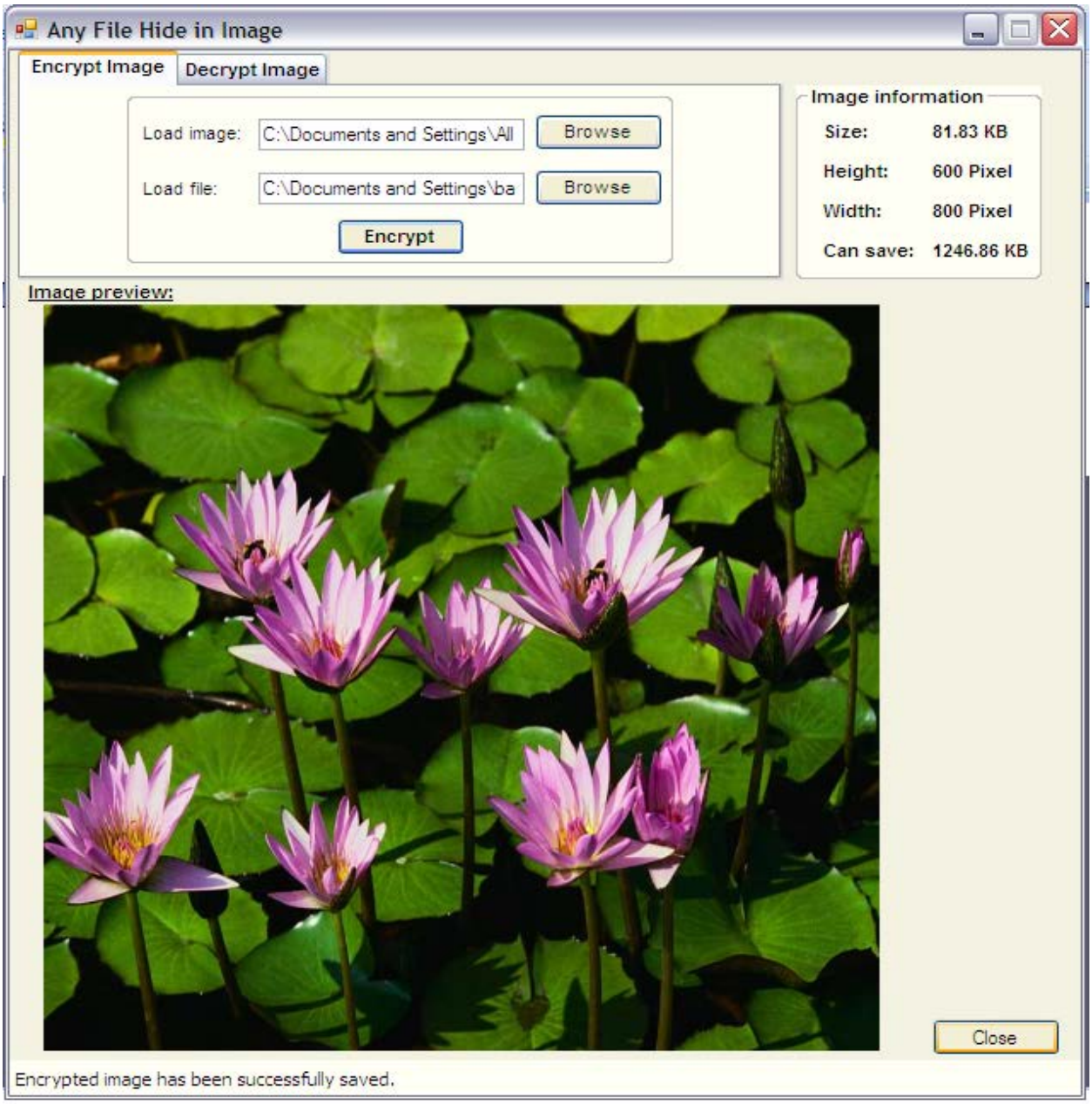


2. Again the file open dialog box will appear, select any type of file whatever you want to hide with the image and click on ok button.



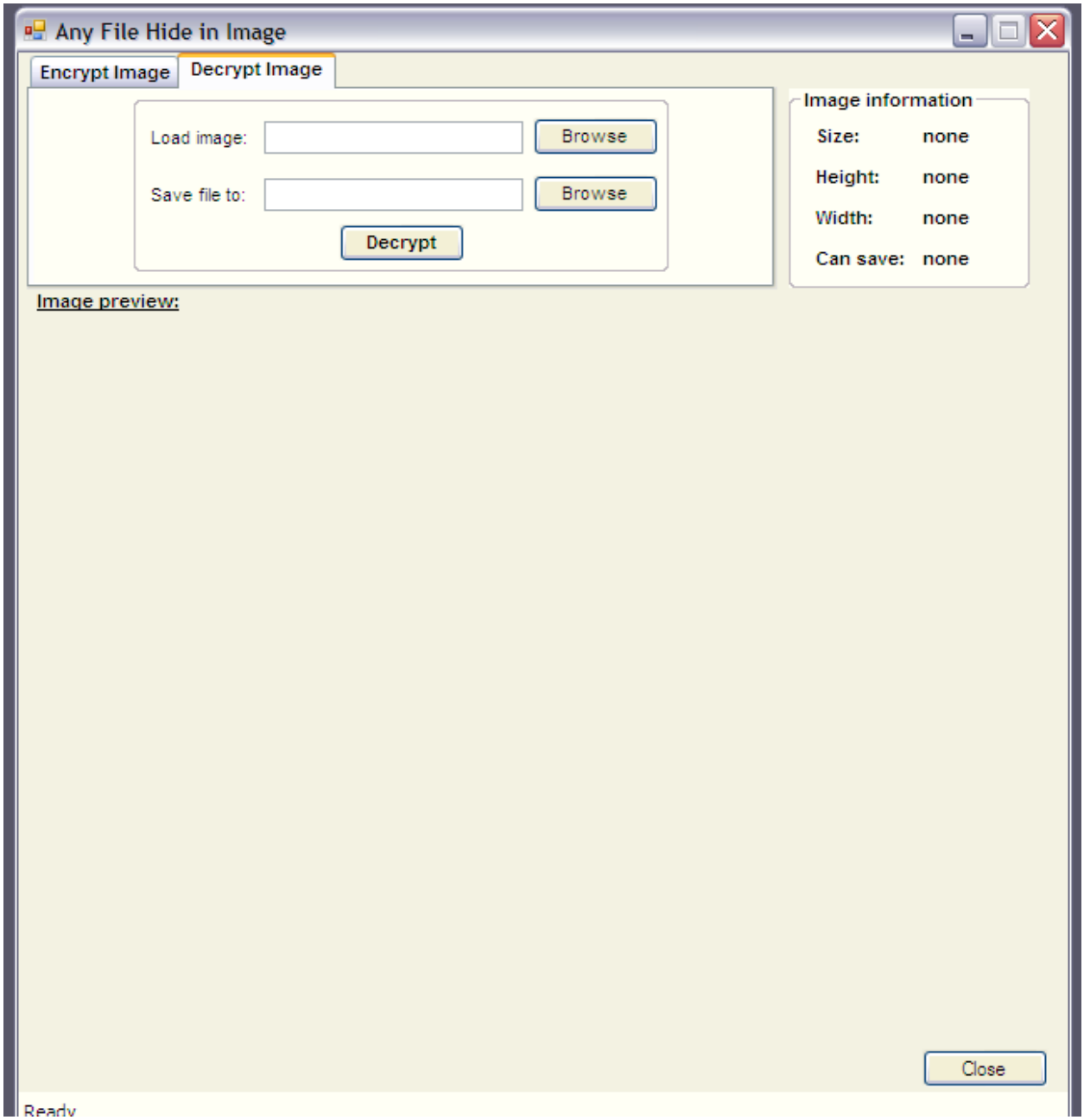
3. The next step is to encrypt the file. Now click on “Encrypt” button, it will open the save dialog box which ask you to select the path to save the New image file and the Image file name. The default format of image file is BMP.



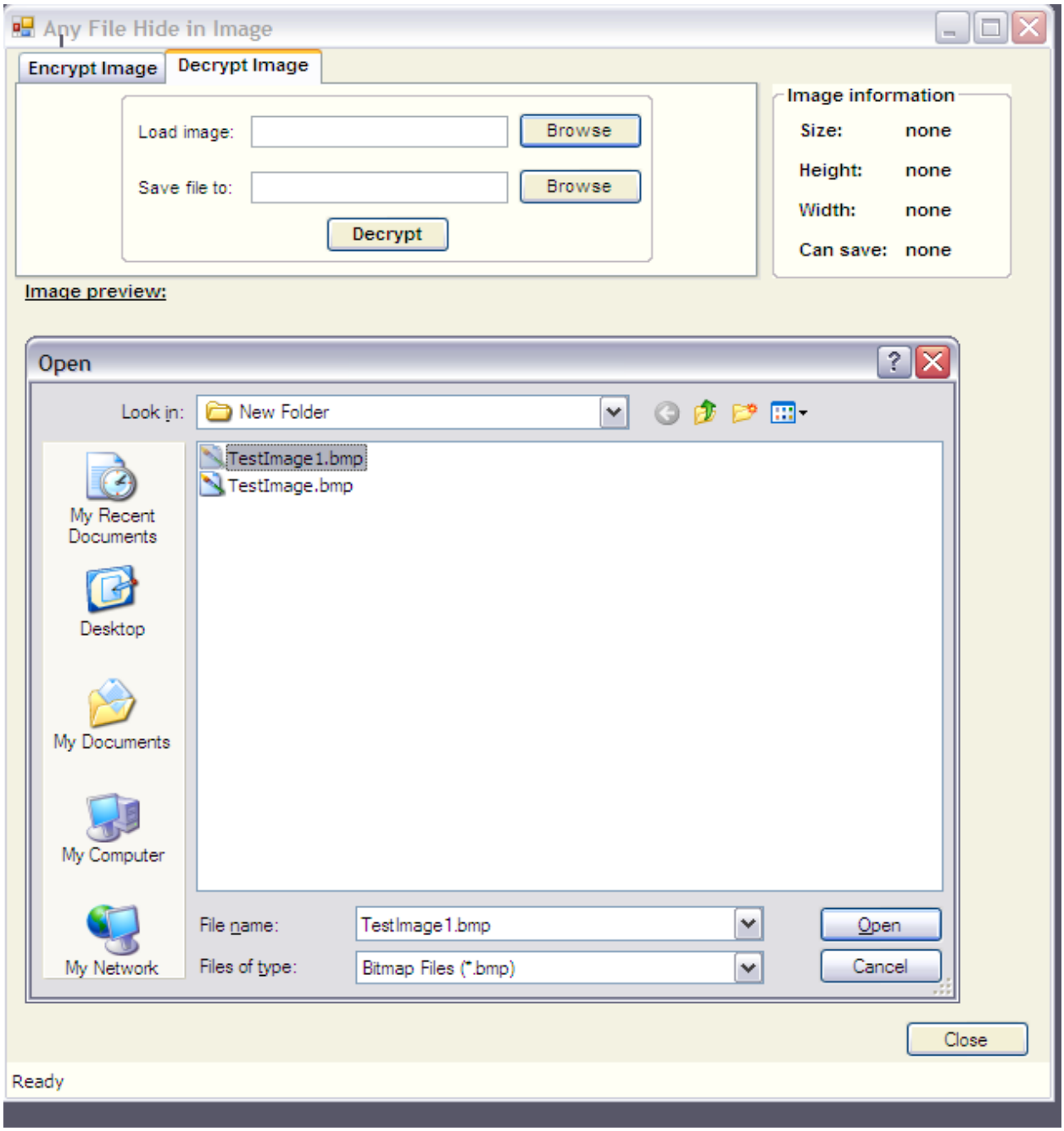

Any File Hide in Image
Encrypt Image
Decrypt Image
Load image:
C:\Documents and Settings\All
Browse
Load file:
C:\Documents and Settings\ba
Browse
Encrypt
Image information
Size:
81.83 KB
Height:
600 Pixel
Width:
800 Pixel
Can save:
1246.86 KB
Image preview:
Close
Encrypted image has been successfully saved.

9.2 Decryption

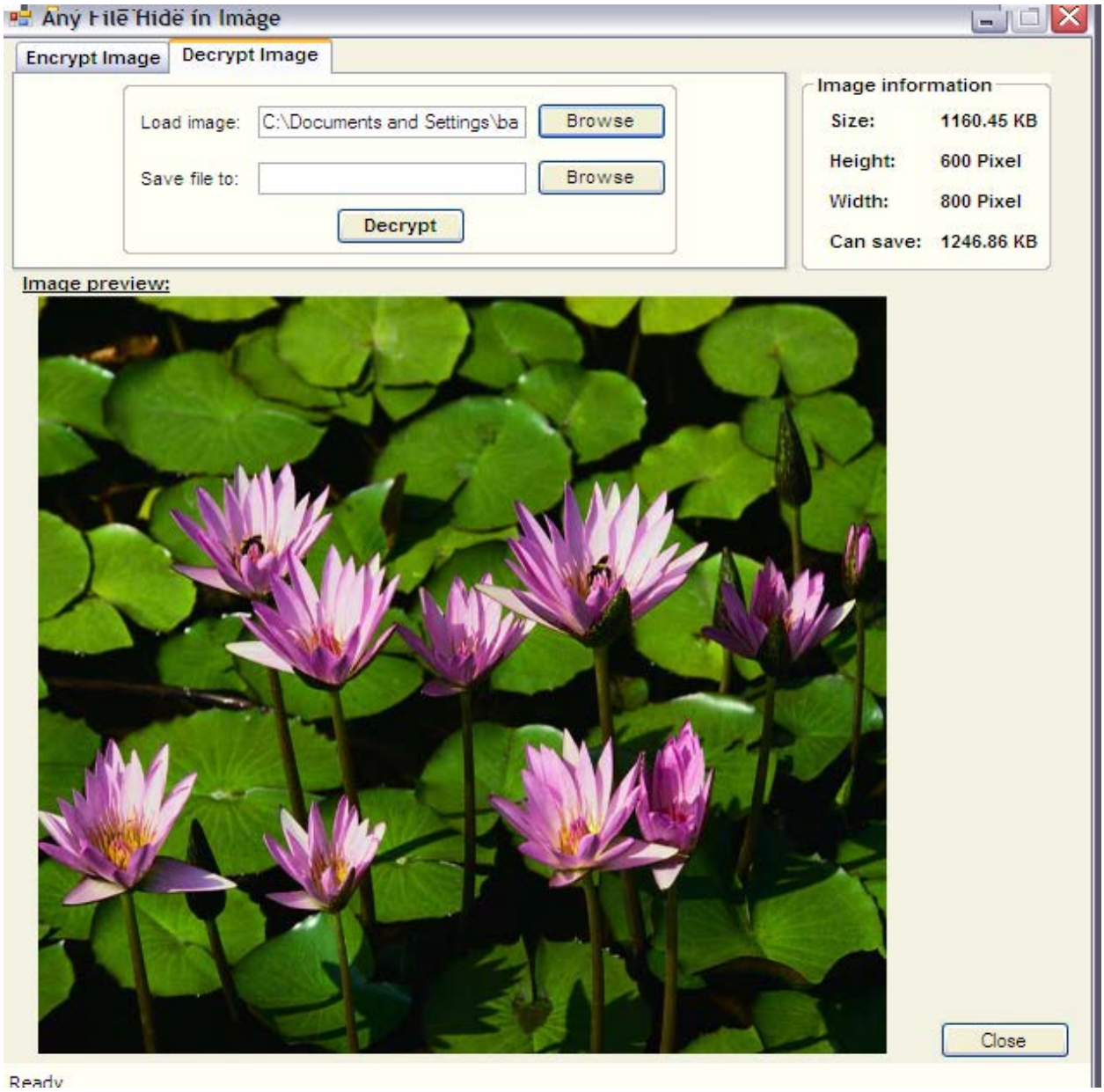
1. Select the Decryption Image tab option.



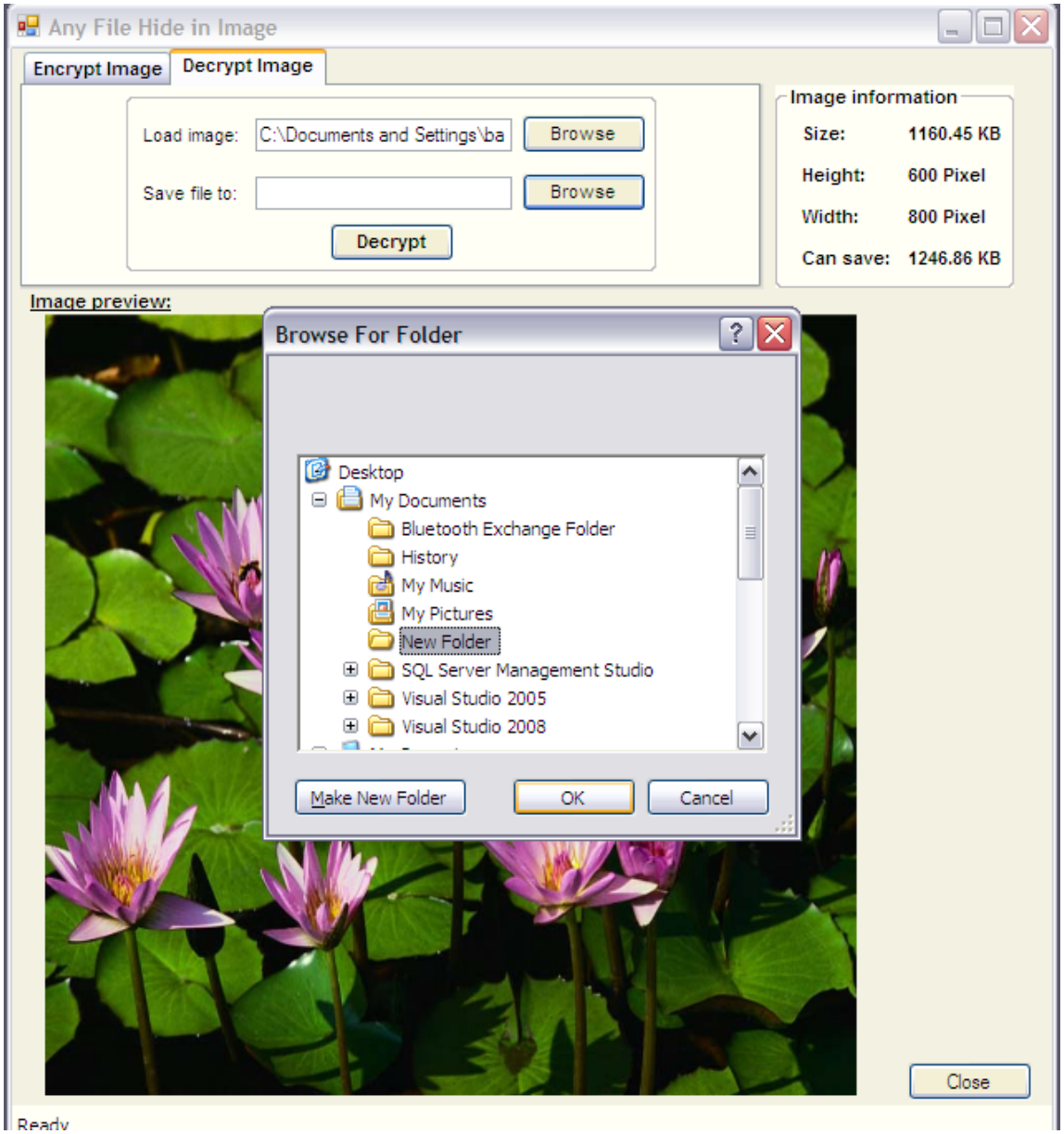
2. Next click on the “Browse” button, which open the Open file dialog box, here you have to select the image which is Encrypted and has hidden information file. Select the image file and click on Open button.



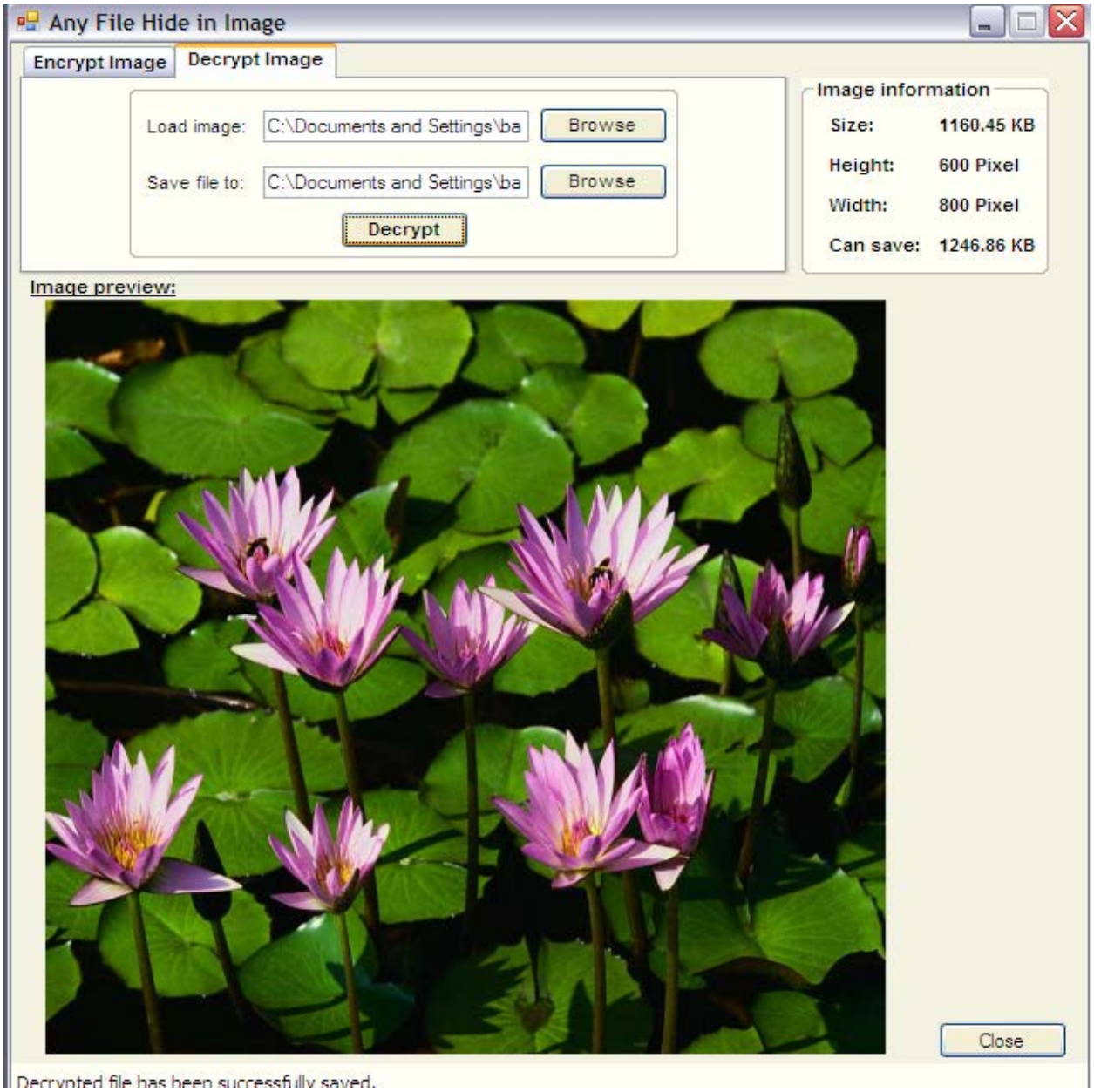
3. The image file displayed as follows:



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.



10. 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”.

11. Bibliography

Websites

Following websites are referring to create this project reports.

- http://www.programmer2programmer.net
- https://msdn.microsoft.com/en-us/library/kx37x362.aspx/
- http://www.codeproject.com
- http://www.asp.net
- http://www.asp123.com

Books

Following books and ebook are used to complete this project reports.

- Professional C#, 2nd Edition, Simon Robinson, K. Scott Allen, Ollie Cornes, Jay Glynn, Zach Greenvoss, Burton Harvey, Christian Nagel, Morgan Skinner, Karli Watson, ISBN: 978-0-7645-4398-2 .
- Professional ASP.NET, Christian Wenz, Jason N. Gaylord, Pranav Rastogi,Scott Hanselman, Todd Miranda, ISBN 978-1-118-31182-0, 2013
- MCAD/MCSD Self-Paced Training Kit: Developing Web Applications with Microsoft® Visual Basic® .NET and Microsoft Visual C#® .NET, 2013, Second Edition
- C# 6.0 in a Nutshell: The Definitive Reference 6th Edition, 2015, O'Reilly Media, Ben Albahari, Joseph Albahari.