A Project Report
on

Image Steganography

Submitted in partial fulfillment of the requirements for the Major Project II of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Anmol Dhall **R780209008**

Under the guidance
Mr. Abhineet Anand
Asst. Professor CIT

Department of Computer Science & Engineering
COLLEGE OF ENGINEERING STUDIES
UNIVERSITY OF PETROLEUM & ENERGY STUDIES
Dehradun- 248007
April 2013

CERTIFICATE

This is to certify that the Project entitled " **Image Steganography** " submitted by

Anmol Dhall **R780209008**

for the partial fulfillment of the requirements of the course **Major Project II** of **Bachelor of Technology in Computer Science & Engineering** degree of **University of Petroleum & Energy Studies, Dehradun** embodies the confide work done by above students under my supervision.

Mr. Abhineet Anand
Asst. Professor CIT

DECLARATION

I, Anmol Dhall bearing the R780209008 respectively hereby declare that this Project work entitled "Image Steganography" was carried out by me under the guidance and supervision of Mr. Abhineet Anand. This Project work is submitted to University of Petroleum & Energy Studies in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering during the Academic Semester Jan 2013 - Apr - 2013. I also declare that, i have not submitted this dissertation work to any other university for the award of either degree or diploma.

Anmol Dhall

Place: Dehradun

Date: 04/18/2013

ABSTRACT

Steganography is the art of hiding information or message in a file or in an image which helps in hiding the information from others. The only difference between the cryptography and steganography is that cryptography conceals the message or data or text not the existence of the message whereas in steganography one can identify the message existence. Carrier file hides the original message. In this project i will discuss how digital images can be used as a carrier to hide messages i.e. how one can hide text message inside an image. In my project I will also study and analyse the steganographic tool's on the basis of there performance.

I proposed an image Steganography tool that will hide a text message inside an image. The method will use the encryption and decryption techniques to hide or unhide the information.

Basically Steganography hides text in such a way that no human being is able to detect the hidden message in an image or in any file. The main use of steganography is hiding a data within the data. When we hide any data in a file what exactly we do is we encrypt that file with the password.

ACKNOWLEDGEMENT

It is a pleasure to thank all those great many people who helped, supported and encouraged us during this project work.

Firstly i express our sincere gratitude to Mr. Abhineet Anand, the guide of the project who carefully and patiently leant his valuable time and effort to give directions as well as to correct various documents with attention and care.

It is a great honor to do this project in this esteemed institution, and we would extend our thanks to the HOD, Prof Manish Prateek and other faculty members who have shared their vast knowledge and experience during our stay.

I do also like to appreciate the consideration of the Project Coordinator, our Faculties and colleagues, which enabled us to balance our work along with this project. It was their attitude that inspired us to do such an efficient and apposite work.

I am indebted to those people across the globe who have shared their knowledge and perspectives in the form of online tutorials, forums and other resources which helped me to a great extend whenever i met with technical obstacles during this endeavour.

I wish to avail this opportunity to express a sense of gratitude and love to all my friends and my family for their unwavering support, strength, help and in short for everything they have done during the crucial times of the progress of our project.

Last but not the least i thank GOD ALMIGHTY for HIS blessings and guidance without which this dream project wouldn't have been reality.

Anmol Dhall

TABLE OF CONTENTS

CHAPTER 1

1.0 INTRODUCTION-

Steganography comes from the Greek words Steganós (Covered) and Graphia (Writing). Steganography usually refers to information or a file that has been concealed inside a digital Image, Video or Audio file.

Steganography brings science to the art of hiding messages in such a way that only the recipient knows the existence of the message whether message resides in an image or in audio file. In cryptography no one can detect the hidden message from an image or from a file. The advantage of steganography over cryptography is that in steganography we can detect the text message which is hidden inside an image. Cryptography protects the contents of a message, whereas steganography protect both messages and communicating parties.

Basically, steganography is the art of hiding information inside information.
For example:

Normal text :

Since everyone can read, encoding text
in neutral sentences is doubtfully effective

After using Steganography:

Since Everyone Can Read, Encoding Text
In Neutral Sentences Is Doubtfully Effective

Hidden message - *Secret inside*

1.1 HISTORY OF STEGANOGRAPHY

It is believed that steganography was first practiced during the Golden Age in Greece. An ancient Greek record describes the practice of melting wax off wax tablets used for writing messages and then inscribing a message in the underlying wood. The wax was then reapplied to the wood, giving the appearance of a new, unused tablet. The resulting tablets could be innocently transported without anyone suspecting the presence of a message beneath the wax. An ancient Greek record describes the practice of melting wax off wax tablets used for writing messages and then inscribing a message in the underlying wood. The wax was then reapplied to the wood, 20 giving the appearance of a new, unused tablet. The resulting tablets could be innocently transported without anyone suspecting the presence of a message beneath the wax. Later on Germans developed microdot technology which FBI Director J. Edgar Hoover referred to as "the enemy's masterpiece of espionage. Microdots are photographs the size of a printed period having the clarity of standard-sized typewritten pages. The first microdots were discovered masquerading as a period on a typed envelope carried by a German agent in 1941. The message was not hidden, nor encrypted. It was just so small as to not draw attention to itself. Besides being so small, microdots permitted the transmission of large amounts of data including drawings and photographs. Another common form of invisible writing is through the use of Invisible inks. Such inks were used with much success as recently as WW-II. An innocent letter may contain a very different message written between the lines. Early in WW-II steganographic technology consisted almost exclusively of invisible inks. Common sources for invisible inks are milk, vinegar, fruit juices and urine. All of these darken when heated.

1.2 OVERVIEW OF THE EXISTING SYSTEM-

Present existing methods allow the Steganography to carry out the hidden messages. The main goal of Steganography is to communicate in a secure manner and in undetectable manner and to avoid drawing suspicion to the transmission of a hidden data. If someone came to know about the secret message being present in the carrier medium, then the method of Steganography become useless. Until recently, information hiding techniques received very much less attention from the research community and from industry than cryptography.

Steganography concepts

Although steganography is an ancient subject, the modern formulation of it is often given in terms of the prisoner's problem proposed by Simmons, where two inmates wish tocommunicate in secret to hatch an escape plan. All of their communication passes through a warden who will throw them in solitary confinement should she suspect any covert communication . The warden, who is free to examine all communication exchanged between the inmates, can either be passive or active. A passive warden simply examines the communication to try and determine if it potentially contains secret information. If she suspects a communication to contain hidden information, a passive warden takes note of the detected covert communication, reports this to some outside party and lets the message through without blocking it. An active warden, on the other hand, will try to alter the communication with the suspected hidden information deliberately, in order to remove the information .

Different kinds of steganography

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily. Image and audio files especially comply with this requirement, while research has also uncovered other file formats that can be used for information hiding.

The four main categories of file formats that can be used for steganography-:

1. Image.
2. Audio/Video.
3. Text.
4. Protocol.

1.3 OVERVIEW OF THE PROPOSED SYSTEM-

I am going to develop a Steganography technique or a tool which hides secret text message or information inside cover image.

Steganography is a useful technique that allows transmission of information over an over the communications channel. Hidden image will be obtained by combining both the secret image and the carrier image.. The hidden image is difficult to detect without retrieval.

CHAPTER 2

2.0 DOMAIN ANALYSIS-

In software engineering, domain analysis, or product line analysis, is the process of analyzing related software systems in a domain to find their common and variable parts. It is a model of wider business context for the system. It is the process by which a software engineer learns enough background information so that he or she can understand the problem and make good decisions during requirement analysis and other stages of software engineering process.

2.1 PROBLEM ANALYSIS-

Problem Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. During analysis, data collected on the various files, decision points and transactions handled by the present system. The success of the system depends largely on how clearly the problem is defined thoroughly investigated and properly carried out through the choice of solution. A good analysis model should provide not only the mechanisms of problem understanding but also the frame work of the solution. Thus it should be studied thoroughly by collecting data about the system. Then the proposed system should be analyzed thoroughly in accordance with the needs. Problem analysis can be categorized into three parts.

2.2 INFORMATION GATHERING-

It encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. It is an early stage in the more general activity of requirements engineering which encompasses all activities concerned with eliciting, analyzing, documenting, validating and managing software or system requirements

Problem Definition

Steganography In Images

For Steganography images are the most popular cover objects used in the present world. Many different image file formats exist in the domain of digital images, most of them for specific applications. For these different image file formats, different steganographic algorithms exist.

LSB (Least Significant Byte) method is usually the best method recommended to hide information or text messages inside images. An image file is simply a file for a computer which shows different colours and intensities of light on different positions of an image.
24 Bit BMP (Bitmap) image is the best type of image file to hide information inside. BMP is recommended because it is the largest type of file available and it normally is of the highest quality. It is a lot easier to hide and mask information inside an image which is of high quality and resolution.
With respect to the size, 24 Bit images are best for hiding information inside of. We may choose to use 8 Bit BMP's or possibly another image format such as GIF, because the posting of large images on the internet may arouse suspicion.
Note-: If you hide information inside of an image file and that file is converted to another image format, it is most likely the hidden information inside will be lost.

Image definition

To a computer, an image is a collection of numbers that constitute different light intensities in different areas of the image. This numeric representation forms a grid and the individual points are referred to as pixels. Most images on the Internet consists of a rectangular map of the image's pixels (represented as bits) where each pixel is located and its colour . These pixels are displayed horizontally row by row. The number of bits in a colour scheme, called the bit depth, refers to the number of bits used for each pixel. The smallest bit depth in current colour schemes is 8, meaning that there are 8 bits used to describe the colour of each pixel. Monochrome and greyscale images use 8 bits for each pixel and are able to display 256 different colours or shades of grey. Digital colour images are typically stored in 24-bit files and use the RGB colour model, also known as true colour. All colour variations for the pixels of a 24-bit image are derived from three primary colours: red, green and blue, and each primary colour is represented by 8 bits. Thus in one given pixel, there can be 256 different quantities of red, green and blue, adding up to more than 16-million combinations, resulting in more than 16-million colours. Not surprisingly the larger amount of colours that can be displayed, the larger the file size.

Image Compression

When working with larger images of greater bit depth, the images tend to become too large to transmit over a standard Internet connection. In order to display an image in a reasonable amount of time, techniques must be incorporated to reduce the image's file size. These techniques make use of mathematical formulas to analyse and condense image data, resulting in smaller file sizes. This process is called compression. In images there are two types of compression: lossy and lossless. Both methods save storage space, but the procedures that they implement differ. Lossy compression creates smaller files by discarding excess image data from the original image. It removes details that are too small for the human eye to differentiate, resulting in close approximations of the original image, although not an exact duplicate. An example of an image format that uses this compression technique is JPEG (Joint Photographic Experts Group).

Lossless compression, on the other hand, never removes any information from the original image, but instead represents data in mathematical formulas. The original image's integrity is maintained and the decompressed image output is bit-by-bit identical to the original image input. The most popular image formats that use lossless compression is GIF (Graphical Interchange Format) and 8-bit BMP (a Microsoft Windows bitmap file).

Compression plays a very important role in choosing which steganographic algorithm to use. Lossy compression techniques result in smaller image file sizes, but it increases the possibility that the embedded message may be partly lost due to the fact that excess image data will be removed. Lossless compression though, keeps the original digital image intact without the chance of lost, although is does not compress the image to such a small file size. Different steganographic algorithms have been developed for both of these compression types and will be explained in the following sections.

Elements of Steganography

Algorithmically, steganography has two processes, one for covering and one for uncovering secret data. The covering process is about hiding overt data into a cover medium, also known as stego or carrier file. In contrast, the uncovering process is just the reverse; it is about extracting the covert data from the carrier file and returning them back to their original state. Fundamentally, modern digital steganography is governed by five key elements. They are as follows [4]:

1. Covert Data: Often known as the payload and refers to the overt data that need to be covertly communicated or stored. The covert data can be anything convertible to binary format, from simple text messages to executable files.

2. Carrier Medium: It is basically a file into which the covert data are concealed. The carrier medium can be any computer-readable file such as image, audio, video, or text file

3. Stego File: Sometimes called package, it is the resulting file which has the covert data embedded into it.

4. Carrier Channel: It denotes the file type of the carrier, for instance, BMP, JPG, MP3, PDF, etc.

5. Capacity: It denotes the amount of data the carrier file can hide without being distorted.

CHAPTER 3

HARDWARE REQUIREMENTS-

Processor	:PentiumIII
Main memory:	
Minimum	: 64MB
Recommended	:128MB
Hard disk:	
Minimum	: 10.2GB
Recommended	: 40GB
Clock Speed	: 866 MHz
Virtual Memory	: 32Bits
Cache memory	: 512KB
Floppy disk drive	: 1.44MB

SOFTWARE REQUIREMENTS-

Programming Language used : Java

CHAPTER 4

4.0 Methodology

There have been many techniques for hiding information or messages in images in such a manner that alteration made to the image is perceptually indiscernible. Commonly approaches are include LSB, Masking and filtering and Transform techniques.

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

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

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

- **Masking and filtering techniques**

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

Masking and filtering techniques for digital image encoding such as Digital Watermarking (i.e.- integrating a companies logo on there web content) are more popular with lossy compression techniques such as (.JPEG). This technique actually extends an images data by masking the secret data over the original data as opposed to hiding information inside of the data. Some experts argue that this is definitely a form of Information Hiding, but not technically Steganography. The beauty of Masking and Filtering techniques are that they are immune to image manipulation which makes there possible uses very robust.

- **Image Generation Technique**

Many techniques have been proposed that encrypt messages so that they are unreadable or as secret as possible. Big Play Maker hides information by converting the secret text message into a larger and a slightly manipulated text format. The same principle can be employed in image creation, in which a message is converted to picture elements and then collected into a complete stego-image. This method cannot be broken by rotating or scaling the image, or by lossy compression. Parts of the message may be destroyed or lost because of cropping, but it is still possible to recover other parts of the message by encoding the message with error correcting information.

Generally, this technique uses pseudo-random images, because if a malicious third party detects a group of images passing through a network without any reason for them being there (i.e., random images), he or she may suspect that the images contain secret information and block their transmission.

- **Distortion Technique**

Distortion techniques require knowledge of the original cover image during the decoding process where the decoder functions to check for differences between the original cover image and the distorted cover image in order to restore the secret message. The encoder, on the other hand, adds a sequence of changes to the cover image . So, information is described as being stored by signal distortion .

Using this technique, a stego-object is created by applying a sequence of modifications to the cover image. This sequence of modifications is selected to match the secret message required to transmit.

- **Transform Domain Techniques**

Transform domain embedding can be defined as a domain of embedding techniques for which a number of algorithms have been suggested. The process of embedding data in the frequency domain of a signal is much stronger than embedding principles that operate in the time domain. It is worth saying that most of the strong steganographic systems today operate within the transform domain.

Transform domain techniques have an advantage over LSB techniques because they hide information in areas of the image that are less exposed to compression, cropping, and image processing. Some transform domain techniques do not seem dependent on the image format and they may outrun lossless and lossy format conversions. The JPEG file format is the most common image file format on the internet owing to the small size of resultant images obtained by using it.

- **Encrypt and Scatter Technique**

This technique attempts to emulate what is known by White Noise Storm which is a combination of spread spectrum and frequency hopping practices. Its principle is so simple; it scatters the message to hide over an image within a random number defined by a window size and several data channels. It uses eight channels each of which represents 1 bit; and consequently, each image window can hold 1 byte of data and a set of other useless bits. These channels can perform bit permutation using rotation and swapping operations such as rotating 1 bit to the left or swapping the bit in position 3 with the bit in position 6. The niche of this approach is that even if the bits are extracted, they will look garbage unless the permutation algorithm is first discovered. Additionally, the encrypt and scatter technique employs DES encryption to cipher the message before being scattered and hidden in the carrier file.

- **BPCS Technique**

This technique which stands for Bit-Plane Complexity Segmentation Steganography, is based on a special characteristic of the Human Visual System (HVS). Basically, the HVS cannot perceive a too complicated visual pattern as a coherent shape. For example, on a flat homogenous wooden pavement, all floor tiles look the same. They visually just appear as a paved wooden surface, without any indication of shape. However, if someone looks closely, every collection of tiles exhibits different shapes due to the particles that make up the wooden tile. Such types of images are called vessel images. BPCS Steganography makes use of this characteristic by substituting complex regions on the bit-planes of a particular vessel image with data patterns from the secret data.

All of the approaches to steganography have one thing in common that they hide the secret message in the physical object which is sent. The following figure shows the steganography process of the cover image being passed into the embedding function with the message to encode resulting in a steganographic image containing the hidden message. A key is often used to protect the hidden message. This key is usually a password,so this key is also used.

Screen Shots:

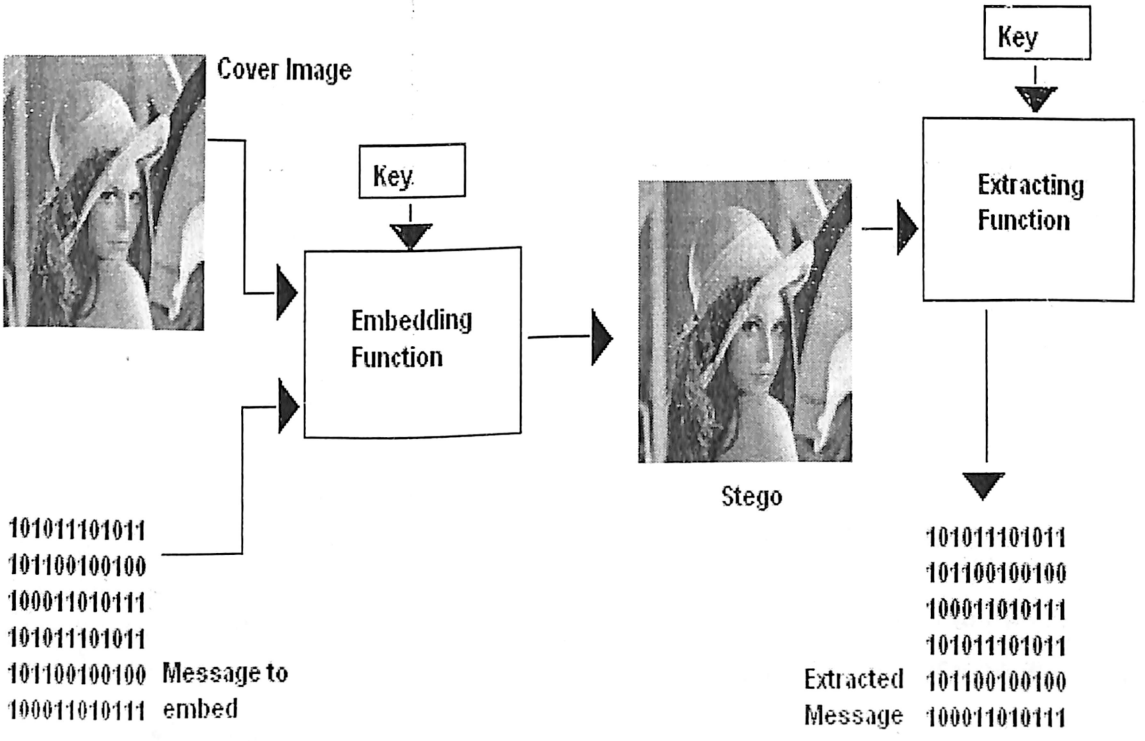


Fig.1.0.Steganography Procedure

The Steganography procedure consists of two main processes:

4.1 Encryption Technique

The process of hiding secret information in any type of image file is known as Encryption. 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.

4.2 Decryption Technique

Decryption is getting the secrete information from 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.

Algorithm

Encryption Algorithm

- Steps-:

1. Select any text message.
2. Choose any key of your choice.
3. Pad the normal message according to the key's length.
4. Take two arrays of size of length of message and key.
5. Let key's length be m and n be the length of padded text.
6. Adding both the arrays and storing result in one.
7. Encrypted Data will be obtained.

Hiding Key Algorithm

- Steps-:

1. Let the length of the key be m.
2. Let array b[] will hold the key.
3. Will add the character with the next character present in key.
4. Last character will be added to the first character obtained.
5. The Hidden Key will be obtained

Decryption of Key and Data Algorithm

- Steps-:

1. Reverse of Encryption.
2. Firstly, will decrypt the key.
3. Hidden Key will be obtained.
4. Will decrypt the message with the help of the key.
5. Hidden data or message will be obtained.

Example

a[]= Anmol // message

b[]= Hello //key

a[i]=a[i]+b[j]

a[]=A+H|n+e|m+l|o+l|l+o // encypted message with key.

m=length of key

Loop for 1to m-1

b[]=H+e|e+l|l+l|l+o|o+H+e //hidden key

Loop for m-1 to 1

(o+H+e) – (H+e)= o

(l+o) – (o)= l

(l+l) – (l)= l

(e+l) – (l)= e

(H+e) – (e)= H

Hidden Key obtained i.e. Hello

Loop 1-m

(A+H)-(H)=A

(n+e)-(e)=n

(m+l)-(l)=m

(o+l)-(l)=o

(l+o)-(o)=l

Hidden message obtained i.e. Anmol

After that we need to patch our information/data in Image using Least Significant Bit Technique.

Least Significant Bit Algorithm

- Steps-:

1. Take an Input Image.
2. Find out the pixel values.
3. Select the pixel on which you want to insert the data.
4. Insert the data values in pixels in binary form.
5. Embedded the binary form into the least significant bit of image.

- **Example**

24-bit image will be having 3 pixel grid as shown below-:

10011001 01101101 01001101

10011010 00111010 10100110

10010110 01001101 10001101

Binary Representation of 500 say is-

10010011

After embedding this binary form into the least significant bit of image.

The grid obtained will be like as shown below-

10011001 01101100 01001100

10011011 00111010 10100110

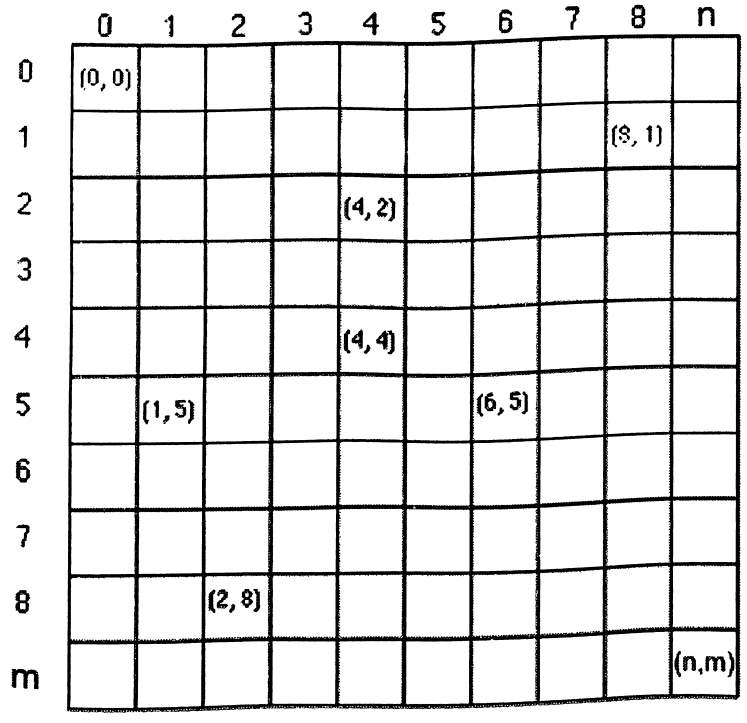
10010111 01001101 10001101

Proposed Solution

I have proposed a novel steganography scheme for hiding any form of digital data into uncompressed digital image files in a random manner. It uses two medium to convey the secret data. An uncompressed image file acting as a carrier image holding the secret data inside the LSBs of its pixels, and an English text made up of several well-structured and syntactically correct English sentences pointing to the location of the carrier pixels, that is, the location of the secret data in the carrier image. Algorithmically, the proposed scheme is based on a randomized algorithm to randomly select the carrier pixels into which the secret data are be concealed, and on a mini-version context-free grammar of the English language to generate correct English sentences that encode the location of the random carrier pixels in the carrier image. The carrier image is processed as a 2D plane with coordinates x and y (x, y) representing pixels' locations. The employed context-free grammar uses a lexicon of English words randomly categorized in 10 categories representing the 10 digits of the decimal numeral system (0...9). These digits are used to generate all possible coordinate values for the carrier pixels.

The Proposed Scheme in Details

The proposed scheme is designed to work on 24-bit True Color uncompressed digital images such as BMP. Basically, the pixels of a 24-bit BMP image are each composed of three 8-bit color components, namely R, G, and B components. The proposed scheme hides the secret data into the three LSBs of each of these color components; thus, the hiding capacity is equal to 9 bits out of 24 bits or 37% of the total size of the carrier image (9/24=0.375=37%). The carrier image is manipulated as a 2D plane geometric object composed of a finite set of points along with their coordinates. These coordinates effectively indicate the locations of the pixels in the carrier image itself. Figure 1 depicts a BMP image as a geometric 2D plane as regarded by the proposed scheme.



In effect, the proposed scheme does not hide the secret data sequentially into the carrier pixels; instead, it randomly selects pixels to carry in the secret data. Thus, shuffling and dispersing the secret data over the carrier image in a random manner. The actual positions of these randomly selected carrier pixels, which interchangeably are their coordinates, are mimicked by an English text composed of English sentences that are grammatically well structured. The English text is generated using a context-free grammar and a lexicon of words organized into 10 categories from category 0 till category 9. Words that constitute the

English sentences are selected from these categories based on the coordinates of the carrier pixels. For instance, coordinates (019,421) are encoded by selecting a word from category 0, a word from category 1, a word from category 9, a word from category 4, a word from category 2, and a word from category 1 respectively. Actually, every coordinate (i.e. pixel location) is represented by an English sentence. As a result, multiple pixels would result in multiple sentences which eventually result in a complete English text. The context-free grammar is essential in order to generate a grammatically correct text, for instance, generating a sentence composed of a determinant, followed by a noun, followed by verb, followed by a preposition. It is worth noting that no common words exit between the categories of the lexicon as it would be later impossible to recover the real coordinates of the carrier pixels out the English text. Finally, together, the carrier image and the generated English text are sent to the receiver who has to use the same algorithm, the same grammar, and the same lexicon to recover the different locations of the carrier pixels and consequently the secret data.

The Context-Free Grammar and the Lexicon

A context-free grammar or CFG is a mathematical system for modeling the structure of languages such as natural languages like English, French and Arabic, or computer programming languages like C++, Java, and C# [13]. Its formalism was originally set by Chomsky [14] and Backus [15], independently of each other. The one of Backus is known as the Backus-Naur Form or BNF for short [16]. In essence, a context-free grammar consists of a set of rules known as production rules that specify how symbols and words of a language can be arranged and grouped together. An example of a rule would specify that in the English language a verb phrase "VP" must always start with a verb then followed by a noun phrase "NP". Another rule would state that a noun phrase "NP" can start with a proper noun "ProperNoun" or a determinant "Det". Following is a sample CFG for a hypothetical language called L, that is a subset of the English language.

NP → Det Nominal

NP → ProperNoun

Nominal → Noun | Nominal Noun

Det → an

Det → the

Noun → apple

The symbols that are used in a CFG are divided into two classes: Terminals that correspond to unbreakable words in the language such as "apple", "the", and "an"; and non-terminals which are variables that can be replaced by terminals and other non-terminal variables to derive and produce sentences of the languages.

Terminals are usually provided by a lexicon of words; whereas, the non-terminals and the production rules are part of the parsing algorithm. Parsing is the process of deriving sentences for the language using the production rules of the CFG and the lexicon [17]. Parsing can also be used to confirm that the structure of a sentence complies with the CFG of the language.

The proposed steganography scheme uses a mini-version CFG of the English language and a lexicon of predefined words to generate correct English sentences that can encode the

coordinates of the carrier pixels. Figure 2 outlines the production rules of the CFG used by the proposed scheme.

$S \rightarrow NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

Fig. 2: CFG of the Proposed Scheme

On the other hand, the lexicon of the CFG which is outlined in Table 1 is organized into 10 categories each of which contains a set of terminal words that belong to the English language. It is worth mentioning that no common words exist between the categories of the lexicon as it would be later impossible to recover the real coordinates of the carrier pixels out of the English text.

Table 1: Lexicon of thetProposed CFG

Category 0	Category 1
Det □ this | that | …	Det □ those | an | …
Pronoun □ I | they | us | …	Pronoun □ he | she | me | …
Preposition □ from | across | about | …	Preposition □ on | above | through | …
Noun □ door | car | memory | …	Noun □ tree | school | board | …
Verb □ play | eat | walk | …	Verb □ study | dance | climb | …
Proper-Noun □ California | John |	Proper-Noun □ Texas | NBA | …

Intel | ...

Category 2 ...

Det □ a | these | ...

Pronoun □ you | it | their | ...

Preposition □ to | towards | along | ...

Noun □ girl | university | roof | ...

Verb □ sleep | like | enroll | ...

Proper-Noun □ Ohio | George | Mike | ...

Category 9

Det □ the | ...

Pronoun □ we | ours | his | ...

Preposition □ among | at | before | ...

Noun □ floor | book | pen | ...

Verb □ move | walk | write | ...

Proper-Noun □ Harvard | Tony | ...

The Proposed Algorithm

Below is the list of steps executed by the proposed algorithm to hide a piece of secret data into an image file.

1. The secret data are preprocessed so that they become suitable for storage inside the carrier image.

a. The secret data, no matter their types and formats – whether text, documents, or executables, are converted into a binary form resulting into a string of bits denoted by D={b0, b1, b2, b3,...bn-1} where bi is a single bit composing the secret data and n is the total number of bits.

b. The string of bits D is organized into chunks of 3 bits, such as D=Chunks={ C0[b0, b1, b2], C1[b3, b4, b5], C2[b6, b7, b8],...Cm-1[bn-3, bn-2, bn-1] }, where Cj is a particular 3-bit chunk, m is the total number of chunks, and n is the total number of bits making up the secret data

.

2. A set of carrier pixels is randomly selected from the carrier image in which every random selected pixel is denoted by Pt(xt, yt) where x and y are the coordinates of pixel P, and t is the index of P.

3. The chunks of the secret data that were created in step 1.b are embedded into the randomly selected pixel of step 2.

a. As the carrier image is a 24-bit colored image, every pixel would have three color components R, G, B, each of length 8 bits. For this reason, every chunk Cj is stored in the three LSBs of each of the color components of every selected carrier pixel such as Pt={ fiveMSBs(Rt) + Cj ; fiveMSBs(Gt) + Cj+1 ; fiveMSBs(Bt) + Cj+2 }, where P is a randomly selected pixel, t is the index of P, and R, G, and B are the three Red-Green-Blue color components of pixel Pt. Furthermore, fiveMSBs(component) is a function that returns the original five most significant bits of the color component "component". The "+" operator concatenates the original five MSBs of the color component with the 3 bits of a particular chunk, making the total number of bits in a given color component equals to 8 bits. In effect,

the first 5 bits are the original five MSBs of the color component and the three LSBs are a particular chunk from the secret data.

4. The coordinates of the randomly selected pixels of step 2 are encoded into English sentences each of which is denoted by Sk.

a. Every single digit of the coordinates x and y such as x={d0,d1,d2,dr-1} and y={d0,d1,d2,dr-1} is mapped into a category number of the lexicon. For instance, P(29, 01) is encoded using the CFG of Figure 2 and the lexicon of Table 1 as "a book from school". In that, "a" is a determinant from category 2, "book" is a noun from category 9, "from" is a preposition from category 0, and "school" is a noun from category 1.

b. Step 4.a is repeated for all selected pixels. Eventually, the number of all generated English sentences would be equal to the number of all randomly selected pixels.

5. The final output comprises two mediums. The first one is the carrier image which houses the secret data into its randomly selected pixels such as IMG={P0,P1,P2,Pt-1}, where P is a carrier pixel and t is the total number of carrier pixels; while, the second one is an English text made up of grammatically correct English sentences such as T={S0,S1,S2,St-1}, where S is an English sentence and t is the total number of English sentences which is also equal to the total number of carrier pixels. Both, the carrier image and the English text are to be sent to the receiver, not necessarily at the same time, however, they should be both present when the receiver is uncovering the secret data.

EXAMPLE

An example is illustrated in this section to demonstrate how the proposed steganography scheme works. It involves all the steps required to cover a secret text message into a 24-bit BMP carrier image file. The carrier image is sampled at 800x600 resolution, and the secret data to hide is a text message denoted by D="kill joe".

The secret message is preprocessed and converted into a binary form.

The secret message in ASCII format can be denoted as DASCII={ k=1101011 i=1101001 l=1101100 l=1101100 space=0100000 j=1101010 o=1101111 e=1100101 }

b. Chunks of 3 bits are generated out of D and are denoted by Chunks={ C0[110], C1 [101], C2[111], C3[010], C4[011], C5[101], C6[100], C7[110], C8[110], C9[001], C10[000], C11[001], C12[101], C13[010], C14[110], C15[111], C16[111], C17[001], C18[01]. The total number of chunks is 19; and therefore, 19 color components are required to store these chunks. Since every pixel has three color components, this requires 19/3 = 6.3 = 7 pixels to store the different chunks of the secret message D.

2. Seven pixels are randomly selected. Their coordinates are respectively: P0(206, 318), P1(407, 192), P2(321, 129), P3(709, 015), P4(501, 000), P5(712, 200), and P6(309, 108).

The color values of these seven pixels are as follows: P0(R=00101111 ; G=00111111 ; B=10101010) , P1(R=11101001 ; G=10110011 ; B=00111011) , P2(R=10100000 ; G=00001111 ; B=00101000) , P3(R=11101111 ; G=11111111 ; B=10000000) , P4(R=11101001 ; G=00110000 ; B=10101111) , P5(R=10101010 ; G=00000000 ; B=11111111) , P6(R=11110010 ; G=11111111 ; B=10000011).

The chunks obtained in step 1.b are stored into the three LSBs of the color components of every pixel of step 2. The results are as follows: P0(R=00101110 ; G=00111101 ;

B=10101111) ; P1(R=11101010 ; G=10110011 ; B=00111101) ; P2(R=10100100 ; G=00001110 ; B=00101110) ; P3(R=11101001 ; G=11111000 ; B=10000001) ; P4(R=11101101 ; G=00110010 ; B=10101110) ; P5(R=10101111 ; G=00000111 ; B=11111001) ; P6(R=11110010 ; G=11111111 ; B=10000011).

The red bits are chunks of the secret message that replaced the original three LSBs of the color components of the selected carrier pixels. The hiding capacity of the algorithm is equal to 9 bits out of 24 bits or 37% of the total size of the carrier image (9/24=0.375=37%). Figure 3 depicts the original carrier image before and after hiding the secret message D inside its pixels.

The coordinates of the randomly selected pixels are encoded into English sentences using the CFG and the lexicon of Figure 2 and Table 1. Since there are seven pixels, there should be seven English sentences. Table 2 outlines the generated English sentences each denoted by S along with the coordinates of the corresponding pixels.

Table 2: The Generated English Sentences

Category/ Word	Category/ Word	Category/ Word	Category/ Word	Category/ Word	Category/ Word
P0(206, 318)	2	0	6	3	1	8
S0	the	boy	went	to	high	school
P1(407, 192)	4	0	7	1	9	2
S1	eat	the	apple	in	the	kitchen
P2(321, 129)	3	2	1	1	2	9

S2	John	write	circle	on	the	wall
P3(709, 015)	7	0	9	0	1	5
S3	dance	in	the	night	with	mike
P4(501, 000)	5	0	1	0	0	0
S4	a	compute r	shipped	to	Atlanta	city
P5(712, 200)	7	1	2	2	0	0
S5	George	does	homewo rk	on	the	desk
P6(309, 108)	3	0	9	1	0	8
S6	these	guys	dive	in	the	ocean

The final output is the carrier BMP image now carrying the secret data, and the English text denoted by T={ "the boy went to high school", "eat the apple in the kitchen", "john write circle on the wall", "dance in the night with mike", "a computer shipped to Atlanta city", "George does homework on the desk", "these guys dive in the ocean" }. It is worth noting that the algorithm does not guarantee that the generated English sentences are all semantically correct as it does not implement a semantic analyzer. It is the job of the user to compile the lexicon with the appropriate words that can semantically work with each other.

As for the uncovering process, it is the reverse of the above process. First, the second medium which is the English text T is decomposed into sentences such as T={S0,S1,S2,St-1}. Then, each sentence is decomposed into words. Then, every word is mapped to a digit pertaining to the category in the lexicon it belongs to. At this point, coordinates of the carrier pixels are generated and are used to extract the secret data from the three LSBs of the pixels

they point to. As a result, a string of bit is formulated which is then converted into ASCII text, revealing the original secret message D, mainly "kill joe".

Advantages of the Proposed Scheme

The proposed steganography scheme has several advantages over other existing schemes. They are listed below.

The first advantage is that the proposed scheme is hard-to-notice, a fact that is promoted by using only three LSBs in every color component, in addition to a well-structured English text to convey secret data. It is by using only three LSBs to hide data, ensures that no visual artifacts are to be produced in the carrier image. Likewise, a grammatically correct text lowers suspicions and totally obscures the fact that a secret communication is taking place.

The second advantage is that the proposed scheme is hard-to-detect, a fact that is promoted by selecting the carrier pixels in a random manner. Using random pixels shuffles the secret data and scatters them all over the carrier image. That way, no one apart from the holder of the English text, the lexicon, and the decoding algorithm can know how to extract the secret data out the carrier image.

The third advantage is that the proposed scheme is hard-to-recover, a fact that is promoted by using two mediums to deliver the secret data, mainly the carrier image and the English text. As a result, the scheme is less susceptible to stego-analysis attacks as third parties often assume that the secret data are hidden in one medium and not between two mediums that complement each other. Using the proposed scheme, the sender can first send one of the mediums, and then later on, send the other one.

The fourth advantage is the capacity of data that can be hidden. As the proposed scheme can hide secret data into the three LSBs of each of the color components of every pixel, the hiding capacity is then equal to 9 bits out of 24 bits or 37% of the total size of the carrier image (9/24=0.375=37%). As a result, a 1MB BMP carrier image can carry 1*37%=0.37MB=378KB, which is around one third of the total image size. All that while retaining the visual quality of the carrier image.

The fifth advantage is that the proposed scheme is binary-based which enables it to operate on text messages as well as images, audio files, video files, applications, word documents,

HTML and XML documents, spreadsheets, PDF documents, and any sort of data that can be converted into a stream of bits. This makes the scheme so generic and suitable for a wide range of applications.

The sixth advantage is that the proposed scheme is multilingual, in that, it can be tailored to support languages other than the English language by just providing the appropriate CFG and lexicon for that language. In fact, almost any natural language in the world has a grammar and a set of words proper to build a lexicon. As a result, the scheme can be used by many users regardless of their native language.

The seventh advantage is that the proposed scheme is mutable, in such a way that it is not bound to a specific lexicon and new lexicons can be compiled and loaded into the algorithm. As a result, new lexicons can derive new sentences for the same coordinate values, fooling off stego-analysts and impeding them from achieving their attacks.

CODING

Image_Filter Class

```
import java.io.*;

/*
 *Image_Filter Class
 */
public class Image_Filter extends javax.swing.filechooser.FileFilter
/*
*Determines if the extension is of the defined types

*/
protected boolean isImageFile(String ext)
{
                return (ext.equals("jpg")||ext.equals("png"));
}

/*
*Determines if the file is a directory or accepted extension

*/
public boolean accept(File f)
{
if (f.isDirectory())
{
return true;
```

```
}

String extension = getExtension(f);
if (extension.equals("jpg")||extension.equals("png"))
{
return true;
}
return false;
}

/*
*Supplies File type description

*/
public String getDescription()
{
return "Supported Image Files";
}

/*
*Determines the Extension

*/
protected static String getExtension(File f)
{
String s = f.getName();
int i = s.lastIndexOf('.');
if (i > 0 &&  i < s.length() - 1)
```

```
return s.substring(i+1).toLowerCase();
return "";
}}
```

Steganography Class

```
import java.awt.Point;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;
import java.awt.image.DataBufferByte;

import javax.imageio.ImageIO;
import javax.swing.JOptionPane;

/*
 *Class Steganography
 */
public class Steganography
{

  /*
   *Steganography Empty Constructor
   */
  public Steganography()
  {
  }

  /*
   *Encrypt an image with text, the output file will be of type .png
```

```
    */
    public boolean encode(String path, String original, String ext1, String stegan,
String message)
    {
        String          file_name    = image_path(path,original,ext1);
        BufferedImage   image_orig   = getImage(file_name);

        //user space is not necessary for Encrypting
        BufferedImage image = user_space(image_orig);
        image = add_text(image,message);

        return(setImage(image,new File(image_path(path,stegan,"png")),"png"));
    }

    /*
    *Decrypt assumes the image being used is of type .png, extracts the hidden
text from an image

    */
    public String decode(String path, String name)
    {
        byte[] decode;
        try
        {
            //user space is necessary for decrypting
BufferedImage image  = user_space(getImage(image_path(path,name,"png")));
            decode = decode_text(get_byte_data(image));
```

```
        return(new String(decode));
      }
      catch(Exception e)
      {
        JOptionPane.showMessageDialog(null,
          "There is no hidden message in this image!","Error",
          JOptionPane.ERROR_MESSAGE);
        return "";
      }
    }

    /*
     *Returns the complete path of a file, in the form: path\name.ext

     */
    private String image_path(String path, String name, String ext)
    {
      return path + "/" + name + "." + ext;
    }

    /*
     *Get method to return an image file

     */
    private BufferedImage getImage(String f)
    {
      BufferedImage  image  = null;
      File       file   = new File(f);
```

```
        try
        {
           image = ImageIO.read(file);
        }
        catch(Exception ex)
        {
           JOptionPane.showMessageDialog(null,
"Image could not be read!","Error",JOptionPane.ERROR_MESSAGE);
        }
        return image;
    }

    /*
     *Set method to save an image file

     */
    private boolean setImage(BufferedImage image, File file, String ext)
    {
       try
       {
          file.delete(); //delete resources used by the File
          ImageIO.write(image,ext,file);
          return true;
       }
       catch(Exception e)
       {
          JOptionPane.showMessageDialog(null,
```

```
"File could not be saved!","Error",JOptionPane.ERROR_MESSAGE);
            return false;
        }
    }

    /*
     *Handles the addition of text into an image

     */
    private BufferedImage add_text(BufferedImage image, String text)
    {
        //convert all items to byte arrays: image, message, message length
        byte img[]  = get_byte_data(image);
        byte msg[] = text.getBytes();
        byte len[]   = bit_conversion(msg.length);
        try
        {
            encode_text(img, len,  0); //0 first positiong
            encode_text(img, msg, 32); //4 bytes of space for length: 4bytes*8bit =
32 bits
        }
        catch(Exception e)
        {
JOptionPane.showMessageDialog(null,"Target File cannot hold
message!","Error",JOptionPane.ERROR_MESSAGE);
        }
        return image;
    }
```

```
/*
 *Creates a user space version of a Buffered Image, for editing and saving
bytes

 */
private BufferedImage user_space(BufferedImage image)
{
  //create new_img with the attributes of image
  BufferedImage new_img = new BufferedImage(image.getWidth(),
image.getHeight(), BufferedImage.TYPE_3BYTE_BGR);
  Graphics2D graphics = new_img.createGraphics();
  graphics.drawRenderedImage(image, null);
  graphics.dispose(); //release all allocated memory for this image
  return new_img;
}

/*
 *Gets the byte array of an image

 */
private byte[] get_byte_data(BufferedImage image)
{
  WritableRaster raster = image.getRaster();
  DataBufferByte buffer = (DataBufferByte)raster.getDataBuffer();
  return buffer.getData();
}
```

```
/*
 *Gernerates proper byte format of an integer

 */
private byte[] bit_conversion(int i)
{
   //originally integers (ints) cast into bytes
   //byte byte7 = (byte)((i & 0xFF00000000000000L) >>> 56);
   //byte byte6 = (byte)((i & 0x00FF000000000000L) >>> 48);
   //byte byte5 = (byte)((i & 0x0000FF0000000000L) >>> 40);
   //byte byte4 = (byte)((i & 0x000000FF00000000L) >>> 32);

   //only using 4 bytes
   byte byte3 = (byte)((i & 0xFF000000) >>> 24); //0
   byte byte2 = (byte)((i & 0x00FF0000) >>> 16); //0
   byte byte1 = (byte)((i & 0x0000FF00) >>> 8 ); //0
   byte byte0 = (byte)((i & 0x000000FF)       );
   //{0,0,0,byte0} is equivalent, since all shifts >=8 will be 0
   return(new byte[]{byte3,byte2,byte1,byte0});
}

/*
 *Encode an array of bytes into another array of bytes at a supplied offset

 */
private byte[] encode_text(byte[] image, byte[] addition, int offset)
{
   //check that the data + offset will fit in the image
```

```
        if(addition.length + offset > image.length)
        {
            throw new IllegalArgumentException("File not long enough!");
        }
        //loop through each addition byte
        for(int i=0; i<addition.length; ++i)
        {
            //loop through the 8 bits of each byte
            int add = addition[i];
            for(int bit=7; bit>=0; --bit, ++offset) //ensure the new offset value
carries on through
both loops
            {
                //assign an integer to b, shifted by bit spaces AND 1
                //a single bit of the current byte
                int b = (add >>> bit) & 1;
                //assign the bit by taking: [(previous byte value) AND 0xfe] OR bit to
add
                //changes the last bit of the byte in the image to be the bit of addition
                image[offset] = (byte)((image[offset] & 0xFE) | b );
            }
        }
        return image;
    }

    /*
     *Retrieves hidden text from an image
```

```
    */
    private byte[] decode_text(byte[] image)
    {
        int length = 0;
        int offset  = 32;
        //loop through 32 bytes of data to determine text length
        for(int i=0; i<32; ++i) //i=24 will also work, as only the 4th byte contains
real data
        {
            length = (length << 1) | (image[i] & 1);
        }

        byte[] result = new byte[length];

        //loop through each byte of text
        for(int b=0; b<result.length; ++b )
        {
            //loop through each bit within a byte of text
            for(int i=0; i<8; ++i, ++offset)
            {
                //assign bit: [(new byte value) << 1] OR [(text byte) AND 1]
                result[b] = (byte)((result[b] << 1) | (image[offset] & 1));
            }
        }
        return result;
    }
}
```

```
import java.io.File;

import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.image.BufferedImage;
import java.awt.event.ActionListener;

import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.ImageIcon;
import javax.swing.JMenuItem;
import javax.swing.JTextArea;
import javax.imageio.ImageIO;
import javax.swing.JOptionPane;
import javax.swing.JFileChooser;

/*
 *Steganography_Controller Class
 */
public class Steganography_Controller
{
//Program Variables
private Steganography_View  view;
private Steganography model;

//Panel Displays
```

```
private JPane  decode_panel;
private JPanel encode_panel;
//Panel Variables
private JTextArea  input;
private JButton encodeButton,decodeButton;
private JLabel image_input;
//Menu Variables
private JMenuItem encode;
private JMenuItem decode;
private JMenuItem exit;

//action event classes
private Encode enc;
private Decode dec;
private EncodeButton encButton;
private DecodeButton decButton;

//decode variable
private String stat_path = "";
private String stat_name = "";

/*
 *Constructor to initialize view, model and environment variables

 */
public Steganography_Controller(Steganography_View aView, Steganography
aModel)
{
```

```
//program variables
view  = aView;
model = aModel;

//assign View Variables
//2 views
encode_panel= view.getTextPanel();
decode_panel= view.getImagePanel();
//2 data options
input= view.getText();
image_input= view.getImageInput();
//2 buttons
encodeButton= view.getEButton();
decodeButton= view.getDButton();
//menu
encode= view.getEncode();
decode= view.getDecode();
exit= view.getExit();

//assign action events
enc = new Encode();
encode.addActionListener(enc);
dec = new Decode();
decode.addActionListener(dec);
exit.addActionListener(new Exit());
encButton = new EncodeButton();
encodeButton.addActionListener(encButton);
decButton = new DecodeButton();
```

```
decodeButton.addActionListener(decButton);

//encode view as default
encode_view();
}

/*
 *Updates the single panel to display the Encode View.
 */
private void encode_view()
{
update();
view.setContentPane(encode_panel);
view.setVisible(true);
}

/*
 *Updates the single panel to display the Decode View.
 */
private void decode_view()
{
update();
view.setContentPane(decode_panel);
view.setVisible(true);
}
/*
 *Encode Class - handles the Encode menu item
 */
```

```
private class Encode implements ActionListener
{
/*
 *handles the click event
 */
public void actionPerformed(ActionEvent e)
{
encode_view(); //show the encode view
}
}

/*
 *Decode Class - handles the Decode menu item
 */
private class Decode implements ActionListener
{
/*
 *handles the click event

 */
public void actionPerformed(ActionEvent e)
{
decode_view(); //show the decode view

//start path of displayed File Chooser
JFileChooser chooser = new JFileChooser("./");
chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
chooser.setFileFilter(new Image_Filter());
```

```
int returnVal = chooser.showOpenDialog(view);
if (returnVal == JFileChooser.APPROVE_OPTION){
File directory = chooser.getSelectedFile();
try{
String image = directory.getPath();
stat_name = directory.getName();
stat_path = directory.getPath();
stat_path = stat_path.substring(0,stat_path.length()-stat_name.length()-1);
stat_name = stat_name.substring(0, stat_name.length()-4);
image_input.setIcon(new ImageIcon(ImageIO.read(new File(image))));
}
catch(Exception except) {
//msg if opening fails
JOptionPane.showMessageDialog(view, "The File cannot be opened!",
"Error!", JOptionPane.INFORMATION_MESSAGE);
}
}
}
}

/*
*Exit Class - handles the Exit menu item
 */
private class Exit implements ActionListener
{
/*
 *handles the click event
 *@param e The ActionEvent Object */
```

```
public void actionPerformed(ActionEvent e)
{
System.exit(0); //exit the program
}
}

/*
 *Encode Button Class - handles the Encode Button item
 */
private class EncodeButton implements ActionListener
{
/*
 *handles the click event

 */
public void actionPerformed(ActionEvent e)
{
//start path of displayed File Chooser
JFileChooser chooser = new JFileChooser("./");
chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
chooser.setFileFilter(new Image_Filter());
int returnVal = chooser.showOpenDialog(view);
if (returnVal == JFileChooser.APPROVE_OPTION){
File directory = chooser.getSelectedFile();
try{
String text = input.getText();
String ext  = Image_Filter.getExtension(directory);
String name = directory.getName();
```

```
String path = directory.getPath();
path = path.substring(0,path.length()-name.length()-1);
name = name.substring(0, name.length()-4);

String stegan = JOptionPane.showInputDialog(view,
"Enter output file name:", "File name",
JOptionPane.PLAIN_MESSAGE);

if(model.encode(path,name,ext,stegan,text))
{
JOptionPane.showMessageDialog(view, "The Image was encoded
Successfully!",
"Success!", JOptionPane.INFORMATION_MESSAGE);
}
else
{
JOptionPane.showMessageDialog(view, "The Image could not be encoded!",
"Error!", JOptionPane.INFORMATION_MESSAGE);
}
//display the new image
decode_view();
image_input.setIcon(new ImageIcon(ImageIO.read(new File(path + "/" + stegan
+ ".png"))));
}
catch(Exception except) {
//msg if opening fails
JOptionPane.showMessageDialog(view, "The File cannot be opened!",
"Error!", JOptionPane.INFORMATION_MESSAGE);
```

```
}
}
}
}

/*
 *Decode Button Class - handles the Decode Button item
 */
private class DecodeButton implements ActionListener
{
/*
 *handles the click event
 *@param e The ActionEvent Object
 */
public void actionPerformed(ActionEvent e)
{
String message = model.decode(stat_path, stat_name);
System.out.println(stat_path + ", " + stat_name);
if(message != "")
{
encode_view();
JOptionPane.showMessageDialog(view, "The Image was decoded
Successfully!",
"Success!", JOptionPane.INFORMATION_MESSAGE);
input.setText(message);
}
else
{
```

```
JOptionPane.showMessageDialog(view, "The Image could not be decoded!",
"Error!", JOptionPane.INFORMATION_MESSAGE);
}
}
}

/*
 *Updates the variables to an initial state
 */
public void update()
{
input.setText(""); //clear textarea
image_input.setIcon(null); //clear image
stat_path = ""; //clear path
stat_name = ""; //clear name
}

/*
*Main Method for testing
 */
public static void main(String args[])
{
new Steganography_Controller(
new Steganography_View("Steganography"),
new Steganography()
);
}
}
```

```
import java.awt.Color;
import java.awt.Insets;
import java.awt.Container;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;

import javax.swing.JMenu;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JTextArea;
import javax.swing.JScrollBar;
import javax.swing.JScrollPane;
import javax.swing.BorderFactory;

/*
 *Class Steganography_View
 */
public class Steganography_View extends JFrame
{
//side variables for window
private static int WIDTH  = 500;
private static int HEIGHT = 400;
```

```
//elements for JPanel
private JTextArea input;
private JScrollBar scroll,scroll2;
private JButton encodeButton,decodeButton;
private JLabel image_input;

//elements for Menu
private JMenu file;
private JMenuItem encode;
private JMenuItem decode;
private JMenuItem exit;

/*
 *Constructor for Steganography_View class
 *@param name Used to set the title on the JFrame
 */
public Steganography_View(String name)
{
//set the title of the JFrame
super(name);

//Menubar
JMenuBar menu = new JMenuBar();
JMenu file = new JMenu("File");file.setMnemonic('F');
encode = new JMenuItem("Encode");
encode.setMnemonic('E'); file.add(encode);
decode = new JMenuItem("Decode");
decode.setMnemonic('D'); file.add(decode);file.addSeparator();
```

```
exit = new JMenuItem("Exit"); exit.setMnemonic('x'); file.add(exit);

menu.add(file);
setJMenuBar(menu);

// display rules
setResizable(true);//allow window to be resized: true?false
setBackground(Color.lightGray);//background color of window:
Color(int,int,int) or Color.name
setLocation(100,100); //location on the screen to display window
setDefaultCloseOperation(EXIT_ON_CLOSE); //what to do on close operation:
exit, do_nothing, etc
 setSize(WIDTH,HEIGHT);//set the size of the window
 setVisible(true);//show the window: true?false
}

/*
 *@return The menu item 'Encode'
 */
public JMenuItem getEncode()
{
 return encode;
}
/*
 *@return The menu item 'Decode'
 */
public JMenuItem getDecode()
{
```

```
 return decode;
}
/*
 *@return The menu item 'Exit'
 */
public JMenuItem getExit()
{
 return exit;
}
/*
 *@return The TextArea containing the text to encode
 */
public JTextArea getText()
{
 return input;
}
/*
 *@return The JLabel containing the image to decode text from
 */
public JLabel getImageInput()
{
 return image_input;
}
/*
 *@return The JPanel displaying the Encode View
 */
public JPanel getTextPanel()
{
```

```
 return new Text_Panel();
}
/*
 *@return The JPanel displaying the Decode View
 */
public JPanel getImagePanel()
{
 return new Image_Panel();
}
/*
 *@return The Encode button
 */
public JButton getEButton()
{
 return encodeButton;
}
/*
 *@return The Decode button
 */
public JButton getDButton()
{
 return decodeButton;
}

/*
 *Class Text_Panel
 */
private class Text_Panel extends JPanel
```

```
{
/*
 *Constructor to enter text to be encoded
 */
public Text_Panel()
{
//setup GridBagLayout
GridBagLayout layout = new GridBagLayout();
GridBagConstraints layoutConstraints = new GridBagConstraints();
setLayout(layout);

input = new JTextArea();
layoutConstraints.gridx = 0;
layoutConstraints.gridy = 0;
layoutConstraints.gridwidth = 1;
layoutConstraints.gridheight = 1;
layoutConstraints.fill = GridBagConstraints.BOTH;
layoutConstraints.insets = new Insets(0,0,0,0);
layoutConstraints.anchor = GridBagConstraints.CENTER;
layoutConstraints.weightx = 1.0;
layoutConstraints.weighty = 50.0;
JScrollPanescroll=new
JScrollPane(input,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
layout.setConstraints(scroll,layoutConstraints);
scroll.setBorder(BorderFactory.createLineBorder(Color.BLACK,1));
add(scroll);
```

```
encodeButton = new JButton("Encode Now");
layoutConstraints.gridx = 0;
layoutConstraints.gridy = 1;
layoutConstraints.gridwidth = 1;
layoutConstraints.gridheight = 1;
layoutConstraints.fill = GridBagConstraints.BOTH;
layoutConstraints.insets = new Insets(0,-5,-5,-5);
layoutConstraints.anchor = GridBagConstraints.CENTER;
layoutConstraints.weightx = 1.0;
layoutConstraints.weighty = 1.0;
layout.setConstraints(encodeButton,layoutConstraints);
add(encodeButton);
//set basic display
setBackground(Color.lightGray);
setBorder(BorderFactory.createLineBorder(Color.BLACK,1));
}
}

/*
 *Class Image_Panel
 */
private class Image_Panel extends JPanel
{
/*
 *Constructor for displaying an image to be decoded
 */
public Image_Panel()
{
```

```
//setup GridBagLayout
GridBagLayout layout = new GridBagLayout();
GridBagConstraints layoutConstraints = new GridBagConstraints();
setLayout(layout);

image_input = new JLabel();
layoutConstraints.gridx = 0;
layoutConstraints.gridy = 0;
layoutConstraints.gridwidth = 1;
layoutConstraints.gridheight = 1;
layoutConstraints.fill = GridBagConstraints.BOTH;
layoutConstraints.insets = new Insets(0,0,0,0);
layoutConstraints.anchor = GridBagConstraints.CENTER;
layoutConstraints.weightx = 1.0;
layoutConstraints.weighty = 50.0;
JScrollPanescroll2=new
JScrollPane(image_input,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDE
D,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
layout.setConstraints(scroll2,layoutConstraints);
scroll2.setBorder(BorderFactory.createLineBorder(Color.BLACK,1));
image_input.setHorizontalAlignment(JLabel.CENTER);
add(scroll2);
decodeButton = new JButton("Decode Now");
layoutConstraints.gridx = 0;
layoutConstraints.gridy = 1;
layoutConstraints.gridwidth = 1;
layoutConstraints.gridheight = 1;
```

```
layoutConstraints.fill = GridBagConstraints.BOTH;
layoutConstraints.insets = new Insets(0,-5,-5,-5);
layoutConstraints.anchor = GridBagConstraints.CENTER;
layoutConstraints.weightx = 1.0;
layoutConstraints.weighty = 1.0;
layout.setConstraints(decodeButton,layoutConstraints);
add(decodeButton);

//set basic display
setBackground(Color.lightGray);
setBorder(BorderFactory.createLineBorder(Color.BLACK,1));
}
}

/*
 *Main Method for testing
 */
public static void main(String args[])
{
new Steganography_View("Steganography");
}
}
```

Resulting figures of this whole procedure are shown below.

After Encryption

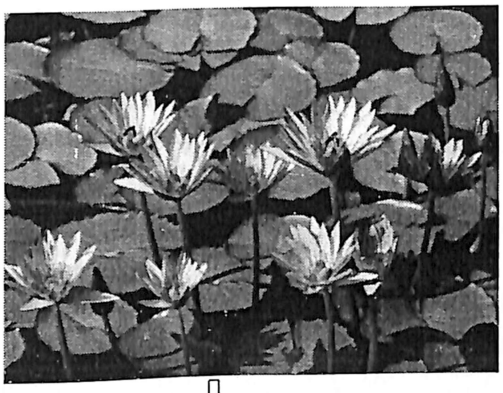
IMAGE FILE

INFORMATION FILE

BMP FILE

After Decryption

BMP FILE



INFORMATION FILE

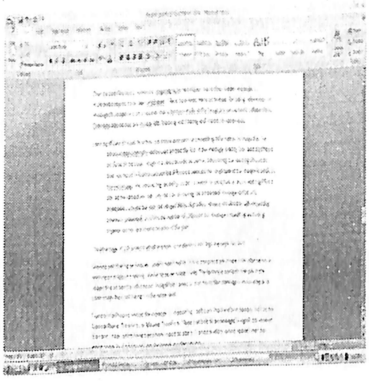
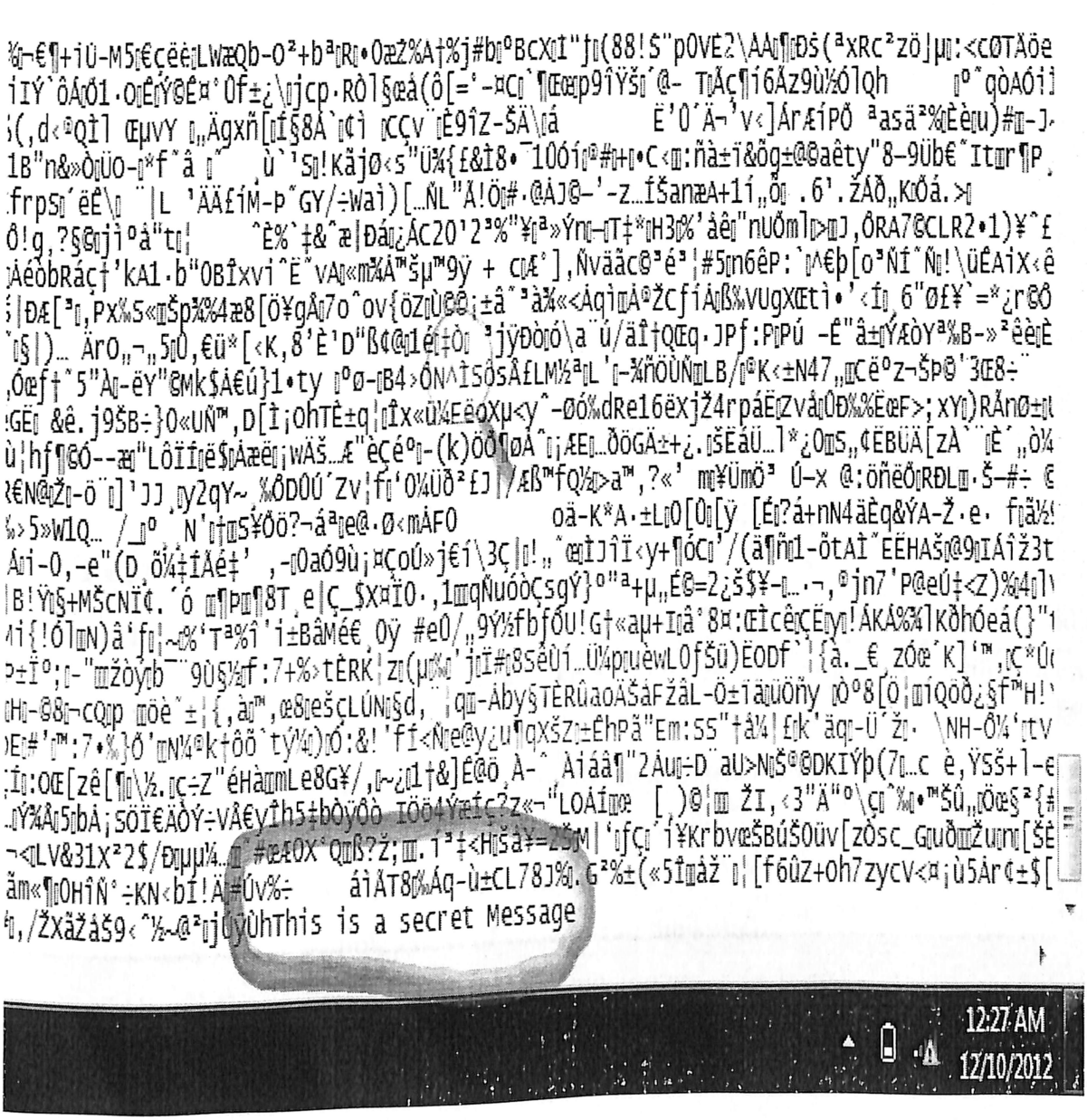


IMAGE FILE

Desired Output:



CHAPTER 5

Testing

5.1 Introduction

Testing is a process of analyzing a system or system components to detect the differences between specified and observed behavior. In other words, testing is a fault detection technique that tries to create failures or erroneous states in a planned way. This allows the developer to detect failures in the system before it is released to the customer.

5.2 Test plan

A Test plan is a plan prepared by the testers to follow a systematic procedure adapted to fulfill the process of testing. This includes the order of testing scenarios implemented.

As our system is concerned, we follow the below procedures:

1. Point out what to be tested and where to be tested:

a) Check the process of Opening a file for data source.

b) Check for the completeness of code to run the system.

c) Decide the portions of code to be tested and not to be tested.

d) Implemented the test scenarios and check the outputs with the specified ones.

2. Fail Criteria:

Here we specify the conditions where the system may fail and add them to test conditions that are to be tested.

3. Test Cases:

The test case Specification deals with the details of the test data, which is used to test the system.

5.3 Testing Activities

The activities of testing include:

Component Inspection: This finds faults in an individual component through the manual inspection of its source code. Inspections can be conducted before or after the unit test.

Usability Testing: This finds differences between the system and the users' expectation of what it should do? Usability testing tests the users' understanding of the system.

Unit Testing: This finds faults by isolating an individual component using test stubs and drivers and by exercising the component using a test case. Unit testing focuses on the building blocks of the software system, that is, objects and subsystems. The most import ant unit testing techniques are:

Equivalence Testing or Black box testing: This black box testing minimizes the number of test cases. The possible inputs are partitioned into equivalence classes, and a test case is selected for each class. Equivalence testing involves two steps:

Identification of the equivalence classes and

Selection of the test inputs.

Path based testing or White box testing: This white box testing technique identifies faults in the implementation of the component. The assumption here is that, by exercising all possible paths through the code at least once, most faults will trigger failures. The identification of paths requires knowledge of the source code and data structures.

State based Testing

Boundary Testing

Integration Testing: This finds faults by integrating several components together. Integration Testing detects faults that have not been detected during unit testing by focusing on small groups of components. Two or more components are integrated and tested and when no new faults are revealed, additional components are added to the group.

System Testing: This focuses on the complete system, its functional and nonfunctional requirements, and its target environment.

All the testing activities stated are to be implemented on large scale projects to get the consistent System to be designed. All of them are not applicable to small projects that do not much analysis done.

User Acceptance Testing: User Acceptance test of a system was the factor for the success of the system. The system under consideration was listed for user acceptance by keeping constant touch with the perspective user of the system at the time of design, development and making changes whenever required. This was done as follows.

Input screen design
- Output design
- Menu drive
- Formats for reports

Program Testing: The software units in a system are modules and routines that are assembled and integrated to perform to specific functions.

Program testing checks for two types: Syntax and logic

o Syntax error is the program statement that violates one or more rules of the language in which it is written.

o Logic error deals with incorrect data fields, out-of-range items, and invalid combinations.

Thus thetest focuses on the individual modules

String Testing: The objective is to ensure that data are correctly transferred from one program in the string to the next. Each module is tested against the entire module with both test and live data before the entire system is ready to be tested.

5.4 Test Cases

A test case is a set of input data and expected results that exercises a component with the purpose of causing failures and detecting faults. A test case has five attributes:

- Name: Name of the test case.
- Location: Full path name of executable.
- Input: Input data or commands.
- Log: Output produced by the test.

The various Test Cases to be handled in the current system are as follows. The main test cases are, test for file existence and test for file format i.e. txt files only.

CHAPTER 6

SYSTEM DESIGN

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

RESULTS AND DISCUSSIONS

Steganographic techniques have obvious uses, some legitimate, some less so, and some are likely illegal.

Advantages of Steganography are:

- It provides a better security for the sharing of data in local area network.
- Important files carrying confidential information can be stored in the server in an encrypted form.
- Using public key or private key can encrypt files.
- No intruder can get any useful information from the original file during transmit.
- It provides a better-secured data storage and transmission both at the system level and network level.

Illegal Use

An annual report on High Technology crime listed nine common types of computer crime:

- Criminal communications
- Fraud
- Hacking
- Electronic payments
- Gambling and pornography
- Harassment
- Intellectual property offenses
- Viruses
- Pedophilia

CONCLUSIONS &RECOMMENDATIONS

Steganography transmits secrets through apparently innocuous covers in an effort to conceal the existence of a secret. Digital image steganography and its derivatives are growing in use and application. In areas where cryptography and strong encryption are being outlawed, citizens are looking at steganography to circumvent such policies and pass messages covertly. As with the other great innovations of the digital age: the battle between cryptographers and cryptanalysis, security experts and hackers, record companies and pirates, steganography and Steganalysis will continually develop new techniques to counter each other.

In the near future, Digital Watermarking will be the most important use of steganographic techniques. To protect the copyrighted works against illegal distribution , digital watermarks provide a way of tracking the owners of these materials. Steganography might also become limited under laws, since governments already claimed that criminals use these techniques to communicate.

The possible use of steganography technique is as following:

- Hiding data on the network in case of a breach.
- Peer-to-peer private communications.
- Posting secret communications on the Web to avoid transmission.
- Embedding corrective audio or image data in case corrosion occurs from a poor connection or transmission.

Future Enhancement

I have analyze the various techniques using the various parameters. Firstly, I have analyze the three existing techniques using the image quality parameter. After analyzing, i found that optimal pixel adjustment technique provide the very less change in image quality than other two investigated methods. Secondly, I have analyze the techniques using the chances of message insertion. Thirdly, I have analyze the three different existing techniques using message length parameter. There is always some space of improvement in mostly proposed techniques. So, in future i will try to propose some new techniques which will improve various image steganography parameters.

REFERENCES

1. en .wikipedia.org/wiki/Steganography

2. ijcaet.net/documents/vol1issue1/IJCAET20120104.p

3. http://www.infosyssec.com/infosyssec/Steganography/techniques.htm

4. Peter Wayner, Disappearing cryptography: information hiding: steganography & watermarking, 3rd Edition, Morgan Kaufmann Publishers, 2009.