

Kingdom of Saudi Arabia
Majmaah University
Ministry of Higher Education
Faculty of science
Dept. of Computer Science



STEGANOGRAPHY USING IMAGES

By

Ghadeer Fahad Alfuhaid

ID:351204862

Supervised by:

T. Hajer brahim

**This Project Documentation is submitted in partial fulfillment for the
requirement for the Degree of B.Sc. in
Computer Science and information
April /2018**

Committee Report Page: This is to certify that the project team has submitted their work on time and that the Committee has reviewed the content and evaluated the work. The following is a committee report page example:

COMMITTEE REPORT		
We certify that we have read this graduation project report as examining committee, examined the student in its content and that in our opinion it is adequate as a project document for B.Sc. in Computer Science and information.		
<u>Supervisor:</u>	<u>Examiner1:</u>	<u>Examiner2:</u>
Name:	Name:	Name:
Signature.	Signature	Signature
.Date: / /	Date: / /	Date: / /

DEDICATION

We all have dreams. But in order to make dreams come into reality, it takes an awful lot of determination, dedication, self-discipline, and effort.

And without the support of our caring and darling people this project will just stay a dream.

I like to dedicate this project:

To my beloved mothers, fathers and family, without you the life is tasteless and the road is endless. You are the most important thing in my world.

To my friends, the ones who made the best memories of my life and the ones who made it worthy to walk the extra mile while we were about to give up.

ACKNOWLEDGMENT

Firstly, we would like to express our sincere gratitude to our project supervisor T. Hajer Brahim, for the continuous support of this project and his patient guidance, enthusiastic encouragement and useful critiques, motivation, and immense knowledge.

We would like to thank the rest of our thesis committee, T.Nora Alneghaimshi, for their guidance helped, insightful comments and encouragement, also for the hard question which incited me to widen my research from various perspectives. Our sincere thanks also go to all the staff members of the department for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project.

Finally, we would like to thank our family for supporting us spiritually during working on the project and our friends who gave us moral support and encouragement throughout the project study.

ABSTRACT

Data hiding is the art of hiding data for various purposes such as; to maintain private data, secure confidential data and so on. Securely exchange the data over the internet network is very important issue. So, in order to transfer the data securely to the destination, there are many approaches like cryptography and steganography. In this project we propose a LSB & DCT-based steganographic method for hiding the data by applying Least Significant Bit (LSB) algorithm for embedding the data into the images which is implemented through the Microsoft .NET framework using the C#.NET.

Keywords: *Least Significant Bit, Discrete Cosine Transform, Steganography.*

TABLE OF CONTENTS

DEDICATION.....	III
ACKNOWLEDGMENT	IV
ABSTRACT.....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES	VIII
LIST OF FIGURES	IX
LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE.....	X
CHAPTER 1 : INTRODUCTION.....	11
1.1 INTRODUCTION	11
1.2 PROJECT DEFENITION.....	11
1.3 STATEMENT OF PROBLEM.....	12
1.4 PROJECT OBJECTIVES	12
1.5 ARCHITECTURE & COMPONENTS.....	12
1.6 PROJECT SCOPE.....	13
1.7 STEGANOGRAPHY ARCHITECTURE.....	13
1.7.1 STEGANOGRAPHY TYPES.....	15
1.7.2 STEGANOGRAPHY ALGORITHMS.....	16
1.8 ORGANIZATION CHART AND RESPONSIBILITIES	18
1.9 SUMMARY.....	18
CHAPTER 2 : LITERATURE REVIEW.....	19
INTRODUCTION	19
2.1 SIMILAR SYSTEMS COMPARISON.....	19
2.2.1 WHITE NOISE STORM.....	19
2.2.2 STEGODOS	20
2.2.3 STEGCURE	20
2.3 SIMILAR TOOLS COMPARISON.....	21
2.4 RECOMMENDATIONS.....	21
CHAPTER 3 : SYSTEM ANALYSIS.....	23
3.1 INTRODUCTION	23
3.2 STAKEHOLDERS	23
3.4 NON-FUNCTIONAL REQUIREMENTS	25

3.5 SEMI-FORMAL REQUIREMENTS	26
3.5.1 GENERAL USE CASE DIAGRAM.....	26
3.5.2 ACTORS	26
3.5.4 HARDWARE AND SOFTWARE REQUIREMENTS	27
3.6 PREDESIGN	30
3.7 CONCLUSION.....	30
CHAPTER 4 : SYSTEM DESIGN	31
4.1 INTRODUCTION	31
4.2 STRUCTURAL VIEW	31
4.2.1 CLASS DIAGRAM	31
4.3 BEHAVIORAL VIEW	32
4.3.1 SEQUENCE DIAGRAM.....	32
4.3.2 ACTIVITY DIAGRAM.....	32
4.3.3 DATA FLOW DIAGRAM	34
CHAPTER 5 : IMPLEMENTATION & TESTING.....	35
5.1 SCREEN SHOTS OF THE SYSTEM:.....	35
5.2 DEFINITION AND THE GOAL OF TESTING	41
5.3 METHODS OF TESTING	43
5.3.1 UNIT TESTING.....	43
5.3.2 VALIDATION TESTING	43
5.3.3 OUTPUT TESTING.....	44
5.3.4 INTEGRATION TESTING	44
5.4 USER ACCEPTANCE TESTING	45
5.5 BLACK BOX AND WHITE BOX TESTING	45
5.6 TEST CASES	46
CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATION	47
6.1 CONCLUSIONS	47
6.2 FUTURE RECOMMENDATIONS	47
REFERENCES.....	48
APPENDICES	49
EMBED FILE CODING	49
EXTRACT FILE CODING	52

LIST OF TABLES

Table 2-1: Similar tools comparison.....	21
Table 3-1: Stakeholders	23
Table 3-2: Use case description – Embedding process.....	24
Table 3-3: Use case description – Extracting process	25
Table 5-1: Test Cases	46

LIST OF FIGURES

Figure 1-1: Pure Steganography	15
Figure 1-2: Secret Key Steganography	16
Figure 1-3: Public Key Steganography	16
Figure 3-1: Use case diagram	26
Figure 4-1: Class diagram	31
Figure 4-2: Sequence diagram -	32
Figure 4-3: System Activities Activity Diagram	33
Figure 4-4: Context Diagram	34
Figure 5-1: Embed File - Encryption Phase	35
Figure 5-2: Extract File - Decryption Phase	36
Figure 5-3: Choose a cover file dialog	37
Figure 5-4: Choose a file to embed dialog	38
Figure 5-5: Save as dialog	38
Figure 5-6: Select Encryption Type Message box	39
Figure 5-7: Decrypt file successfully message	39
Figure 5-8: Hide file successfully message	40
Figure 5-9: Test Information Flow	42

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

LSB	Last Significant Bit
DCT	Discrete Cosine Transform

CHAPTER 1 : INTRODUCTION

1.1 INTRODUCTION

The growing use of Internet needs to take attention while we send and receive personal information in a secured manner. For this, there are many approaches that can transfer the data into different forms so that their resultant data can be understood if it can be returned back into its original form. This technique is known as encryption. However, a major disadvantage of this method is that the existence of data is not hidden. If someone gives enough time then the unreadable encrypted data may be converted into its original form.

A solution to this problem has already been achieved by using a “steganography” technique to hide data in a cover media so that other cannot notice it. The characteristics of the cover media depends on the amount of data that can be hidden, the perceptibility of the message and its robustness.

In this document, I propose a new system for hiding data stands on many methods and algorithms for image hiding where I store on data file, called sink file in an image file called as container image. The primary objective is to use steganography techniques so as to provide more security and simultaneously using less storage.

1.2 PROJECT DEFENITION

In this project, we propose to develop a system to hiding data by using "STEGANOGRAPHY" technique as I used many methods stands on some techniques to have at the back-end a software for hiding data based on hiding algorithms.

After studying the data hiding algorithms we found many ways to hiding data by using the multimedia files and the main question for me was "Where hidden data hides?" as we found by our search to know where the data hides it's important to know what is the file type of the data that it shall be hidden and the cover file type so it is possible to alter graphic or sound files slightly without losing their overall viability for the viewer and listener. With audio, you can use bits of file that contain sound not audible to the human ear. With graphic images, you can remove redundant bits of color from the image and still produce a picture

that looks intact to human eye and is difficult to discern from its original. It is in those bits that stego hides its data.

By the final of our research we developed a software uses an algorithm, to embed data in an image; The purposed system is called "Steganography", the aim of this project is to encrypt the data; the meaning of encrypt is to hide the data over an image using different steganographic algorithms, in this system LSB is the algorithms that we use to hiding the data.

1.3 STATEMENT OF PROBLEM

This project addresses the security problem of transmitting the data over internet network, the main idea coming when we start asking that how can we send a message secretly to the destination? The science of steganography answers this question. Using steganography, information can be hidden in carriers such as images, audio files, text files, videos and data transmissions.

In this document, we proposed some methods and algorithms of an image steganography system to hide a digital text of a secret message.

1.4 PROJECT OBJECTIVES

In this project we primarily concentrated on the data security issues when sending the data over the network using steganographic techniques.

The main objectives of our project are to product security tool based on steganography techniques to hider message carried by stego-media which should not be sensible to human beings and avoid drawing suspicion to the existence of hidden message.

1.5 ARCHITECTURE & COMPONENTS

In the proposed system we concentrate on finding some algorithm to hide the data inside images using steganography technique. An algorithm is designed to hide all the data inputted within the image to protect the privacy of the data. Then, the system is developed based on the new steganography algorithm.

This proposed system provides the user with two options encrypt and decrypt the data, in encryption the secret information is hiding in with image file, and on the other side the decryption is getting the hidden information from the stego image file, and also the user can show the image size after and before the encryption.

The processes of encryption and decryption of the data file consists of:

- Providing security for the data to be transmitted through network using steganography.
- Proposing an approach for hiding the data within an image using a steganographic algorithm which provides better accuracy and quality of hiding.

Microsoft Techniques is used through the .NET framework to extensively analyze the functions of the LSB algorithm in steganography. Texts and other file formats are encrypted and embedded into an image file which is then transferred to the destination.

1.6 PROJECT SCOPE

Our project scope is developed for hiding information in any image file to ensure the safety of exchange the data between different parties and provide better security during message transmission.

The scope of the project is implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save image and extruded file. We will use LSB technique; the proposed approach is to use the suitable algorithm for embedding the data in an image files; we will show a brief of this algorithm that we used to hiding data.

1.7 STEGANOGRAPHY ARCHITECTURE

Steganography is the art of hiding and transmitting data through apparently innocuous carriers in an effort to conceal the existence of the data, the word Steganography literally means covered or hiding writing as derived from Greek. Steganography has its place in security. It is not intended to replace cryptography but supplement it. [1]

Hiding a message with Steganography methods reduces the chance of a message being detected. If the message is also encrypted then it provides another layer of protection.

Therefore, some Steganographic methods combine traditional Cryptography with Steganography; the sender encrypts the secret message prior to the overall communication process, as it is more difficult for an attacker to detect embedded cipher text in a cover. It has been used through the ages by ordinary people, spies, rulers, government, and armies. There are many stories about Steganography. [1]

For example, ancient Greece used methods for hiding messages such as hiding In the field of Steganography, some terminology has developed. The adjectives 'cover', 'embedded', and 'stego' were defined at the information hiding workshop held in Cambridge, England. The term "cover" refers to description of the original, innocent message, data, audio, video, and so on. Steganography is not a new science; it dates back to ancient times. [1]

Hidden information in the cover data is known as the "embedded" data and information hiding is a general term encompassing many sub disciplines, is a term around a wide range of problems beyond that of embedding message in content. The term hiding here can refer to either making the information undetectable or keeping the existence of the information secret. [1]

Information hiding is a technique of hiding secret using redundant cover data such as images, audios, movies, documents, etc. This technique has recently become important in a number of application areas. For example, digital video, audio, and images are increasingly embedded with imperceptible marks, which may contain hidden signatures or watermarks that help to prevent unauthorized copy. It is a performance that inserts secret messages into a cover file, so that the existence of the messages is not apparent. [1]

Research in information hiding has tremendously increased during the past decade with commercial interests driving the field. Although the art of concealment "hidden information" as old as the history, but the emergence of computer and the evolution of sciences and techniques breathe life again in this art with the use of new ideas, techniques, drawing on the computer characteristics in the way representation of the data, well-known computer representation of all data including (Multimedia) is binary these representations are often the digital levels and areas and change values-aware of slight not aware or felt by Means sensual of human such as hearing, sight, the advantage use of these properties to hide data in multimedia by replace the values of these sites to the values of data to be hidden,

taking into account the acceptable limits for the changeover, and not exceeded to prevent degradation media container with a change becomes aware and felt by human. It should be noted here that although the art of hidden information come in the beginning of the computer and its techniques However, the seriousness of the work in the stenography as a stand-alone science started in 1995. [1]

1.7.1 Steganography Types

As it is known there is much communication between people and organizations through the use of the phone, the fax, computer communications, radio, and of course all of these communications should be secure. There are basically three Steganography types: -

- Pure Steganography.
- Secret key Steganography.
- Public key Steganography.

1.7.1.1 Pure Steganography

Pure Steganography is a Steganography system that doesn't require prior exchange of some secret information before sending message; therefore, no information is required to start the communication process: the security of the system thus depends entirely on its secrecy. [1]

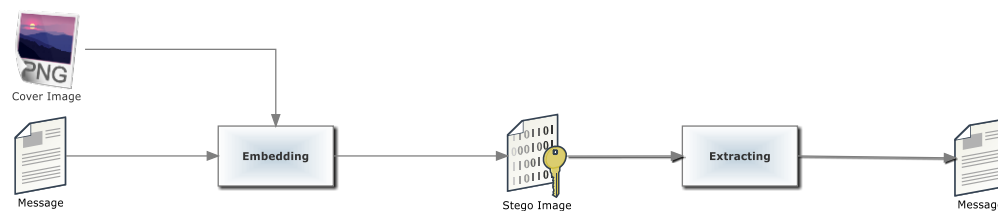


Figure 1-1: Pure Steganography

In most applications, pure Steganography is preferred, since no stego-key must be shared between the communication partners, although a pure Steganography protocols don't provide any security if an attacker knows the embedding method. [1]

1.7.1.2 Secret Key Steganography

A secret key Steganography system is similar to a symmetric cipher, where the sender chooses a cover and embeds the secret message into the cover using a secret key. If

the secret key used in the embedding process is known to the receiver, he can reverse the process and extract the secret message. [1]

Anyone who doesn't know the secret key should not be able to obtain evidence of the encoded information. [1]

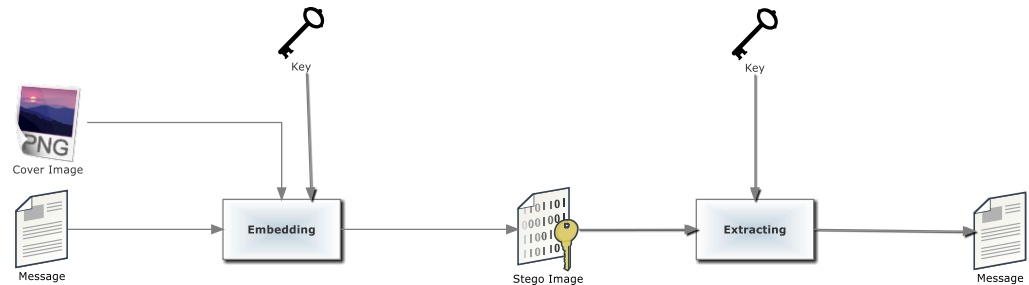


Figure 1-2: Secret Key Steganography

1.7.1.3 Public Key Steganography

Public key Steganography does not depend on the exchange of a secret key. It requires two keys, one of them private (secret) and the other public: the public key is stored in a public database, whereas the public key is used in the embedding process. The secret key is used to reconstruct the secret message. [1]

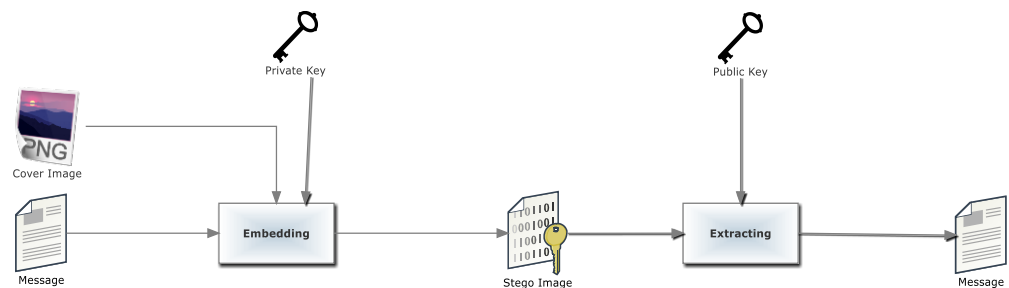


Figure 1-3: Public Key Steganography

1.7.2 Steganography Algorithms

For encryption and decryption of text messages using the secret keys steganographic system uses algorithms known as steganographic algorithms. The mostly used algorithms for embedding data into images are:

- LSB (Least Significant Bit) Algorithm (Our domain)
- JSteg Algorithm

- F5 Algorithm

1.7.2.1 LSB Algorithm

LSB embedding is the most common technique to embed message bits DCT coefficients. This method has also been used in the spatial domain where the least significant bit value of a pixel is changed to insert a zero or a one. A simple example would be to associate an even coefficient with a zero bit and an odd one with a one-bit value.

In order to embed a message bit in a pixel or a DCT coefficient, the sender increases or decreases the value of the coefficient/pixel to embed a zero or a one. The receiver then extracts the hidden message bits by reading the coefficients in the same sequence and decoding them in accordance with the encoding technique performed on it.

The advantage of LSB embedding is that it has good embedding capacity and the change is usually visually undetectable to the human eye. If all the coefficients are used, it can provide a capacity of almost one bit per coefficients using the frequency domain technique. On the other hand, it can provide a greater capacity for the spatial domain embedding with almost 1 bit per pixel for each color component. However, sending a raw image such as a Bitmap (BMP) to the receiver would create suspicion in and of itself, unless the image file is very small. Fridrich et al. proposed a steganalysis method which provides a high detection rate for shorter hidden messages [2].

Westfeld and Pfitzmann proposed another steganalysis algorithm for BMP images where the message length is comparable to the pixel count. Most of the popular formats today are compressed in the frequency domain and therefore it is not a common practice to embed bits directly in the spatial domain. Hence, frequency domain embeddings are the preferred choice for image steganography [2].

1.8 ORGANIZATION CHART AND RESPONSIBILITIES

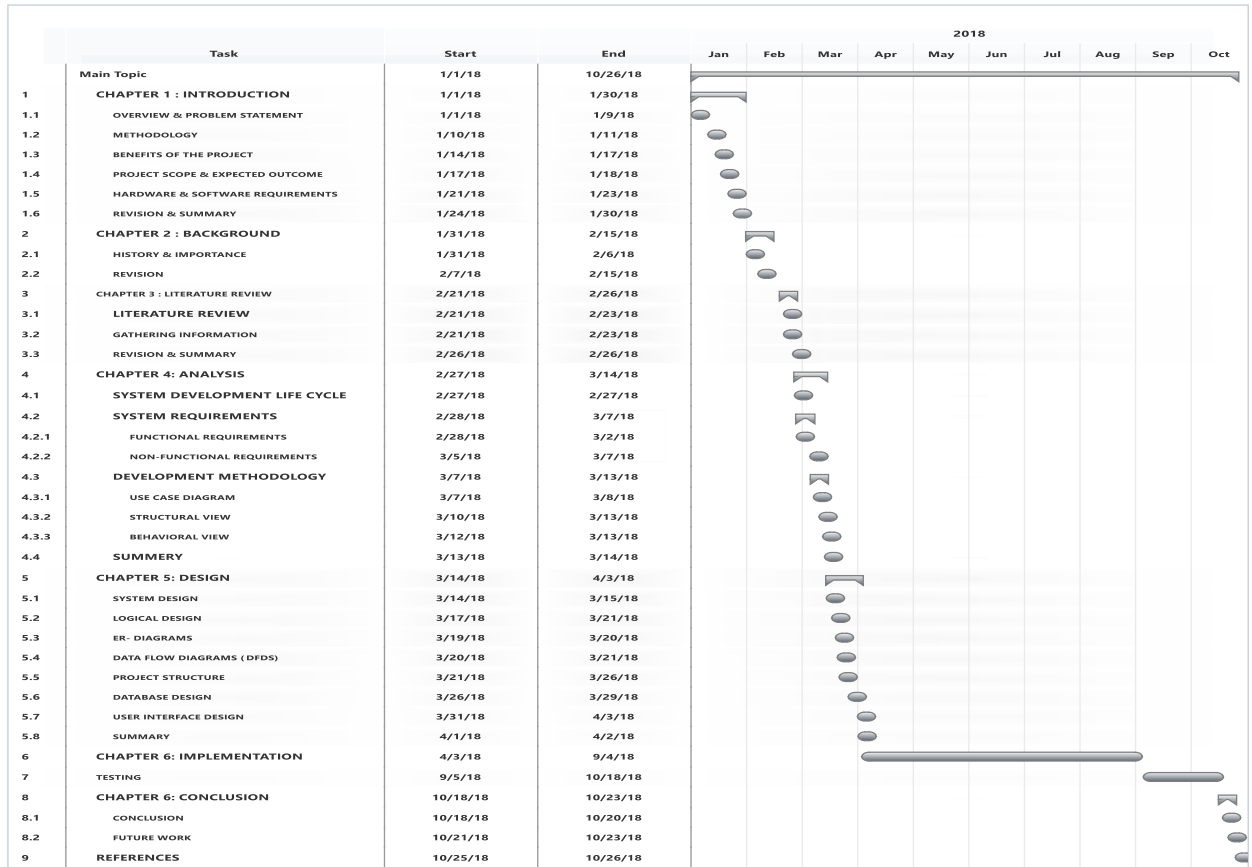


Figure 1-1 : Organizational chart

1.9 SUMMARY

This chapter gave an introduction about steganography and the meaning of hiding data; also, the problem research and the importance of Steganography in transmit the data over the network, for that reason in this document we propose a system to hiding and embedding the data inside images. Also, it provided what is the purpose and objectives of our project.

CHAPTER 2 : LITERATURE REVIEW

INTRODUCTION

In this chapter, we will provide an overview of steganography using LSB to hide the files inside images using the C#.NET technical computing language, it is development environment is used in a variety of domains, such as image and signal processing, C# offers many “toolboxes”, and a simple interface to high-performance libraries, one of the most advantages is a large user community with lots of free code and knowledge sharing and the ability to process both still images and video. It is popular because of its ease and simplicity. Also, we will mention some programs that have the same approach it is using an encryption. Then, we will give the recommendation that help to develop our program.

In this project, use a method of encrypting any data file in an image file. This process of hiding the data helps to sharing the information with others over the internet network without any potential risk. The proposed system will help to hide the content with in the image and encryption of data file with in the image will help to make the document much securer.

In this research, developed the proposed system by using steganographic algorithm which is LSB and a technique for hiding capacity and efficiency of hiding the message with in an image.

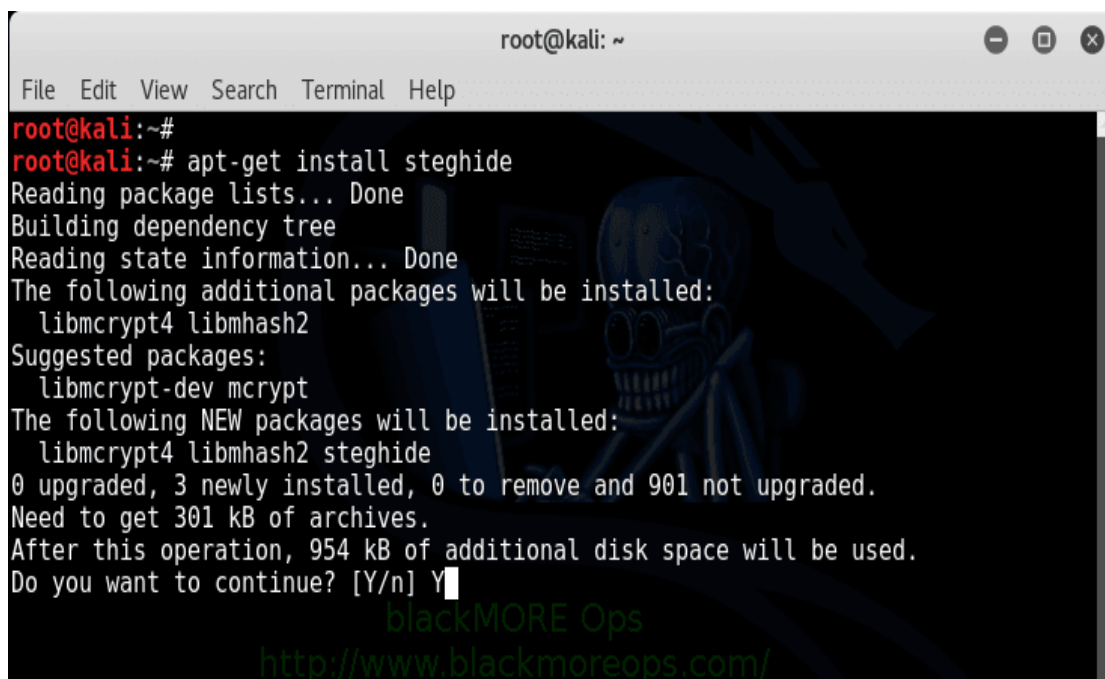
2.1 SIMILAR SYSTEMS COMPARISON

2.2.1 White Noise Storm

White Noise Storm is a DOS based tool that could easily embed secret messages in cover images without any degradation. However, the integrity of the cover image could be severely affected by noise. The tool uses LSB steganography technique to embed secret messages in PCX files. The main disadvantage of this tool is the loss of many bits that can be used to hold information. Additionally, it uses large cover images to store information that it could be stored in a smaller cover images using other tools [3].

2.2.2 StegoDos

StegoDos is public domain software that only works on 320x200 pixel images with 256 colors. The tool uses LSB steganography technique to hide secret messages. The main disadvantage of the tool is the size restriction that limits the user's cover image to 320x200 pixels in-order to have a stego-image that is similar to the original one. Another disadvantage is the dependence on the end-of- file character to end the message that does not have any significance work since the message after retrieval appears to contain garbage [3].

A screenshot of a terminal window titled 'root@kali: ~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows the command 'apt-get install steghide' being executed. The output includes package list reading, dependency tree building, and a list of additional packages (libmcrypt4, libmhash2) and suggested packages (libmcrypt-dev, mcrypt) that will be installed along with steghide. It also shows the disk space requirements and a confirmation prompt 'Do you want to continue? [Y/n]' which has been answered with 'Y'. At the bottom, there is a green watermark that reads 'blackMORE Ops' and a URL 'http://www.blackmoreops.com/'.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~#
root@kali:~# apt-get install steghide
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libmcrypt4 libmhash2
Suggested packages:
  libmcrypt-dev mcrypt
The following NEW packages will be installed:
  libmcrypt4 libmhash2 steghide
0 upgraded, 3 newly installed, 0 to remove and 901 not upgraded.
Need to get 301 kB of archives.
After this operation, 954 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
blackMORE Ops
http://www.blackmoreops.com/
```

Figure 2-1 : StegoDos

2.2.3 StegCure

StegCure uses three different LSB steganography techniques. In compared with the other tools, StegCure offers a better security and has a user-friendly functionality with interactive graphical user interface (GUI) and integrated navigation capabilities. Also, it can prevent any attacks by restricting the user to one attempt to retrieve the secret message [4].



Figure 2-2: StegCure

2.3 SIMILAR TOOLS COMPARISON

Table below shows a comparison between our proposed system tool and other image steganography tools.

	White Noise Storm [3]	StegoDos [3]	StegCure [4]	Proposed System
Capacity of Secret Message	Limited hiding capacity	Limited hiding capacity	Limited hiding capacity	Optimum hiding capacity
Image Size	-	320x200	-	Variable size
Image Format	PCX	Lossless (Ex. GIF and BMP)	GIF	BMP & PNG
Efficiency	Low	Low	Medium	Medium
Shared Key	No	Yes	No	Yes

Table 2-1: Similar tools comparison

2.4 RECOMMENDATIONS

After studying similar tools to our proposed system LSB substitution is used to embed the message into an image. It works by adjusting the LSB of the carrier image's pixels whereas, the last bit of each byte in the image is changed to a bit of the secret message that is known standard LSB (1-LSB). Also, use 2-LSB method that differs from the standard LSB method by allowing more data to be hidden into the cover image. The idea of this method is almost similar to the standard LSB, except that it replaces the

2-LSB of each byte of the cover image instead of one bit.

The LSB insertion differs depending on the number of bits in an image. In 8-bit images, the last bit of each byte in the image is changed to a bit of the secret message. However, it has a major limitation, which is embedding only small size data into images. While in 24-bit images, the last bit of each RGB component is changed which allows more data to be hidden.

CHAPTER 3 : SYSTEM ANALYSIS

3.1 INTRODUCTION

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

3.2 STAKEHOLDERS

Stakeholders must be able to make decisions, but need not be human: "An stakeholder might be a person, a company or organization, a computer program, or a computer system — hardware, software, or both." Actors are always stakeholders, but not all stakeholders are actors. The following Table 3.1: Stakeholders and its Description.

Stakeholder	Description
User	That have major controlling of the whole systems process and able to access or use all modules of the application.
Program	The Application itself which can perform the user operation in main process encrypt (hide image in another image), decrypt (unhide image from Stego-image).

Table 3-1: Stakeholders

- **Choose an image** :Select the picture, open the image, the image review.
- **Processing image** :typing image in the image box pressing the button for the treatment process

- **Hide image** :The program hides image.

A use case is a written description of how users will perform tasks on your website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

The use case description of our project as follows;

Actors	User
Description	The user wants to hide image in another image file
Data	Cover image, Image to covered (Message)
Response	<ol style="list-style-type: none"> 1.Select the Embed from the tabs. 2.Select the algorithm type. 3.Select the cover file by click browse until the open file dialog "Choose a Cover" file appears. 4.From the "Choose a Cover File" dialog choose the cover file and click open. 5.Select the embed file by click browse until the open file dialog "Choose a File To Embed" appear. 6.From the " Choose a File To Embed " dialog choose the embed file and click open. 7.Select the output file by click browse until the save file dialog "Save as .." appear. 8.Select the path where you want to save the file and type the file name then click save. 9.Finally click on the embed button to hiding the embed file information inside the cover file. 10.After the hiding data, message box will appear to tell user the hiding operation is successfully.
Comments	In case the cover image can't hide the image that user trying to hide the system will show alert message to the user.
Stimuli	Start hiding process by clicking on Embed button.

Table 3-2: Use case description – Embedding process

Actors	User
Description	The user wants to extract image from stego-image file
Data	Stego-Image
Response	<p>To start extracting a file you must the following steps:</p> <ol style="list-style-type: none"> 1.Select the Extract from the tabs. 2.Select the algorithm type. 3.Select the cover file by click browse until the open file dialog "Choose a Cover" file appears. 4.From the "Choose a Cover File" dialog choose the cover file and click open. 5.Select the output file by click browse until the save file dialog "Save as .." appear. 6.Select the path where you want to save the file and type the file name then click save. 9.Finally click on the extract button to extracting the embed file information. 10.After the extracting data, message box will appear to tell user the extracting operation is successfully.
Comments	The system will show message after the extracting process to let the user know that the process is end.
Stimuli	Start hiding process by clicking on Extract button.

Table 3-3: Use case description – Extracting process

3.4 NON-FUNCTIONAL REQUIREMENTS

- For user interfaces we take in consideration that they should has a standard look and being user friendly at the same time to make sure that users' attention will not be distracted and interface to provide more flexibility and scalability.
- The program will be in the English language
- The program must be fast in processing
- The program must to hide the image within the image and then extract image from the image properly.
- All function must be works well then system will be a high quality

3.5 SEMI-FORMAL REQUIREMENTS

3.5.1 General Use Case Diagram

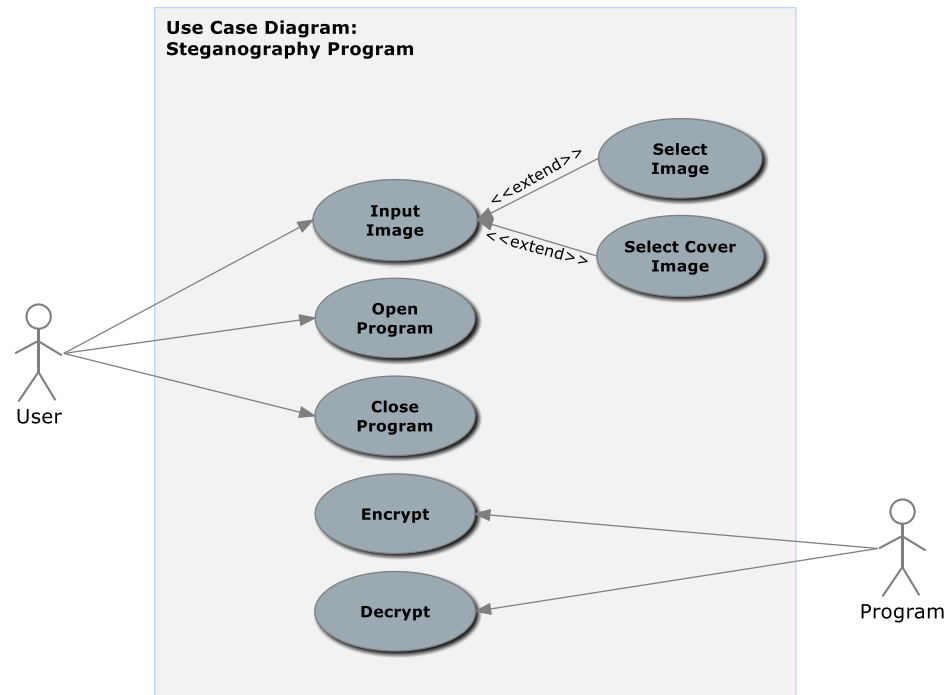


Figure 3-1: Use case diagram

3.5.2 Actors

An actor in the Unified Modeling Language (UML) "specifies a role played by a user or any other system that interacts with the subject." "An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject."

There are two actors in our application:

1. User
2. Program

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases.

3.5.4 Hardware and Software Requirements

The execution phase was developed based upon three phases. The different phases are encryption, decryption and implementation phases. We require few hardware and software interfaces for implementing these phases.

The software interface which is implemented in this project is done using the Microsoft .NET running in the Windows environment. The main aim of the project is to improve the data security when the data is transmitted using the transmission medium. This can be done by embedding the secret data into the image file and then transmitting the encrypted data through the transmission medium. Then, the carrier image file is decrypted at the destination by using the secret key.

For implementing the above procedure, we used the Microsoft .NET framework to create the steganographic application. The proposed method in this project can be used for both "encryption" and "decryption" of the message from the digital image.

3.5.4.1 Microsoft .NET

Microsoft .NET is a framework developed by Microsoft in the year 2002. The main aim of the .NET framework is to build web and user interactive GUI (Graphical User Interface) applications. The Windows forms web application classes that are used for creating new windows form based applications. Microsoft .NET is a user friendly language which helps us build the required web application easily.

- .NET framework acts as a common platform for building, organizing, and running web applications and web services.
- .NET has common libraries like ASP.NET, ADO.NET and Windows Forms.
- .NET supports multiple languages like C, C#, Visual Basic, Jscript.
- .NET is a user friendly language and is easy to learn compared to other languages for example JAVA because in .NET the coding is very easier.
- .NET supports additional data sources of ADO.NET like Oracle and ODBC.

- Using .NET platform we can build web applications as required, the applications are highly secure because it uses access control lists and security identifiers.
- One of the main important sections of Microsoft .NET environment is CLR. The Common Language Runtime (CLR) is heart to .NET framework.
- CLR is same as JVM (JAVA Virtual Machine) in .NET, .NET program runs only on platforms that support CLR.

Now, in this project I have chosen Microsoft .net platform for building the windows based steganographic application. The main components of .NET which used in this project are Visual C# (Thai, 2003).

3.5.4.2 CLR: (COMMON LANGUAGE RUNTIME)

As mentioned above, the "Common Language Runtime" is an important component of .NET platform. The CLR handles the object layouts and manages references to objects. It makes the designing architecture simpler and easier. I used Microsoft Visual C# for structuring the application. CLR is efficient when C# language is used for writing programs. The various advantages when the C# language is employed are:

- Complete Object oriented design
- Provides high security and stronger to break the code
- Uses the logics of visual basic and C++ languages
- The syntax is simple is similar to languages C, C++ (Microsoft, 2010, MSDN).

3.5.4.3 Windows Forms

Visual Studio Windows application is supported by the Microsoft .NET frame work and it allows a rich set of classes so that the Windows application can be built by any .net programming language like Visual Basic .net and Visual C#.

Here, in this project used visual c# to create the windows based application. The windows based application differs with Web form application. In windows based application, we create some type of forms which are used for user interface. These forms are known as "Windows forms".

- Using windows forms we can easily drag and drop the controls like radio buttons and text boxes etc., using the Windows forms designer.
- Windows forms are also known as win forms in which the standard built in controls can be used.
- By using the custom Graphical User Interface (GUI) controls we can modify the controls and also add new ones (MSDN, 2010).

3.5.4.4 VISUAL C#

Visual C# is the one of the component in Microsoft Visual studio which works similar to Visual basic. Creating applications using Visual C# is easier compared to the JAVA.

- In order to create the application, we need to use the designer tool and tool box so that required tool like radio button, text boxes etc., can be placed.
- After designing the next phase is to relate them with specified functions. This can be done using visual C# coding.
- The coding can be done by double clicking in the elements on the designer tool.
- The program or code which is written using C# language is saved with “.CS” extension.
- The code can be compiled using “Debug” option. When we click on debug option the .NET architecture creates the class file.
- After creation of class file, Common Language Runtime (CLR) converts the class file into the machine language that is compatible with the hardware since CLR supports cross-language integration.
- The code as such can be compiled on any other operating system.
- Microsoft visual studio provides a bunch of tools for creating Windows as well as web applications.

3.6 PREDESIGN

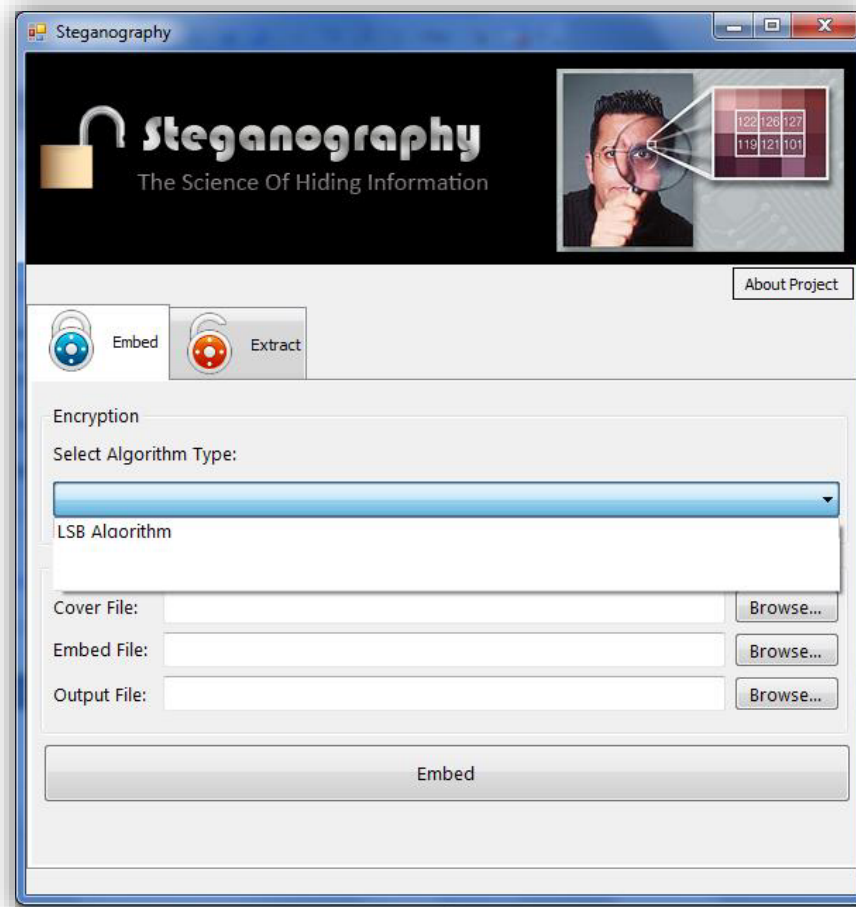


Figure 3.3: Embed File - Encryption Phase (Main interface)

3.7 CONCLUSION

In this chapter, we have described the introduction about system analysis and design. We discuss the result of software requirements by functional requirement, non- functional requirement. Furthermore, we also explained use case diagram which consists of two description actor and use case. Finally, we describe the architecture design phase through system design.

CHAPTER 4 : SYSTEM DESIGN

4.1 INTRODUCTION

Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software is determined, software developers will design or employ designers to develop a plan for a solution. It includes construction component and algorithm implementation issues which shown in as the architectural view. During this chapter we will introduce some principles that are considered through the Software design.

4.2 STRUCTURAL VIEW

Structural diagrams are used to describe the relationships between classes. Structural view can be described using class diagram.

4.2.1 Class Diagram

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid.

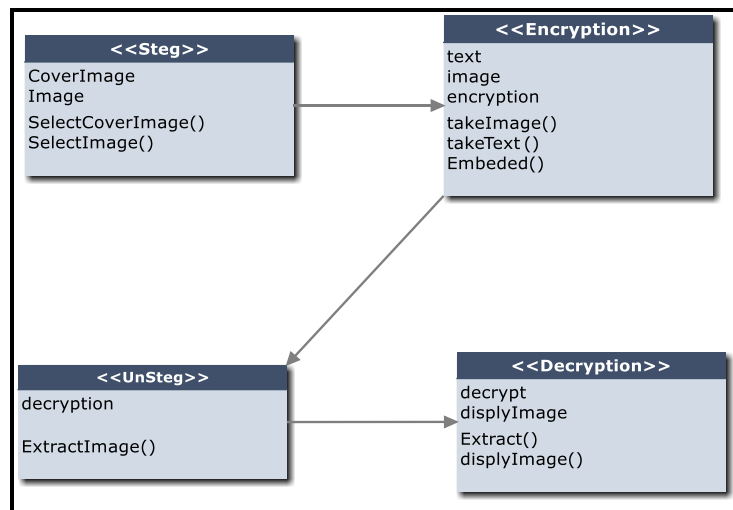


Figure 4-1: Class diagram

Class diagrams are more prevalent than object diagrams. Normally you will build class diagrams plus occasional object diagrams illustrating complicated data structures or message-passing structures.

4.3 BEHAVIORAL VIEW

As we mentioned previously that activity diagram, and sequence diagram provide the behavioral view for our project. Behavioral diagrams are used to describe the interaction between the actors and the system. All the activities that are performed by the actors and the system are introduced in some way.

4.3.1 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

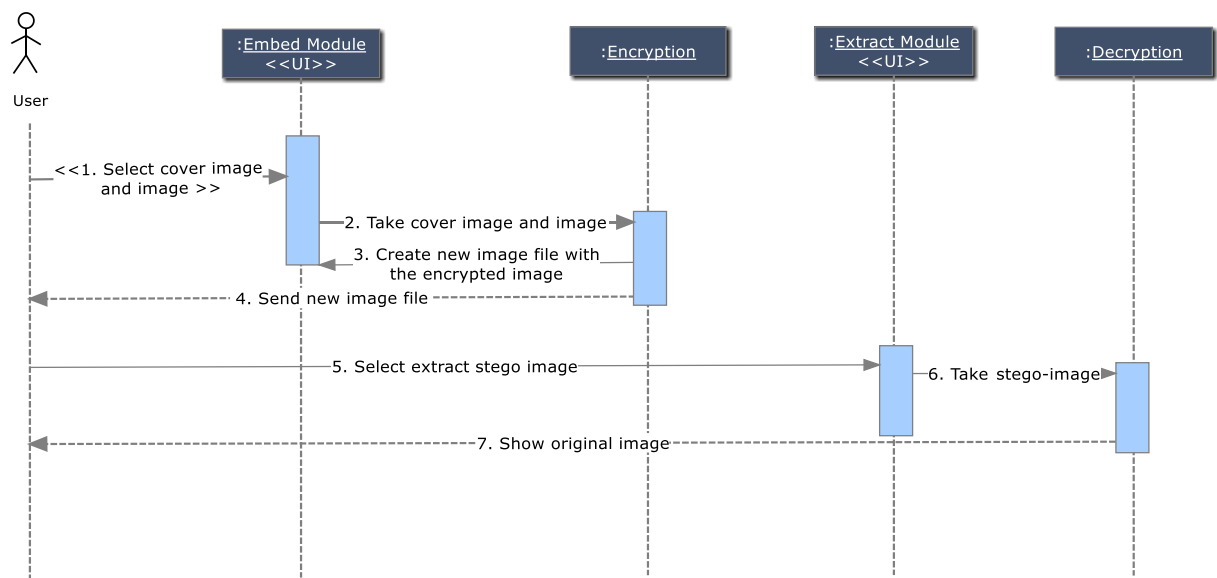


Figure 4-2: Sequence diagram -

4.3.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-

step workflows of components in a system. An activity diagram shows the overall flow of processes.

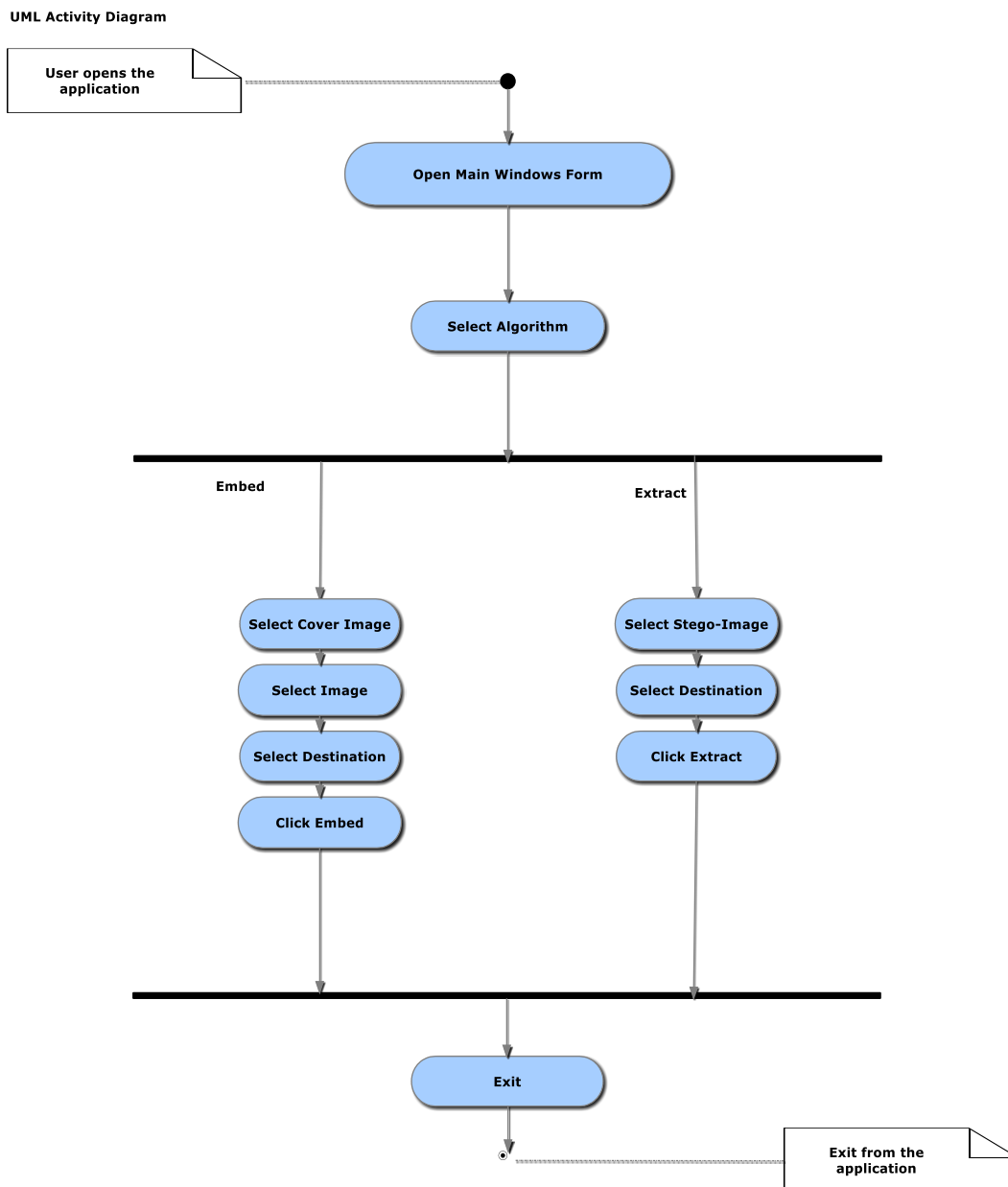


Figure 4-3: System Activities Activity Diagram

4.3.3 Data Flow Diagram

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 information 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.



Figure 4-4: Context Diagram

CHAPTER 5 : IMPLEMENTATION & TESTING

5.1 SCREEN SHOTS OF THE SYSTEM:

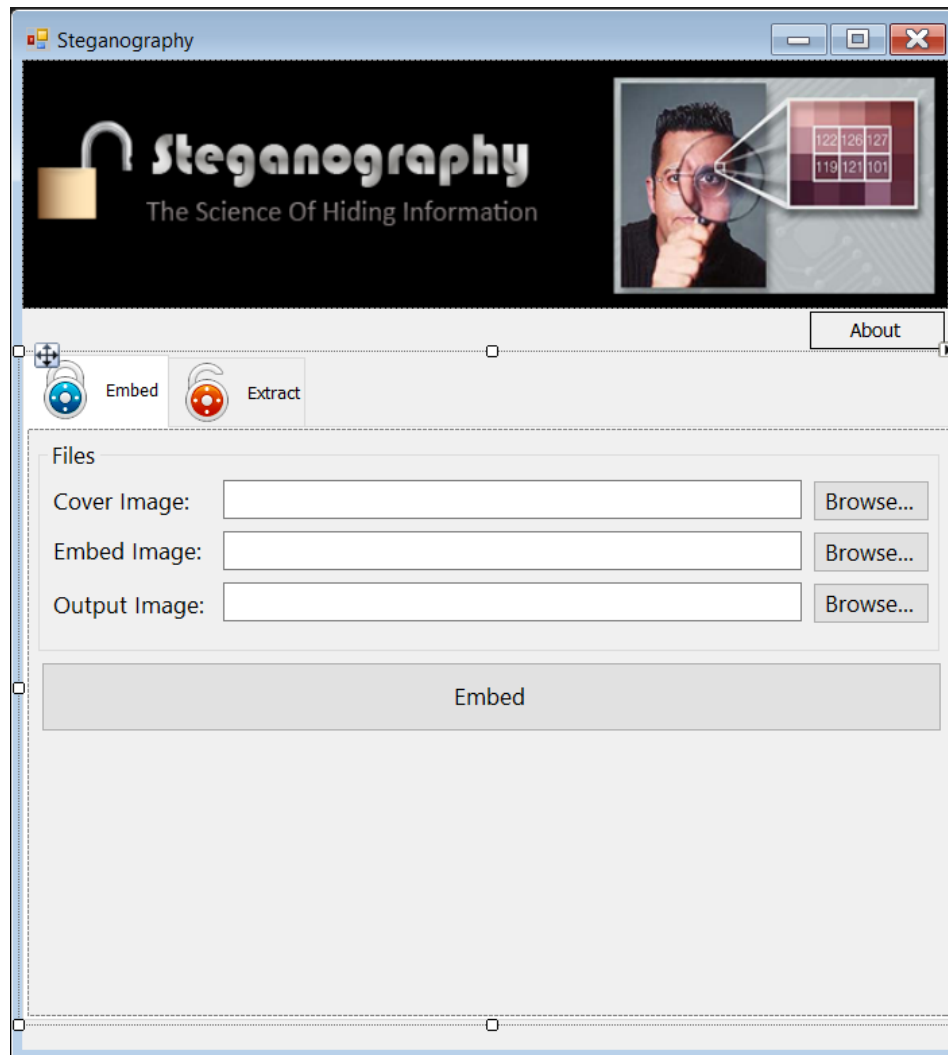


Figure 5-1: Embed File - Encryption Phase

To start hiding a file you must the following steps:

1. Select the Embed from the tabs.
2. Select the algorithm type.
3. Select the cover file by click browse until the open file dialog "Choose a Cover" file appears.
4. From the "Choose a Cover File" dialog choose the cover file and click open.

5. Select the embed file by clicking browse until the open file dialog "Choose a File To Embed" appears.
6. From the "Choose a File To Embed" dialog, choose the embed file and click open.
7. Select the output file by clicking browse until the save file dialog "Save as .." appears.
8. Select the path where you want to save the file and type the file name, then click save.
9. Finally, click on the embed button to hide the embed file information inside the cover file.
10. After hiding the data, a message box will appear to tell the user the hiding operation is successful.

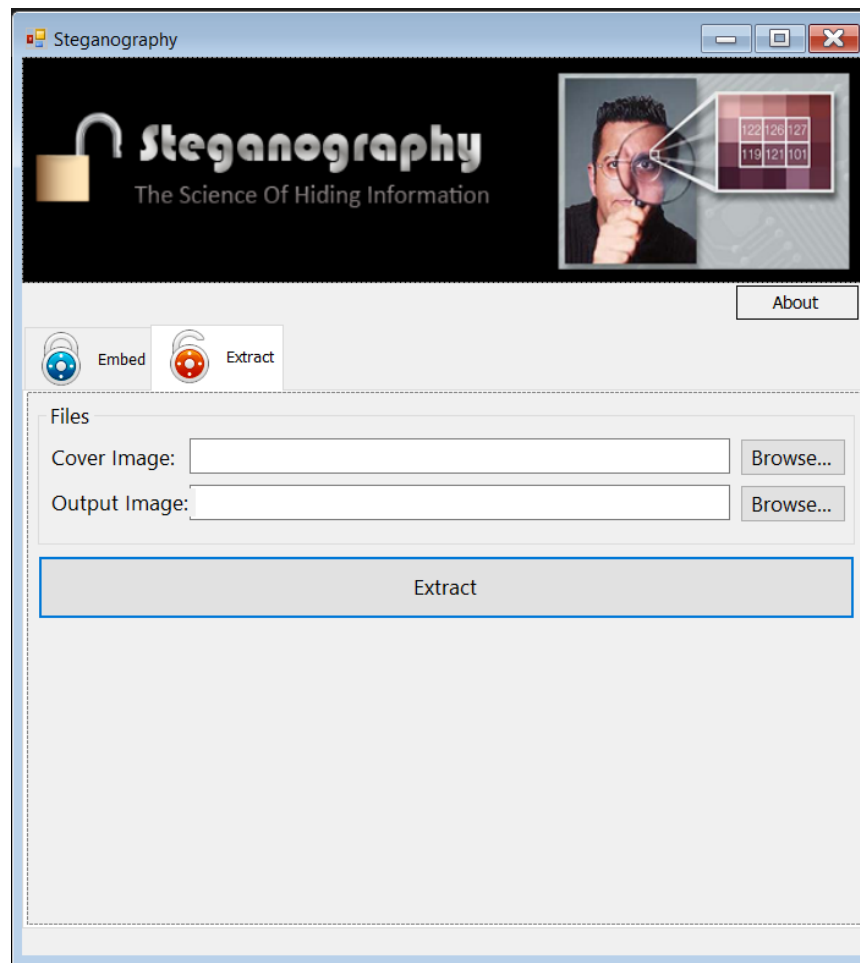


Figure 5-2: Extract File - Decryption Phase

To start extracting a file, you must follow the following steps:

1. Select the Extract from the tabs.

2. Select the algorithm type.
3. Select the cover file by clicking browse until the open file dialog "Choose a Cover" file appears.
4. From the "Choose a Cover File" dialog choose the cover file and click open.
5. Select the output file by clicking browse until the save file dialog "Save as .." appears.
6. Select the path where you want to save the file and type the file name then click save.
9. Finally click on the extract button to extracting the embed file information.
10. After the extracting data, message box will appear to tell user the extracting operation is successfully.

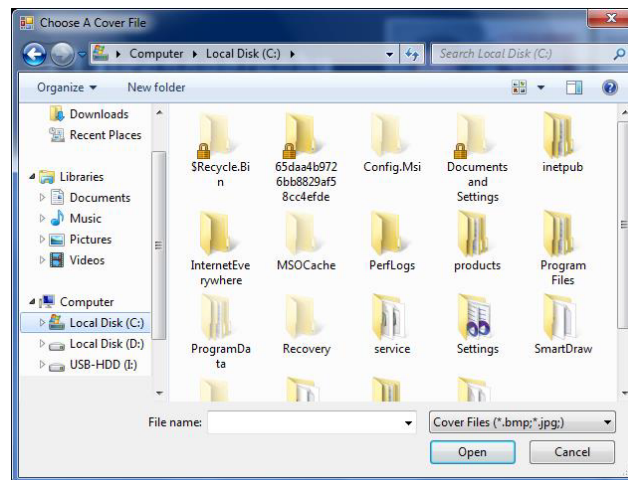


Figure 5-3: Choose a cover file dialog

From this screen you must select the cover file by choosing the right path and selecting the file and then clicking the open button.

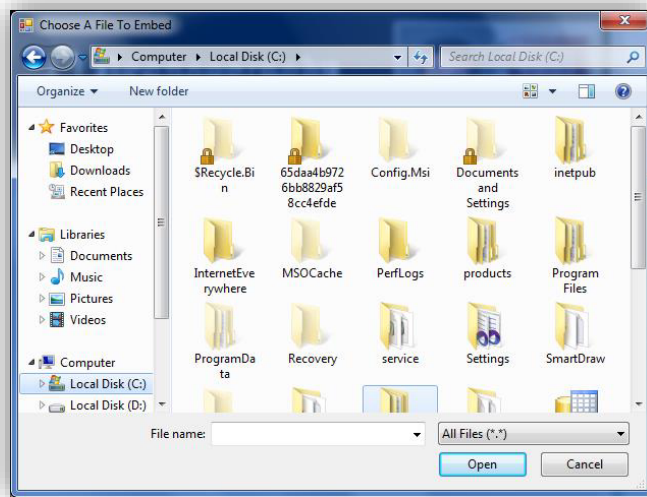


Figure 5-4: Choose a file to embed dialog

From this screen you must select the embed file by choose the right path and select the file and then click open button.

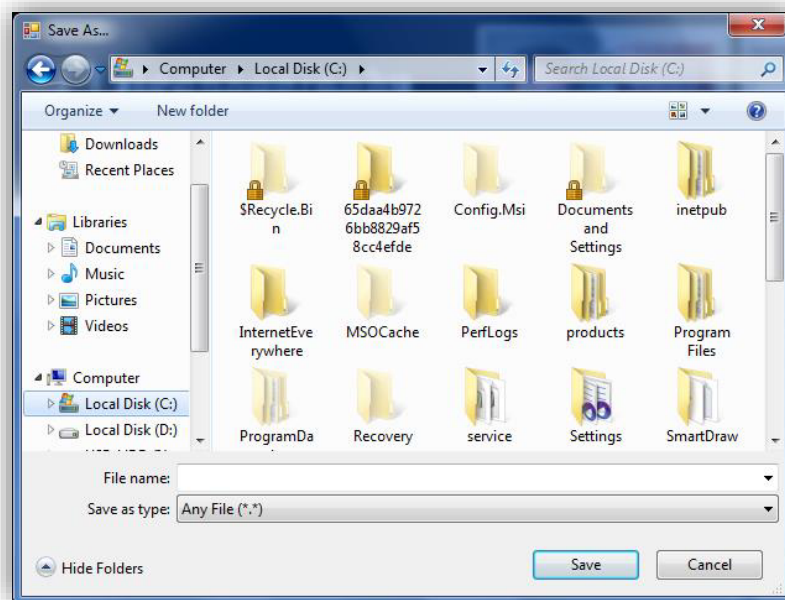


Figure 5-5: Save as dialog

From this screen you must select the right path and type the file name to save the file in the selected path then click save button.

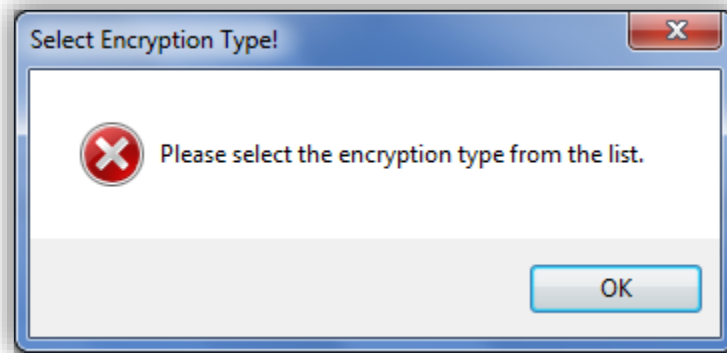


Figure 5-6: Select Encryption Type Message box

Select encryption type message box to notify the user of select the encryption type from the list.

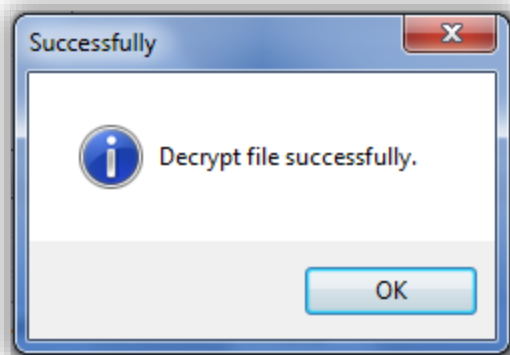


Figure 5-7: Decrypt file successfully message

Successfully hide file message box to notify the user of the extract operation is successfully.

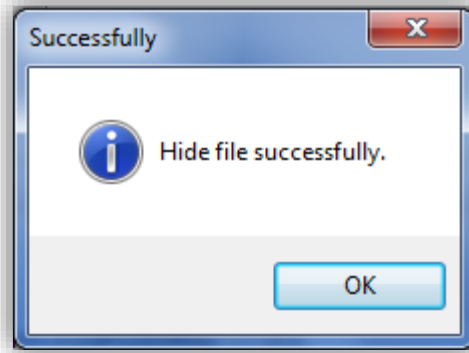


Figure 5-8: Hide file successfully message

Successfully hide file message box to notify the user of the embed operation is successfully.

5.2 DEFINITION AND THE GOAL OF TESTING

Process of creating a program consists of the following phases: 1. defining a problem; 2. designing a program; 3. building a program; 4. analyzing performances of a program, and 5. final arranging of a product.

According to this classification, software testing is a component of the third phase, and means checking if a program for specified inputs gives correctly and expected results. So the main aim of testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running in different operating environments.

Software testing (*Figure 3.1*) is an important component of software quality assurance, and many software organizations are spending up to 40% of their resources on testing. For life-critical software (e.g., flight control) testing can be highly expensive. Because of that, many studies about risk analysis have been made. This term means the probability that a software project will experience undesirable events, such as schedule delays, cost overruns, or outright cancellation.

There are a many definitions of software testing, but one can shortly define that as: A process of executing a program with the goal of finding errors. So, testing means that one inspects behavior of a program on a finite set of test cases (a set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement) for which valued inputs always exist.

In practice, the whole set of test cases is considered as infinite, therefore theoretically there are too many test cases even for the simplest programs. In this case, testing could require months and months to execute. So, how to select the most proper set of test cases? In practice, various techniques are used for that, and some of them are correlated with risk analysis, while others with test engineering expertise.

Testing is an activity performed for evaluating software quality and for improving it. Hence, the goal of testing is systematical detection of different classes of errors (error can be defined as a human action that produces an incorrect result) in a minimum amount of time and with a minimum amount of effort.

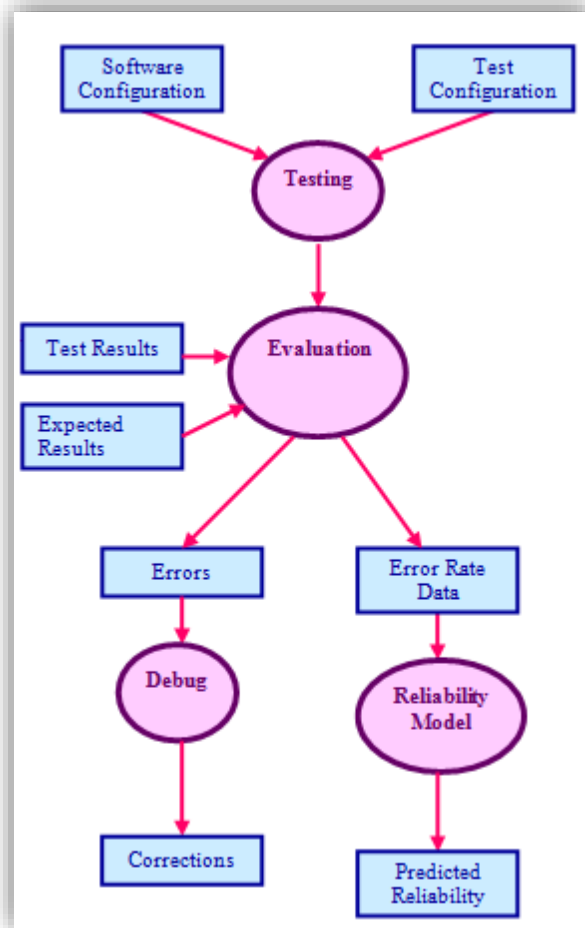


Figure 5-9: Test Information Flow

There are different types of approaches for testing a .NET framework based application. The types of testing are:

- Unit testing
- Validation testing
- Integration testing
- User acceptance testing
- Output testing

- Black box and white box testing.

5.3 METHODS OF TESTING

5.3.1 Unit Testing

Unit testing is the approach of taking a small part of testable application and executing it according to the requirements and testing the application behavior. Unit testing is used for detecting the defects that occur during execution (MSDN, 2010).

When an algorithm is executed, the integrity should be maintained by the data structures. Unit testing is made use for testing the functionality of each algorithm during execution.

Unit testing can be used in the bottom up test approach which makes the integration test much easier. Unit testing reduces the ambiguity in the units. Unit testing uses regression testing, which makes the execution simpler. Using regression testing, the fault can be easily identified and fixed.

In this project, the purposed system of hiding the data using different phases likes encryption, decryption, etc. So, for getting the correct output all the functions that are used are executed and tested at least once making sure that all the control paths, error handling and control structures are in proper manner.

Unit testing has its applications for extreme programming, testing unit frame works and good support for language level unit testing.

5.3.2 Validation Testing

Validation is the process of finding whether the product is built correct or not. The software application or product that is designed should fulfill the requirements and reach the expectations set by the user. Validation is done while developing or at the final stage of development process to determine whether it is satisfies the specified requirements of user.

Using validation test the developer can qualify the design, performance and its operations. Also the accuracy, repeatability, selectivity, Limit of detection and quantification can be specified using Validation testing (MSDN, 2010).

5.3.3 Output Testing

After completion of validation testing the next process is output testing. Output testing is the process of testing the output generated by the application for the specified inputs. This process checks weather the application is producing the required output as per the user's specification or not.

The output testing can be done by considering mainly by updating the test plans, the behavior of application with different type of inputs and with produced outputs, making the best use of the operating capacity and considering the recommendations for fixing the issues (MSDN, 2010).

5.3.4 Integration Testing

Integration testing is an extension to unit testing, after unit testing the units are integrated with the logical program. The integration testing is the process of examining the working behavior of the particular unit after embedding with program. This procedure identifies the problems that occur during the combination of units.

The integration testing can be commonly done in three approaches:

- Top-down approach
- Bottom-up approach
- Umbrella approach

5.3.4.1 Top-down approach:

In the top-down approach the highest level module should be considered first and integrated. This approach makes the high level logic and data flow to test first and reduce the necessity of drivers.

One disadvantage with top-down approach is its poor support and functionality is limited (MSDN, 2010).

5.3.4.2 Bottom-up approach:

Bottom-up approach is opposite to top-down approach. In this approach, the lowest level units are considered and integrated first. Those units are known as utility units. The utility units are tested first so that the usage of stubs is reduced. The disadvantage in this method is that it needs the respective drivers which make the test complicated, the support is poor and the functionality is limited (MSDN, 2010).

5.3.4.3 Umbrella approach:

The third approach is umbrella approach, which makes use of both the top - bottom and bottom - top approaches. This method tests the integration of units along with its functional data and control paths. After using the top - bottom and bottom-top approaches, the outputs are integrated in top - bottom manner.

The advantage of this approach is that it provides good support for the release of limited functionality as well as minimizing the needs of drivers and hubs. The main disadvantage is that it is less systematic than the other two approaches (MSDN, 2010).

5.4 USER ACCEPTANCE TESTING

User acceptance testing is the process of obtaining the confirmation from the user that the system meets the set of specified requirements. It is the final stage of project; the user performs various tests during the design of the applications and makes further modifications according to the requirements to achieve the final result.

The user acceptance testing gives the confidence to the clients about the performance of system.

5.5 BLACK BOX AND WHITE BOX TESTING

Black box testing is the testing approach which tells us about the possible combinations for the end-user action. Black box testing doesn't need the knowledge about the interior connections or programming code. In the black box testing, the user tests the application by giving different sources and checks whether the output for the specified input is appropriate or not.

White box testing is also known as "glass box" or "clear box" or "open box" testing. It is opposite to the black box testing. In the white box testing, we can create test cases by checking the code and executing in certain intervals and know the potential errors. The analysis of the code can be done by giving suitable inputs for the specified applications and using the source code for the application blocks.

The limitation with the white box testing is that the testing only applies to unit testing, system testing and integration testing (MSDN, 2010).

These are the different testing approaches that can be used for testing the application which is developed using Microsoft Visual studio (.NET).

5.6 TEST CASES

STEP	TEST CONDITION	ACTION	EXPECTED RESULT	ACTUAL RESULT
1. Embed File				
1.1	Select Cover Image	Click	Check browse images	Done
1.2	Select Embed Image	Click	Check browse images	Done
1.3	Select Output Image	Click	Check browse images	Done
1.4	Click Embed Button	Click	Check embed function	Done
2. Extract				
2.1	Select Cover Image	Click	Check browse images	Done
2.2	Select Output Image	Click	Check browse images	Done
2.3	Click Extract Button	Click	Check extract function	Done

Table 5-1: Test Cases

CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATION

6.1 CONCLUSIONS

Although only some of the main image steganographic techniques were discussed in this document, one can see that there exists a large selection of approaches to hiding information in images. All the major image file formats have different methods of hiding messages, with different strong and weak points respectively. Where one technique lacks in payload capacity, the other lacks in robustness. For example, the patchwork approach has a very high level of robustness against most type of attacks, but can hide only a very small amount of information.

Least significant bit (LSB) in both BMP and GIF makes up for this, but both approaches result in suspicious files that increase the probability of detection when in the presence of a warden.

The proposed approach in this project uses a new steganographic approach called image steganography. The application creates a stego image in which the personal data is embedded inside the cover file image.

Used the Least Significant Bit algorithm in this project for developing the application which is faster and reliable and compression ratio is moderate compared to other algorithms.

6.2 FUTURE RECOMMENDATIONS

The major limitation of the application is designed for images cover files. It accepts only images as a carrier file.

The future work on this project is to improve the compression ratio of the image to the text. This project can be extended to a level such that it can be used for the different types of multimedia files.

REFERENCES

- [1] Rosziati Ibrahim and Teoh Suk Kuan, Steganography Imaging System (SIS): Hiding Secret Message inside an Image
http://www.iaeng.org/publication/WCECS2010/WCECS2010_pp144-148.pdf
- [2] Zaidoon Kh. AL-Ani, A.A.Zaidan, B.B.Zaidan and Hamdan.O.Alanazi, Overview: Main Fundamentals for Steganography
<http://arxiv.org/ftp/arxiv/papers/1003/1003.4086.pdf>
- [3] Bere Sachin Sukhadeo, User Aware Image Tag Refinement
<http://www.ijcsmr.org/eetecme2013/paper19.pdf>
- [4] Youssef Bassil , A Simulation Model for the Waterfall Software Development Life Cycle, 2011
<http://arxiv.org/ftp/arxiv/papers/1205/1205.6904.pdf>
- [5] ANALYSIS MODEL WATERFALL MODEL
<http://www.scribd.com/doc/87322736/Analysis-Model-Waterfall-Model>
- [6] Data Flow Diagram Symbols
<http://www.idi.ntnu.no/~sif8035/pdf/flesn/notation.pdf>
- [7] Donald S. Le Vie, Jr., Understanding Data Flow Diagrams
http://ratandon.mysite.syr.edu/cis453/notes/DFD_over_Flowcharts.pdf
- [8] B. Beizer, Software Testing Techniques. London: International Thompson Computer Press, 1990.
- [9] B. Beizer, Black Box Testing. New York: John Wiley & Sons, Inc., 1995.
- [10] A. Bertolino, "Chapter 5: Software Testing," in IEEE SWEBOK Trial Version 1.00, May 2001.
- [11] Testing Overview and Black-Box Testing Techniques
<http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf>
- [12] Jovanović, Irena , Software Testing Methods and Techniques,
<http://www.internetjournals.net/journals/tir/2009/January/Paper%2006.pdf>
- [13] Hong Cai1 and Sos S.Agaian2, BREAKING F5 IN COLOR IMAGES WITH LOW EMBEDDING RATES
- [14] Yao Wang, DCT and Transform Coding
- [15] N. F. Johanson and S.Jajodia. "Exploring Steganography: Seeing The Unseen," IEEE CJ, February 1998, pp. 26-43.
- [16] L. Y. Por et al., "StegCure: A Comprehensive Steganographic Tool Using Enhanced LSB Scheme," WSEAS Transactions on Computers, vol. 7, no. 8, August 2008, pp. 1309-1318.

APPENDICES

EMBED FILE CODING

```
#region Embedded the file into the cover by using Hiding methos
    public static Bitmap EncryptLayer(Bitmap OriginalBitmapPicture, long
startPosition, long endPosition, bool blnHideFileNameData, long lngFileNameLength,
byte[] BytesOfHiddenFileData, string fileName, long lngfileSize)
    {
        // Get the height and width of the original bitmap picture
        int height = OriginalBitmapPicture.Height;
        int width = OriginalBitmapPicture.Width;

        // Declare the index of the LSB as constant
        const int LSB = 7;

        // Save the original bitmap image in another bitmap picture called
StegoBitmapPicture to make the updates
        Bitmap StegoBitmapPicture = OriginalBitmapPicture;

        // Declare the variables i, j to use them for get the pixels of the
image
        int i = 0, j = 0;

        // ArrStegoData to store the binary value of every byte of the char of
the file that we want to hide
        bool[] ArrStegoData = new bool[8];

        // blnArrOfRedByte, blnArrOfGreenByte, blnArrOfBlueByte to store
boolean values of the binaries of the RGB of the original file image
        bool[] blnArrOfRedByte = new bool[8];
        bool[] blnArrOfGreenByte = new bool[8];
        bool[] blnArrOfBlueByte = new bool[8];

        // pixel to get the values of the original image pixel
        Color pixel = new Color(); // Pixel (R,G,B) = (0-255, 0-255, 0-255)

        // ArrOfRedByte, ArrOfGreenByte, ArrOfBlueByte to store the binary
values of the RGB of the original image pixel
        byte ArrOfRedByte, ArrOfGreenByte, ArrOfBlueByte;

        bool blnCheckIfFinishedHidingTheFileName = false;

        // Check if hide the file name data are stored or not - if it hided
then the value must be false
        // else if the value is true then start hidding the file name data
        if (blnHideFileNameData)
        {
            int intFileNameCharIndex = 0;
            int intFileDataCharIndex = 0;

            for (i = 0; i < height; i++)
                for (j = 0; j < width; j++) // we multiply by 3 not * because
we store in byte everytime
```

```

    {
        //Put here we shall take in our considration the picture
length if we arrived to the last pixel then

        if (intFileNameCharIndex == lngFileNameLength) // finished
hiding the file information
        {
            blnCheckIfFinishedHidingTheFileName = true;
        }

        if (blnCheckIfFinishedHidingTheFileName == false)
        {

            // Convert the byte for every char of the file name
ConvertByteToBoolean((byte)fileName[intFileNameCharIndex], ref ArrStegoData);

            // Get the pixel of the input image
            pixel = OriginalBitmapPicture.GetPixel(j, i);

            //Get the values of RGB of the pixel
            ArrOfRedByte = pixel.R;
            ArrOfGreenByte = pixel.G;
            ArrOfBlueByte = pixel.B;

            // Convert the bytes of the RGB colors to binary values
            ConvertByteToBoolean(ArrOfRedByte, ref
blnArrOfRedByte);
            ConvertByteToBoolean(ArrOfGreenByte, ref
blnArrOfGreenByte);
            ConvertByteToBoolean(ArrOfBlueByte, ref
blnArrOfBlueByte);

            // Now we want to store the 8 bits of the file name
that we want to hide by changing the LSB of every color byte
            // but on every pixel we can store 3 bits of the byte
            // so here to save the sorting we must divide the store
process on three stages
            // first store first three values from j = 0 to j = 2
            // second the next three bits from j =3 to j=5
            // finally the final two bits on j =6 and 7

            if (j % 3 == 0)
            {
                blnArrOfRedByte[LSB] = ArrStegoData[0];
                blnArrOfGreenByte[LSB] = ArrStegoData[1];
                blnArrOfBlueByte[LSB] = ArrStegoData[2];
            }
            else if (j % 3 == 1)
            {
                blnArrOfRedByte[LSB] = ArrStegoData[3];
                blnArrOfGreenByte[LSB] = ArrStegoData[4];
                blnArrOfBlueByte[LSB] = ArrStegoData[5];
            }
            else
            {
                blnArrOfRedByte[LSB] = ArrStegoData[6];

```

```

        blnArrOfGreenByte[LSB] = ArrStegoData[7];
        intFileNameCharIndex++;
    }

    // Get the new updated pixel of the image after hide
the file name
    Color newPixelColor =
Color.FromArgb((int)ConvertBooleanToByte(blnArrOfRedByte),
(int)ConvertBooleanToByte(blnArrOfGreenByte),
(int)ConvertBooleanToByte(blnArrOfBlueByte));
    StegoBitmapPicture.SetPixel(j, i, newPixelColor);
}
else
{
    if (intFileDataCharIndex ==
BytesOfHiddenFileData.Length) // finished hiding the file information
    {
        goto endPosition;
    }
}

ConvertByteToBoolean((byte)BytesOfHiddenFileData[intFileDataCharIndex], ref
ArrStegoData);

    pixel = OriginalBitmapPicture.GetPixel(j, i);

    ArrOfRedByte = pixel.R;
    ArrOfGreenByte = pixel.G;
    ArrOfBlueByte = pixel.B;

    ConvertByteToBoolean(ArrOfRedByte, ref
blnArrOfRedByte);
    ConvertByteToBoolean(ArrOfGreenByte, ref
blnArrOfGreenByte);
    ConvertByteToBoolean(ArrOfBlueByte, ref
blnArrOfBlueByte);

    if (j % 3 == 0)
    {
        blnArrOfRedByte[LSB] = ArrStegoData[0];
        blnArrOfGreenByte[LSB] = ArrStegoData[1];
        blnArrOfBlueByte[LSB] = ArrStegoData[2];
    }
    else if (j % 3 == 1)
    {
        blnArrOfRedByte[LSB] = ArrStegoData[3];
        blnArrOfGreenByte[LSB] = ArrStegoData[4];
        blnArrOfBlueByte[LSB] = ArrStegoData[5];
    }
    else
    {
        blnArrOfRedByte[LSB] = ArrStegoData[6];
        blnArrOfGreenByte[LSB] = ArrStegoData[7];
        intFileDataCharIndex++;
    }
    Color newPixelColor =
Color.FromArgb((int)ConvertBooleanToByte(blnArrOfRedByte),

```

```

(int)ConvertBooleanToByte(blnArrOfGreenByte),
(int)ConvertBooleanToByte(blnArrOfBlueByte));
        StegoBitmapPicture.SetPixel(j, i, newPixelColor);

    }
}
endPosition:

//int initm = j;

long lngtempFileSize = lngfileSize, lngtempFileNameSize =
lngFileNameLength;

ArrOfRedByte = (byte)(lngtempFileSize % 100);
lngtempFileSize /= 100;
ArrOfGreenByte = (byte)(lngtempFileSize % 100);
lngtempFileSize /= 100;
ArrOfBlueByte = (byte)(lngtempFileSize % 100);

Color flenColor = Color.FromArgb(ArrOfRedByte, ArrOfGreenByte,
ArrOfBlueByte);
StegoBitmapPicture.SetPixel(width - 1, height - 1, flenColor);

ArrOfRedByte = (byte)(lngtempFileNameSize % 100);
lngtempFileNameSize /= 100;
ArrOfGreenByte = (byte)(lngtempFileNameSize % 100);
lngtempFileNameSize /= 100;
ArrOfBlueByte = (byte)(lngtempFileNameSize % 100);

Color fnlenColor = Color.FromArgb(ArrOfRedByte, ArrOfGreenByte,
ArrOfBlueByte);
StegoBitmapPicture.SetPixel(width - 2, height - 1, fnlenColor);

return StegoBitmapPicture;
}
#endregion

```

EXTRACT FILE CODING

```

#region Get the embeded file from the cover file by using the unhiding method
public static void DecryptLayer(int width, int height, Bitmap
StegoBitmapPicture, string strOutputFileName)
{
    int i, j = 0;

    bool[] ArrStegoData = new bool[8];

    bool[] blnArrOfRedByte = new bool[8];
    bool[] blnArrOfGreenByte = new bool[8];
    bool[] blnArrOfBlueByte = new bool[8];

    byte ArrOfRedByte, ArrOfGreenByte, ArrOfBlueByte;

```

```

        Color pixel = new Color();
        // Get the Stored file size that we saved in the encryption on the
pixel (width - 1, height - 1)
        pixel = StegoBitmapPicture.GetPixel(width - 1, height - 1);
        long fSize = pixel.R + pixel.G * 100 + pixel.B * 10000;

        // Get the Stored file name size that we saved in the encryption on the
pixel (width - 2, height - 1)
        pixel = StegoBitmapPicture.GetPixel(width - 2, height - 1);
        long fNameSize = pixel.R + pixel.G * 100 + pixel.B * 10000;

        byte[] res = new byte[fSize];

        string resFName = "";
        byte temp;

        int intFileNameCharIndex = 0;
        int intFileDataCharIndex = 0;

        bool blnCheckIfFinishedHidingTheFileName = false;
        bool blnCheckIfFinishedHidingTheFileData = false;

        //Read file name:
        for (i = 0; i < height; i++)
            for (j = 0; j < width; j++)
            {
                pixel = StegoBitmapPicture.GetPixel(j, i);

                if (intFileNameCharIndex == fNameSize)
                {
                    blnCheckIfFinishedHidingTheFileName = true;
                }

                if (blnCheckIfFinishedHidingTheFileName == false)
                {
                    ArrOfRedByte = pixel.R;
                    ArrOfGreenByte = pixel.G;
                    ArrOfBlueByte = pixel.B;

                    ConvertByteToBoolean(ArrOfRedByte, ref blnArrOfRedByte);
                    ConvertByteToBoolean(ArrOfGreenByte, ref
blnArrOfGreenByte);
                    ConvertByteToBoolean(ArrOfBlueByte, ref blnArrOfBlueByte);

                    if (j % 3 == 0)
                    {
                        ArrStegoData[0] = blnArrOfRedByte[7];
                        ArrStegoData[1] = blnArrOfGreenByte[7];
                        ArrStegoData[2] = blnArrOfBlueByte[7];
                    }
                    else if (j % 3 == 1)
                    {
                        ArrStegoData[3] = blnArrOfRedByte[7];
                        ArrStegoData[4] = blnArrOfGreenByte[7];
                        ArrStegoData[5] = blnArrOfBlueByte[7];
                    }
                }
            }

```

```

else
{
    ArrStegoData[6] = blnArrOfRedByte[7];
    ArrStegoData[7] = blnArrOfGreenByte[7];
    temp = ConvertBooleanToByte(ArrStegoData);
    resFName += (char)temp;
    intFileNameCharIndex++;
}
}
else
{
    if (intFileDataCharIndex == fSize) // finished hiding the
file information
    {
        blnCheckIfFinishedHidingTheFileName = true;
        goto endPosition;
    }

    if (blnCheckIfFinishedHidingTheFileData == false)
    {
        pixel = StegoBitmapPicture.GetPixel(j, i);

        ArrOfRedByte = pixel.R;
        ArrOfGreenByte = pixel.G;
        ArrOfBlueByte = pixel.B;

        ConvertByteToBoolean(ArrOfRedByte, ref
blnArrOfRedByte);
        ConvertByteToBoolean(ArrOfGreenByte, ref
blnArrOfGreenByte);
        ConvertByteToBoolean(ArrOfBlueByte, ref
blnArrOfBlueByte);

        if (j % 3 == 0)
        {
            ArrStegoData[0] = blnArrOfRedByte[7];
            ArrStegoData[1] = blnArrOfGreenByte[7];
            ArrStegoData[2] = blnArrOfBlueByte[7];
        }
        else if (j % 3 == 1)
        {
            ArrStegoData[3] = blnArrOfRedByte[7];
            ArrStegoData[4] = blnArrOfGreenByte[7];
            ArrStegoData[5] = blnArrOfBlueByte[7];
        }
        else
        {
            ArrStegoData[6] = blnArrOfRedByte[7];
            ArrStegoData[7] = blnArrOfGreenByte[7];
            temp = ConvertBooleanToByte(ArrStegoData);
            res[intFileDataCharIndex] = temp;
            intFileDataCharIndex++;
        }
    }
}
}

```

```
        endPosition:  
        File.WriteAllBytes(strOutputFileName + resFName, res);  
    }  
#endregion
```