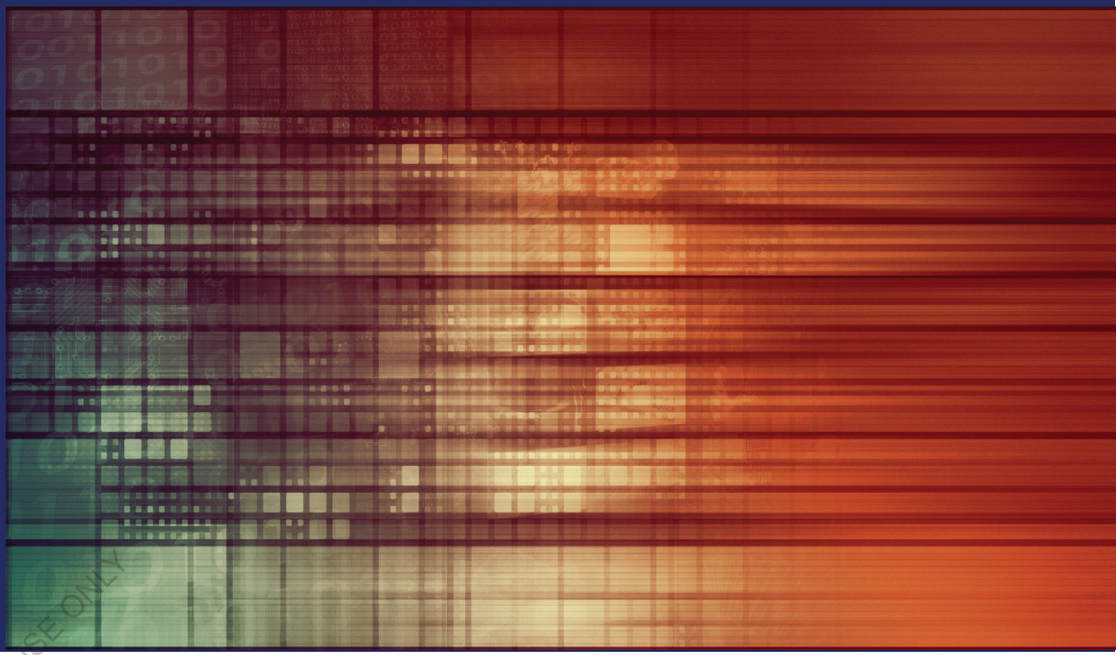
Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the internet. For hiding secret information in images, there 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 Steganography techniques are more suitable for which applications.



Mehtab Alam

Steganography

Art of hiding Information

PhD Scholar in Computer Science and Engineering, with Master of Technology in Information Security and Cyber Forensics and Bachelor of Technology in Information Technology. Residing in New Delhi, India.
My areas of interest are Internet of Things (IoT), Smart Cities, Edge Computing, Fog Computing and other similar technologies.

9 786203 869446

LAP LAMBERT Academic Publishing

Mehtab Alam

Steganography

Mehtab Alam

Steganography

Art of hiding Information

LAP LAMBERT Academic Publishing

Imprint
Any brand names and product names mentioned in this book are subject to trademark, brand or patent protection and are trademarks or registered trademarks of their respective holders. The use of brand names, product names, common names, trade names, product descriptions etc. even without a particular marking in this work is in no way to be construed to mean that such names may be regarded as unrestricted in respect of trademark and brand protection legislation and could thus be used by anyone.

Cover image: www.ingimage.com

Publisher:
LAP LAMBERT Academic Publishing
is a trademark of
Dodo Books Indian Ocean Ltd., member of the OmniScriptum S.R.L Publishing group
str. A.Russo 15, of. 61, Chisinau-2068, Republic of Moldova Europe
Printed at: see last page
ISBN: 978-620-3-86944-6

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 Steganography techniques are more suitable for which applications.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

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.

Steganography become more important as more people join the cyberspace revolution. Steganography is the art of concealing information in ways that prevents the detection of hidden messages. Steganography include an array of secret communication methods that hide the message from being seen or discovered.

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

The growing possibilities of modern communications need 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 use 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.

1.1 Detecting Steganography:

The art of detecting Steganography is referred to as **Steganalysis.** 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 JPED and your suspect file is also a JPED 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.

1.2 Objective

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

The other 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.

1.3 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.

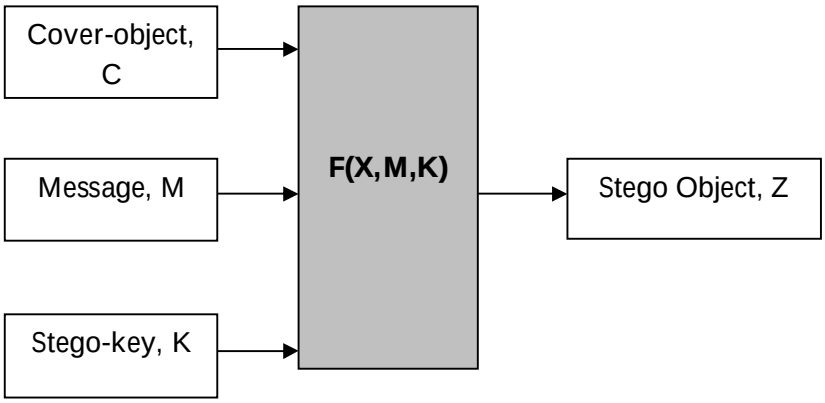
The main goal of this projects it to communicate securely in a completely undetectable manner and to avoid drawing suspicion to the transmission of a hider data. There has been a rapid growth of interest in Steganography.

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, cipher text, 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 knows the corresponding decoding key will be able to extract the message from a *cover-object*. The *cover-object* with the secretly embedded message is then called the *Stego-object*.

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

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

- Network protocols such as TCP, IP and UDP
- Audio that using digital audio formats such as wav, midi, avi, mpeg, mpi and voc
- File and Disk that can 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

1.4 Steganography Vs Cryptography:

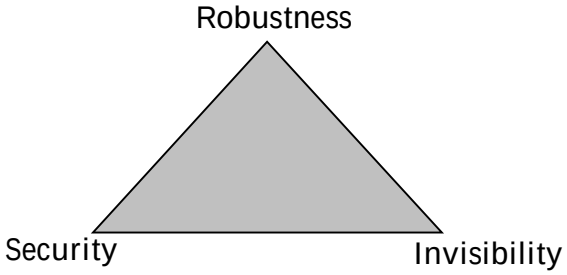
Basically, the purpose of cryptography and Steganography is to provide secret communication. However, Steganography is not the same as cryptography. Cryptography hides the contents of a secrete message from a malicious people, whereas Steganography even conceal the existence of the message. In cryptography,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

1.5 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 (Stegnographic 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 sin ominous 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, Stegnographic 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.

1.6 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.

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 including 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.

2.1 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.

2.2 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 colors 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 layer) of information, we should change the last bit of pixels, in other hands we have 3 bits in each pixel so we have 3*height*width bits memory to write our information. But before writing our data we must write name of data (file), size of name of data & size of data. We can do this by assigning some first bits of memory (8st layer).

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

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

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 linguist. 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 simpler and user friendly.

FEASIBILTY STUDY

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations.

Generally, feasibility studies precede technical development and project implementation.

4.1 Five Common Factors for Feasibility Study

The acronym TELOS refers to the five areas of feasibility - Technical, Economic, Legal, Operational, and Scheduling.

4.1.1 Technology and system feasibility

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. in order to estimate whether the new system will perform adequately or not. Technological feasibility is carried out to determine whether the company has the capability, in terms of software, hardware, personnel and expertise, to handle the completion of the project.

When writing a feasibility report the following should be taken to consideration:

- A brief description of the business to assess more possible factor/s which could affect the study
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problems

At this level, the concern is whether the proposal is both *technically* and *legally* feasible (assuming moderate cost).

4.1.2 Economic feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action.

Cost-based study: It is important to identify cost and benefit factors, which can be categorized as follows:

1. Development costs
2. Operating costs.

This is an analysis of the costs to be incurred in the system and the benefits derivable out of the system.

Time-based study: This is an analysis of the time required to achieve a return on investments. The future value of a project is also a factor.

4.1.3 Legal feasibility

Determines whether the proposed system conflicts with legal requirements, e.g. a data processing system must comply with the local Data Protection Acts.

4.1.4 Operational feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

4.1.5 Schedule feasibility

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. You need to determine whether the deadlines are mandatory or desirable.

4.2 Other Feasibility Factors

4.2.1Market and real estate feasibility

Market feasibility studies typically involve testing geographic locations for a real estate development project, and usually involve parcels of real estate land. Developers often conduct market studies to determine the best location within a jurisdiction, and to test alternative land uses for given parcels. Jurisdictions often require developers to complete feasibility studies before they will approve a permit application for retail, commercial, industrial, manufacturing, housing, office or mixed-use project. Market Feasibility takes into account the importance of the business in the selected area.

4.2.2 Resource feasibility

This involves questions such as how much time is available to build the new system, when it can be built, whether it interferes with normal business operations, type and amount of resources required, dependencies,

4.2.3 Cultural feasibility

In this stage, the project's alternatives are evaluated for their impact on the local and general culture. For example, environmental factors need to be considered and these factors are to be well known. Further an enterprise's own culture can clash with the results of the project.

4.2.4 Financial feasibility

In case of a new project, financial viability can be judged on the following parameters:

- Total estimated cost of the project
- Financing of the project in terms of its capital structure, debt equity ratio and promoter's share of total cost
- Existing investment by the promoter in any other business
- Projected cash flow and profitability

4.3 Output

The feasibility study outputs the **feasibility study report**, a report detailing the evaluation criteria, the study findings, and the recommendations.

HARDWARE and SOFTWARE REQUIRMENT

5.1 Hardware Requirement

Hardware requirement are the basic need of the system or the package, which is been developed and will be deployed upon the system, which should have these basic components or fulfill these basic hardware needs of these package.

The following hardware is recommended for the user.

- CPU Intel Pentium IV or above
- RAM 32MB or above
- Monitor Color monitor
- Hard disk 8 GB or above
- CD –Drive Any CD ROM
- Input device Key board, Mouse

5.2 Software Requirement

- Operating System Windows XP or above
- Language .Net Framework 3.5
- Frontend/Technology VB.Net

SYSTEM REQUIRMENT SPECIFICATION (SRS)

Steganography is a very old technique of hiding the data. This software is all about hiding the data. This software is made according to the modern need of hiding data. It uses various new techniques for hiding the data. The basic advantage of this product is that it is not specific for a particular type of either hidden file or carrier file.

6.1 Product Feature

- Carrier file is image file.
- Various file format is supported
  - Text - .txt, .rtf, .doc, .docx
  - Image - .bmp, .gif, .jpeg
- Hidden object can be of any text format.
- Multiple file encryptions are supported.
- File compression is supported.
- Image conversion is supported.
- User manual, installation file, help manual is also provided with his product.

6.2 User Classes and Characteristics

Administrator: They have full control over the software. Apart from using the basic task, they have full control over the user management and they can also view the log file.

End User: They can perform the Stegnographic task but have no control over the user management and log file.

6.3 Operating Environment

Operating System: Windows XP, 2007, Vista.
Software Requirement: VB.NET.
Recommended Configuration: 256 MB Ram or higher, 10 MB Disk Space.
Screen Resolution: 1024 x 768.

6.4 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.

DATA FLOW DIAGRAM (DFD)

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

Data Flow Diagram serves two purposes:

1. To provide annunciation of how data are transformed as they move through the system.
2. To depict the functions that transforms the data flow.

DFDs are a excellent mechanism for communicating with the customer during requirement analysis and are widely used for the representation of external and top-level internal design specification. In the latter situations, DFDs are quite valuable for subsystem, files and data links. The DFD methodology is quite effective, especially when the required design is unclear. In the process, many levels of DFDs are created depending upon the level of details needed

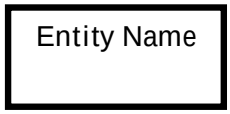
.

The Level 0 DFD is also called Context Level DFD. It depicts the overview of the entire system. The major external entities, a single process and the output stores constitute the level-0 DFD. Though this diagram does not depict the system in detail, it represents the overall inputs, process and output of the entire system at a very high level.

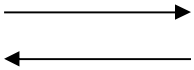
The Level 0 DFD is now expended into a level 1 model. It should be noted that information flow continuity is maintained between level 0 and level 1. The process represents at DFD level 1 further refined into lower levels. This further refinement is continued until an easily implement able program component is reached.

SYMBOLS USED IN DFD

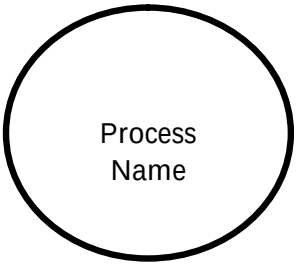
An external entity



Dataflow



Processes

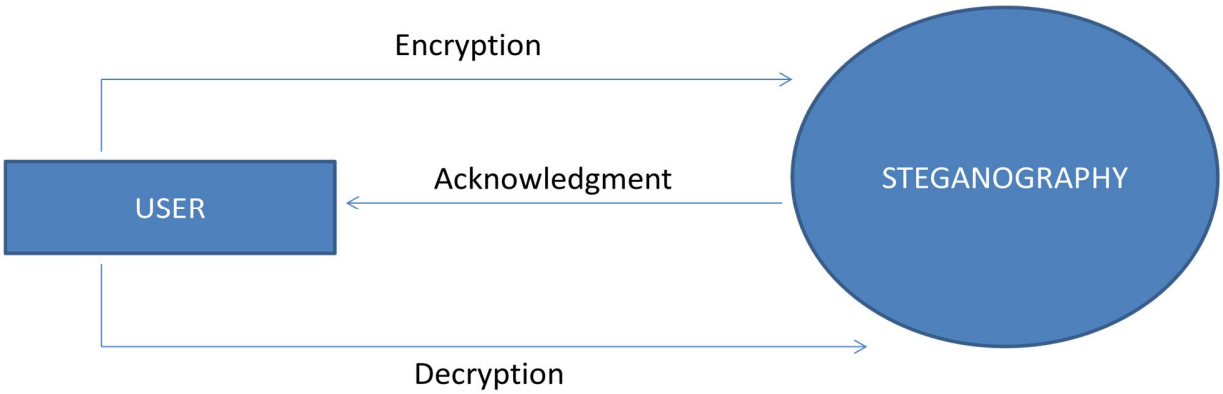


Data Store

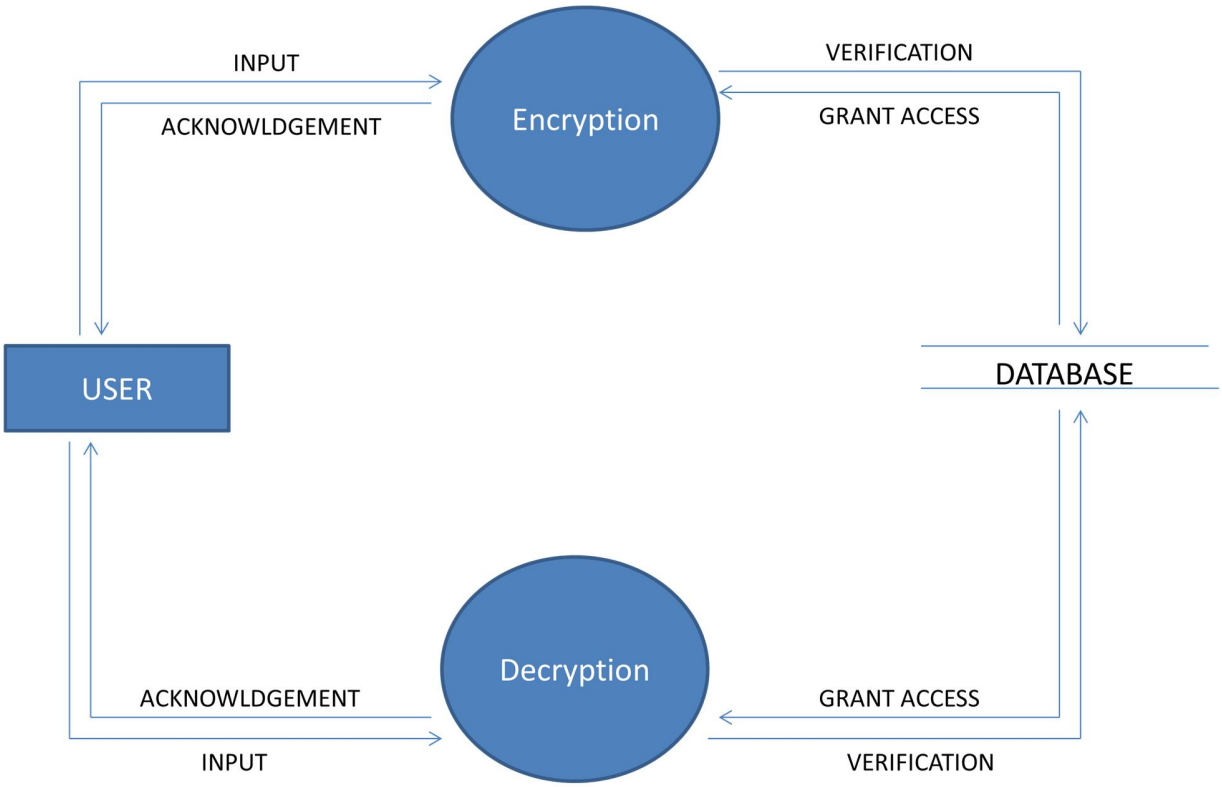
Context level Diagram: -

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, it only shows that the data is stored and access from the database.

7.1 Context Level DFD



7.2 Level 1 DFD



SYSTEM 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).

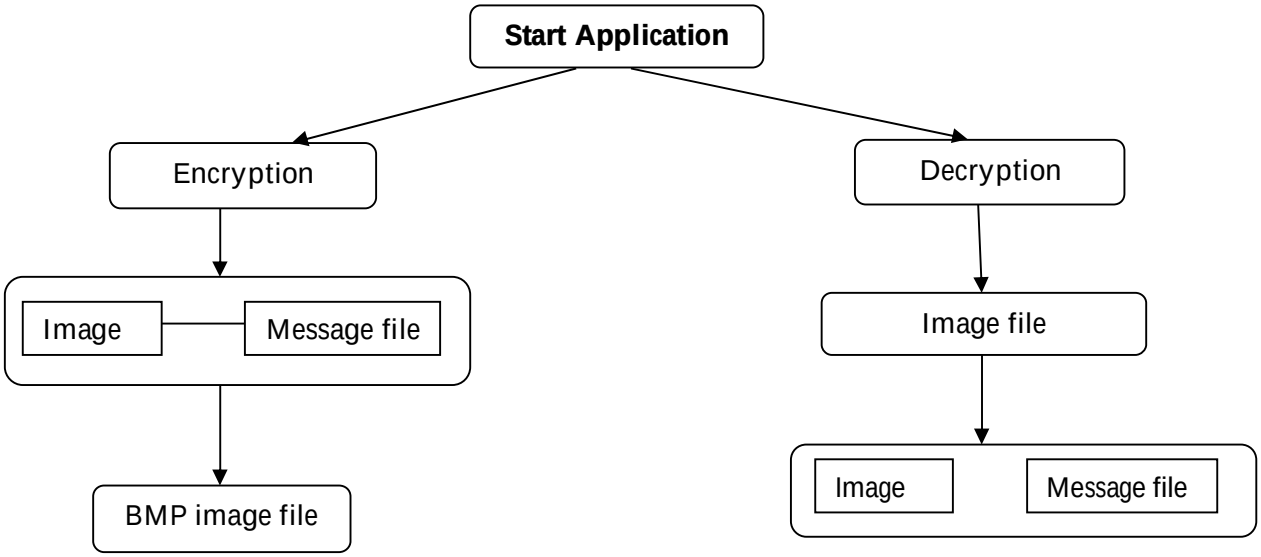
The algorithm used for Encryption and Decryption in this application provides using several layers lieu of using only LSB layer of image. Writing data starts from last layer (8st or LSB layer); because significant of this layer is least and every upper layer has doubled significant from its down layer.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

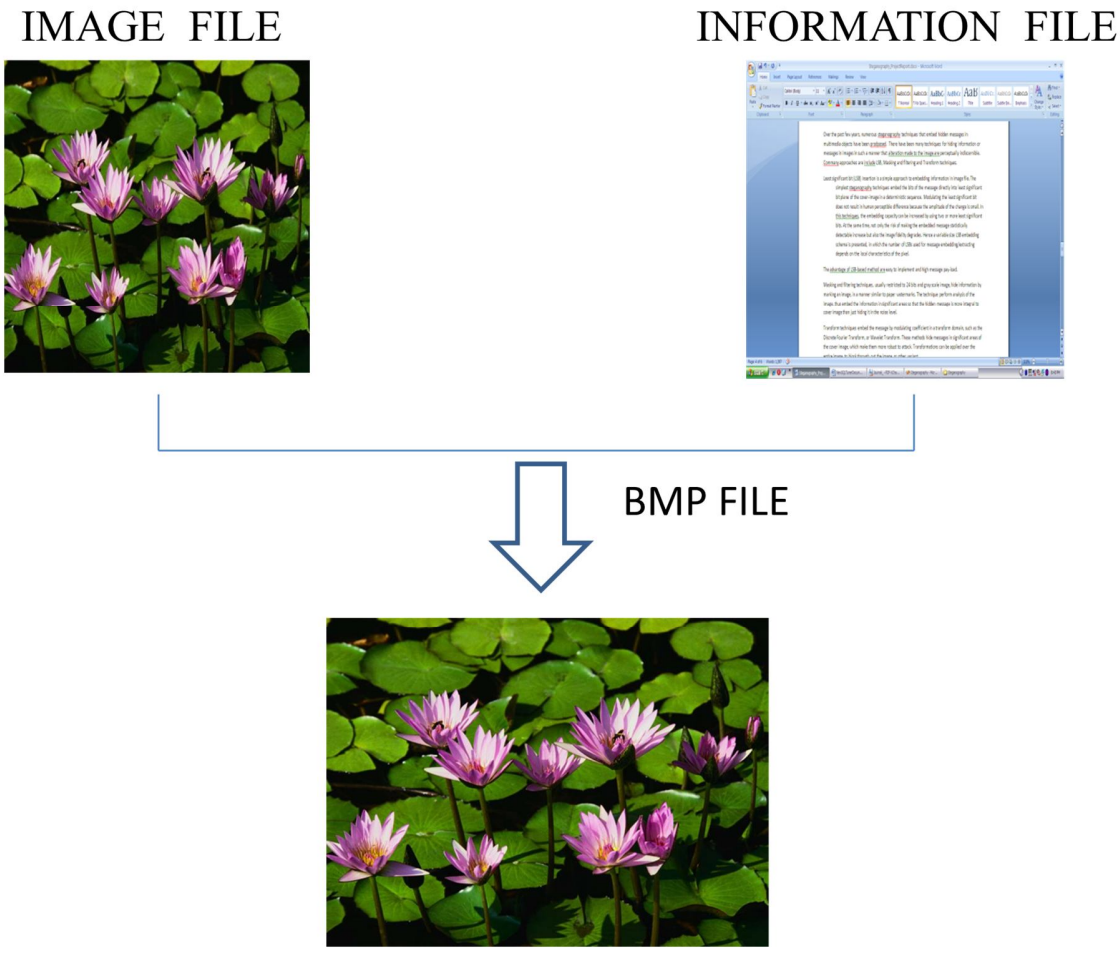
The decrypt module is used to get the hidden information in an image file. It take the image file as an output, and give two file at destination folder, one is the same image file and another is the message file that is hidden it that.

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.

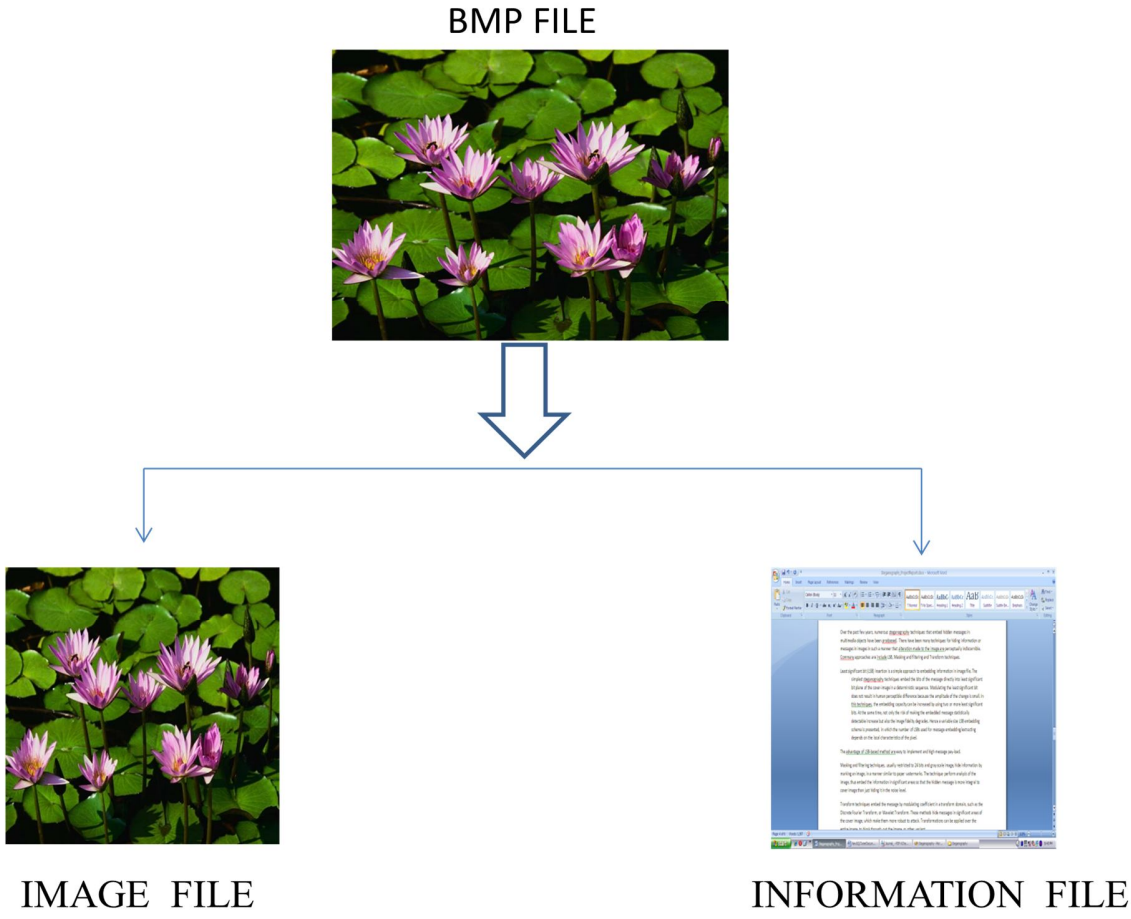
The graphical representation of this system is as follows:



8.1 Encryption Process



8.2 Decryption Process



CODING

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
namespace Text2Image
{
public partial class FrmSteganography : Form
{
public FrmSteganography()
{
InitializeComponent();
}
//public values:
Stringloaded TrueImagePath,loadedFilePath,saveToImage,DLoadImagePath,DSaveFile
Path;
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
```

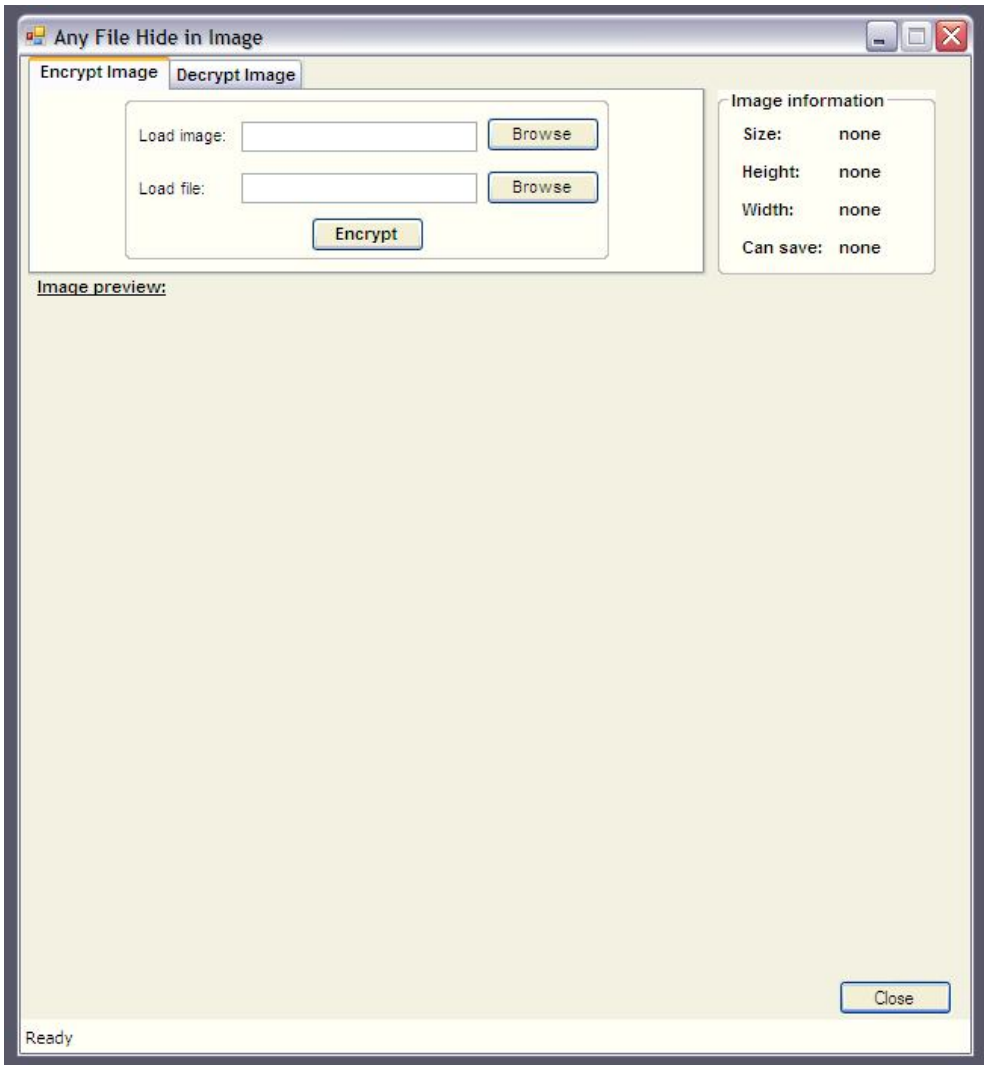
```
File.WriteAllBytes(DSaveFilePath + "\\" + resFName, res);
toolStripStatusLabel1.Text = "Decrypted file has been successfully saved.";
Application.DoEvents();
}
private void byte2bool(byte inp, ref bool[] outp)
{
if(inp>=0 && inp<=255)
for (short i = 7; i >= 0; i--)
{
if (inp % 2 == 1)
outp[i] = true;
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("MOHAMMAD TAUSIF");
}
}
}
```

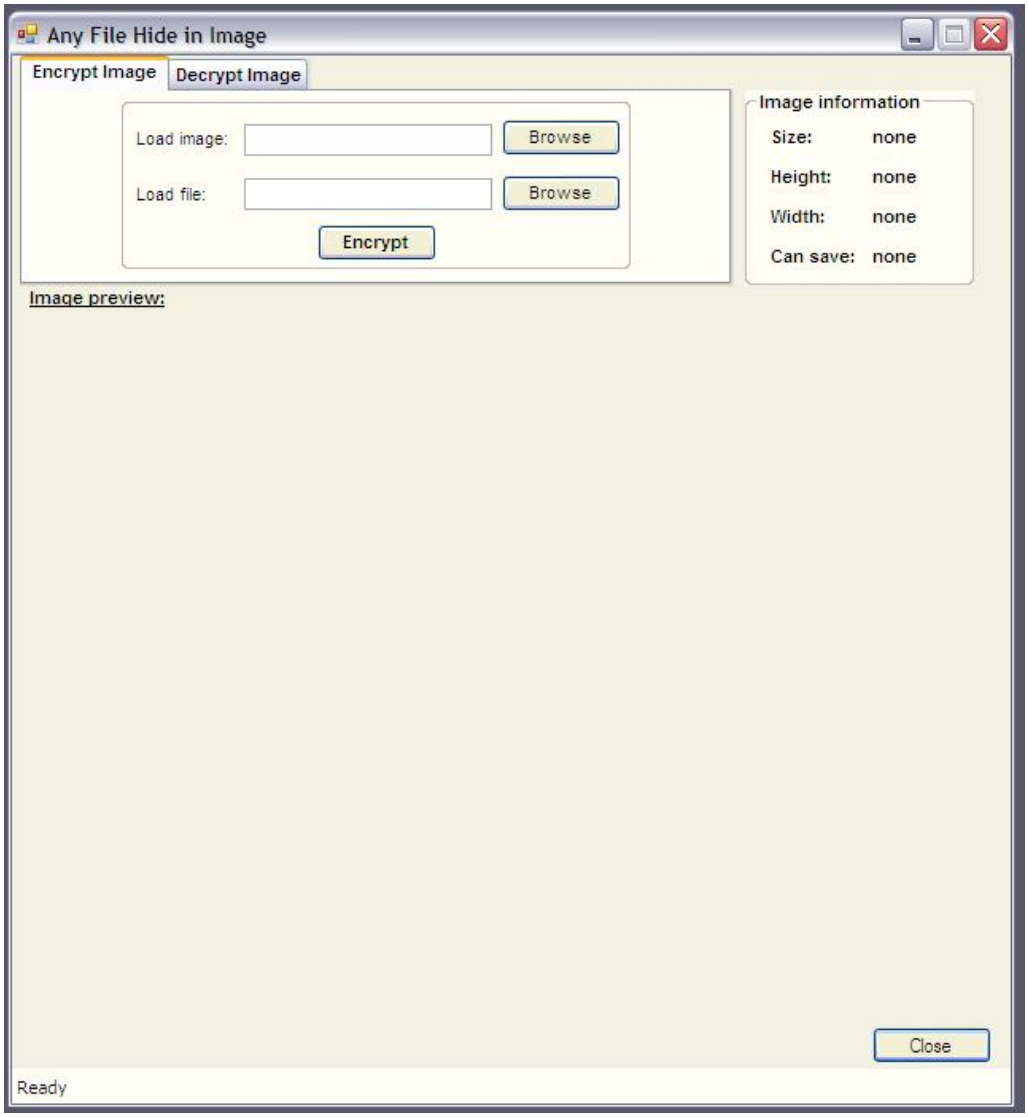
USER MANNUAL

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.

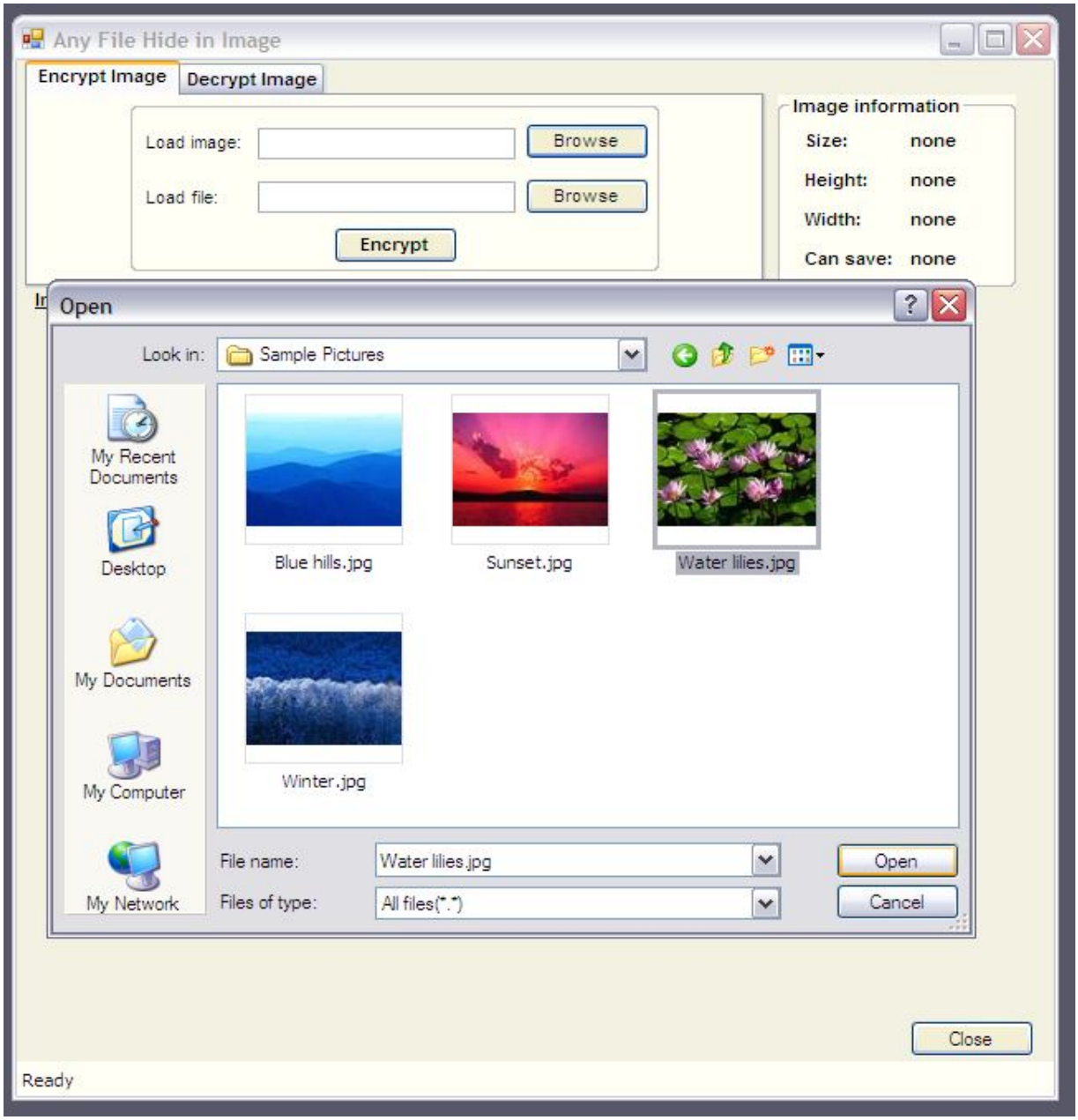


Encryption

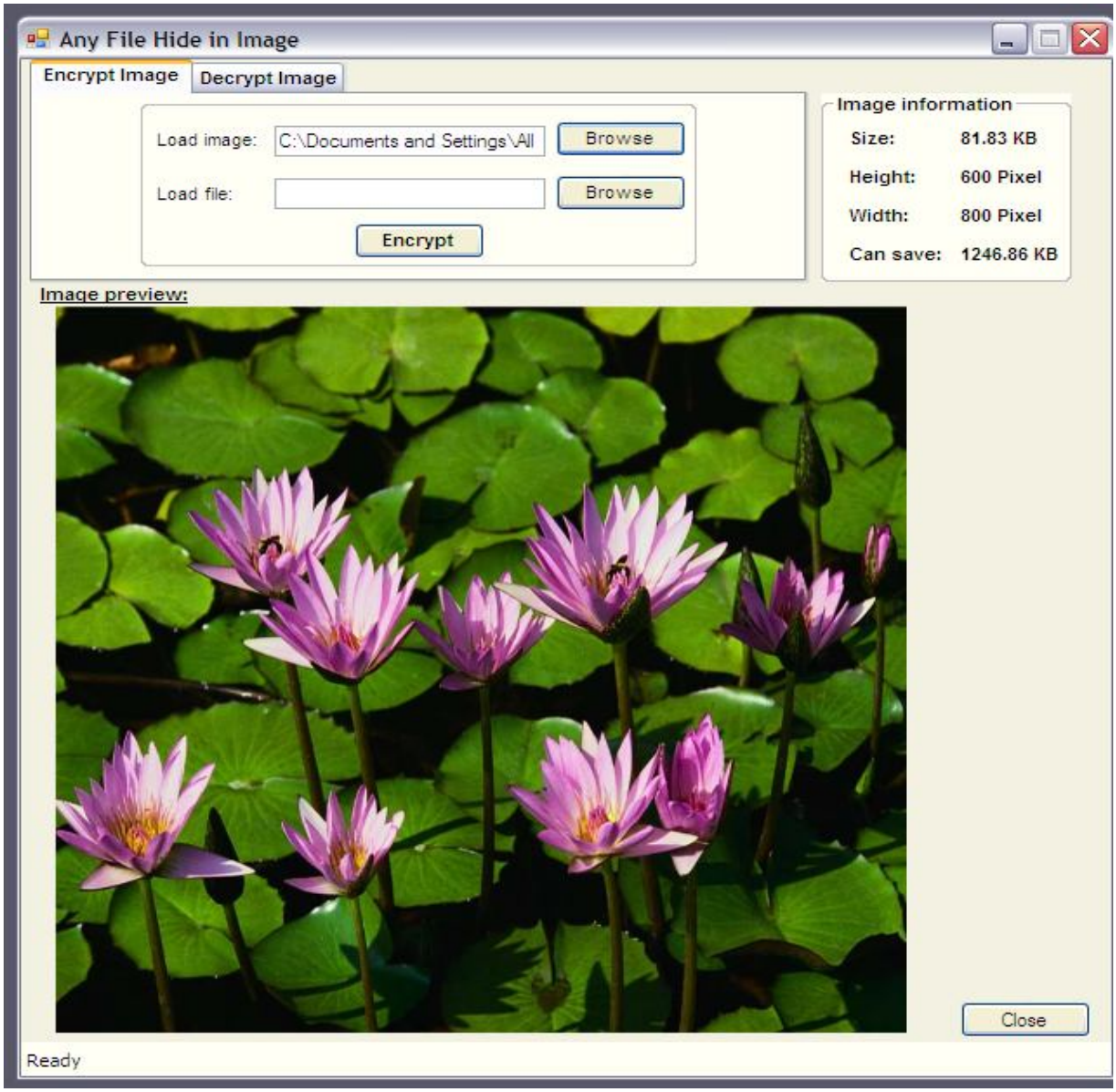
Step 1: For Encryption select Encrypt Image tab button.



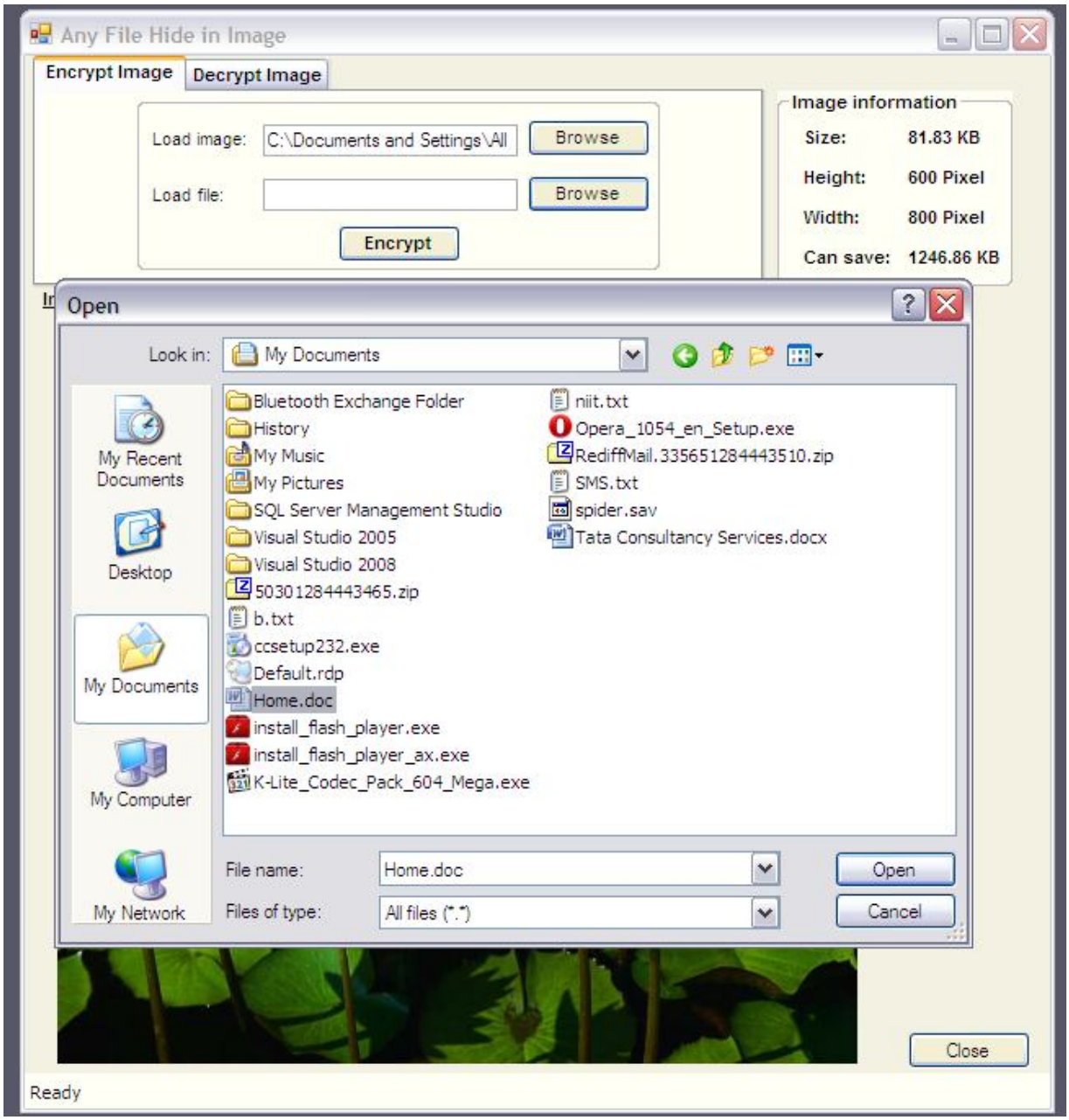
Step 2: To load image click on button "**Browse"** that is next to the "**Load Image"** text box. The file open dialog box will displays as follows, select the image file, which you want to use for hiding information and click Open button.



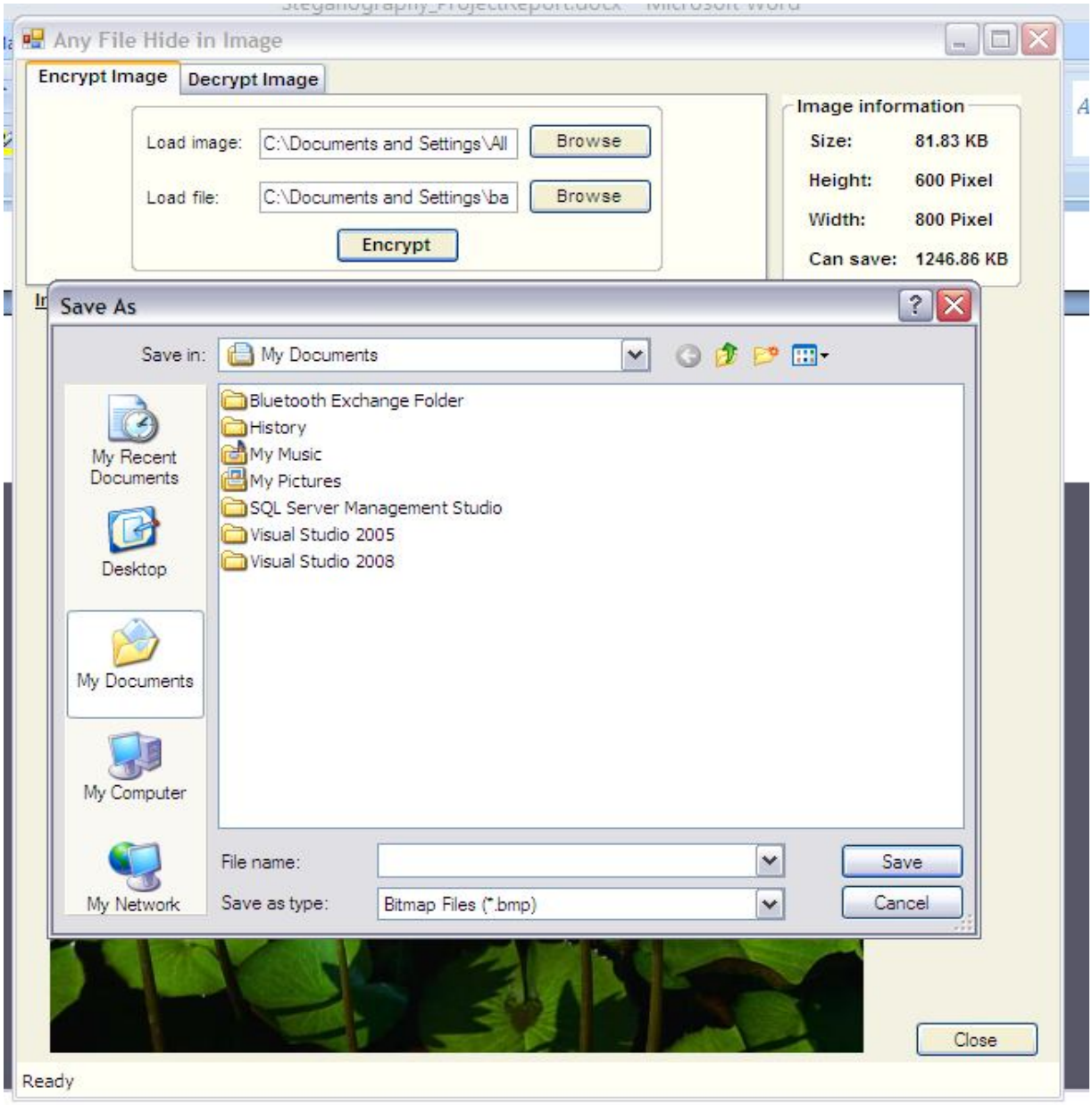
Step 3: Image will open and displays as follows. Next, click "**Browse"** button that is next to the "L**oad File",** button.



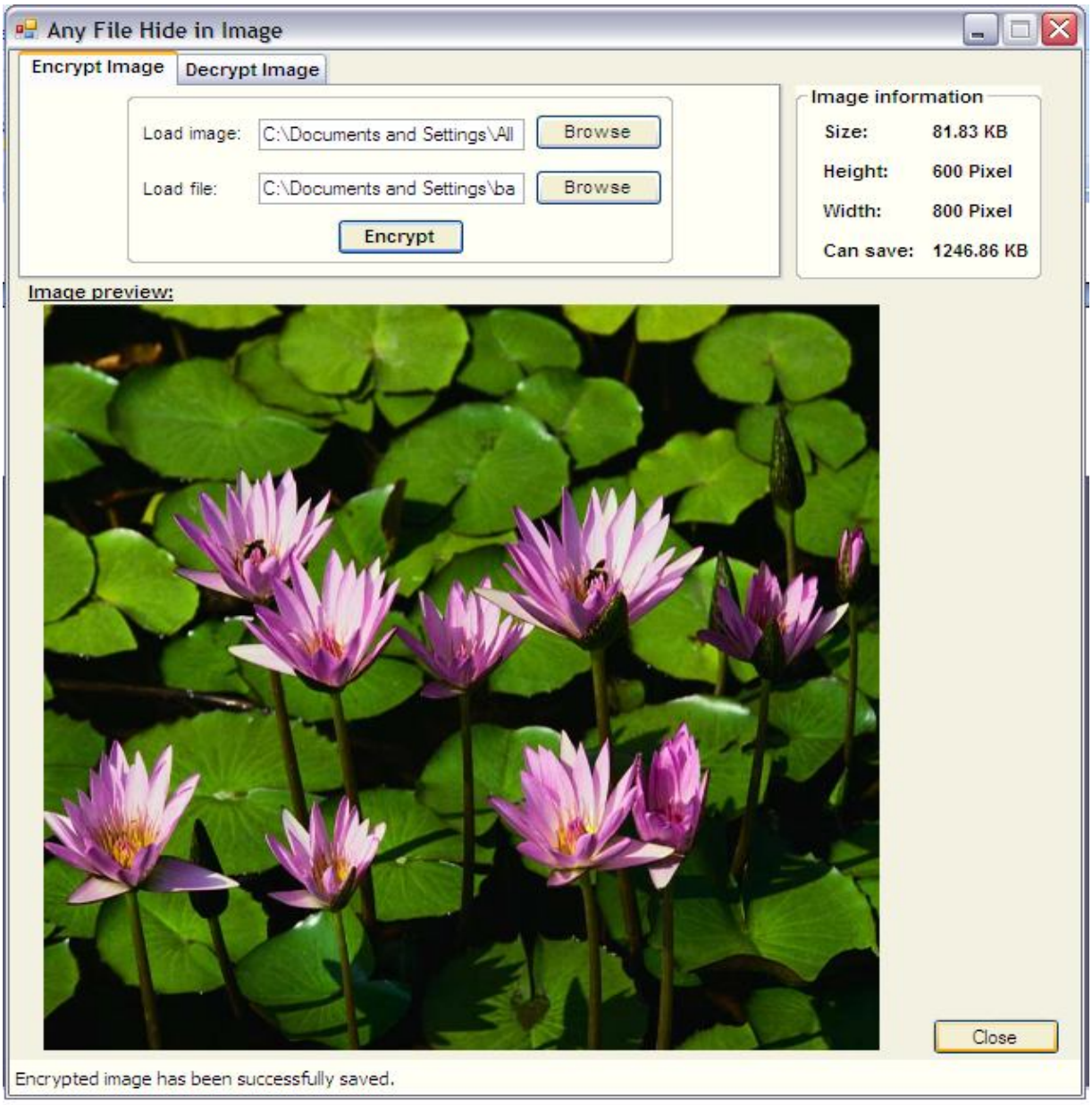
Step 4: Again the file open dialog box will appear, select any type of file whatever you want to hide with the image and click on ok button.



Step 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.

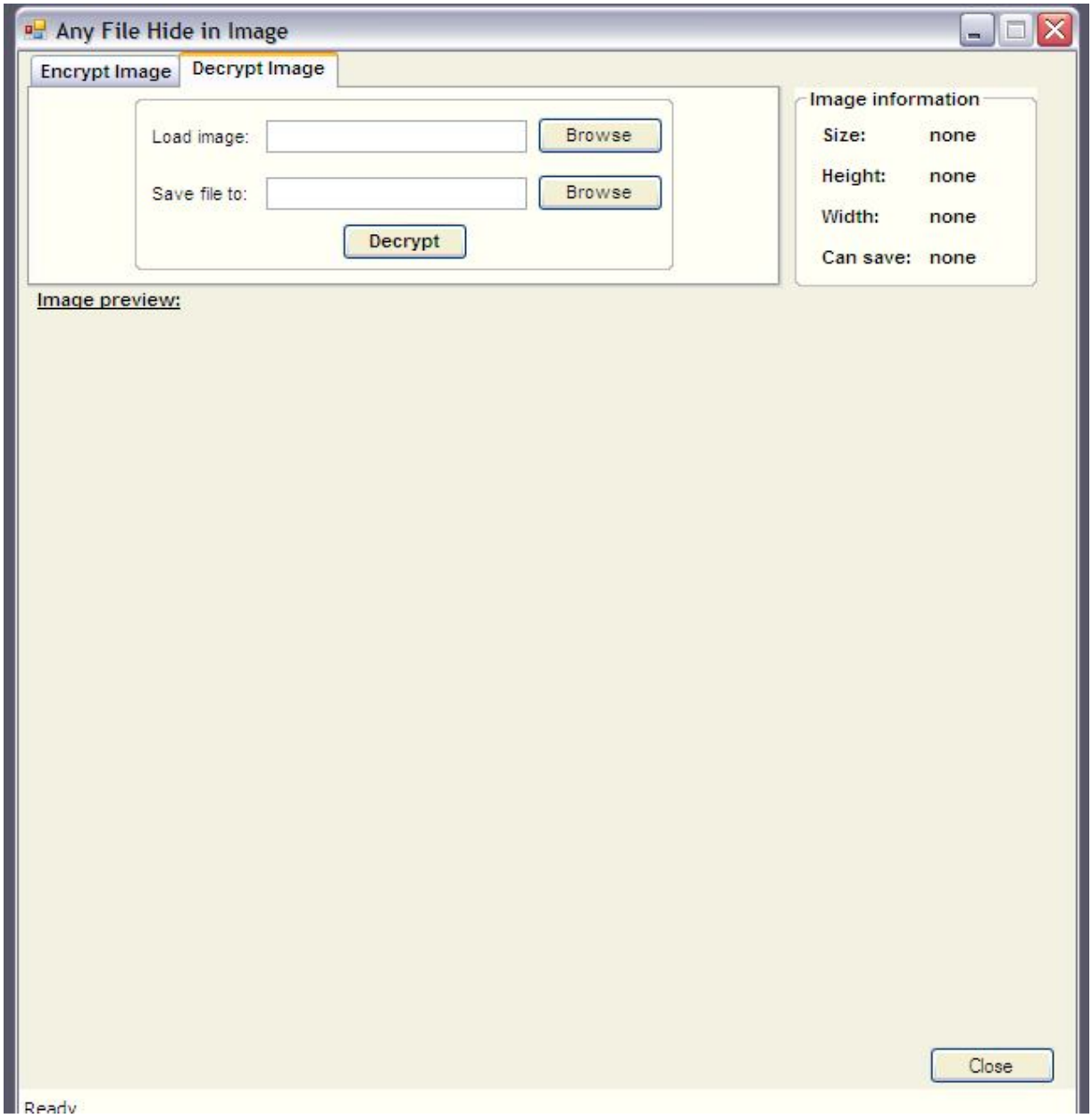


Step 6: The Encryption of the document is successfully done.

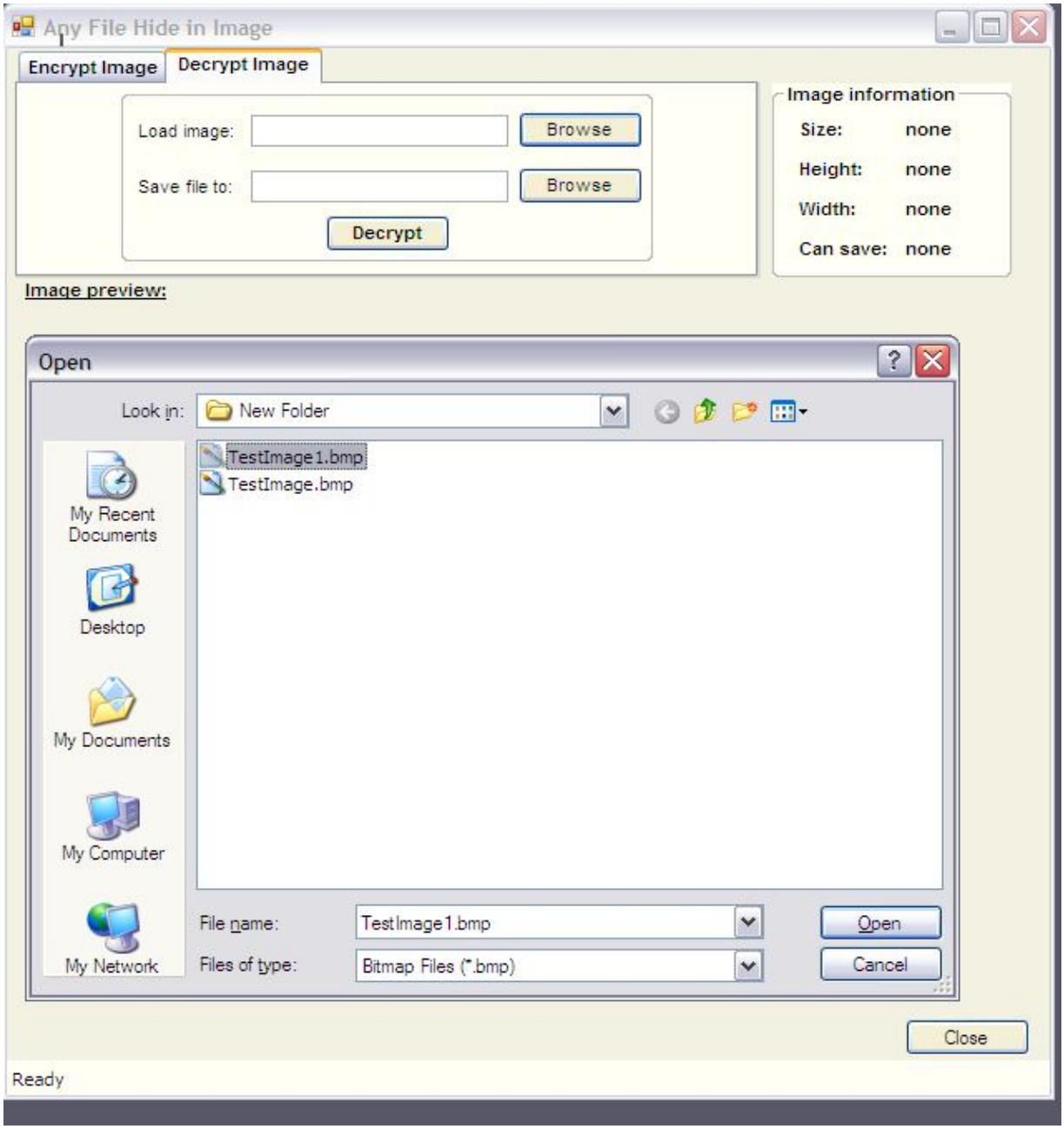


Decryption

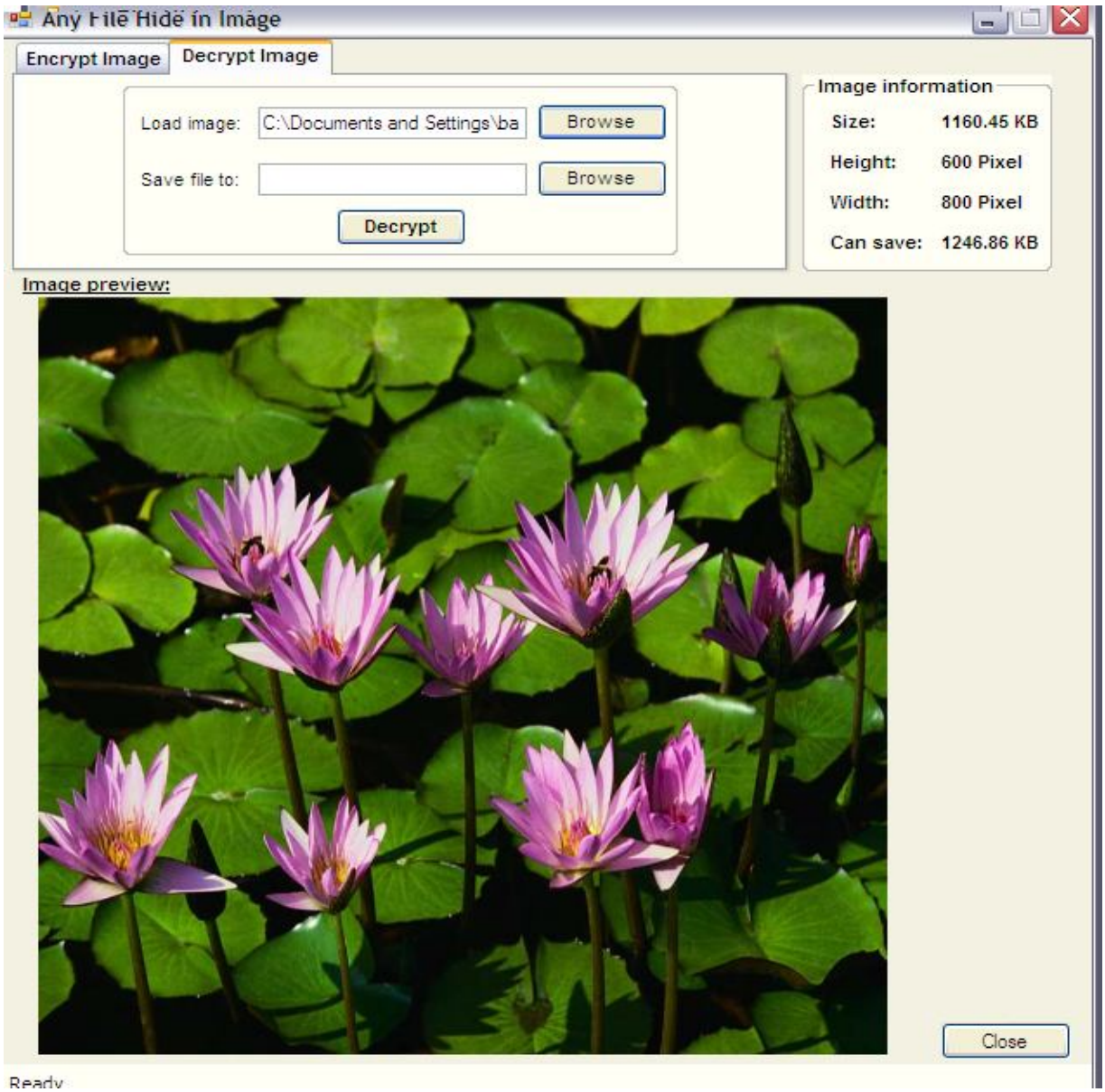
Step 1: Select the "**Decryption"** image tab button.



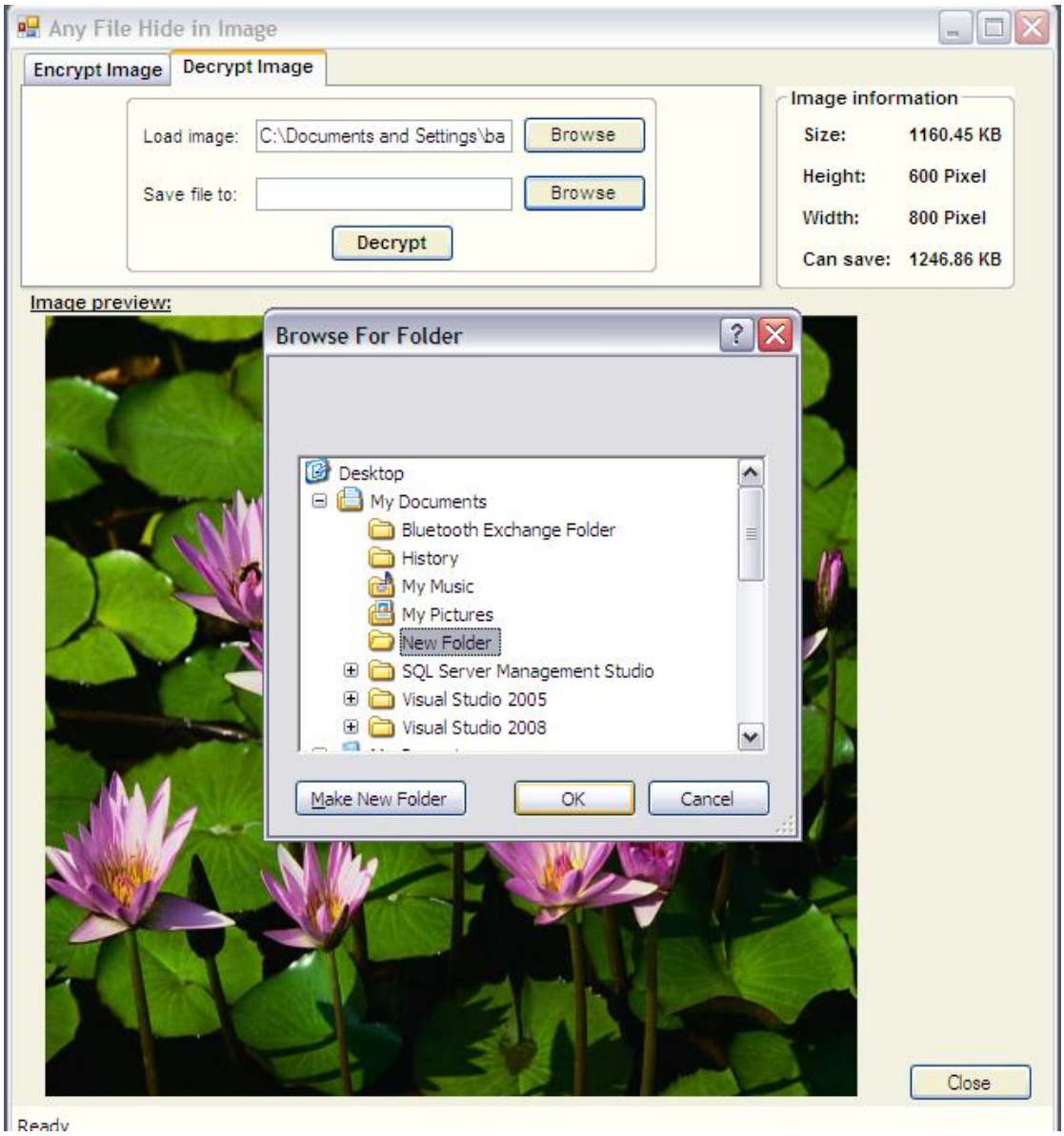
Step 2: Next click on the "**Browse**" button, which open the Open file dialog box, here you have to select the image which is Encrypted and has hidden information file. Select the image file and click on Open button.



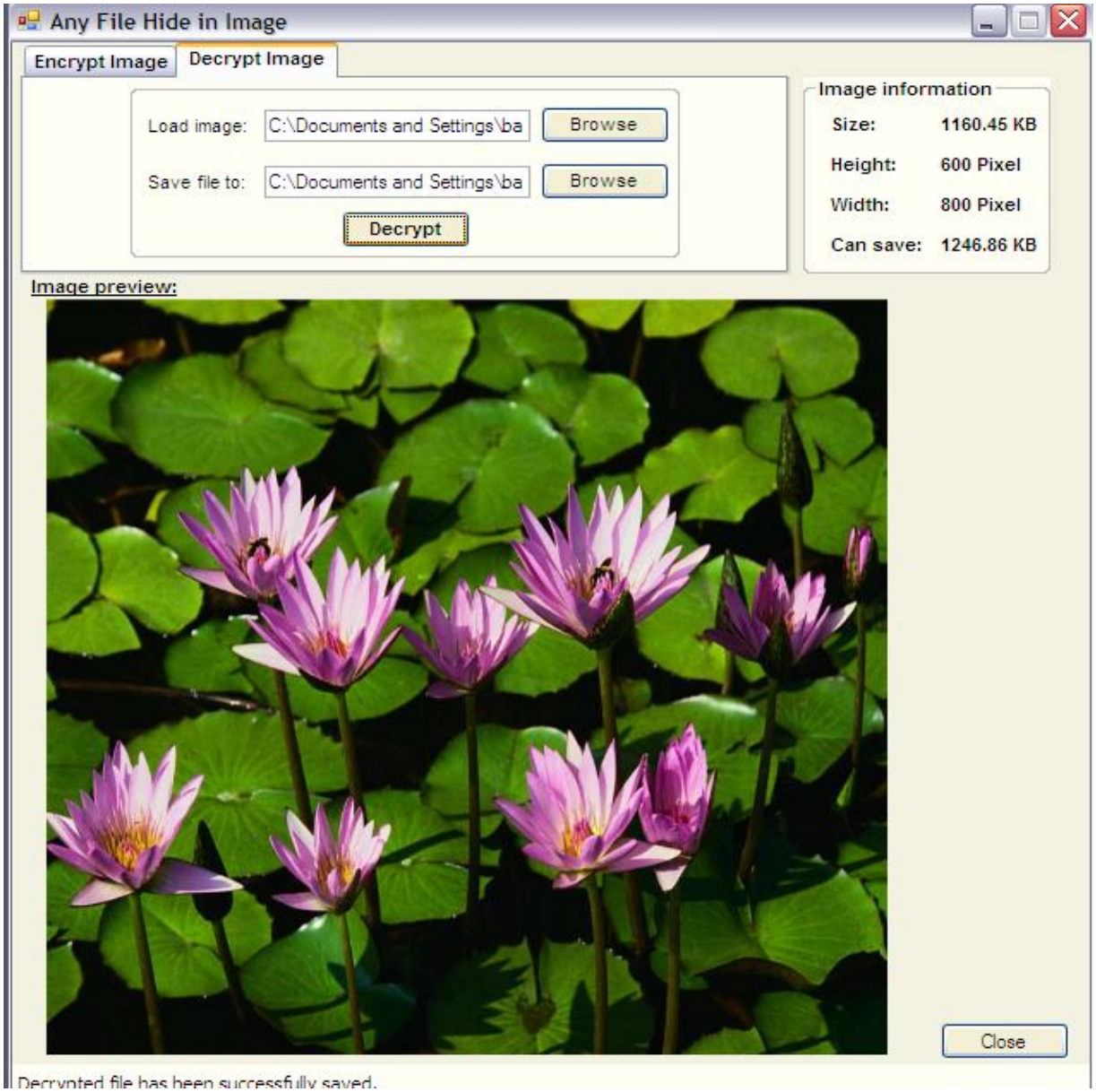
Step 3: The image file displayed as follows:



Step 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 asks you to select the path or folder, where you want to extract the hidden file. Select the folder and click on Ok button.



Step 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.



SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

11.1 Testing the whole System

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

11.2 Testing Issue

- Client GUI consideration.
- Target environment and platform diversity consideration.
- Distributed processing consideration.

11.3 Testing 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**
- **Decrypt**

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

Decryption is getting the secrete information from image file. The objective of the testing is to discover errors. To fulfil this objective a series of test step unit, integration, validation and system tests were planned and executed

11.4 Type of Testing

11.4.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developers to modify the source code without immediate concerns about how such changes might affect the functioning of other units or the program as a whole. Once all of the units in a program have been found to be working in the most efficient and error-free manner possible, larger components of the program can be evaluated by means of integration testing.

11.4.2 Integrated Testing

Integration testing, also known as integration and testing (I&T), is a software development process which program units are combined and tested as groups in multiple ways. In this context, a unit is defined as the smallest testable part of an application. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

There are two major ways of carrying out an integration test, called the bottom-up method and the top-down method. Bottom-up integration testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds. In top-down integration testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems and networks.

11.4.3 Validation Testing

Software validation is achieved through the series of tests that demonstrate confirmative with requirement. Thus the proposed system and considerations has been tested by validation and found to be working satisfactorily.

11.4.4 Output Testing

Asking the user about the format required by them to test the output generated by the system under considerations. It can be considered in two ways, one on the screen and the other is printed format. The output format on the screen is found to be correct as the format designed in the system design.

11.5 Testing Result

All tests should be traceable to customer requirements. The focus of testing will shift progressively from programs. Exhaustive testing is not possible. To be more effective, testing should be one, which has probability of finding errors.

The following are the attributes of good test:

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be “best of breeds”.
- A good test should neither too simple nor too complex.

11.6 Test Cases

Test Cases	Test Data	Expected Result	Status
Encrypting			
Browsing Image	<Path of image to be used>	Image path should be shown in the text box given, and image should be displayed on the software window.	Pass
Browsing Document	<Path of document to be used>	Document path should be shown in the text box given.	Pass
Encrypting	<Path of image to be saved>	Success message should be displayed in bottom left corner of software window.	Pass
Decrypting			
Browsing Image	<Path of image to be used>	Image path should be shown in the text box given, and image should be displayed on the software window.	Pass
Defining Path to Save	<Desired path to save encrypted message>	Path should be shown in the text box given.	Pass
Decrypting	<Path of image to be used>	Success message should be displayed in bottom left corner of software window. And document should be present at desired path.	Pass

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 there. 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”.

REFERENCES

1. http://www.asp.net
2. http://www.asp123.com
3. http://www.wikipedia.org
4. Mastering C# (Paperback).
5. SQL Server Bible (Paperback).
6. Professional C#, 2nd Edition (Paperback).
7. Professional ASP.NET (Paperback).
8. MCAD/MCSD Self-Paced Training Kit: Developing Web Applications with Microsoft® Visual Basic® .NET and Microsoft Visual C#® .NET, Second Edition.
9. MCAD/MCSE/MCDBA Self-Paced Training Kit: Microsoft SQL Server 2000 Database Design and Implementation, Exam 70-229, Second Edition.
10. Alam, M. (2011). Online Banking (1st ed.). New Delhi. LAP LAMBERT Academic Publishing, ISBN: 978-620-3-86302-4, DOI: 10.6084/m9.figshare.14612127.
11. Alam, M. (2012). Electronic Ticket Machine (1st ed.). New Delhi. LAP LAMBERT Academic Publishing, ISBN: 978-620-3-86332-1, DOI: 10.6084/m9.figshare.14661354.
12. Herbert Schildt (2000) 'Visual Basic 6.0' Tata McGraw Hill
13. Jamie Jaworsky 'Visual Basic 6.0'Techmedia
14. Alam, M. (2013). Just Shop-Shopping (1st ed.). New Delhi. LAP LAMBERT Academic Publishing, ISBN: 978-620-3-58124-9, DOI: 10.6084/m9.figshare.14431382.
15. Alam, M. (2013). Core ePortal (1st ed.). New Delhi. Glasstree Bookstore, ISBN: 978-1-6671-9827-9, DOI: 10.20850/9781667198279.
16. Alam, M., & Khan, M. (2013). E-Cops (1st ed.). New Delhi. LAP LAMBERT Academic Publishing, ISBN: 978-620-3-86368-0, DOI: 10.6084/m9.figshare.14662479.