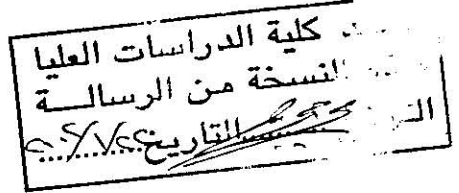


**HIDING AN IMAGE IN ANOTHER BY USING
STEGANOGRAPHY**

Handwritten notes in Arabic script, including the number 29 and other illegible characters.



**BY
Abdulrahman Omar Alkhalifa**

**Supervisor
Dr. Ahmad Al-Jaber**

**Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science**

**Faculty of Graduate Studies
University of Jordan**

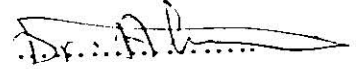
January 2002

This Thesis was successfully defended and approved on 13/1/2002.

Examination committee

Signature

Dr. Ahmad Al-Jaber
Assoc. Prof. Of Algorithm Analysis, Chairman



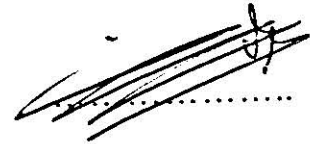
Dr. Riyadh Jabri
Assoc. Prof. Compiler Design



Dr. Ahmad Al-Sharieh
Asst. Prof. Of Parallel Processing



Dr. Mohammed Al- Zoubi
Asst. Prof. of Graphics and Pattern Recognition



Dr. Talib Al-Sarie
Prof. Of Numerical Analysis



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

فمكث غير بعيد فقال أحطت بما لم تحط به وجئتك من
سبأ نبأ يقين.

بِسْمِ اللَّهِ
الْحَمْدُ لِلَّهِ
الْعَظِيمِ

الآية ﴿٢٢﴾ سورة النمل

The Contents

Items	Pages
Chapter 1 : Information Hiding	1
1.1 Introduction	1
1.2 Subject Review	3
1.3 Importance of data Hiding	5
1.4 Data Hiding Classification	7
1.5 Steganography Definitions	8
1.6 Digital Watermarking Definitions	10
1.7 Difference between Digital Watermarking and Steganography	11
1.8 Data Hiding Features	11
1.9 Aim of the Theses and Outlines	12
Chapter Two: Models of Steganography	14
2.1 Introduction	14
2.2 Preliminaries and Terminology	14
2.2.1 Steganography Terminology	14
2.2.2 Steganalysis Preliminaries	19
2.2.3 Public Key Steganography	20
2.3 Classification of Steganography Techniques	22
2.3.1 Substitution Systems	24
2.3.2 Transform Domain Techniques	38
2.3.3 Spread Spectrum and Information Hiding	39
2.3.4 Statistical Steganography	40
2.3.5 Distortion Techniques	41
2.3.6 Cover Generation Techniques	42
2.4 Good Steganography	42
2.5 Difference between Cryptography and Steganography	44
2.6 Steganography System Security	47
2.6.1 Similarity Function	49
2.6.2 Entropy Test Function	50
2.6.2.1 Discrete Probability Distribution	52
2.7 Steganalysis Definitions	52
2.8 Attack and Steganalysis	52
2.9 Types of Attacks	54
2.9.1 Passive Attacker	55
2.9.2 Active Attacker	56
2.9.3 Malicious Attackers	58

Items	Pages
Chapter 3:New Algorithm For Gray Image Steganography System	
3.1 Introduction	60
3.2 Gray Image Steganography System	60
3.2.1 Image Histogram	61
3.2.2 Digital Images and its Palettes	63
3.2.2.1 Gray-level palettes	64
3.2.2.2 256 Color Palettes	66
3.3 GISS (Gray Image Steganography System)	68
3.4 GISS Keys	69
3.4.1 Short Key	70
3.4.2 Long Key	70
3.5 GISS Cover Image Size	71
3.6 GISS Stages	72
3.6.1 Media-Test Stage:	74
3.6.2 Splitting stage	75
3.6.3 Positioning and evaluating stage	76
3.6.4 Jump and Hide stage	79
3.7 Extracting stage	83
3.8 GISS (Gray Image Steganography System) Result	100
3.9 GISS Security	105
3.10 Comparison	106
Chapter four : Discussion and Conclusion	106
4.1 Discussion	107
4.2 Conclusions	
4.3 Suggestions for Future Work	
References	
Appendix1	
Appendix2	

Table of Algorithm

Algorithm	Ref #	Pages
Embedding process: Least significant bit substitution	2.1	25
Extracting process: least significant bit substitution	2.2	26
Computing the index j_i using pseudorandom permutation	2.3	32
Embedding algorithm	3.1	71
Media-Test Algorithm	3.2	73
Split Algorithm	3.3	74
Abstraction Search Algorithm	3.4	75
Jump and Hide Algorithm	3.5	77
Extracting Algorithm	3.6	81

The Tables

The Table	Ref #	Page s
Gray level palette	(3.1)	65
256 Color Palette	(3.2)	67
The long key	(3.3)	86
The Position File of Stego-Image	(3.4)	87
Long Key (INH)	(3.5)	94
Long Key (OUTH)	(3.6)	95
The INP Position File of Stego-Image	(3.7)	95
The OUTP Position File of Stego-Image	(3.8)	96

Table of Figures

The figure	Ref #	Pages
The growth of information hiding publications	(1.1)	2
The classification of data hiding	(1.2)	7
The Embedding Model	(2.1)	15
General steganography system(Stego System)	(2.2)	16
Steganography classification	(2.3)	23
The existing substitution systems	(2.4)	24
640X480 gray scale cover-image	(2.5)	27
The cover – image histogram	(2.6)	28
100X100 gray scale embedded-image	(2.7)	28
The embedded – image histogram	(2.8)	29
The distribution of Embedded – image pixels over Cover image	(2.9)	29
The Stego– image (Embedded with Cover- image)	(2.10)	30
The Stego – image histogram	(2.11)	30
Cryptography Sketch	2.12	44
Steganography Sketch	2.13	46
Combined Security System	2.14	47
(a) Histogram of eight Laplace filtered image; (b)Histogram of eight Laplace filtered image after applying an existing steganography system	(2.15)	55
Visual Data (Image)	(3.1)	62
The histogram of The Image	(3.2)	62
A Gray Image	(3.4)	64
A gray scale palette	(3.5)	65
A Color Image	(3.6)	66
The Color Palette	(3.7)	67

Seventh Bytes short key	(3.8)	70
The Embedded Image	(3.9)	83
The Embedded Image Histogram	(3.10)	84
The Cover Image	(3.11)	85
The Cover Image Histogram	(3.12)	85
The Stego Image	(3.13)	88
The Stego Histogram	(3.14)	89
The Distribution of Embedded Image Over Cover Image	(3.15)	90
The Embedded Image	(3.16)	90
The Embedded Image Histogram	(3.17)	91
The Cover1 Image	(3.18)	91
The Cover Image Histogram	(3.19)	92
The Stego-Image	(3.20)	97
The Stego Histogram	(3.21)	98
The Distribution of Embedded Image Over Cover Image	(3.22)	99
Probability Distribution of figure (3.9)	(3.23)	102
The Stego Image Probability Distribution of figure (3.11)	(3.24)	102
The Cover Image Probability Distribution of figure (3.16)	(3.25)	103
The Stego-Image Probability Distribution of figure (3.18)	(3.26)	104

Abstract

Hiding an Image In Another By Using Steganography

**By
Abdulrahman Omar Alkhalifa**

**Supervisor
Dr. Ahmad Al-Jaber**

Steganography is the science that deals with hiding an object of a specific identity within another one, which is either similar or different from its counterpart. This differs from cryptography, which deals with encrypting a message instead of hiding through the application of certain technique. In this thesis we introduce a new algorithm “Gray Image Steganography System (GISS)” for hiding images. These new methods are compared with the previous ones “Least Significant Bit (LSB)”. The comparison shows better results. The similarity parameter and entropy function show too close result to a high degree of measurement. What resulted by the algorithms of the present study assure the superiority of the present method.

The system of GISS (Gray Image Steganography System) functions by hiding a gray scale color image in a larger one with the same properties and characteristics that are declared where the key is secret and contracted by the two sides. This system consists of five stages Media-Test stage, Splitting stage, Position Evaluating stage, Hiding stage and extraction stage.

The system of GISS is established to be easily applied by the user through providing the system with the original image with an image intended to be hidden. The purpose behind this is to let the system do the processes of embedding and hiding to produce an image ready to send. This image is

called stego-image. When this image is received at a specific destination, the second site extracts the embedded image from the stego-image.

The present study provides evidence that the system of GISS is workable and efficient.

To My Family, Parents and Sisters

Acknowledgements

I would like to express my sincere appreciation to my research supervisor, The Dean of King Abdulla II School for Information Technology, for giving me the major steps to go on to explore the subject, shearing with me the ideas in my research "*Information Hiding In Image Media File*", and perform the points that I felt were important. I wish also to thank The Head and the faculty of Computer Science Department for their available advice.

Information Hiding

1.1 Introduction to Information Hiding

Information hiding represents a class of processes used to embed data into various forms of media such as image, audio, or text. The embedded data should be invisible to a human observer [Bender W., et al, 1996]

Information hiding is a large topic covers two areas, the first one is called Steganography and the second one is digital watermarking. The word Steganography comes from the Greek words, “stegein” “grajein” which literally means “covered “ which literally means, “covered writing”. It conceals a message where that is the object of the communication, for example, sending an image hidden in another image.

Digital watermarking means embedding information into multimedia data that should be imperceptible but unremovable. It extends some information that may be considered attributes of the cover, such as copyright.

Until recently, information-hiding techniques received less attention from industry than cryptography, but this is changing rapidly.

Within the past several years, there has been an exponential increase in the research community and industry’s focus towards information hiding.

Steganography includes a vast number of methods of secret communications that conceal the very existence of the message. Among these methods are invisible inks, microdots, character arrangement (other than the cryptographic methods of permutation and substitution), digital signatures, covert channels and spread-spectrum communications [T. Aura, 1996].

1.2 Subject Review

Throughout history, a multitude of methods and variations has been used to hide information. David Kahn's *The Code-breakers* provides an excellent account of this history. Bruce Norman recounts numerous tales of cryptography and Steganography during times of war in *Secret Warfare: The Battle of Codes and Ciphers* [D. Kahn, 1996].

One of the first documents describing Steganography is from the Histories of Herodotus. In ancient Greece, text was written on wax covered tables. In one story Demeratus wanted to notify Sparta that Xerxes intended to invade Greece. To avoid capture, he scraped the wax off of the tablets and wrote a message on the underling wood. He then covered the tablets with wax again. The tablets appeared to be blank and

unused so they passed inspection by centuries without question.[Johnson N.F. 1999]

Another historical fact, Histiaeus wished to inform his friends that it was time to begin a revolt against the Medes and the Persians. He shaved the head of the most trusted slave and tattooed a message on the head, waited till his hair grew back, and sends him along. The message would be undetected until the head was shaved again [Khan David,1996].

In Tudor England, when Mary Queen of Scots was imprisoned at Chartley Castle, she sent secret message to the Catholics including the French Ambassador, by hiding the letters inside the empty beer barrels that left the castle [Khan David,1996].

Another common form of invisible writing is through the use of invisible inks. Such inks were used with much success as recently as 2nd World War. An innocent letter may contain a very different message written between the lines. Common sources for invisible inks are milk, vinegar and fruit juices. All of these darken when heated. With the improvement of technology and the ease as to decoding of these invisible inks, more sophisticated had to be “developed” much as photographs are developed with a number of chemicals in processing labs [Johnson N.F. 1999].

Null ciphers (unencrypted message) were also used, in this method the first letter of each word of the sent message spells out the hidden message [Khan David,1996]. With every discovered of a message hidden using an existing application, a new Steganography application is being devised. It is simple to encode a message by varying lines, colors or other elements in picture. Electronics and Computers take such a method to new dimensions.

1.3 Importance of data Hiding

The threat by some governments to ban cryptography has led to a surge of interest in steganography. Steganography is destined to become more important as more people join the Cyberspace revolution and as the existing governments of personal privacy purposes [Franz Elke et al. , 1996]. If the government prohibits the use of cryptography, you can still send encrypted messages by hiding the encrypted message in another innocuous file using steganography techniques. Also there are often cases when sending encrypted message is not possible, either because you are working for a company that does not approve of encrypted email or perhaps the local government does not approve of encrypted

communication (a reality in some parts of the world). This is where steganography can come into play.

The main driving force is concern over protecting copyright; as audio, video and other works become available in digital form, book and unauthorized copying which will undermine the music, film, book and software publishing industries. There has therefore been significant recent research into “watermarking” (hidden copyright messages) and “fingerprinting” (hidden serial numbers or a set of characteristics that tend to distinguish an object from other similar objects). The idea is that the latter can be used to detect copyright violators and the former to prosecute them.

By the current surveys the scientific community outside the military world has not yet examined the field of steganography in detail, in multilevel security systems (such as the one used by the army). One wants sometimes to declassify some information from, say, and “top secret “to” confidential “or even” public. Unfortunately, this is not as easy as it seems, especially if you want to downgrade image.

1.4 Data Hiding Classification

During this study many conflicts appear about the general classification of data hiding. Some authors separate steganography from watermarking, others classify digital watermarking as a one type of steganography. Figure (1.1) shows a classification that is adopted during this study.

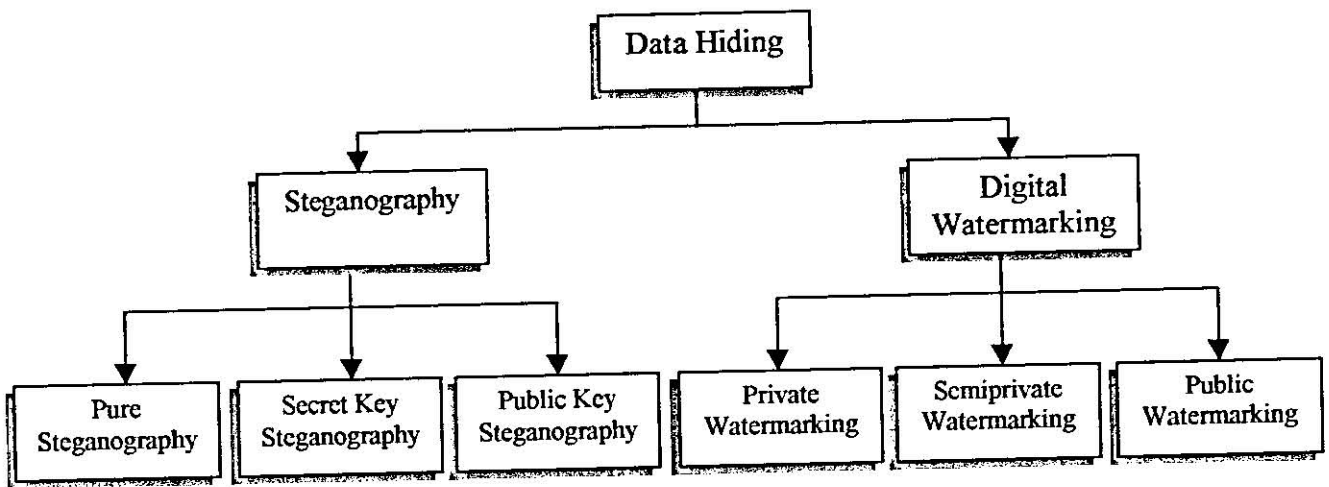


Figure (1.2): The classification of data hiding

In this classification, the main category of data hiding – which is separated into two classes – is the steganography and Digital Watermarking.

In the next section, we will present some definitions of steganography and digital watermarking and the difference between them

1.5 Steganography Definitions

Any system, method or technique that deals with, processing information (data), and put this data in shapes or forms of media under the condition that it must be not visible in its new form for human observer are called hiding systems for information.

Systems are defined to be known, the following are the most widely used definitions in Steganography:

Def 1.5.1.

Steganography includes a vast number of methods of secret communication that conceal the very existence of the message. Among these methods are invisible inks, microdots, character arrangement (other than the cryptographic methods of permutation and substitution), digital signatures, covert channels and spread-spectrum communication.

Def 1.5.2

Steganography is the art and science of communication in a way, which hides the existence of the communication [Johnson N.F. 1999].

Def 1.5.3

Steganography is the art of passing information in a manner that the existence of the message is unknown [Johnson N.F. and Jajodia S. 1998].

Def 1.5.4

Steganography: In an ideal world users would all be able to openly send encrypted mail or files to each other with no fear of reprisals.

Def 1.5.5

Steganography encompasses methods of transmitting secret messages through innocuous cover carriers in such a manner that the very existence of the embedded message is undetectable [Johnson N.F. and Jajodia S. 1998].

Def 1.5.6

Steganography is in the (especially military) literature is also referred to as transmission security TRANSEC for short.

552062

1.6 Digital Watermarking Definitions

Def. 1.6.1

Digital watermarking is employed in an attempt to provide proof of ownership and identify copying and distribution of multimedia information [Johnson Neil F. et al., 1999].

Def. 1.6.2

Digital watermarking providing means of placing additional information within digital media so if copies are made, the rightful ownership may be determined [Johnson N.F. ,1999].

Def 1.6.3

A watermark is added to a multimedia content by embedding an imperceptible signature into multimedia data [Neubauer Chr. ,et al.,1998].

Def 1.6.4

Digital watermarking is a mechanism for binding content identification to content in a persistent manner [Lacy Jack, et al.,1998].

1.7 Difference between Digital Watermarking and Steganography

There are main different points between Steganography and digital watermarking.

Steganography is an invisible to the human eye, related to point-to-point communication between two parties., and must not be known. The existence of the hidden data is not known to the parties, so they will not have the interest in getting the embedded data.

While Digital Watermarking is an invisible to the human eye too, related to point-to-many point communication between one party and many other parties. It is a popular application to give proof of ownership of digital data by embedding copyright.

1.8 Data Hiding Features

Data hiding should be capable of embedding data in a host signal with the following restriction and features [Bender W., et al.,1996]:

1. The host signal should be nonobjectionally degraded and the embedded data should be minimally perceptible.
2. The embedded data should be directly embedded into the media, rather than in to a header or wrapper.

3. The embedded data should be immune to modification ranging from international and intelligent attempt for removing these data.
4. Asymmetrical coding of the embedded data is desirable

1.9 Aim of the Theses and Outline

The aim of this work is to design an efficient algorithm to hide a small gray scale images in to bigger gray scale without affecting them so as to avoid drawing suspicion to the transmitted images.

The proposed algorithm involves an idea that differs from the ideas mentioned in the literature. Most of the exiting works are used to hide a bit of information inside the least significant bit of the byte (pixel) in the cover media, but this is not suitable if it's used to hide a lot of information such as an image inside another image. In contrast, the proposed system replaces some of the bytes in the cover image with new bytes from the embedded image to be hidden using some developed methods like (pixels position techniques).

In chapter 2, we give a theoretical background about steganography, including the types of the steganography, steganography techniques the type of the steganography attacks and finally how steganography be classified as secure one. In Chapter 3, we will describe

the new proposed algorithm, its theoretical background, the basic idea of it (algorithm) with its results, and finally evaluates the proposed algorithm. In Chapter 4, we will contain the discussion and conclusion and some ideas for future work.

Models of Steganography

2.1 Introduction

Steganography has its place in security. It is not intended to replace cryptography but it can be considered as a supplement to this science. 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 steganography methods combine traditional cryptography with steganography; the sender encrypts the secret message prior to the embedding process. Such combination increases the security of the overall communication process, as it is more difficult for an attacker to detect embedded ciphertext (which itself has a rather random appearance) in a cover.

However strong steganography systems do not prior encryption stage.

2.2 Preliminaries and Terminology

2.2.1 Steganography Terminology

A general Steganography system is shown in Figure (2.1). In this system, there are numbers of known terminology, which are illustrated as follows: [Pfitzmann Birgit, 1996].

Figure (2.1) shows the basic model of a steganography system called the “Embedded Model”,

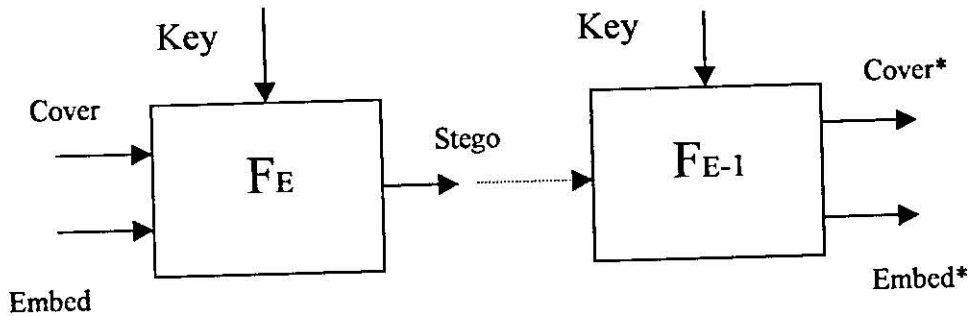


Figure (2.1) The Embedding Model

Where:

F_E : is the steganographic function "Embedding",

F_{E-1} : Steganographic function "Extracting",

Cover: is the cover data in which "Embed" will be hidden.

Embed: is the message to be embedded.

Key: is the parameter of F_E .

Stego: is the cover data with embedded message.

The input "cover" represent the untreated original data, "Embed" is the one which will be embedded into "cover" by the function (F_E), the resulting data called "Stego" contain the message "Embed", the operation (F_{E-1}) extracts the embedded data to "Embed*" and also produces an

output “Cover*”. Naturally, “Emb*” should be equal to “Emb” and in most cases “cover*” is the same as “stego” [T. Aura, 1996].

The main goals of steganography are:

1. To avoid drawing suspicion to the transmission of a hidden message. If suspicion is raised, then this goal is defeated [T. Aura, 1996].
2. To hide messages inside other “harmless” to even detect that there is a second secret message present [W. Benderet, al, 1996].

Steganography is divided into three main types, They are described in the following sections:

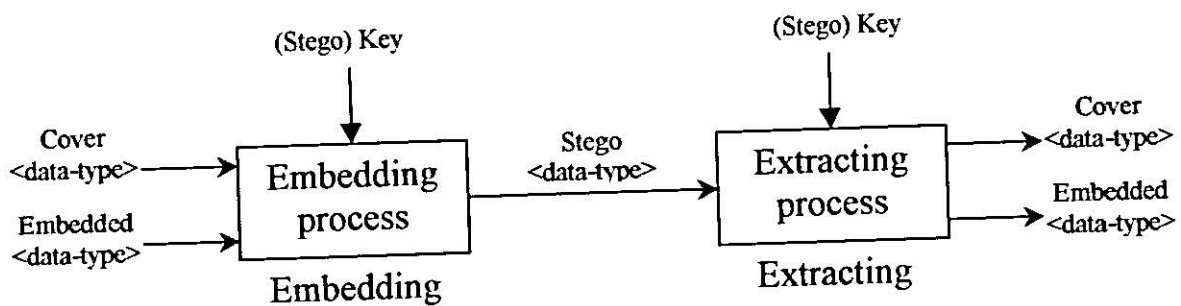


Figure (2.2) : General steganography system (Stego System)

1. Embedded <data-type>: Something to be hidden in something else
 M : refers to the set of possible message, m refers to the secret message, the length of m is by $L(m)$, and the bits forming m by $m_i, 1 \leq i \leq L(m)$.

2. Cover <data-type> : An input to the stego-system in which the embedded will be hidden.

C : refers to the set of possible covers used in the embedded step, c refers to a cover, it can be represented by a sequence of numbers c_i of length $L(c)$ (i.e., $1 \leq i \leq L(c)$). In the case of digital images, a sequence can be obtained by vectorizing the image.

$c_i = 0$ or 1 for binary images

A binary image is an image in which each pixel assumes one of only two discrete values. Essentially, these two values correspond to on and off.

Looking at an image in this way makes it easier to distinguish structural features. For example, in a binary image, it is easy to distinguish objects from the background. A binary image is stored as a two-dimensional matrix of 0's (which represent off pixels) and 1's (which represent on pixels). The on pixels are the foreground of the image, and the off pixels are the background.

Binary image operations return information about the form or structure of binary images only. To perform these operations on another type of image, you must first convert it to binary.

Or,

$0 \leq c_i \leq 255$ for 256 quantized image

A Quantized image : Quantization in general, is the most effective approach to reducing the number of colors in an image. This approach involves cutting the RGB (Red, Green and Blue) color cube into a number of smaller regions, and then mapping all colors that fall within a given region to the value at the center of that region. Quantization ensures that the colors in the output image do not deviate too much from those in the input image, and also enables you to control the maximum number of output colors.

The index of all cover-element c_i will use the symbol j for such an index. If the index itself is indexed by some set, the notation j_i will be used. When referring to j_i th cover-element I means c_{j_i}

3. Stego <data-type> : The output form the stego system ; something that has the embedded message hidden in it.

S: refers to the set of possible stego-objects, s refers to the stego object, the length of s is defined by $L(s)$.

4. Stego key or simply key: Additional secret data that may be needed in the stego system. The same key (or related one) is usually needed to extract the embedded message process.

K: refers to the set of stego-key. k refers to a stego-key

5. The process of hiding the embedded message is called embedding.

E : refers to the embedded message

6. Getting the embedded message out of the stegomessage again is called extracting.

D: refers to the extracting process.

7. The party from whom the embedded message is hidden is called stegoanalyst.

8. The key has been generated often depending on one or more security parameters. The standard case where the same key is used in embedding and extracting is called symmetric.

9. An entity or person that embeds and extracts is called an embedder and an extractor.

2.2.2 Steganalysis Preliminaries

Some of the basic Steganalysis definitions are [W. Bender, 1996 and Kahn David, 1996]

1. Passive warden: It is a spy in the communication channel that can do nothing.
2. Active warden: It is a spy that is allowed to modify (slightly) the data being sent between two persons.
3. Malicious warden: It is the spy that can alter the persons message with impunity, perhaps composing entire message.
4. Stego-only attack: Only the stego-object is available for analysis.

5. **Known cover attack:** At some point, the hidden message may become known to the attacker. Analyzing the stego-object for patterns that correspond to the hidden message may be beneficial for future attacks against that system. Even with the message, this may be very difficult and may even be considered equivalent to the stego-only attack.
6. **Chosen stego attack:** The steganography tool (Algorithm) and stego-object are known.
7. **Chosen message attack:** The steganalyst generates stego-object from some steganography tool or algorithm from a chosen message. The goal in this attack is to determine corresponding patterns in the stego-object that may point to the use of specific steganography tools or algorithm.
8. **Known stego attack:** The steganography algorithm (tool) is known and both the original and stego-object are available.

2.2.3 Public Key Steganography

As in public key cryptography, public key steganography does not rely on the exchange of a secret key. Public key steganography systems require the use of two keys, one private and one public key; the public key is stored in a public database. While the public

key is used in the embedding process, the secret key is used to reconstruct the secret message [S. Katzenbeisser and F. Petitcolas, 2000].

One way to build a public key steganography system is the use of a public key cryptosystem. Public key steganography utilizes the fact that the decoding function D in a steganography system can be applied to any cover c , whether or not it already contains a secret message (recall that D is a function on the entire set C). In the latter case, there are random elements of M that will be the result, we will call them "natural randomness" of the cover. If one assumes that this natural randomness is statistically indistinguishable from ciphertext produced by some public key cryptosystem, a secure steganography system can be built by embedding ciphertext rather than unencrypted secret messages.

A protocol that allows public key steganography has been proposed by Anderson [Anderson Ross, 1996] ; it relies on the fact that encrypted information is random and enough to "hide in plain sight".

Craver [Craver Scott, 1998] extended this protocol to simulate pure steganography using both public and private key steganography. In most applications, pure steganography is

preferred, since no stego-key must be shared between the communication partners, although a pure steganography protocol does not provide any security if an attacker knows the embedding method. By implementing a key-exchange protocol using public key steganography, the users can exchange a secret key k which they can later use in a secret key steganography system. Since no stego-key must be known in prior, we can refer to the communication process as pure steganography, although it does not conform to Definition (2.1).

2.3 Classification of Steganography Techniques

In the last section, we pointed out the different types of steganography. There are several techniques implemented in steganography, where any one of these techniques may involve different steganography types mentioned above. There are several approaches to classify steganography systems. One could categorize them according to the type of covers used for secret communication. A classification according to the cover modifications applied in the embedding process is another possibility [Katzenbeisser S. and Petitcolas F., 2000.].

This study undertakes the second approach, and groups of steganographic methods in six categories as shown in Figure (2.2).

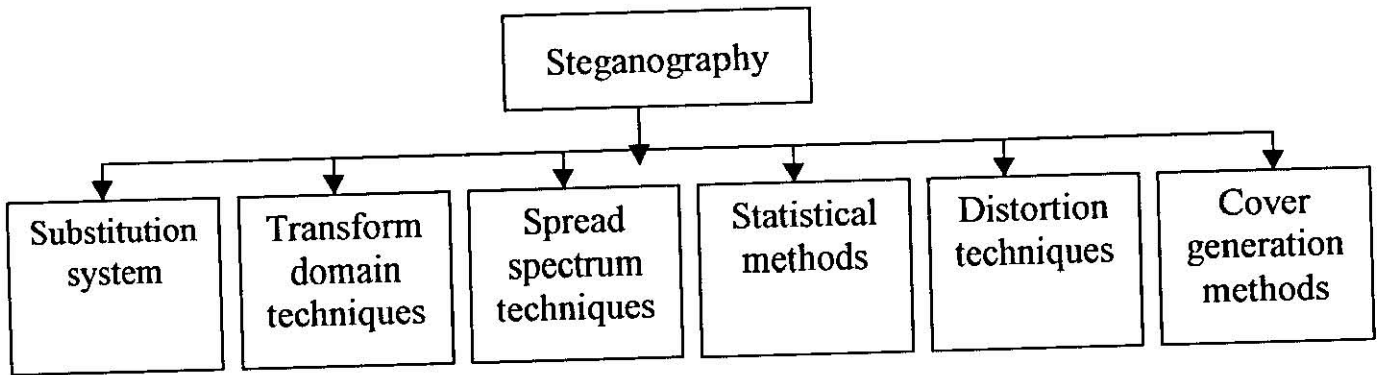


Figure (2.3): Steganography classification

These methods are **Substitution systems** substitute some parts of a cover with a secret message, **Transforme domain techniques** embed secret information in a transform space of the signal (e.g., in the frequency domain), **Spread spectrum techniques** adopt ideas from spread spectrum communication, **Statistical methods** encode information by changing, several statistical properties of a cover and use hypothesis testing in the extraction process, **Distortion techniques** store information by signal distortion and measure the deviation from the original cover in the decoding step, **Cover generating methods** encode information in the way that a cover for secret communication is created.

In the following sections these six categories will be discussed in more detail, but emphasizing the first since it is the one used in this work..

2.3.1 Substitution Systems

A number of methods exist for hiding information in various media. These methods range from Least Significant Bit (LSB) coding, also known as bitplane or noise insertion tools-to modification of image properties [Katzenbeisser S. and Petitcolas F., 2000].

Basic substitution systems try to encode secret information by substituting insignificant parts of the cover by secret message bits; the receiver can extract the information if he has knowledge of the position where secret information has been embedded, Figure (2.4) shows the existing methods. Since only minor modifications are made in the embedding process, the sender assumes that they will not be noticed by a passive attacker.

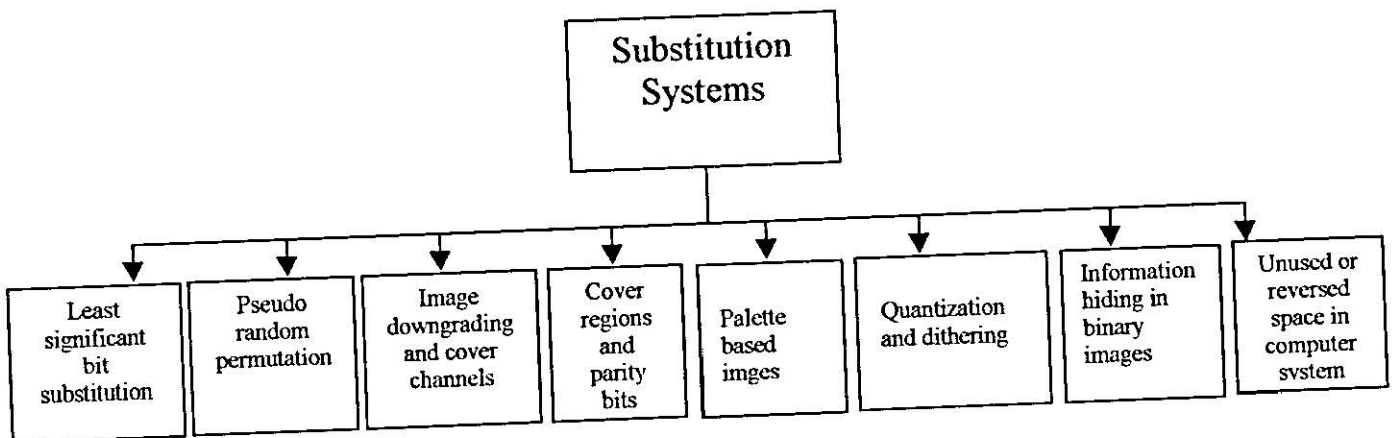


Figure (2.4): The existing substitution systems

2.3.1 A. Least significant bit substitution

Sample tools used in this group include StegoDos, S-Tools, Hide and Seek and White Noise Storm [Johnson, N. F., 1999]. The image formats typically used in such steganography methods are lossless and that data can be directly manipulated and recovered. Some of these programs apply compression and encryption in addition to steganography services. These services provide better security of the hidden data. Even so, the bitplane methods are rather brittle and vulnerable to corruption due to small changes to the carrier.

The embedding process consists of choosing a subset $\{j_1, \dots, j_l(m)\}$ of cover-elements and performing the substitution operation $c_{j_i} \rightarrow m_i$ on them, which exchanges the LSB of c_{j_i} by m_i (m_i can be either 1 or 0). In the extraction process, the LSB of the selected cover-element is extracted and lined up to reconstruct the secret message. The basic scheme is presented in Algorithm 2.1 and 2.2 [Katzenbeisser S. and Petitcolas F, 2000].

Algorithm 2.1: Embedding process: Least significant bit substitution

For $i=1, \dots, l(c)$ do

$s_i \leftarrow c_i$

end for

for $i=1, \dots, l(c)$ do

compute index j_i , where to store i th message bit

$s_i \leftarrow c_{j_i} \leftrightarrow m_i$

end for

Algorithm 2.2: Extracting process: least significant bit substitution

for $i=1, \dots, l(m)$ do

compute index j_i , where to store i th message bit

$m_i \leftarrow \text{LSB}(c_{j_i})$

end for

In order to be able to decode the secret message, the receiver must have access to the sequence of elements indices used in the embedding process. In the simplest case, the sender uses all cover elements for information transfer, starting at the first element. Since the secret message will normally have less bit than $l(c)$, the embedding process will be finished long before the end of the cover. In this case, the sender can leave all other elements unchanged. This can, however, lead to a serious security problem: the first part of the cover will have different statistical properties than the second part, where no modifications have been made.

To overcome this problem, for instance some programs enlarge the secret message with random bits in an attempt to create an equal change in randomness at the beginning and the end of the cover. The embedding

process thus changes for more elements than the transmission of the secret would require. Therefore the probability that an attacker will suspect secret communication increases. For more information, see Appendix (2).

The image in Figure (2.5) illustrates the two way stego system (Embedding , Extracting)



Figure (2.5) 640X480 gray scale cover-image

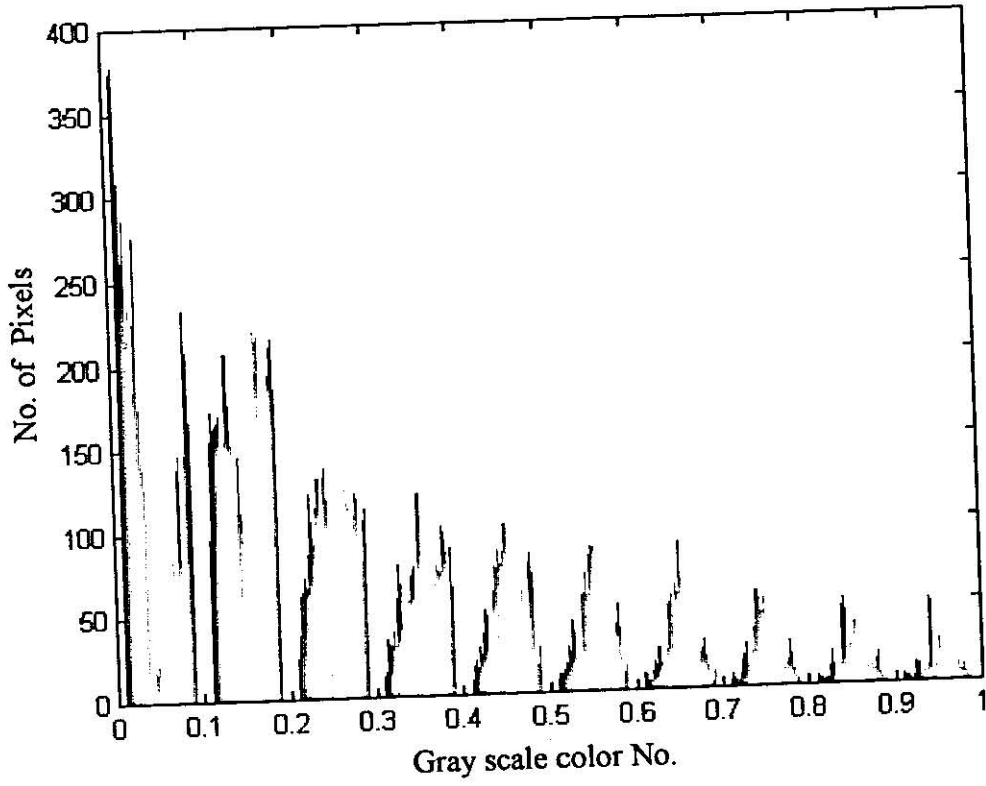


Figure (2.6) the cover – image histogram

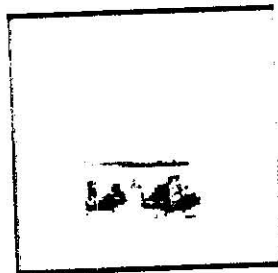


Figure (2.7) 100X100 gray scale embedded-image

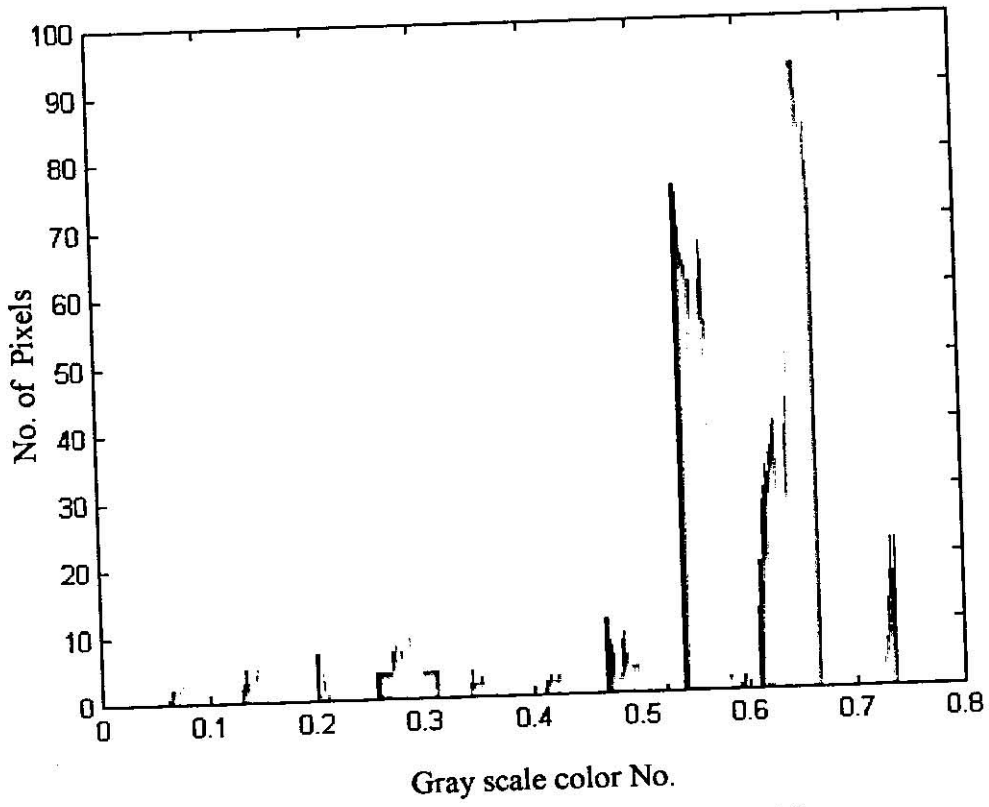


Figure (2.8) the embedded – image histogram

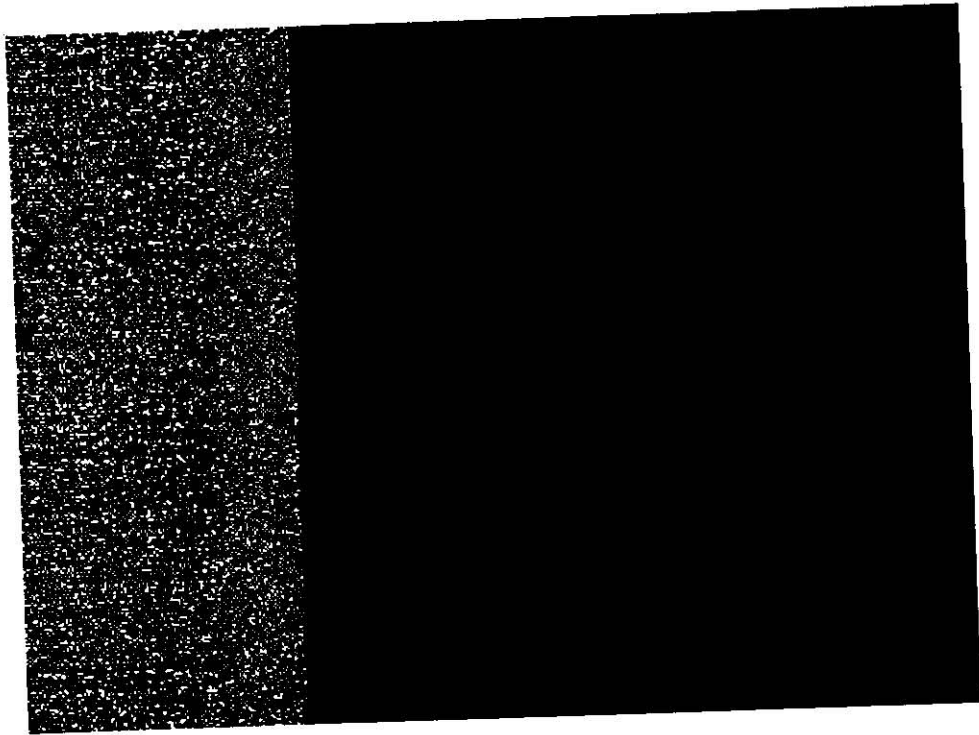


Figure (2.9) the distribution of Embedded – image pixels over Cover image



Figure (2.10) the Stego- image (Embedded with Cover- image)

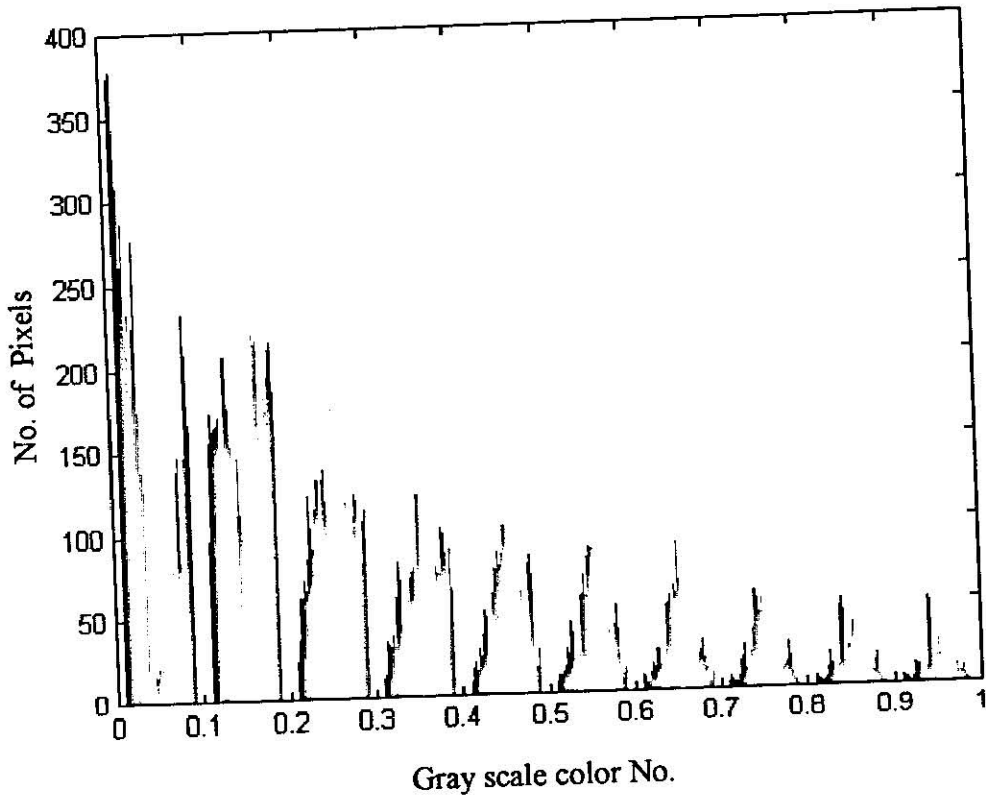


Figure (2.11) the Stego – image histogram

2.3.1 B. Pseudorandom permutation

A more sophisticated approach is the use of a pseudorandom number generator to spread the secret message over the cover in a rather random manner; a popular approach is the random interval method. Thus the distance between two embedded bits is determined pseudorandomly.

This technique further increases the complexity for an attacker, since it is not guaranteed that subsequent message bits are embedded in the same order.

By using pseudorandom number generator a sequence $j_1, \dots, j_{l(m)}$ of elements indices are created which are used to store the n th message bit in these elements with index j_n .

Collision appears (one index could appear more than once in the sequence). Since there are no restriction to the output of the pseudorandom number generator, then it is possible to insert more than one message bit into the same cover-element, thereby corrupting some of them.

To overcome the problem of collision, Aura [Aura Tumoas, 1996] uses the basic substitution scheme of Algorithms 2.1 and 2.2 and calculates the index j_i , via a pseudorandom permutation of the set $\{1, \dots, l(c)\}$. Suppose the number $l(c)$ can be expressed as a product of two numbers, X and Y , and h_k is an arbitrary cryptographically secure

hash function depending on a key K . Let k_1 , k_2 and k_3 be three secret keys. It can then be shown that Algorithm 2.3 outputs a different number j_i for each input i ($1 \leq i \leq XY$), (i.e., it produces a pseudorandom permutation of the set $\{1, \dots, l(c)\}$), provided that the algorithm is evaluated with input $i=1, \dots, l(c)$ [Katzenbeisser S. and Petitcolas F, 2000].

Algorithm 2.3: Computing the index j_i using pseudorandom permutation

$$v \leftarrow i \text{ div } X$$

$$u \leftarrow i \text{ mod } X$$

$$v \leftarrow (v + h_{k_1}(u)) \text{ mod } Y$$

$$u \leftarrow (u + h_{k_2}(v)) \text{ mod } X$$

$$v \leftarrow (v + h_{k_3}(u)) \text{ mod } Y$$

$$j_i \leftarrow vX + u$$

In Algorithm (2.3) the stego-key K is split into three pieces. K_1, k_2 and k_3 . In the embedding process, the i th message bit in the element with index j_i , which is computed according to Algorithm 2.3 is stored. Collision does not occur, since Algorithm 2.3 does not produce duplicate elements indices.

To reconstruct the position of the secret message bits, the extractor must have access to the three keys k_1 , k_2 and k_3 . However,

this method needs a considerable amount of computation time, since the chosen hash function must be evaluated $3l(m)$ times.

2.3.1 C. Image downgrading and cover channels

Image downgrading is a special case of substitution system in which images act both as secret message and covers. Given a cover-image and a secret image of equal dimensions, the sender exchanges the four least significant bits of the cover's grayscale (or color) values with the four most significant bits of the secret image. The receiver extract the four least significant bits out of the stego-image, thereby gaining access to the most significant bits of the secret image. In many cases, four bits are sufficient to transmit a rough approximation of the secret image [Katzenbeisser S. and Petitcolas F,2000].

By this method, the degradation of the cover will be high although in Katzenbeisser S. and Petitcolas F,2000] they stated that "the degradation will not be noticeable", but this is not true in practical applications.

2.3.1 D. Cover regions and parity bits

Any nonempty subset of $\{c_1, \dots, c_l(c)\}$ is called a cover-region. By dividing the cover in several disjoint regions, it is possible to store

one bit of information in a whole cover-region rather than in a single element. A parity bit of a region can be calculated by equation (2.1)

$$b(l) = \sum_{j \in l} \text{LSB}(c_j) \bmod 2 \dots \dots \dots (2.1) \quad [\text{Katzenbeisser S. and Petitcolas F, 2000}]$$

In the embedding step, $l(m)$ disjoint cover-regions $I_i, (1 \leq i \leq l\{m\})$ are selected, each encodes one select bit m , in the parity bit $b(I_i)$. If the parity bit of one cover-region I_i does not match with the secret bit m ; to encode, one LSB of the values in I_i is flipped. This will result in $h(I_i) = m_i$.

In the decoding process, the parity bits of all selected regions are calculated and lined up to reconstruct the message. Again, the cover-regions can be constructed pseudorandomly using the stego-key as a seed.

Although the method is not more robust than simple bit substitution, it is conjectured to be more powerful in many cases.

2.3.1 E. Palette-based images

Generally, there are two ways to encode information in a palette-based images (image palette is described later in chapter three): either the palette or the image data can be manipulated [Katzenbeisser S.

and Petitcolas F,2000]. The LSB of the color vectors could be used for information transfer, just like the substitution methods presented. Alternatively, since the palette does not need to be stored in any way, information can be encoded in the way the colors are stored in the palette. Since there are $N!$ different ways to sort the palette, there is enough capacity to encode a small message. However, all methods which use the order of a palette to store information, are not robust, since an attacker can simply sort the entries in a different way and destroy the message.

One proposes a slightly different technique, which does not need the palette to be sorted:

for every pixel, the set of closest colors is calculated. Starting with the closest color, the sender proceeds to find the next closest color until a color is found where its parity matches with the secret bit to encode. Once such a color is found, the pixel is changed to this new color.

Yet another steganographic application reduces the total number of color values in a picture to $\lfloor N/2 \rfloor$ using some dithering method, and doubles the entire palette; thereby all doubled entries are slightly modified. After this preprocessing stage, each color value of the dithered image corresponds to two palette entries from which one is chosen according to a secret message bit.

2.3.1 F. Quantization and dithering

Dithering and quantization of digital images can be used for embedding secret information. Some steganographic systems operate on quantized images. The difference e_i between adjacent pixels x_i and x_{i+1} is calculated and fed into a quantizer Q that outputs a discrete approximation Δ_i of the difference signal $x_i - x_{i-1}$. Thus, in each quantization step a quantization error is introduced [Katzenbeisser S. and Petitcolas F,2000].

For steganographic purpose the quantization error in a predictive coding scheme can be utilized; specifically, we adjust the difference signal Δ_i so that it transmits additional information. In this scheme, the stego-key consists of a table, which assigns a specific bit to every possible value, Δ_i .

In order to store the i th message bit in the cover-signal, the quantized difference signal Δ_i is computed. If Δ_i does not march (according to the secret table) with the secret bit to be encoded, Δ_i is replaced by the nearest Δ_i where the associated bit equals the secret message bit. The resulting values Δ_i are the fed into the entropy coder. At the receiver side, the message is decoded according to the difference signal Δ_i and the setgo-key.

2.3.1 G. Information hiding in binary images

Binary images contain redundancies in the way black and white pixels are distributed. Although the implementation of a simple substitution scheme is possible, these systems are highly susceptible to transmission errors and are highly susceptible to transmission errors and are therefore not robust.

In this method, a binary image is divided into rectangular image blocks B_{ij} . Let $P_0(B_i)$ be the percentage of black pixels in the image block B_i and $P_1(B_i)$ the percentage of white pixels, respectively [Katzenbeisser S. and Petitcolas F,2000]. Basically, one block embeds a 1, if $P_1(B_i) > 50\%$ and a 0, if $P_0(B_i) > 50\%$. In the embedding process the color of some pixels is changed so that the desired relation holds. Modifications are carried out of those pixels whose neighbors have the opposite color; in sharply contrasted binary images, modifications are carried out at the boundaries of black and white pixels. These rules assure that the modifications are not generally noticeable.

A different embedding, presented by Matsui and Tanaka, uses the lossless compression system which is used to encode information in a facsimile document.

2.3.1 H. Unused or reserved space in computer systems

Taking advantage of unused or reversed space to hold covert information provides means of hiding information without perceptually degrading the carrier. For example the way operating systems store files typically results in unused space that appears to be allocated to a file.

Another method of hiding information in file system is to create a hidden partition. These partitions are not seen if the system is started normally. However, in many cases, running a disk configuration utility (such as DOS FDISK) exposes the hidden partition [Katzenbeisser S. and Petitcolas F,2000].

2.3.2 Transform Domain Techniques

Transformation domain methods hide messages in a significant area of the cover image, which makes them more robust to attacks, such as compression, cropping some image processing, than the LSB approach. However, while they are more robust to various kinds of signal processing, they remain imperceptible to the human sensory system. Many transform domain variations exist. One method is to use the Discrete Cosine Transformation (DCT) as a vehicle to embed information in images; another would be the use of wavelet transforms. Transformation can be applied over the entire image, to

blocks throughout the image, or other variations. However, trade-off exists between the amount of information added to the image and the robustness obtained.

2.3.3 Spread Spectrum and Information Hiding

Spread Spectrum (SS) communication technologies have been developed since the 1950s in an attempt to provide means of low-probability-of-intercept and antijamming communications. Spread Spectrum techniques are defined as " means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information; the band spread is accomplished by means of a code which is independent of the data, and a synchronized reception with the code at the receiver is used for despreading and subsequent data recovery". Although the power of the signal to be transmitted can be large, the signal-to-noise ratio in every frequency band will be small. Even if parts of the signal could be removed in several frequency bands, enough information should be present in the other bands to recover the signal. This situation is very similar to a steganography system which tries to spread a secret message over a cover in order to make it impossible to perceive. Since spread signals tend to be difficult to remove, embedding methods based on SS should provide a considerable level of robustness.

In information hiding, two special variants of SS are generally used: direct sequence and frequency-hopping scheme. In direct-sequence scheme, the secret signal is spread by a constant called chip rate, modulated with a pseudorandom signal and added to the cover. On the other hand, in the frequency-hopping schemes, the frequency of the carrier signal is altered in a way that it hops rapidly from one frequency to the another. SS are widely used in the context of watermarking [Katzenbeisser S. and Petitcolas F,2000].

2.3.4 Statistical Steganography

Statistical steganography techniques utilize the existence of "1-bit" steganographic schemes, which embed one bit of information in a digital carrier. This is done by modifying the cover in such a way that some statistical characteristics change significantly if a "1" is transmitted. Otherwise the cover is left unchanged. So the receiver must be able to distinguish unmodified covers from modified ones.

In order to construct an $l(m)$ -bit stego-system from multiple "1-bit" stego-systems, a cover is divided into $l(m)$ disjoint blocks $B_1, \dots, B_{l(m)}$. A secret bit, m_i , is inserted into the i th block by placing "1" into B_i if $m_i=1$. Otherwise, the block is not changed in the embedding process. The detection of a specific bit is done via a test

function which distinguishes modified blocks from unmodified blocks:

$$f(B_i) = \begin{cases} 1 & \text{If 1 block } B_i \text{ was modified in the embedding process} \\ 0 & \text{otherwise} \end{cases}$$

The function f can be interpreted as a hypothesis-testing function. We test the null hypothesis "block B_i was not modified" against the alternative hypothesis "block B_i was modeled". The receiver successively applies f to all cover-blocks B_i in order to restore every bit of the secret message.

This technique is difficult to apply in many cases. First a good test statistic must be found which allows distinction between modified and unmodified cover-blocks. Additionally, the distribution must be known for a "normal" cover in most cases. This is quite a difficult task.

2.3.5 Distortion Techniques

In contrast to substitution technique systems, distortion requires the knowledge of the original cover in the decoding process. In order to hide information, a sequence of modifications is made to the cover in order to get a Stego-object. A sequence of modifications is chosen in such a way that it corresponds to a specific secret message want to be transmitted. The receiver measures the difference to the original

cover in order to reconstruct the sequence of modifications applied by the sender, which corresponds to the secret message.

In many applications, such systems are not useful, since the receiver must have the access to the original cover. If the attacker also has access to them, he can easily detect the cover modifications and has evidence for a secret communication. If the embedding and extraction functions are public and do not depend on a stego-key, it is also possible for the attacker to reconstruct secret message entirely.

2.3.6 Cover Generation Techniques

In contrast to all embedding methods presented above, where secret information is added to a specific cover by applying an embedding algorithm, some steganographic applications generate a digital object only for the purpose of being a cover for secret communication.

2.4 Good Steganography

A good Steganography system should fulfill the same requirements posed by the "*Kerckhoff principle*" in cryptography [Katzenbeisser S. and Petitcolas F,2000].

Cryptography and Steganography are a contest between two adversaries. The first one is the designer of the system and the second one is the opponent who attempts to recover the original information. This process is called cryptanalysis in cryptography and steganalysis

in Steganography. The ground rules of this struggle between the designer and opponent were formulated in the nineteenth century by Kerckhoffs in six points which are [Konheim Alan G.,1981]:

- k₁**. The system should be, if not theoretically unbreakable, unbreakable in practice.
- k₂**. Compromise of the system should not inconvenience the correspondent.
- k₃**. The method for choosing the particular member (key) of the cryptographic system be used should be easy to memorize and change.
- K₄**. Ciphertext should be transmittable by telegraph.
- K₅**. The apparatus should be portable.
- K₆**. Use of the system should not require a long list of rules or mental strain.

This means that the security of the system has to be based on the assumption that the "enemy" has full knowledge of the design and Implementation details of the steganographic system. The only missing Information for the "enemy" is a short easily exchangeable random number sequence, the secret key, and without the secret key, the "enemy" should not have the slightest chance or even become suspicious that an observed communication channel which is hidden might take place.

However really good steganography is much more difficult and usage of most of the currently available steganographic tools might be quite easily detected using sufficiently careful analysis of the transmitted data.

Difference between Cryptography and Steganography

Although steganography differs from cryptography, many of the techniques and wisdom from the more thoroughly researched discipline can be borrowed. Covert information is not necessarily secure and secure information is not necessarily covert information.

The two are fundamentally different. The distinction between the two is made clear in the following discussion.

In cryptography, information is secured by transforming original data into encrypted data with an enciphering scheme. Figure (2.12) depicts the encryption process that produces the output, or ciphertext. The cipher text should be meaningless as to what it is truly representing.

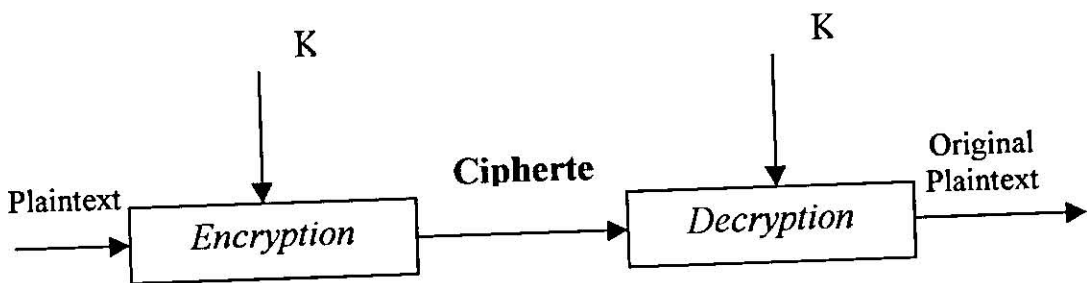


Figure (2.12) Cryptography Sketch

Steganography, on the other hand, leaves the original data unchanged and conceal it. The original information is hidden using an embedding technique into an innocent cover medium, as shown in Figure (2.12). To an observer, the cover medium appears normal unchanged. By applying the reverse of the original embedding technique, the original data is recovered.

Past cryptography history has shown that the adversary usually knows that communication is occurring and is able to intercept it. The adversary is often aware that the information is encrypted and that in most cases will break the encryption algorithm at any cost. Thus, cryptography's underlying security is based on the difficulty of breaking the encryption algorithm. With sufficient time and resources, this decryption task has usually been achieved [H. Brase, 1982].

The modern system of cryptography is categorized two classes, block cipher and stream cipher.

In block cipher, the plaintext (message) must be assembled into block length depend on the cryptographic algorithm designer (system designer), the corresponding ciphertext block depend on the cryptographic key.

In stream cipher, the plaintext is encoded into numbers (usually binary), and a key stream of binary numbers are combined with the plaintext to form the ciphertext. The receiver is supplied with one

steganography system then we have a double security on our data, which implies the increase of the protection applied on our information.

However, some security can be implemented by combining the two sciences as shown in Figure (1.6). The combination of these two techniques has become an everyday practice for many of the steganographic systems [6].

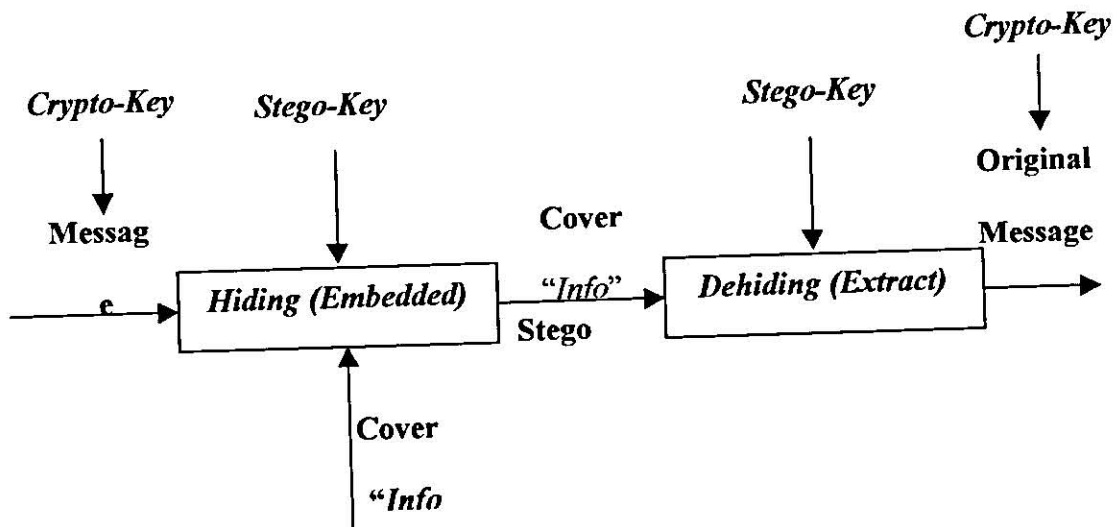


Figure (2.14) Combined Security System

2.6 Steganography System Security

Although breaking a steganography system normally consists of three parts: detecting, extracting, and disabling embedded information. A system is already insecure if an attacker is able to prove the existence of a secret message. In developing a formal security model of steganography, we must assume that an attacker has unlimited computation power and is able and willing to perform a

variety of attacks. If he cannot confirm his hypothesis that a secret message is embedded in a cover, then a system is theoretically secure [Katzenbeisser S. and Petitcolas F.,2000].

The goal of a secure steganographic methods is to prevent an observant intermediary from even obtaining knowledge of the mere presence of the secret data[Etting J. Mark,1998]

The two necessary conditions for secure steganography are [Zollner J.,and eat,1998]:

1. Key remains unknown to the attacker
2. The attacker does not know the actual cover.

We can define a secure steganography algorithm in terms of four requirements [Katzenbeisser S. and Petitcolas F., 2000]

- * Messages are hidden using public algorithm and a secret key; the secret key must identify the sender uniquely;
- * Only a holder of the correct key can detect, extract, and prove the existence of the hidden message. Nobody else should be able to find any statistical evidence of a message's existence;
- * Even if the enemy knows (or is able to select) the contents of one hidden message, he should have no chance of detecting others;
- * It is computationally infeasible to detect hidden messages.

There are two basic methods used to prove the security of steganography system, these methods are:

1. Similarity function
2. Entropy test function

2.6.1 Similarity Function

The embedding process is defined in a way that a cover and the corresponding stego-object are perceptually similar. Formally, perceptual similarity can be defined via a similarity function:

Definition 2.3 [Katzenbeisser S. and Petitcolas F.,2000]: (Similarity function) Let C be a nonempty set. A function $\text{sim} : C^2 \rightarrow [-\infty, 1]$ is called similarity function on C , if for $x, y \in C$, then

$$\text{sim}(x, y) \begin{cases} = 1 & x \Leftrightarrow y \\ < 1 & x \neq y \end{cases}$$

In the case of digital images, the correlation between two images can be used as a similarity function. Therefore most practical steganographic systems try to fulfill the condition $\text{sim}\{c, E\{c, m\}\} \approx 1$ for all $m \in M$ and $c \in C$. Where c is the cover image, E is the embedded one, and m is the secret message.

For every communication process, a cover is randomly chosen. The sender could also look through the database of usable covers and select one that the embedding process will change the least.

Such a selection process can be done via the similarity function sim . In the encoding phase, the sender chooses a cover c with the property in equation 2.2

$$C = \max \text{sim}(x, E(x, m)) \dots \dots \dots (2.2)$$

The sender could select one, best suitable for communication. Such a technique, called selection method of invisibility [Katzenbeisser S. and Petitcolas F., 2000],

2.6.2 Entropy Test Function

An information-theoretic model for steganography was proposed by Cachin [Cachin Christian, 1998], which gave a definition of the security of steganographic systems. The main idea is to refer to the selection of a cover as a random variable C with probability distribution P_C . The embedding of a secret message can be seen as a function defined in C ; let P_s be the probability distribution of $E_k(c, m, k)$, that is the set of all stego-objects produced by the steganographic system. If a cover c is never used as a stego-object, then $P_s(c) = 0$. In order to calculate P_s , probability distribution on K

and M must be imposed. Using the definition of relative entropy $U(P_1||P_2)$ between two distribution P_1 and P_2 are defined on the set W . As in equation

$$U(P_1 || P_2) = \sum_{w \in W} P_1(w) \log_2 (P_1(w)/P_2(w)) \dots \dots \dots (2.3)$$

which measures the inefficiency assuming that the distribution is P_2 where i th true distribution is P_1 - the impact of the embedding process on the distribution P_C can be measured. Specifically, we define the security of a steganography system in terms of $U(P_c||P_s)$.

Definition 2.4: (Perfect security) Let \mathfrak{S} be a steganography system, P_s

the probability distribution of the stego-cover sent via the channel, and P_c the probability distribution of C .

\mathfrak{S} is called ϵ -secure against passive attackers if

$$U(P_c||P_s) \leq \epsilon$$

and perfectly secure if $\epsilon=0$

Since $U(P_C||P_S)$ is zero if and only if both probability distributions are equal, we can conclude that a steganography system is (theoretically) perfectly secure, if the process of embedding a secret message in a cover does not alter the probability distribution of C . A perfectly secure system can be constructed out of a one time.

2.6.2.1 Discrete Probability Distribution

If a variable X can assume a discrete set of value X_1, X_2, \dots, X_k with respective probabilities P_1, P_2, \dots, P_k where $P_1 + P_2 + \dots + P_k = 1$, we say that a discrete probability distribution for X has been defined. The function $p(X)$ which has the respective values P_1, P_2, \dots, P_k for $X = X_1, X_2, \dots, X_k$, is called the probability function or frequency function of X . Because X can be assumed certain values with given probabilities, it is often called a discrete random variables [Spiegel R. Murray, 1991].

2.7 Steganalysis Definitions

- * Steganalysis is the art of discovering and rendering useless such covert messages [Etting J. Mark, 1998].
- * Steganalysis are the methods in determining the existence and potential locations of hidden information [Johnson, N. F. and Jajodia S., 1998].

2.8 Attack and Steganalysis

New terminology with respect to attacks and breaking steganography schemes is similar to cryptography terminology; however, there are some significant differences.

Just as cryptography applies cryptanalysis in an attempt to decode Or crack encrypted messages, the steganalyst is one who applies steganalysis in an attempt to detect the existence of hidden information. With cryptography, comparison is made between portions of the plaintext (possibly none) and portions of the ciphertext. In steganography, comparisons may be made between the cover-media, the stego-media, and possible portions of the message. The end result in cryptography is the ciphertext, while the end result in steganography is the stego-media.

The message in steganography may or may not be encrypted, if it is encrypted, then if the message is extracted, the cryptanalysis techniques may be applied.

In order to define attack techniques used for steganalysis, corresponding techniques are considered in cryptanalysis.

There are somewhat parallel attacks are available to the steganalyst, these are stego-only, known cover, known message, chosen stego, and chosen message. A stego-only-attack is similar to the ciphertext only attack where only the stego-medium is available for analysis. If the “Original” cover-media and both available, then a known cover attack is available. The stegoanalysis may use a known message attack when the hidden message is revealed at some later data, an attacker may attempt to analyze the stego-media for future attacks.

Even with the message, this may be very difficult and may even be considered equivalent to the stego-only attack..

The chosen stego attack is one where the steganography tool (algorithm) and stego-medium are known. A chosen message attack is one where the steganalyst generates stego-media from some steganography tool or algorithm from a known message.

The goal of this attack is to determine corresponding patterns in the stego-media that may point to the use of specific steganography tools or algorithms [E. Denning, 1983].

2.9 Types of Attacks

There are three main types of attacks, passive, active and malicious attacks.

As in most steganography systems, they change parts of the cover media causing changes in the covers statistical properties, the embedding process does not pay attention to this fact. A passive attacker could exploit this fact and break the system.

Active attackers are able to change a cover during the communication process. It is a general assumption that an active attacker is not able to change the cover and its semantics entirely; but only make minor changes so that original and the modified cover-object stay perceptually or semantically similar. An attacker is malicious if he forges message or

starts steganography protocols under the name of one communication partner. So, during the design of a steganography system attention has to be paid to the presence of passive attacker, since by detecting that a message was hidden then the other two attacks will start.

2.9.1 Passive Attacker

The passive attacker can detect the existence of a secret message by using the discrete Laplace operator[20]. By this operator it is possible to detect secret messages in grayscale images.

$$\nabla^2 P(x,y) = p(x+1,y) + p(x-1,y) + p(x,y+1) + p(x,y-1) - 4p(x,y) \dots (2.4)$$

Evaluating (2.4) at every point (x,y) gives the "Laplace filtered" image. Since we can expect neighboring pixels to have a similar color, the histogram of $P(x,y)$ is tightly clustered around zero.

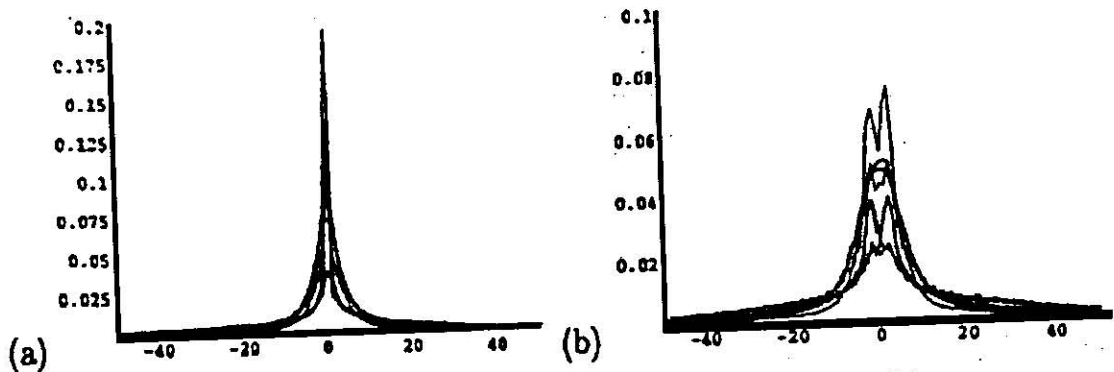


Figure (2.15) : (a) Histogram of eight Laplace filtered image;

(b) Histogram of eight Laplace filtered image after applying an existing steganography system

Figure (2.15a) shows eight histogram of Laplace filtered grayscale image printed in one coordinate system. Figure (2.15b) shows the Histogram of the same image after applying an existing steganography algorithm. Since the embedding process adds noise To the picture, which is statistically quite different from the true random noise, the new histogram differs extremely. Laplace filtering does not prove the existence of a secret, but it will provide strong evidence that the picture was subject to modification.

2.9.2 Active Attacker

Steganography systems are extremely sensitive to cover modifications, such as image processing techniques (like smoothing, filtering, and image transformations) in the case of digital image. But even a lossy compression can result in total information loss., Lossy compression techniques try to reduce the amount of information by removing imperceptible signal components and so often remove the secret information which has previously been added.

An active attacker, who is not able to extract or prove the existence of a secret message, thus can simply add random noise to the transmitted cover and so try to destroy the information. In the case of digital images, an attacker could also apply image processing techniques or convert the

image to another file format. All of these techniques can be harmful to the secret communication. Another practical requirement for a steganography system, therefore, is robustness. A system is called robust if the embedding information cannot be altered without making drastic changes to the stego-object.

Definition 2.5 [Katzenbeisser S. and Petitcolas F,2000]:

(Robustness) Let \mathfrak{S} be a steganography system and \mathfrak{a} be a class of mapping $C \rightarrow C$. \mathfrak{S} is \mathfrak{a} -robust if for all $p \in \mathfrak{a}$

$$D_k(p\{E_k(c,m,k),k\}) = D_k(E(c,m,k),k) = m \dots\dots\dots(2.5)$$

Equation (2.4) represents the formula in the case of a secret steganography system and

$$D(p(E(c,m))) = D(E(c,m)) = m \dots\dots\dots(2.6)$$

in the case of a pure steganography system, regardless of the choice of $m \in M$, $c \in C$, and $k \in K$

It should be clear that there is a trade-off between security and robustness. The more robust a system will be against modification of the cover, the less secure it can be.. Since robustness can only be achieved by

redundant information encoding which will degrade the cover heavily and possibly alter the probability distribution P_s .

An ideal system would be robust to all "a-similarity-preserving" mapping (i.e., mapping $p:C \rightarrow C$ with the property $\text{sim}(C,p(C)) \geq a$ and $a \approx 1$). However, such system is difficult to engineer and would have a low bandwidth due to the robustness of the encoding. On the other hand, a system is called a-weak, if for every cover any similarity-preserving mapping exists so that the encoded information is not recoverable in the sense of (2.5) and (2.6).

Generally, there are two approaches in making steganography robust. First, by foreseeing possible cover modifications the embedding process itself be made robust so that modifications will not entirely destroy secret information. A second approach tries to reverse the modifications that have been applied by the attacker to cover, so that the original stego-object can be restored.

2.9.3 Malicious Attackers

In the presence of a malicious attacker, robustness is not enough. If the embedding method is not dependent on some secret information shared by sender and receiver, an attacker can forge messages, since the

recipient is not able to verify the correctness of the sender's identity.

Thus, to avoid such an attack, the algorithm must be robust and secure.

New Algorithm For Gray Image Steganography System

3.1 Introduction

In all previously reviewed substitution methods, a bit of information from the embedded-object was hidden in a byte of the cover object. In the proposed system, a bit from the embedded-object will be hidden in a byte from the cover-object by using jump and hide technique.

The proposed system called Gray Image Steganography System (GISS) takes the embedded-object as input, which is in our case a gray-scale image. The cover-object will be a gray scale image too. The system produces an output stego-object, which is the resulting image.

3.2 Gray Image Steganography System

The Gray Image Steganography System is based on a number of general concepts such as data structures, image histogram and color palette. Formal definitions and clarifications will be given in the following literatures.

3.2.1 Image Histogram

An *image histogram* is a chart that shows the distribution of intensities in an indexed or intensity image. The image histogram function creates this plot by making equally spaced bins, each representing a range of data values. It then calculates the number of pixels within each range.

In general, the histogram of a digital image is an approximation of the distribution of colors in an image. To obtain the image histogram, each color intensity r_k presented in an image has frequency of occurrence n_k , that is, the number of pixels in the image that has color r_k . [Gomes And et. al., 1997]

The histogram function is as follows [Gomes And et. al., 1997]:

$$H(r_k)=n_k$$

where

r_k is the k th gray or color level.

n_k is the number of pixels in the gray or color image

k is the range of gray or color levels (i.e., 0,1,...255).

The Figure (3.2) illustrates simple histogram to every visual data.



Figure (3.1): Visual Data (Image)

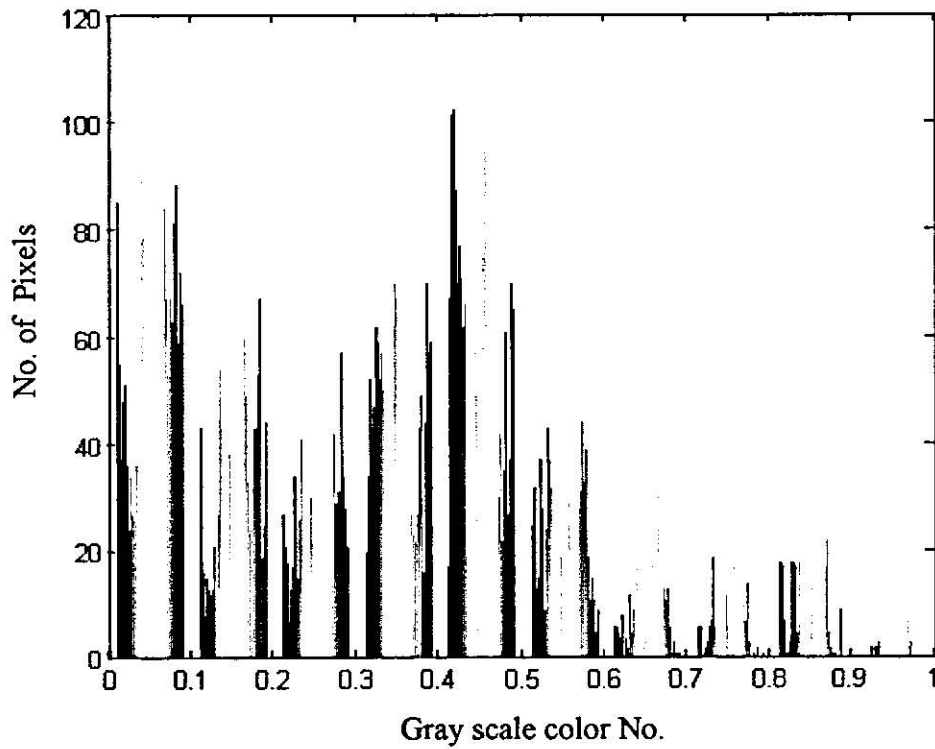


Figure (3.2): the histogram of The Image

3.2.2 Digital Images and Their Palettes

In order to represent and manipulate images on the computer, we must define appropriate mathematical models. Thus a natural mathematical model for describing an image is a function on a two-dimensional $f(x,y)$ where x and y denote spatial coordinates and the value of f at any point (x,y) is the value of the color [Gomes And et. al., 1997].

A digital image can be considered as a matrix whose row and column indices identify a point in the image where the corresponding matrix element value identifies the gray or color level at that point. The elements of such a digital array are called image elements that are picture elements pixel or pels.

In order to convert continuous images to discrete images, an image must be digitized both spatially and in amplitude. Digitization of the spatial coordinates (x,y) is called image sampling and amplitude digitization is called color or gray level quantization. In other words, the discretization of an image's color space is called quantization. The discretization of the color map is called a palette.

Every palette consists of two parts: a palette specifying N colors as a list of indexed pairs (i,cli) , assigning a color vector cli to every index i , and the actual image data which assigns palette index to every pixel rather than the color value itself. If only a small number of color

values are used throughout the image, this approach greatly reduces the file size.

3.2.2.1 Gray-level palettes

In gray-level images, a digital image will have a 256 palette. Each palette represents a gray shaded color. Each image palette consists of the three primary colors (Red, Green, and Blue). The value of these colors are equal as shown in table(3.1). Figure (3.4) shows a gray level image palette of the image in Figure (3.4).



Figure (3.4): A Gray Image

Table (3.1): Gray level palette

Palette No.	Red	Green	Blue
0	0	0	0
1	1	1	1
2	2	2	2
.	.	.	.
.	.	.	.
↓	↓	↓	↓
255	255	255	255

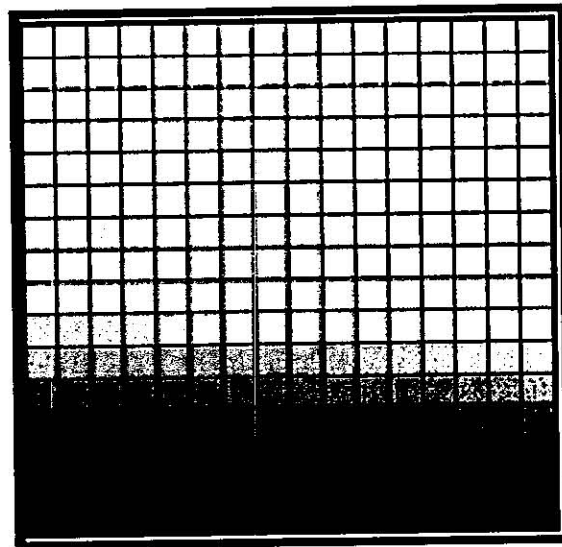


Figure (3.5): A gray scale palette

3.2.2.2 (256) Color Palettes

In a 256 color image, there will be also a 256 palette of the three primary colors (Red, Green, Blue) but in the case of color images the values of these components are not equal and they depend on the continuous image discretization as shown in table (3.2). Figure (3.6) shows the palettes of a 256-color image of that in figure (3.6).



Figure (3.6): A Color Image

Table (3.2): 256 Color Palette

Palet No.	Red	Green	Blue
0	000	000	000

1	004	012	012
2	012	012	004
3	012	012	012
4	020	012	012
.	.	.	.
.	.	.	.
↓	↓	↓	↓
255	255	255	255

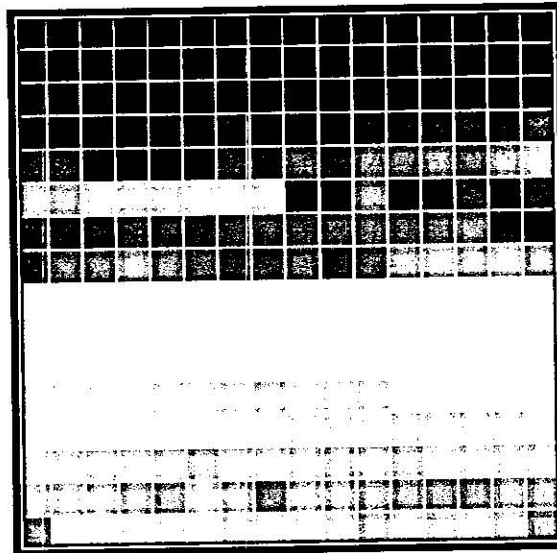


Figure (3.7): The Color Palette

For more Information see Appendix (1)

3.3 GISS (Gray Image Steganography System)

The embedding process of GISS has four stages: Media-Test stage, Splitting stage, Position Evaluating stage and Hiding stage

Like all Steganography methods, the **GISS** has a key, which is used to extract the embedded-image from the Stego-image. Two types of keys were used, the first key is a seven byte length, which is considered as a secret key transferred separately. The second one is (765-byte or 1530-byte) length key, which will be saved in the same image (Stego-image) and the extractor will get it from the same image.

- **Media-Test Stage:** In this stage, the **GISS** will check the size of the embedded-image against the cover-image. The size of embedded-Image must be smaller than the size of the cover-image, test whether the cover image is suitable to embed the embedded image or not.
- **Splitting stage :** In this stage, we will be splitting the color palette of embedded image into one or two output table that contain matching and unmatching colors values that are depending on cover image color values.
- **Position Evaluating stage:** To generate the position file that contain the embedded pixels position, so if we have one table for matching color the position file is unique, otherwise two position

files will be generated that holds both the positions of matching and unmatching pixels.

- **Hiding stage:** this is the final stage to hide the positions of pixels files in cover image, which the result calls stego-image.

3.4 GISS Keys

Two major types of keys are used in this system. The first type is a set of seven bytes (short key). The second one is 765 bytes (long key). Short keys are sent to separate it from a stego-image, while the long key is embedded inside the stego-image.

3.4.1 Short Key

Figure (3.8) shows the short key components:

- The short key contains a set of seven bytes.
- The first two bytes represent the size of INP position file.
- The second two bytes represent the size of OUTF position file.
- The fifth byte hold the value (0 or 3) represent the existing position file ,which :
 - 00 value represents the INP position file.
 - 11 value represents the INP and OUTF position file.

N_{EH} : Embedded image height size

INH palette size: the palette size that consists the matching color value numbers which is calculated by $(256*2*16)$ or less.

OUTH palette size: the palette size that consists the unmatching color value numbers which is calculated by $(256*2*16)$ or less .

Jump: the value that is specified to separate between cover pixels to hide embedded pixels.

gap: the value that is specified to separate between IN position pixels file and OUT position pixels file.

3.6 GISS Stages

The embedding process is performed using four main stages as described before, in the next sections these stages are described in detail.

The general embedding algorithm of the **GISS** is as illustrates below:

Algorithm 3.1 Embedding algorithm

Input : The covered-image, the embedded-image.

Output : The stego-image.

Process: Media Test and Embedding

Step 1: *Select the embedded-image and select the cover-image.*

Step 2: *Perform the Media-test stage if the result is true go to step 3, otherwise return to step 1 to select new cover-image.*

Step 3: *Calculate the histogram of both embedded and cover images and generate the INH and OUTH tables that holds the histogram color values for embedded image.*

Step 4: *Perform the Abstraction Search Algorithm and produce the position file that contains the pixels positions.*

Step 5: *Using the Jump and Hide Algorithm to perform the embedding process.*

Step 6: *End*

3.6.1 Media-Test Stage:

The first step in the embedded algorithm is the test cover stage. The stage includes:

- 1. Size test:** by applying the equation (3.1), we can determine if the embedded image is suitable for embedding in the cover or not.
- 2. Pallete test:** This test is done to decide whether the selected cover-image is suitable to hold the embedded-image or not. This test is done by calculating the histograms of both images and making a quick comparison between the distribution of pixels according to their histogram. If the distribution of the embedded image is near to the distribution of the cover image then this test is passed, otherwise, the test fails and the cover-image must be changed see Algorithm (3.2).

The palette test is done as follows:

- Calculate the histogram of both the cover-image and embedded-image.

-Compare between the two histograms as follows:

if $H(\text{cover-image}) \geq H(\text{embed-image})$ then increment the counter by one.

- If the value of counter > 150 then it is acceptable otherwise it is refused.

Algorithm 3.2: Media-Test Algorithm

Input : The cover image and the embedded image

Output: Return true or false

Process : Embedding Image Suitability

Step 1: If the size of Embedded image is greater than or equal to the
Cover image then goto step 5

Step 2: Compute the histogram of Embedded and Cover images.

Step 3 If the color values in histogram table of Embedded are
greater than the Cover histogram values set flag to false and
goto step 5

Step 4 Set flag to true

Step 5 End

3.6.2 Splitting stage

In this stage we calculate the number of color pixels of embedded image over the cover image pixels value and determine the tables that hold all color values, which agree with the color value of color image called INH table for matching colors and OUTH table for unmatched colors.

The Algorithm (3.5) illustrates the splitting strategy, which take the histogram of both cover and embedded images. The output results are INH and OUTH tables.

Algorithm 3.3: Split Algorithm

Input: The Histogram of Cover Image (HC) and Histogram of Embedded Image (HE).

Output: (INH) table contains all matching color, (OUTH) table contains unmatching color and (COL) represent the color palette

Process : Splitting Image Colors

Step 1 : For $i = 1$ to size of embedded Histogram (HE) **Do**

If no. of color in HE (i) < no. of color HC (j) **Do**

Increment matching color X by 1,

Put the no. of color of HE(i) in INH (i) table,

Put the color i in COL(i) table.

Else

Increment unmatching color Y by 1,

Put the no. of color of HE(i) in OUTH (i) table

End If

End For

Step 2 : END

3.6.3 Positioning and evaluating stage

In this stage, the position of pixels of the embedded image are found. The search algorithm starts by finding each position of pixel depending on its color in the embedded image, and finally the algorithm generate the dynamic table called POS that contains the color and its position of pixels, and this table stored in a file called position file see Algorithm (3.4).

Algorithm 3.4: Abstraction Search Algorithm

Input: The Cover-Image (C) and Embedded-Image (E), (COL) the color palette.

Output: Dynamic table (POS) contains the position of embedded pixels.

Process : Embedding Pixels Position

Step 1 : Mark all position of dynamic table (POS) with nil value

Step 2 : For $i=1$ to no. of color in COL palette **Do**

 Select the COL(i) from COL palette

```
For j=1 to size of cover-image (C) Do
    If COL(i) is equal to C(j) Do
        Increment the matching counter X by 1,
        Put the matching color and its position in the POS table
    End If
    If matching color X is equal to the no. of the
    same color in the embedded-image (E) Do
        Stop searching on this color
    End If
End For
End For
```

Step 3: END

3.6.4 Jump and Hide stage

The important stage in the GISS is the jump and hide algorithm, where its main function is to hide each embedded position file inside the media of cover image. The first location of position file that contains the color and its positions embedded on cover image bit by byte and the second location of positions file is embedded in the next block and so on until the whole values of position file embedded entirely. The steps below illustrates the jump and hide technique, and are expressed in Algorithm (3.5)

1. Get the (INP) position file and (OUTP) position file
2. Check if (OUTP) position file equal to zero then goto step 4
3. Embed (OUTP) position file and public key (histogram of 'OUTP' position file) in the Cover image.
4. Jump selected number of positions and locate the empty area in cover image.
5. Embed (INP) position file and public key (histogram of 'INP' position file) in the Cover image

Algorithm 3.5: Jump and Hide Algorithm

Input: POSE : is the position file

CC : is the cover Media

Key : No. of Jumps between pixels.

HistE :the histogram of Embedded file

Output:

CE: the stego-image (Embedded with cover image)

Process : Jump and Hide

Step1: If POSE = OUT position file then

The gap = (cover size of (CC) - ((IN) position file size + (OUT) position file size + histogram of (IN) + histogram of (OUT)) / 2,

Jump a given number of gap position

Step2: For $i= 1$ to size of HistE

Get the color $i:= \text{HistE}(i,1)$, and get the No. of color $i:= \text{HistE}(i,2)$,

Embedd the color $i:= \text{HistE}(i,1)$ and the No. of color $i:= \text{HistE}(i,2)$ in cover image CE(i) bit by bit

Step3 : For $i= 1$ t size of POSE do

Get the i^{th} position value and put it in X variable

Call MaxPosBit algorithm to compute the No. of byte of the X value,

For $j=1$ to No. of byte in X value.

Embed the j^{th} bit of POSE(j) in CE(i) then

Jump a given number of key position.

End for

End for

Step 4: End

3.7 Extracting stage

To start the extracting process the extractor must have the stego-image and the stego-key. From the first and second byte of this key, the size of the embedded-image will be known as N , and the size of the cover-image M will be extracted from the third and fourth byte. After specifying the cover and embedded image size, The retrieving process is done by reading the (Least Significant Bit) LSB of the first bit plus (sixth byte of jump value) of each first byte in the cover image and collect these bits to generate the values of the position file.

From the fifth byte of the key, the extractor will know the number of position file as follow:

1. If the value of this byte equal to zero that means only one position file is embedded in the stego-image (INP).
2. If the value of this byte equal to three that means there are two position files embedded in the stego-image (INP and OUTP).

From the above, if the second stage is satisfied, we need to read the seventh byte (Gap value) to determine the location of position file (OUTP). So we retrieve the values of this file by reading LSB of each bytes plus (jump value) from stego-image area.

The process will continue until all the bits of the embedded-image are extracted.

The remaining unused area of the stego-image has been used to hide the long key. The extractor will retrieve (765 bytes for each position file) bit from the (LSB) of the bytes of the unused area that will create the long key.

The last stage is to re-transform the extracted image to the original-image using the long-key.

The steps bellow shows the extracting operation

1. Determine the short key and output of the position of (INP) position file (OUTP) position file if exist.
2. Extracting the (INH) and (OUTH) palette.
3. Extracting the (INP) and (OUTP) position file.
4. Build the extracting position file that contain both the histogram of (INH) and (OUTH) file with each pixel of position files INP and OUTP.
5. Generate from the extracting position file the embedded image.

The Algorithm (3.6) shows the retrieving process of GISS

Algorithm 3.6: Extracting Algorithm

Input: C: Stego Image

Key : no. of jump

Gap : the number of byte separate between INP and OUTP

SiE: size of Embedded image

SiC : size of Cover image

Output :

EM: Embedded Image

Process : Extraction

Step 1 : Initialize the temporary table (PS) to hold the Embedded palette

Initialize the temporary table (ES) to hold the Embedded position file values that will extracted from Stego-Image

Step 2 : If Gap value greater than zero then jump a given number of Gap position

Step 3 : For $i = 1$ to 765 // size of long key //

Get the i th bit from $C(i)$

For $j = 1$ to 16 // word (2 bytes) //

Extract the first bit from i th byte from $C(i)$

Set the j th bit to j th position of $PS(i)$ byte

End For

End For

Step 4 : For $i = 765$ to SiE

 Get the i th byte from $C(j)$

 For $j = 1$ to 16 // word (2 bytes) //

 Extract the first bit from i th byte of $C(i)$

 Set the j th bit to j th position of $ES(j)$ byte

 End For

End For

Step 5 : For $j = 1$ to size of PS

 Get the i th color from $PS(i)$

 Get the no. of i th color from $PS(j)$

 For $j = 1$ to no. of i th color from $PS(i)$

 Set the i th color from $PS(j)$ that is in the position $PS(j)$ to EM
 ($PS(j)$)

 End For

End For

Step 6 : End

3.8 GISS (Gray Image Steganography System) Result

In order to implement the conceptual framework discussed in previous sections, using Matlab (ver. 5.3) Application programming language and in addition to the modules involved with implementing the algorithms of embedding, hiding, and random jump generation GISS has been built.

The result has been illustrated in the following examples:

Example 1 :

The gray scale embedded image size = 100 X 100 pixels is as shown in Figure (3.9).

The gray scale cover image size = 640 X 480 pixels is as shown in Figure (3.11)

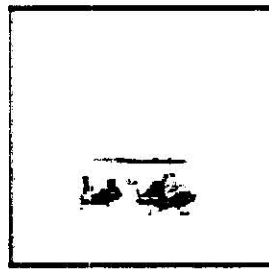


Figure (3.9): The Embedded Image

The histogram of the embedded image is shown in figure (3.10)

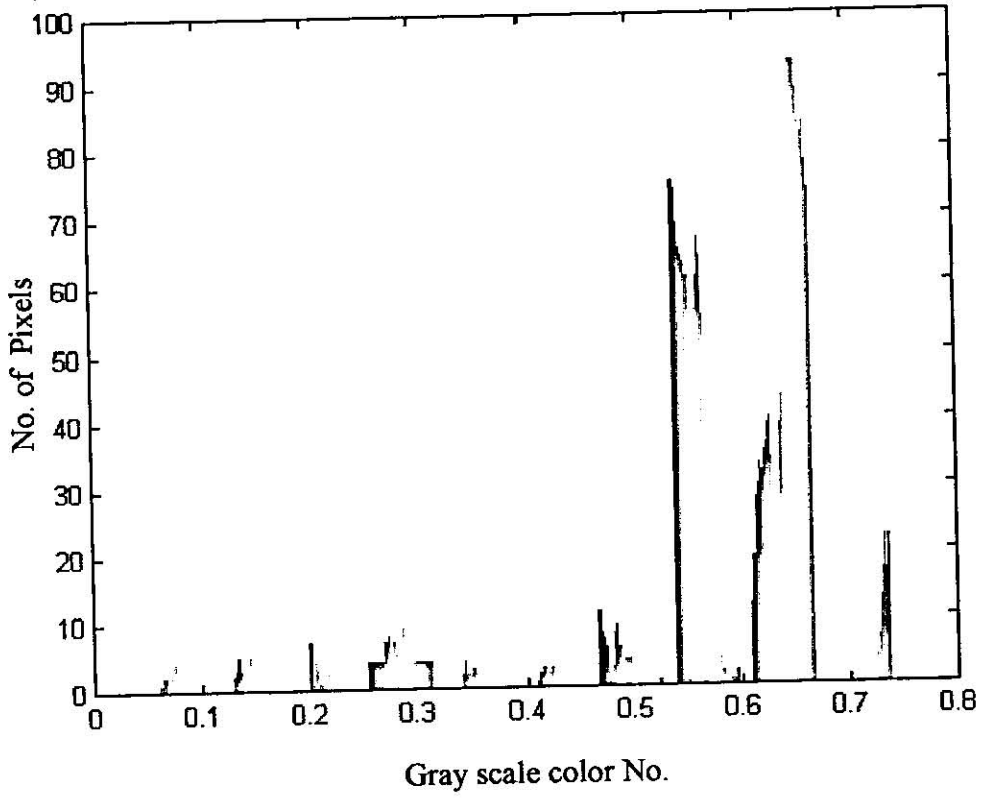


Figure (3.10): The Embedded Image Histogram



Figure (3.11): The Cover Image

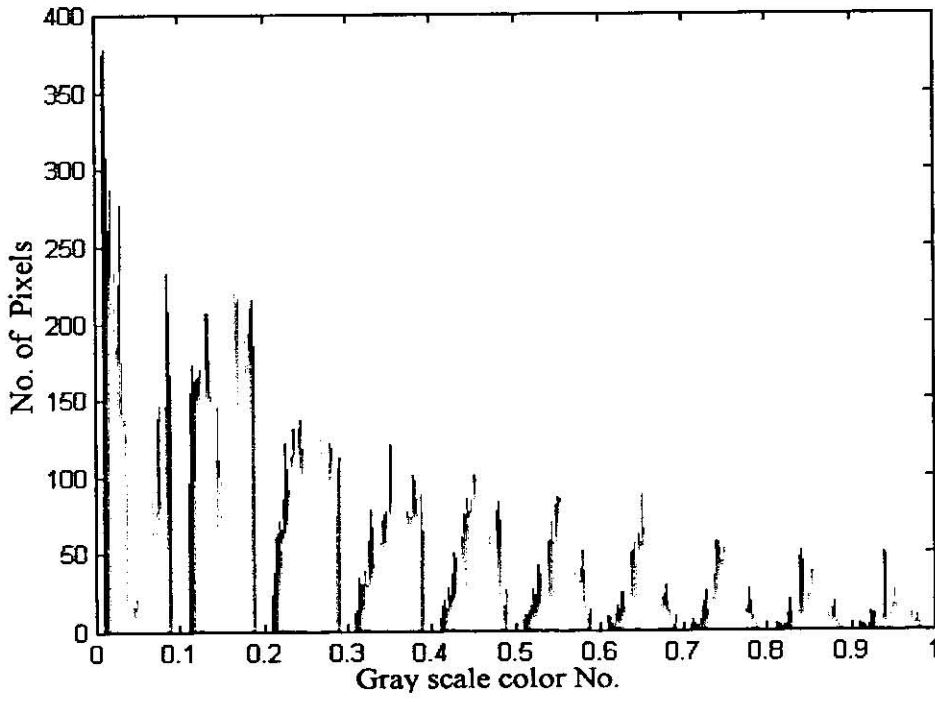


Figure (3.12): The Cover Image Histogram

552062

By applying media test stage, the two tests are passed. The first test (size test) is passed by using equation (3.1), since the size of cover image is greater than the size of equation result. The equation coefficients are illustrated below:

NEW=100, NEH=100, INH palette size = 765, OUTH palette size = 0, jump=20000, gap = 0

So, (the equation result=200,763)<(cover width X cover height =307,206)

The second test is passed since the number H(cover image) that is greater than H(embedded image) = 256 which is greater than 150.

After applying the splitting stage the long key is generated (INH),

Table (3.3)

Table (3.3): the long key

1	4	2	8	2	5	3	1	2	6	5	16	107	197	321	251	1
1	4	8	6	2	3	5	4	2	9	5	11	108	170	316	266	1
1	3	3	4	3	3	5	6	4	1	3	11	163	117	329	357	1
1	3	5	2	5	1	8	7	8	6	3	13	176	139	258	343	1
3	7	8	5	2	5	2	6	5	3	3	20	190	184	241	334	1
2	5	5	3	5	405	4	3	4	3	3	21	153	355	290	189	0
2	4	7	3	1	1	3	3	2	6	5	22	188	390	242	62	0
2	5	7	5	5	4	4	1	3	4	6	25	252	376	224	25	0
6	7	5	1	3	7	6	1	5	3	5	36	239	374	210	14	0
4	5	5	3	4	1	2	5	2	5	16	58	222	343	197	2	0

The abstraction search algorithm is then applied, and the result is the position file. Table (3.4) illustrate sample of this file.

Table (3.4) : The Position File of Stego-Image

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	5760	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0
12	5768	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	6260	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	5670	0	0	0	0	0	0	0	0	0
19	5769	5960	6875	0	0	0	0	0	0	0
20	5669	6576	0	0	0	0	0	0	0	0
21	5272	5473	0	0	0	0	0	0	0	0
22	3973	5160	0	0	0	0	0	0	0	0
23	3476	4173	5260	5460	5770	5860	0	0	0	0
24	4074	5360	5668	6673	0	0	0	0	0	0
25	3276	3376	5660	6672	0	0	0	0	0	0
26	5560	5575	6060	6477	0	0	0	0	0	0
27	4073	5060	6165	0	0	0	0	0	0	0
28	5373	6476	6775	0	0	0	0	0	0	0
29	2676	3974	5573	5674	6577	6772	6773	0	0	0
30	3673	5475	5673	5677	5774	0	0	0	0	0
31	3675	3874	5376	6873	0	0	0	0	0	0
32	3076	3176	5773	5877	6574	0	0	0	0	0
33	2975	2976	3774	5075	5173	6475	6572	0	0	0
34	2875	4070	4975	5671	6160	0	0	0	0	0
35	4760	6472	0	0	0	0	0	0	0	0
36	3175	3275	3475	3571	3868	3969	4660	6575	0	0
37	3375	5375	6674	0	0	0	0	0	0	0
38	3775	5271	5273	5276	6474	0	0	0	0	0
39	3075	3576	3675	4860	5676	5771	6177	6360	0	0
40	3575	4960	5076	5176	5977	0	0	0	0	0
41	3574	3674	5275	5775	6077	6378	6580	0	0	0
42	3673	3966	4768	4875	5371	6375	6675	0	0	0
43	3472	3474	6175	6176	6573	0	0	0	0	0
44	3374	3573	3771	5577	6277	0	0	0	0	0
45	3071	3274	3676	3871	5372	6275	6460	6973	0	0

After the position file is generated, the next stage is the jump and hide stage. In this stage, the jump and hide algorithm tries to embed the position file (INP) and long key (INH) into cover image, so the final result is stego-image. Figure (3.13) and Figure (3.14) illustrate the stego-image and there histogram.

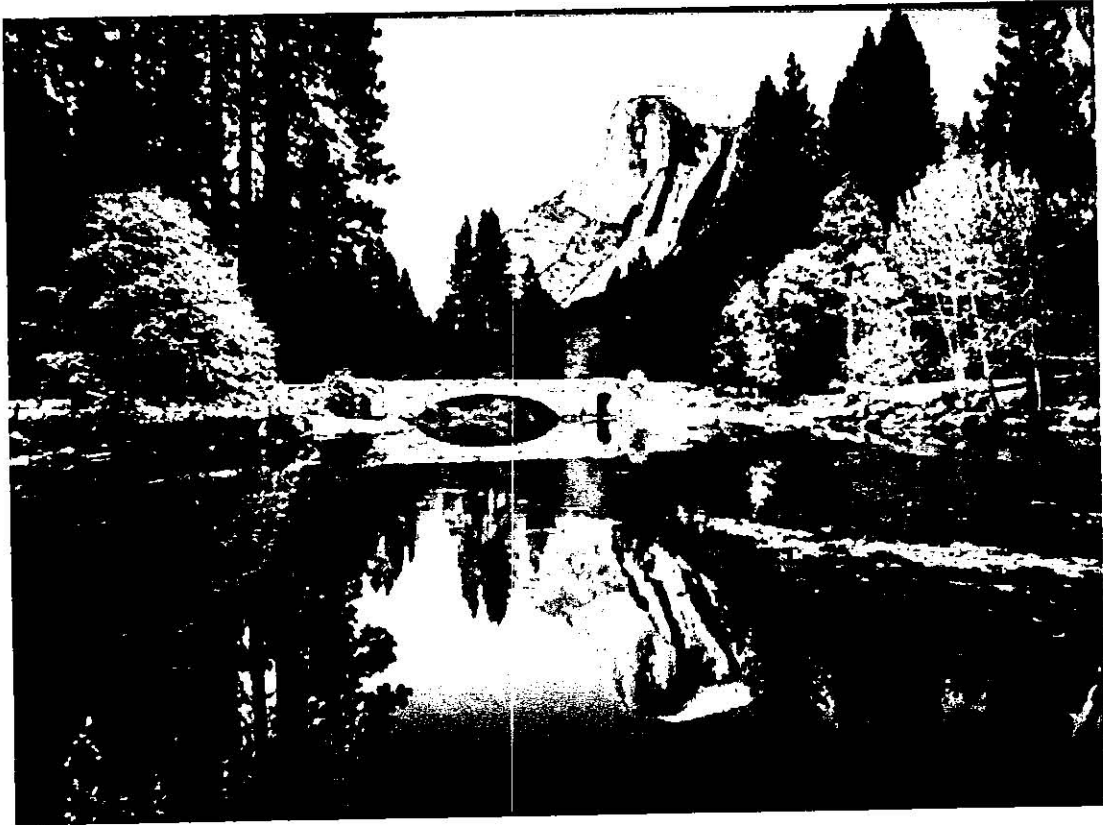


Figure (3.13): The Stego Image

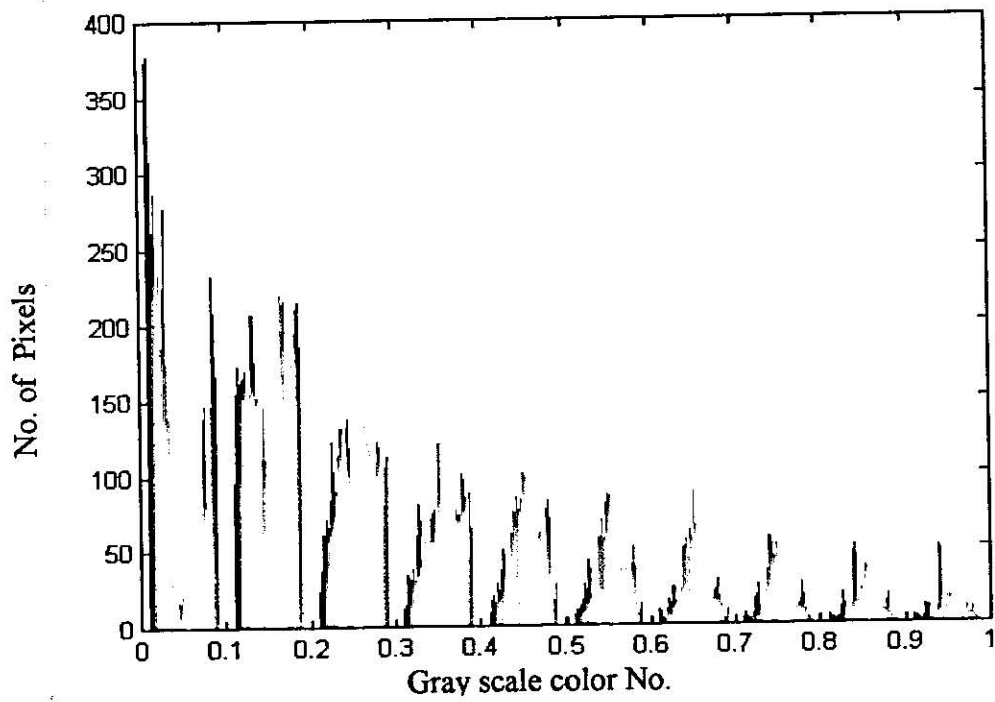


Figure (3.14):The Stego Histogram

We can distinguish the distribution of embedded image over cover image clearly in Figure (3.15)

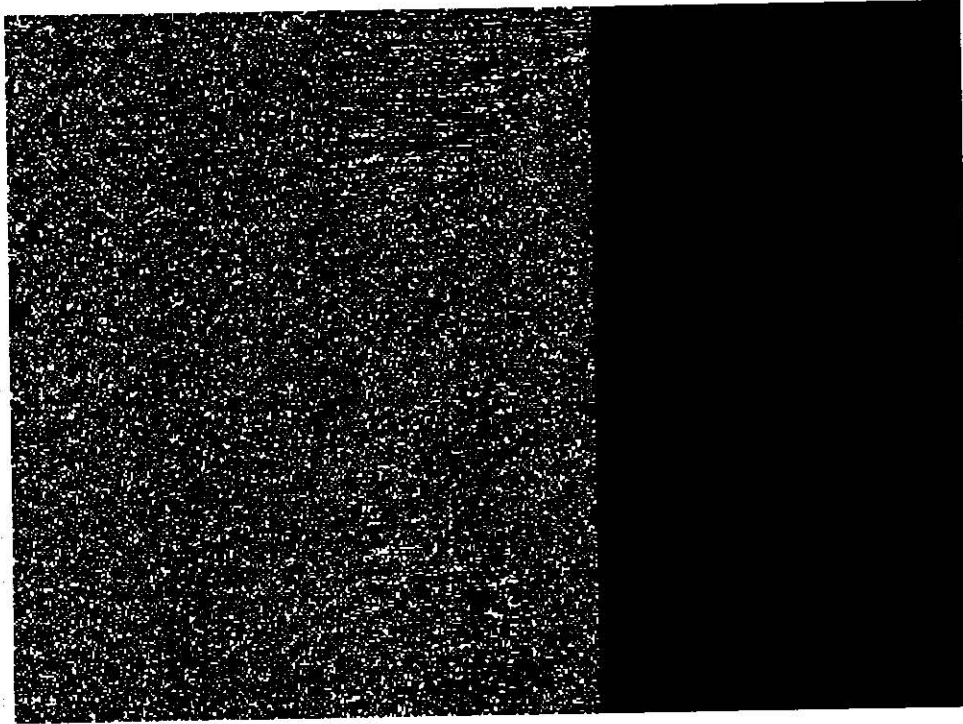


Figure (3.15): The Distribution of Embedded Image Over Cover Image

To extract the embedded image from stego-image, the extractor need to know the stego key (short key).

The stego key =

0	165	0	0	00	2	0
---	-----	---	---	----	---	---

- The first two byte represent the size of INP position file.
- The third and forth byte represent the size of OUTP position file

- The fifth byte represents the type of the position file, in this case the (00) denote to the INP file position.
- The sixth byte represents the jump value
- The last byte represents the gap

From this information the process of extraction starts to extract the embedded image and the long key.

Example 2 :

A gray scale embedded image size = 100 X 100 pixels is as shown in Figure (3.16).

The gray scale cover image size = 640 X 480 pixels is as shown in Figure (3.18)

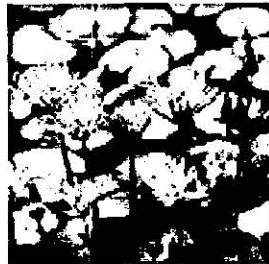


Figure (3.16): The Embedded Image

The histogram of the embedded image is shown in figure (3.17)

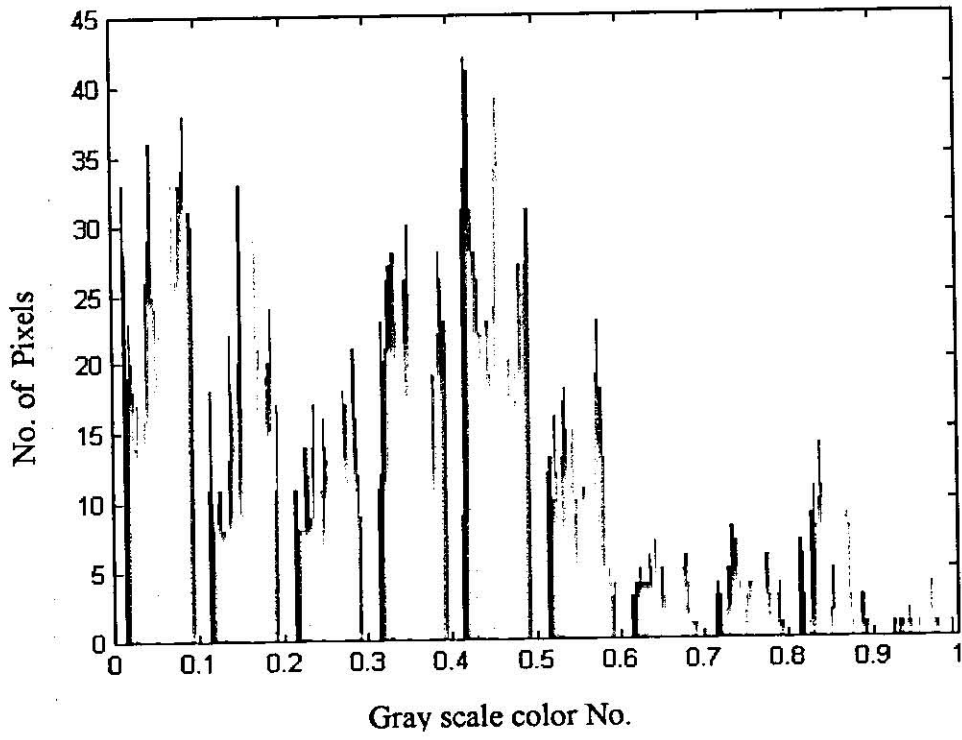


Figure (3.17): The Embedded Image Histogram



Figure (3.18): The Cover1 Image

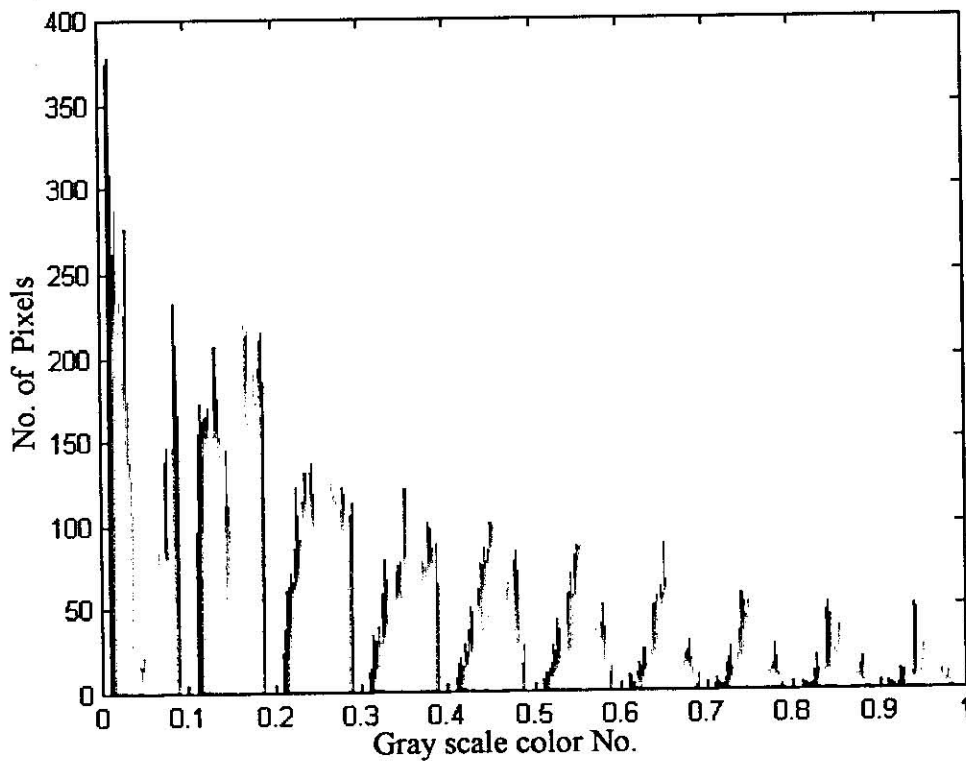


Figure (3.19): The Cover Image Histogram

The media test stage will be applied, the two tests are passed. The equation coefficients are illustrated below:

$N_{EW}=100$, $N_{EH}=100$, INH palette size = 765, $OUTH$ palette size = 765, $jump=20000$, $gap = 256$

So, (the equation result= $201,788$) < (cover width X cover height = $307,206$)

The second test is passed since the number $H(\text{cover image})$ that is greater than $H(\text{embedded image}) = 156$ which is greater than 150.

The long key is generated (INH) and (OUTH) Tables (3.5) and (3.6) show the color values of embedded image.

Table (3.5) : Long Key (INH)

7	60	81	39	41	39	48	47	57	77
89	78	74	43	23	13	12	11	13	9
5	11	32	51	70	35	28	35	37	44
58	71	114	88	50	33	30	16	12	10
7	10	8	17	39	46	78	36	25	35
34	41	68	82	126	100	59	34	28	14
13	7	12	4	15	16	115	56	90	48
41	31	54	43	48	91	73	68	51	43
17	18	8	7	11	13	5	14	245	37
33	30	28	42	48	68	76	112	65	56
32	29	14	3	9	7	9	11	17	420
77	50	36	45	39	50	67	96	94	72
55	42	22	16	11	7	8	9	10	21
54	45	43	29	38	48	46	60	111	98
74	49	21	23	11	6	13	8	7	8
52	53	46	30	36	48	51	53	111	108
70	51	28	26	8	12	7	9	7	12
60	50	38	34	49	26	55	88	123	94
75	43	31	21	11	7	7	7	5	13
44	27	29	41	37	52	76	103	101	62
29	28	17	13	4	6	12	12	12	0
107	57	350	17	128	11	85	72	58	0

Table (3.6) : Long Key (OUTH)

7	12	2	4	2	1
1	13	4	4	1	0
7	12	4	1	1	0
23	10	4	3	1	0
22	10	4	1	1	0

The abstraction search algorithm is then applied, and the result is the position file. Tables (3.7) and (3.8) illustrate sample of these files.

Table (3.7) : The INP Position File of Stego-Image

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	2278	3800	5732	5857	7116	8097	9201	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	2076	2061	2300	2728	2917	3026	3265	3456	3655	3900
8	128	157	1093	1353	1453	1897	1979	2082	2666	3400
9	29	57	131	245	303	794	893	1042	1242	1261
10	23	102	145	146	197	198	199	205	226	446
11	44	48	51	54	95	96	97	98	100	124
12	2	3	24	49	50	55	56	83	84	99
13	4	5	25	28	53	104	123	154	324	348
14	27	30	203	248	345	346	350	394	454	455
15	45	249	396	425	655	954	982	1142	1394	1553
16	47	250	251	253	351	354	445	503	505	507
17	1	22	153	349	1297	1391	1805	2981	3057	3588
18	195	604	706	1406	1505	2199	3220	3260	3598	3684
19	397	654	880	941	974	1054	1253	1389	1677	2224
20	202	353	804	2201	2666	3222	3260	3892	4030	4193
21	125	693	1641	2095	3159	3210	3218	3279	3387	3519
22	975	1279	1495	2451	2798	3059	3225	3585	3790	3883
23	231	232	1288	1455	1578	1882	2195	2698	2966	3733
24	225	504	1393	2367	2551	2717	2960	3063	3277	3278
25	257	325	352	398	1392	1496	1996	1997	2181	2370
26	855	940	1141	1895	2053	2098	2157	2263	2264	2295
27	293	608	653	1139	1206	1289	1290	1995	2156	2301
28	292	524	645	1041	1241	2024	2262	2457	2998	3077
29	9	26	126	294	399	424	452	753	754	1183
30	58	187	223	875	1555	1654	2004	2158	2828	3065
31	52	252	308	553	603	1292	1896	2052	2957	3188
32	79	178	545	1154	1341	2694	2794	3093	3283	3546
33	493	844	1795	2499	2521	2698	2718	3066	3067	3095
34	106	152	206	400	2138	2399	3092	3096	3282	3291
35	6	309	593	707	1040	2369	2767	3097	3213	3288
36	186	295	393	453	874	904	976	2383	2566	2690
37	109	188	209	1039	1184	1504	2265	2789	3178	3447
38	1055	1143	1355	1489	2159	2365	2891	2958	3199	3392
39	83	546	605	756	955	1185	1240	1408	1545	1546
40	82	194	277	299	970	2055	3079	3166	3189	3555
41	300	475	526	873	1083	1694	1950	2266	2800	2852
42	122	278	296	297	302	953	1043	1304	1753	2256
43	110	323	872	939	1340	1409	2198	2299	2960	3436
44	298	871	1004	1140	1404	1478	1680	2160	2279	2617
45	81	193	575	607	675	2599	2987	3167	4231	4336

Table (3.8) : The OUPF Position File of Stego-Image

	1	2	3	4	5	6	7	8	9	10
1	1442	1542	1633	1642	2327	2527	6058	0	0	0
2	9741	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	1342	2283	2382	2482	6687	7298	8651	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	744	942	1175	2223	4183	4414	4879	5630	6159	7200
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0
222	0	0	0	0	0	0	0	0	0	0
223	461	1354	1847	2125	2129	2149	2943	3041	3043	3044
224	0	0	0	0	0	0	0	0	0	0
225	366	668	1926	1929	2128	2132	2330	2530	6062	6534
226	367	368	1747	2270	2371	3037	6367	6735	6829	6927
227	62	269	667	2232	2234	3374	6533	6928	6938	7134
228	1547	2532	2845	6767	6839	6936	7034	7233	7436	7536
229	1647	5444	6566	6734	6737	6830	6837	7074	7229	7374
230	767	7035	0	0	0	0	0	0	0	0
231	6736	6831	6832	6937	0	0	0	0	0	0
232	1447	6733	6929	7735	0	0	0	0	0	0
233	3270	3437	3638	7332	0	0	0	0	0	0
234	6627	6628	6629	6833	0	0	0	0	0	0
235	2034	4946	6864	7036	0	0	0	0	0	0
236	1177	2627	3337	6626	0	0	0	0	0	0
237	6567	0	0	0	0	0	0	0	0	0
238	1378	3471	7234	0	0	0	0	0	0	0
239	2433	0	0	0	0	0	0	0	0	0
240	5544	6930	0	0	0	0	0	0	0	0
241	6371	0	0	0	0	0	0	0	0	0
242	0	0	0	0	0	0	0	0	0	0
243	0	0	0	0	0	0	0	0	0	0
244	7834	0	0	0	0	0	0	0	0	0
245	0	0	0	0	0	0	0	0	0	0
246	0	0	0	0	0	0	0	0	0	0
247	0	0	0	0	0	0	0	0	0	0
248	0	0	0	0	0	0	0	0	0	0
249	0	0	0	0	0	0	0	0	0	0
250	0	0	0	0	0	0	0	0	0	0
251	0	0	0	0	0	0	0	0	0	0
252	7474	0	0	0	0	0	0	0	0	0

After the position file is generated, the next stage is the jump and hide stage, The jump and hide Algorithm try to embed the position files (INP and OUTP) and long key (INH and OUTH) into the cover image, so the final result is stego-image. Figure (3.20) and Figure (3.21) illustrate the stego-image and its histogram, respectively



Figure (3.20): The Stego-Image

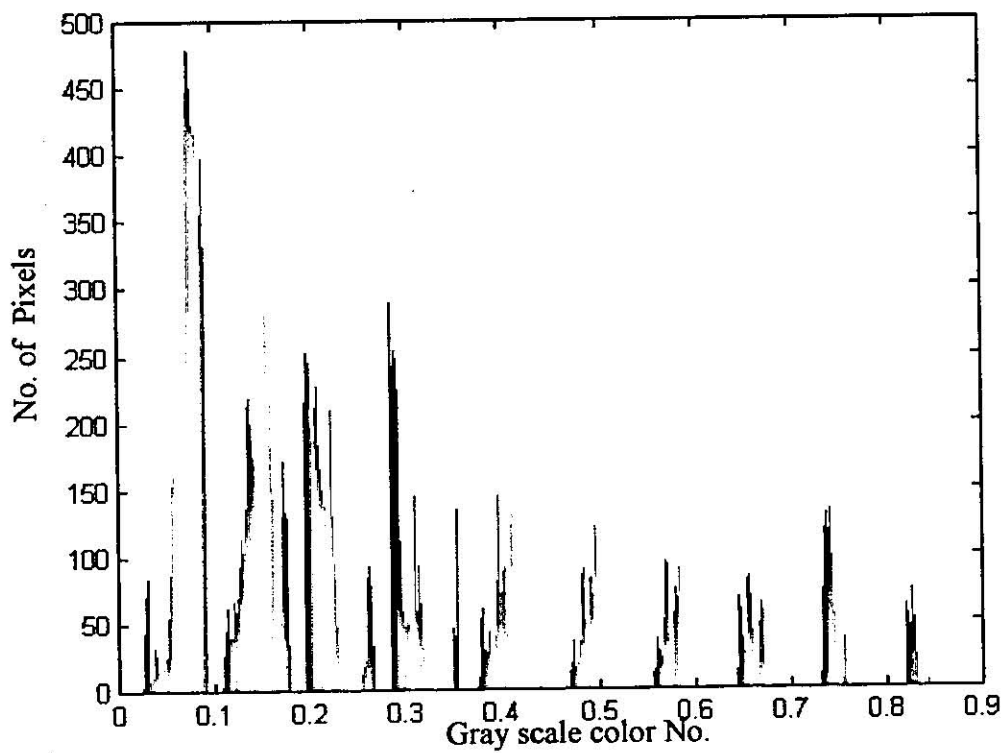


Figure (3.21): The Stego Histogram

We can distinguish the distribution of embedded image over cover image clearly in the Figure (3.22)

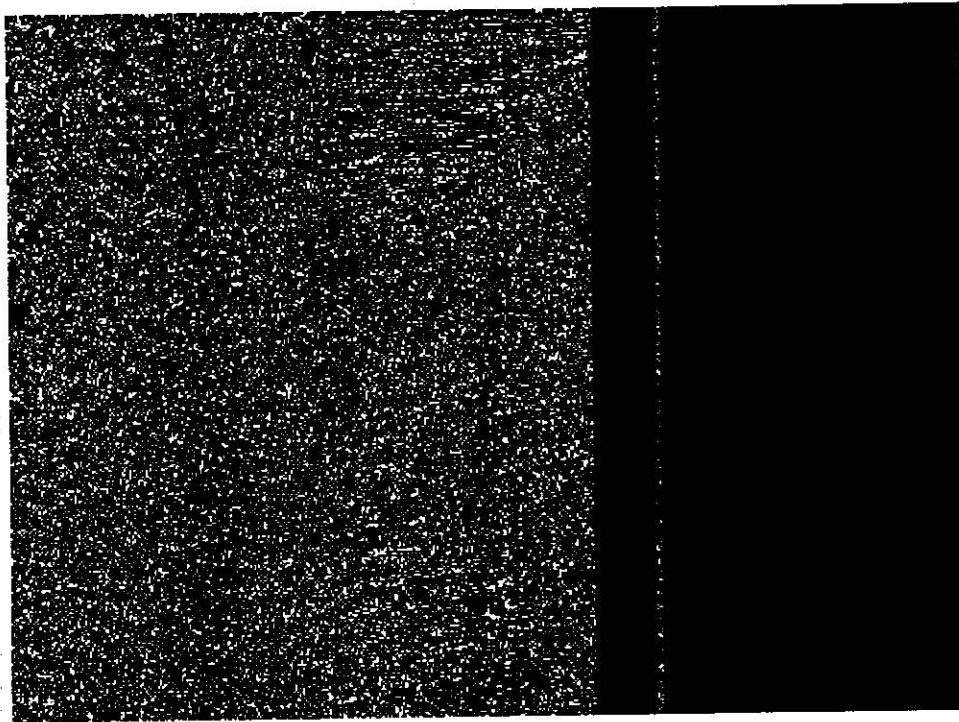


Figure (3.22): The Distribution of Embedded Image Over The Cover Image

To extract the embedded image from stego-image, the extractor need to know the stego key (short key).

The stego key =

0	218	0	26	11	2	256
---	-----	---	----	----	---	-----

- The first two byte represent the size of INP position file.
- The third and forth byte represent size of OUTP position file.

- The fifth byte represents the type of the position files, in this case the (11) denote to the INP and OUTP position files.
- The sixth byte represents the jump value
- The last byte represents the gap

From this information, the process of extraction starts to extract the embedded image and the long key.

3.9 GISS Security

As mentioned in Section (2.6), the security of a steganography system is measured using two methods of measuring. The first is the similarity function and the second one is entropy test function.

In this section the results of these tests are applied to show how fair GISS is secure.

First the similarity function is calculated between the pixels of cover-image and stego-image by using the correlation function.

The second test is the entropy test, the probability distribution of the cover-image and stego-image is calculated by the following equation.

$$P(r_k) = n_k/n.$$

where

r_k is the k th gray or color level.

n_k is the number of pixels in the gray or color image

n is the total number of pixels in the image

$k= 0, 1,2,\dots, 255$

$P(r_k)$ gives an estimate of the probability of occurrence of gray or color level r_k .

After calculating both probabilities the equation (3.2) is applied to get the entropy between the two images.

$$U(P_C \| P_S) = \sum_{n=0}^{255} P_C(n) \log_2 \frac{P_C(n)}{P_S(n)}$$

Below are the results of these two tests that were applied to the four examples given below.

Example 1:

- 1- Applying the Correlation test. The correlation between the cover-image shown in Figure (3.9) and stego-image shown in Figure (3.11) = 0.989839008
- 2- Applying the entropy test, the probability distribution of both the cover-image Figure (3.9) and the stego-image Figure (3.11) is = 0.017406319.

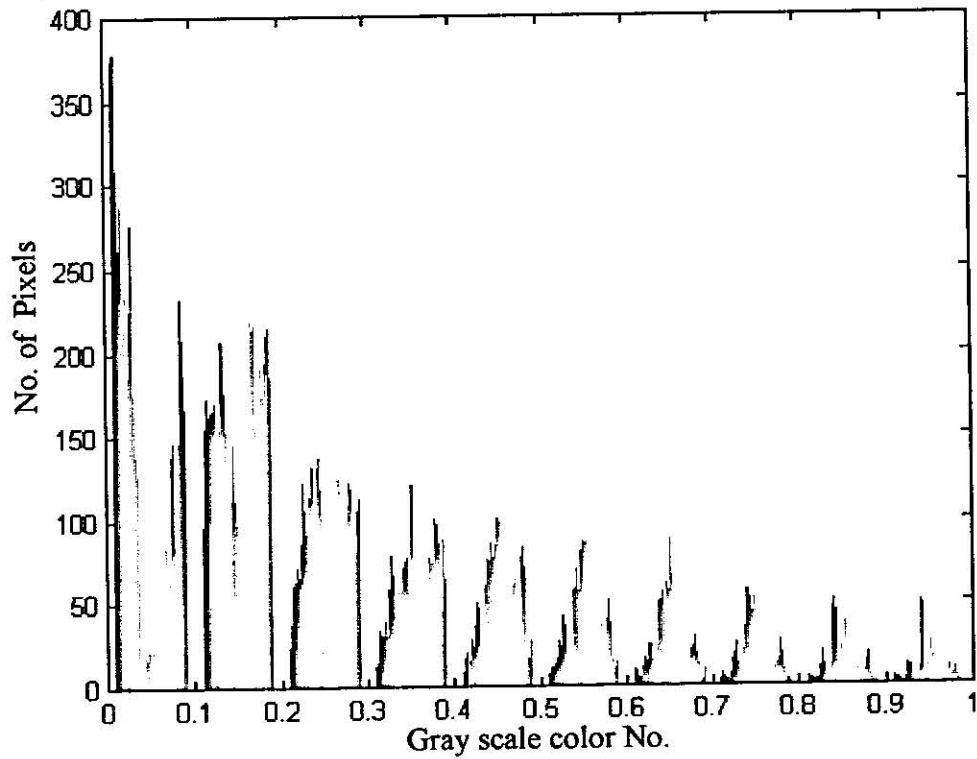


Figure (3.23): cover Image Probability Distribution of Figure (3.9)

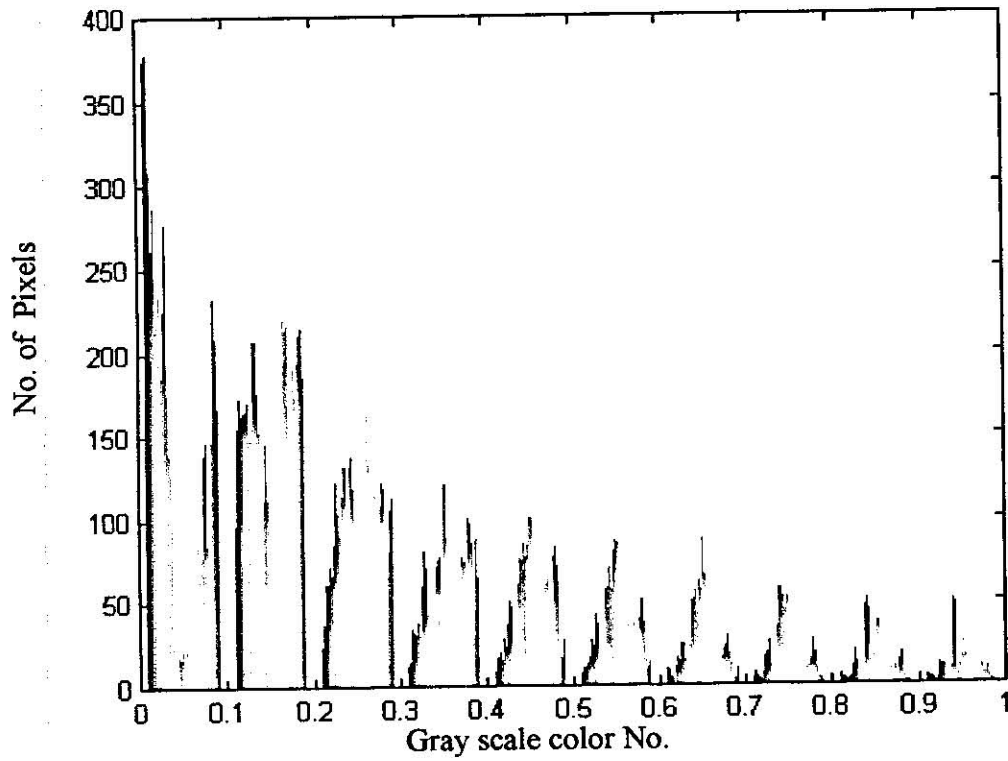


Figure (3. 24):The Stego Image Probability Distribution of Figure

(3.11)

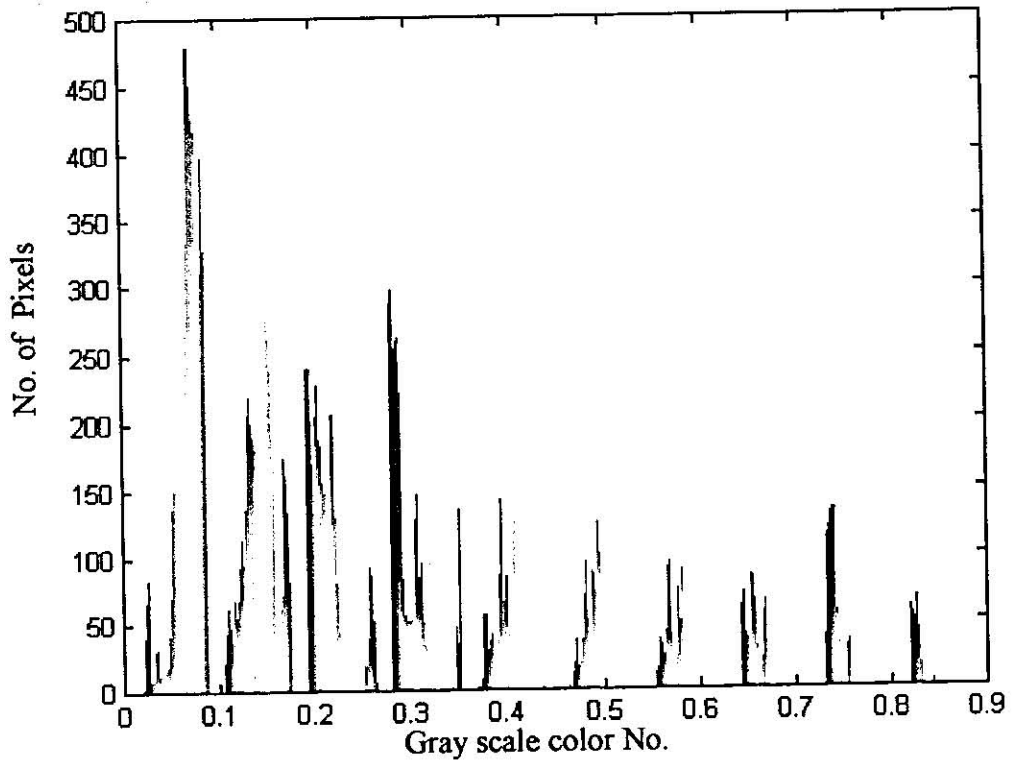


Figure (3. 26):The Stego-Image Probability Distribution of Figure (3.18)

3.10 Comparison between (Gray Image Stego. System) and (Substitution system) (LSB technique).

Example 1 comparison:

System	Similarity	Entropy
GISS	0.989839008	0.017406319
LSB	0.981339008	0.019300311

Example 2 comparison:

System	Similarity	Entropy
GISS	0.987164850	0.061254874
LSB	0.982213740	0.068479515

The results mentioned above show the differences between the proposed system (GISS) and the LSB method. These results clarify the fact that the proposed system is more accurate than LSB method.

Discussion and Conclusion

4.1 Discussion

Hiding an image in another presents a challenge, mainly because of the size of the data of the embedded image. However, in this study it was found that the result obtained overcomes this problem.

The image file has a big size of empty bits inside bytes to be utilized. Images have sizes, a lower size image (100 x 100) can easily be embedded in (640x480) if it is compared with other files that cannot be resized (Constant or minimal constant) like any office file, that does not make the hiding invisible to recognize. Following are several discussion points regarding the **GISS**.

1-By applying the two tests (similarity and entropy test) on the **GISS** it proved that the system is secured since the correlation between the cover and stego images is nearly equal to 1 and the entropy is of the order 10^{-3} (i.e. two experiments results 0.989 and 0.987)

As mentioned in section (2.8), there are four main security points, which determine whether a steganography algorithm is a secured one or not. Three of these points are satisfied in **GISS**, the message in image is embedded using public algorithm, with a secret key (stego-key), and only the holder of this key can extract and prove the existence of the embedded image.

2-For the three attacks mentioned in (section 2.11), the first attack (passive attacker) will not detect whether the existed is a stego-image or not. The **GISS** like most of other existing steganography systems is not robust against modification of the data, or have limited robustness because the main goal of steganography is to avoid drawing suspicion and if so then the goal will be defeated. Robustness is used mostly in watermarking since the attacker will know in advance that there is something hidden as a watermark inside the product and he tries to remove it. As a result the **GISS** fails against active or malicious attackers, since the first one adds noise to the stego-image, which could cause a miss-extracting of the embedded-image. The second attack modifies the semantic of the stego-image, which alters the description of the stego-image, and consequently leads to miss-extracting of the embedded-image.

Conclusions

Using cryptography will support the security immunity of the system without causing any additional distortion.

The system proved to be a good system used to hide a gray image in gray images by the cluttering the position of pixel of embedded image into cover image location of information. In the following are some points concluded from this study.

- 1- The **GISS** can be defined as a secret key steganography since it fulfills definition 2.2 of the secret key steganography. It shares a secret key between the sender and the receiver. In this system there is no need for the knowledge of original cover in the extracting process.
- 2- If the stego-image draws suspicion, then the strength of **GISS** lies in the complexity of the obtaining of the 7-bytes key set, since at the worst case it is 2^{56} . These are obtained as follows:
 - To obtain the embedded width and height (2 bytes), it needs 2^{16} possibilities.
 - To obtain the cover width height (2 bytes), it needs 2^{16} possibilities.
 - To obtain the positions files (1 byte), it needs 2^8 possibilities.
 - To obtain the jump (1 byte), it needs 2^8 possibilities.
 - To obtain the gap (1 byte), it needs 2^8 possibilities.
- 3- Embedded the position of pixel increase the complexity of obtaining the embedded image over stego-image, so the **GISS** is powerful system to use this technique.

4.3 Suggestions for Future Work

We propose to embed a color image inside a gray or another color image, but this requires sending the palette of the embedded color image which result in a large key. Thus if a method of data compression is used in conjunction with the steganography it may lead to good results. Increasing the size of embedded image over the cover image needs to obtain the new methods that satisfies good hidden metrology. Some of images that are noised by some of unexpected information may be used as a cover media, so we can use special technique to hide any embedded image in this noised information by an ability method that depends on an image processing filter.

Bearing in mind the above, the following can be researched:

1. Hiding in compression video files (.AVI, MPEG, and .DAT).
2. Hiding in GIF Animation files, which are widely used in design for web pages.
3. Hiding in compression audio files like the (MP3), which are widely used in the Internet.

References

1. Anderson,R.,1996."***Stretching the Limits of Steganography***", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, , pp. 39-48.
2. Anderson,R. J. Needham, R M. and Shamir, A.,1998. "***The Steganography File System***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 73-82.
3. Aura,T.,1996. "***Practical Invisibility in Digital Communication***", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, pp. 39-48.
4. Baase,S., 1988. "***Computer Algorithms***", Addison-Wesley Publishing Company.
5. Bender, W., Gruhl, D., Morimoto N., 1996. "***Techniques for Data Hiding***", IBM Systems Journal, Vol. 35, No. 3&4.
6. Brase, C., Henry and Brase Carrinne Pellillo,1982. "***College Algebra***", D.C. Health and Company, USA.
7. Cachin,C., 1998. "***An Information-Theoretic Model for Steganography***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 306-318.
8. Craver, S., 1998. "***On Public-Key Steganography in the Presence of an Activer Warden***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 355-368.
9. Davern, P. and Scott M.,1996. "***Fractal Based Image Steganography***", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, pp. 279-294.
10. Denning, D.,1983. "***Cryptography and Data Security***", Addison-Wesley Publishing Company.
11. Etting, J., 1998. "***Steganalysis and Game Equilibria***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 319-328.

12. Franz, E., Jerichow, A., 1996. "**Computer Based Steganography**", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, pp. 7-22.
13. Gomes, J., and Velho, L., 1997. "**Image Processing for Computer Graphics**", Prentice Hall, USA, pp 60-63.
14. Gruhla, D., and Bender, W., 1998. "**Information Hiding to Foil the Casual Counterfeiter**" Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 1-15.
15. Johnson, N. F., 1999. "**Steganography**" , An internet survey, <http://ise.gmu.edu/~csis>.
16. Johnson, N. F. and Jajodia, S., 1998. "**Steganalysis of Image Created Using Current Steganography Software**", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 273-289.
17. Johnson, N., Duric, Z., 1999. "**A Role for Digital Watermarking in Electronic Commerce**", ACM Computer Surveys, <http://www.acm.com>.
18. Johnson, N., 1999. "**An Introduction to Watermark Recovery from Images**", SANS Intrusion Detection and Response, Proceedings, San-Diego, <http://www.nitv.net>.
19. Kahn, D., 1996. "**The History of Steganography**", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, pp. 1-5.
20. Katzenbeisser, S. and Petitcolas, F. 2000 "**Information Hiding Techniques for Steganography and Digital Watermarking**", Artech House, USA.
21. Konheim, A., 1981. "**Cryptography a Prime**", Wiley-Interscience publication.
22. Lacy, J., 1998. "**Intellectual Property Protection Systems and Digital Watermarking**", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 158-168.
23. Marvel, L., 1998 "**Reliable Blind Information Hiding for Images**", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 48-61.

24. Neubauer, C., 1998. "***Continuous Steganographic Data Transmission Using Uncompressed Audio***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 208-217.
25. Pfitzmann, B., 1996. "***Information Hiding Terminology***", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, pp. 347-350.
26. Rimell, J., 1997. "***Data Hiding Inside TIFF images***", John's College, Cambridge, England.
27. Smith, J., and Comiskey, B., 1996. "***Modulation and Information Hiding in Images***", Information Hiding: First International Workshop, Proceedings, vol. 1174 of Lecture Notes in Computer Science, Springer, pp. 207-226.
28. Spiegel, R., 1961. "***Theory and Problems of Statistics***", Schaum's Outline Series, McGraw-Hill International Book Company.
29. Westfeld, A., 1998. "***Steganography in a Video Conferencing System***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 32-47.
30. Zollner, J., Federrath, H., 1998. "***Modeling the security of Steganography System***", Information Hiding: Second International Workshop, Proceedings, vol. 1525 of Lecture Notes in Computer Science, Springer, pp. 344-354.

Image Processing Principles

Image Processing

Image processing is computer imaging where the application involves a human being in the visual loop. In other words, the images are to be examined and acted upon by people.

Image Enhancement

Image enhancement techniques are used to emphasize and sharpen image features for display and analysis. Image enhancement is the process of applying these techniques to facilitate the development of a solution to a computer-imaging problem. Consequently the enhancement methods are application specific and are often develop empirically.

Image Noise:

What is noise? Noise is any undesirable signal. Noise is everywhere and thus we have to learn to live with it. Noise gets introduced into the data via any electrical system used for storage, transmission, and/or processing. In addition, nature will always plays a "noisy" trick or two with the data under observation.

When encountering an image corrupted with noise you will want to improve its appearance for a specific application. The techniques applied are application-oriented. Also, the different procedures are related to the types of noise introduced to the image. Some examples of noise are : Gaussian or White, Rayleigh, Shot or Impulse, periodic, sinusoidal or coherent, uncorrelated, granular.



Image with (salt & pepper) noise



Original image

because they believed that an improvement of that magnitude would be visible.

Some definitions of PSNR use $2552/\text{MSE}$ rather than $255/\text{RMSE}$. Either formulation will work because we are interested in the relative comparison, not the absolute values. For our assignments we will use the definition given above.

The other important technique for displaying errors is to construct an error image which shows the pixel-by-pixel errors. The simplest computation of this image is to create an image by taking the difference between the reconstructed and original pixels. These images are hard to see because zero difference is black and most errors are small numbers, which are shades of black. The typical constructions of the error image multiples the difference by a constant to increase the visible difference and translate the entire image to a gray level. The computation is

$$E(i,j) = 2 [f(i,j) - F(i,j)] + 128$$

You can adjust the constant (2) or the translation (128) to change the image. Some people use white (255) to signify no error and difference from white as an error, which means that darker pixels are bigger errors. [A.N. Netravali and B.G. Haskell, 1995].

Steganography in Images

Information can be hidden in many different ways in images. Straight message insertion can be done, which will simply encode every bit of information in the image. More complex encoding can be done to embed the message only in "noisy" areas of the image that will attract less attention. The message may also be scattered randomly throughout the cover image

The most common approaches to information hiding in images are:

1. Least significant bit (LSB) insertion
2. Masking and filtering techniques
3. Algorithms and transformations

Each of these can be applied to various images, with varying degrees of success. Each of them suffers to varying degrees from operations performed on images, such as cropping, or resolution decrementing, or decreases in the color depth.

Least Significant Bit Insertion (LSB)

The least significant bit insertion method is probably the most well known image steganography technique. It is a common, simple approach to embedding information in a graphical image file. Unfortunately, it is extremely vulnerable to attacks, such as image manipulation. A simple conversion from a GIF or BMP format to a lossy compression format such as JPEG can destroy the hidden information in the image.

When applying LSB techniques to each byte of a 24-bit image, three bits can be encoded into each pixel, as each pixel is represented by three bytes. Any changes in the pixel bits will be indiscernible to the human eye. For example, the letter A can be hidden in three pixels. Assume the original three pixels are represented by the three 24-bit words below:

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

The binary value for the letter A is (10000011). Inserting the binary value of A into the three pixels, starting from the top left byte, would result in:

(0010011**1** 11101000**0** 11001000**0**) (0010011**0** 11001000**0** 11101000**0**)

(11001000**0** 0010011**1** 11101001)

The emphasized bits are the only bits that actually changed. The main advantage of LSB insertion is that data can be hidden in the least and second least bits and still the human eye would be unable to notice it. When using LSB techniques on 8-bit images, more care needs to be taken, as 8-bit formats are not as forgiving to data changes as 24-bit formats are. Care needs to be taken in the selection of the cover image, so that changes to the data will not be visible in the stego-image. Commonly known images, such as famous paintings, like the *Mona Lisa* should be avoided. In fact, a simple picture of your dog would be quite sufficient.

When modifying the LSB bits in 8-bit images, the pointers to entries in the palette are changed. It is important to remember that a change of even one bit could mean the difference between a shade of red and a shade of blue. Such a change would be immediately noticeable on the displayed image, and is thus unacceptable. For this reason, data-hiding experts recommend using Grey-scale palettes, where the differences between shades is not as pronounced. Alternatively, images consisting mostly of one color, such as the so-called Renoir palette, named because it comes from a 256-color version of Renoir's "Le Moulin de la Galette"[Bender W., et al, 1996].

2.2.1.1 Masking and Filtering

Masking and filtering techniques hide information by marking an image in a manner similar to paper watermarks. Because watermarking techniques are more integrated into the image, they may be applied without fear of image destruction from lossy compression. By covering, or masking a faint but perceptible signal with another to make the first non-perceptible, we exploit the fact that the human visual system cannot detect slight changes in certain temporal domains of the image.

Masking techniques are more suitable for use in lossy JPEG images than LSB insertion because of their relative immunity to image operations such as compression and cropping.

552062

2.2.1.2 Algorithms and Transformations

JPEG images use the discrete cosine transform (DCT) to achieve compression. DCT is a lossy compression transform, because the cosine values cannot be calculated precisely, and rounding errors may be introduced. Variances between the original data and the recovered data depends on the values and methods used to calculate the DCT.

Images can also be processed using Fourier transformation and wavelet transformation. Other properties such as luminance can also be utilized. The

HVS has a very low sensitivity to small changes in luminance, being able to discern changes of no less than one part in thirty for random patterns. This figure goes up to one part in 240 for uniform regions of an image.

The Patchwork method explored by Bender et. al. is based on a pseudorandom, statistical process that takes advantage of the human weaknesses to luminance variation. Using *redundant pattern encoding* to repeatedly scatter hidden information throughout the cover image, Patchwork can hide a reasonably small message many times in a image. In the Patchwork method, n pairs of image points (a,b) are randomly chosen. The brightness of a is decreased by one and the brightness of b is increased by one. For a labeled image, the expected value of the sum of the differences of the n pairs of points is then $2n$. Bender shows that after JPEG compression, with the quality factor set to 75, the message can still be decoded with an %85.

This algorithm is more robust to image processing such as cropping and rotating, but at the cost of message size. Techniques such as Patchwork are ideal for watermarking of images. Even if the image is cropped, there is a good probability that the watermark will still be readable. Other suggested methods, also attempt to mark labels into the images by altering the brightness of pixel blocks of the image by a selected value k . The value of k depends on the lower quality JPEG compression version of the labelled block. This method is fairly resistant to JPEG compression, depending on the size of the pixel blocks used, and offers low visibility of the label. Unfortunately, it is not very suitable to real-time applications. Other techniques encrypt and scatter the hidden throughout the image in some pre-determined manner. It is assumed that even if the message bits are extracted, they will be useless without the algorithm and stego-key to decode them. Although such techniques assist the protection against hidden message extraction, they are not immune to destruction of the hidden message through image manipulation.

الملخص

إخفاء الصور بالتشفير

إعداد

عبد الرحمن عمر ال خليفة

المشرف

الدكتور أحمد الجابر

يمكن تعريف STEGANOGRAPHY على انه العلم الذي يختص بإخفاء رسالة ذات طابع محدد بأخرى تتشابه أو تختلف عن مثيلتها, وهذا العلم يختلف طبعا عن علم التشفير CRYPTOGRAPHY والذي يختص بتشفير الرسالة وليس إخفاءها باستخدام تقنيات محددة وكثيرة.

اعتمدنا في هذه الأطروحة على إنشاء خو
ارزميات وطرق جديدة للإخفاء بعد دراسة الطرق المتبعة السابقة وكانت النتائج جيدة مقارنة بتلك الطرق حيث اظهر مقياس التناظر SIMILARITY وعامل ENTROPY قيما تقترب كثيرا من القيم العليا للقياس, وهذه النتائج تجعل من الخوارزميات المنشئة لها الأفضلية بالتطبيق والاستخدام.

أن نظام (GRAY IMAGE STEGANOGRAPHY SYSTEM) GISS المقترح هو نظام يقوم على أساس إخفاء صورة ذات تدرج رصاصي اللون (GRAY SCALE COLOR IMAGE) بأخرى أكبر حجما منها لها نفس الخصائص لذا يعتبر هذا النظام من طرق الكتابة المخفية السرية حيث تكون الخوارزمية معلنة والمفتاح هو سري وغير معلن و تتداوله جهتين المستفيدتين (المرسله و المستقبلة)

يتضمن النظام خمسة مراحل وهي مرحلة الفحص وتشخيص خصائص الصورة (MEDIA-TEST STAGE), مرحلة فصل خصائص الصور

(SPLITTING STAGE), مرحلة تعيين مواقع النقاط الصورية (PIXELS
(PICTURE ELEMENT), مرحلة إنشاء ملف المواقع (POSITION AND
(EVALUATING STAGE), مرحلة التضمين (JUMP AND HIDE
(EXTRACTING STAGE) و مرحلة فصل الصورة المضمنة (STAGE).

قد بني GISS ليكون سهل الاستخدام من قبل المستخدم حيث يقوم
المستخدم بتزويد النظام بالصورة الأصلية والصورة المراد تضمينها وإخفائها
ليقوم النظام بعملية التضمين لينتج صورة مهيأة للإرسال يطلق عليها
STEGOIMAGE. وبعد وصولها للمستقبل المحدد يقوم المستخدم الآخر
بإخراج الصورة المضمنة من صورة STEGOIMAGE.

عند مراجعة خصائص STEGANOGRAPHY, نرى بأن GISS نظام قابل
للتطبيق والعمل وفق تلك الشروط والخصائص