LSB EMBEDDING METHODS
ON STILL IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

CEM OLCAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

APRIL 2010

Title of the Thesis : **LSB Embedding Methods on Still Images**

Submitted by **Cem OLCAY**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Mehmet R. TOLUN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

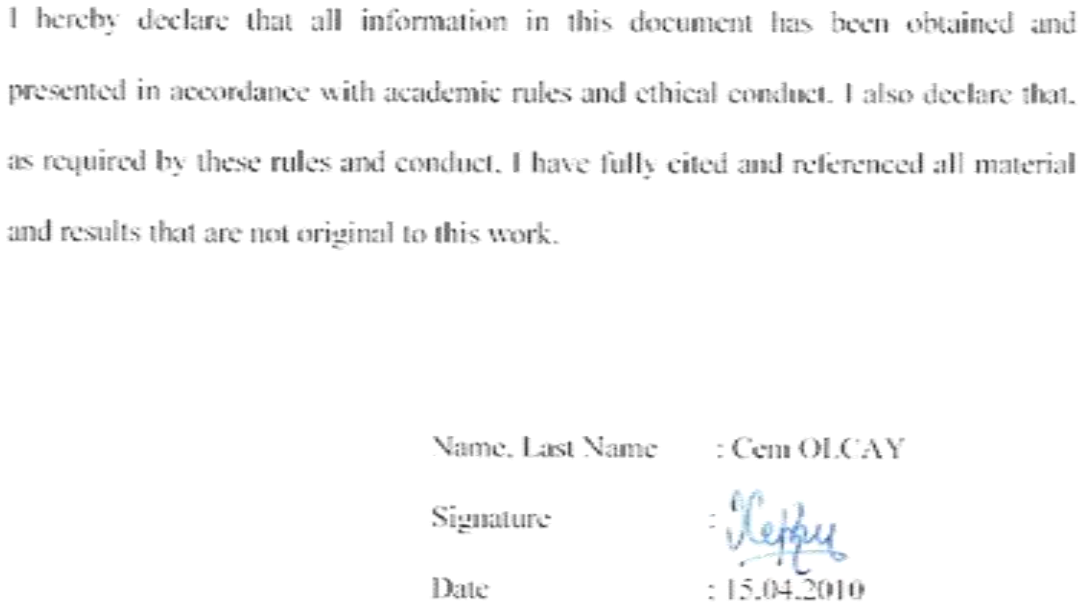
Dr. Nurdan SARAN
Supervisor

Examination Date : April 15, 2010

Examining Committee Members

Y.Doç.Dr. Reza HASSANPOUR	(Çankaya Univ.)
Dr. Nurdan SARAN	(Çankaya Univ.)
Y.Doç.Dr. Zülfükar SAYGI	(TOBB ETU)

STATEMENT OF NON-PLAGIARISM



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Cem OLCAY

Signature :

Date : 15.04.2010

ABSTRACT

LEAST SIGNIFICANT BIT EMBEDDING METHODS ON STILL IMAGES

OLCAY, Cem

M.Sc., Department of Computer Engineering

Supervisor : Dr. Nurdan SARAN

April 2010, 74 pages

Steganography is the science of hiding message or file in a cover media which can be text, image, video, or other digital media formats. Digital images, the most published media on internet, may be divided into two groups as lossless compressed/uncompressed and lossy compressed. One of the most preferred steganographic methods is setting the last bit of pixel in still images according to bits of information that will be hidden. Least Significant Bit (LSB) methods are most suitable for uncompressed/lossless compressed images. In this study, LSB embedding methods such as LSB Replacement, Matching, Chan’s Method, Modulus Function Technique, 2/3 Embedding Efficiency, and Matrix Embedding using Hamming Codes are examined. Moreover most used steganalysis methods such as Visual Attack, Chi-Square Test, Raw Quick Pairs (RQP), and Regular and Singular

Groups (RS) are briefly described. Effectiveness of these embedding methods is compared against steganalysis methods.

Keywords: Steganography, Steganalysis, Chan’s Method, RS, Hamming Code

ÖZ

İMGELERDE EN ÖNEMSİZ BİTE GÖMME YÖNTEMLERİ

OLCAY, Cem

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez yöneticisi : Dr. Nurdan Saran

Nisan 2010, 74 sayfa

Steganografi, metin, imge, video gibi dijital medya formatlarını kullanarak mesaj veya dosya gizleme bilimidir. İnternette en çok yayımlanan medya olan dijital imgeleri sıkıştırılmamış/kayıpsız sıkıştırılmış ve kayıplı sıkıştırılmışlar olarak ikiye ayırabiliriz. İmgelerde en çok tercih edilen steganografi yöntemlerinden biri piksel değerlerinin son bitlerini saklanacak bilgiye göre ayarlamaktır. En önemsiz bite gizleme yöntemleri en çok sıkıştırılmamış ve kayıpsız sıkıştırılmış imgeler için uygundur. Bu çalışmada en önemsiz bite gizleme yöntemlerinden yerdeğiştirme, eşleştirme, modüler foksiyonu tekniği, Chan'ın yöntemi, 2/3 verimli gömme ve Hamming kodlarını kullanarak matris gömme yöntemleri açıklanmıştır. Bunlardan başka, en çok kullanılan steganaliz yöntemlerinden olan görsel ataklar, Ki kare testi, RQP ve RS analiz kısaca tanımlanmıştır. Açıklanan gömme yöntemlerinin steganaliz yöntemlerine karşı verimliliği karşılaştırılmıştır.

Anahtar Kelimeler: Steganografi, Steganaliz, Chan’ın Yöntemi, RS, Hamming Kodu

TABLE OF CONTENTS

APPENDICES:

LIST OF TABLES

TABLES PAGES

LIST OF FIGURES

FIGURES PAGES

CHAPTER 1

INTRODUCTION

Communicating secretly is always a matter of human throughout history. Different methods have been developed to hide or share information. One of them is to convert message into scramble form. If someone has the scrambled message, he/she can try to convert the message. This form of hiding information is called as cryptography. Success of cryptography methods are judged depending on different usage of them. Encryption is the process of making message unreadable, called as ciphertext, using an algorithm called as cipher, and a key between sender and receiver. Decryption is the process of reading secret message hidden in ciphertext using the cipher and the key. Other form of secretly communication is hiding message using a cover object. This message can be encrypted or original. This type of data hiding methods can be divided into two different forms called as steganography and watermarking. Watermarking is the process of hiding message into a cover object to declare its identity, in a way that is difficult to remove. Digital watermark identifies the associated rights of media during distribution and later. Watermarks are called as robust if they resists to all kind of manipulations like cropping, editing, resizing, compressing, and etc. Text or logo of copyright owner can be seen in image, video or text file, called as visible watermarking. Digital data may not be unseen to declare the ownership of media called as invisible watermarking. Another type of data hiding science is called as steganography. Steganography uses cover objects to hide message into them. Digital cover objects can be a text, an image, a video or another digital media formats. The aim of steganography may be summarized as someone should not to realize cover object carries a message. After embedding hidden data to cover object, changes in stego object (an object that carries hidden message), should not be detected by Human Visual System (HVS). Differences between steganography and watermarking may be listed as follow:

- Steganography uses any object to hide message but watermarked object carries own identity information.
- Steganographic techniques will do fewer changes on cover object to make the message unrealizable, watermarking techniques will make the watermark irremovable. Also, changes on watermarked objects should not defect the quality of media.
- Steganography techniques fail if someone relies that an object carries a hidden message. So the objectives is to make minimum changes on cover object in this way the attacker may not detect the existence in a cover object. Main case is to protect the communication channel. So, the minimum changes on cover object that can be undetectable for attacker is matter.

1.1 Outline of Thesis

In the following chapter, history of steganography, forms of steganography, how to hide message in text, image, audio, video, and other digital files are explained.

In the third chapter image file formats, coding and compression algorithms are described. Comparison methods of original and manipulated images such as Mean Squared Error (MSE), Peak Signal Ratio (PSNR), Weighted PSNR (WPSNR), Histogram analysis, are briefly described.

Data embedding methods that use LSB techniques for uncompressed/lossless compressed images such as replacement method, modulus function method, matching method, Chan’s method, and matrix embedding using Hamming codes are described in Chapter four. Also, in this chapter data embedding methods into lossy compressed image formats as jpeg and data embedding into bit planes are explained.

Chapter five is about analysis methods of still images. Visual attacks and filtering methods to find where the secret message is explained and some important statistical analysis methods; Regular and Singular group’s analysis (RS), Raw Quick Pair analysis (RQP), and Chi-Square Steganalysis methods are described.

Compression algorithms are used to minimize the message size; this reduces the defects on cover image. Some compression algorithms such as Winrar, Huffman coding are examined and their effects on steganography are illustrated on some different stego objects in Chapter six. Some LSB methods hide 1 or more bits in a pixel to have more embedding capacity. Some LSB methods need more than 1 pixel to embed 1 bit data. So, these methods may cause less change on image. Steganalysis methods need different percentage of changes on object to catch stego objects. For a fixed size image, embedding capacity of LSB methods is compared according to the resistance to the steganalysis methods on images.

CHAPTER 2

STEGANOGRAPHY

Steganography is the science of hiding message in a way that anyone does not suspect that there is a hidden message in a stego media. The word steganography is derived from the Greek words “stegos” meaning “cover” and “grafia” meaning “writing” defining it as “covered writing” [1].

Modern formulation of steganography is first defined by Simmons as 'The Prisoners' Problem' in 1984 [2]. Steganography has a long history since 484 BC- 425 BC.

2.1 The Prisoners’ Problem

Assume that Alice and Bob are two prisoners and want to develop an escape plan. Let Wendy be the warden [3]. The only way for Bob to communicate with Alice is through the warden. If Wendy should notice anything suspicious, they will be put in solitary confinement. So they have to communicate in a manner that does not attract attention. Steganography is a way that they communicate invisibly.

Figure 2.1 shows the components of steganography to send a secret message. The cover object can be picture, sound, video, text and any object which is not suspicious. Alice uses a cover media to hide message. Alice and Bob know how to encode and decode stego object and they share the stego key. Encoding is the process of hiding message in cover media and decoding is the process of finding secret message. Alice uses a stego-key to code secret message and Bob uses stego-key to read the message. For example, if the cover object is a text, encoder decides how to and where to hide message using a method in it. Assume that this method hides message in the i.th letters of words in the cover object. Decoder knows which words hiding secret message letters. Stego-key may define the order of hidden letters. The stego object is a combination of cover object, an encoding algorithm, and a stego-key. This stego object goes though the recipient Bob. All messages are checked by warden. If warden realize something suspicious about stego-object, warden can destroy message which is called as active attack Warden may create own fake object and send to recipient, called as malicious attack, so decoder becomes ineffective. If warden does not attack to object and passes it to recipient, Bob uses decoder and shared stego-key to extract the hidden message [4].

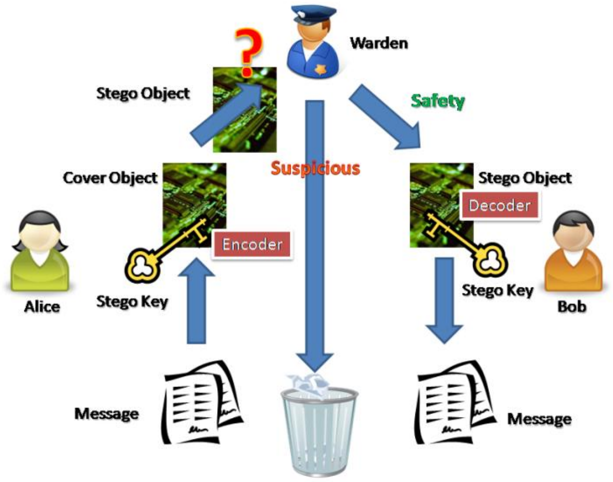


Figure 2.1 Prisoners’ Problem

2.2 History

The earliest record of steganography found from Histories of Herodotus is about covering secret message written wood with wax to warn Spartans against invasion of Xerxes in BC 480 [13].

Another history record is tattooed head. In ancient Greece, a Greek shaved head of a slave and tattooed message. After hair grows back, slave was sent to encourage Aristagoras of Miletus to battle against the Persian king.

Fruit juice, urine and milk were used as invisible inks in Ancient Romans. By heating the letter the secret message is reconstructed. Other chemicals are used in World War II such as copper sulfate solution on handkerchief, which can be visible with ammonia fume [14].

Microfilms were used as stego object during the Franco-Prussian War (1870 -1871).

During World War I, the reason of the large number of cigar orders by two German spies was discovered by the British censor. German spies were using the cigar orders, numbers and types to code ship movements. After the British censor exposed the spies, they were captured and executed [4].

A Cardan grille, introduced by Renaissance mathematician Gerolamo Cardano in 1550, is an important tool for reading and hiding of a message [4] [13]. The grille is a paper or cardboard with holes at selected places. The card is laid over the page of texts that contains a hidden message to read messages. Only the secret letters that appear through the holes in the grille are read.

2.3 Forms of Steganography

Steganography methods can be used for different media types as cover such as text, audio, xml, video, image files and etc. Each of these cover media has different technique and embedding capacity to hide secret message. Figure 2.2 shows how to group steganographic methods [7].

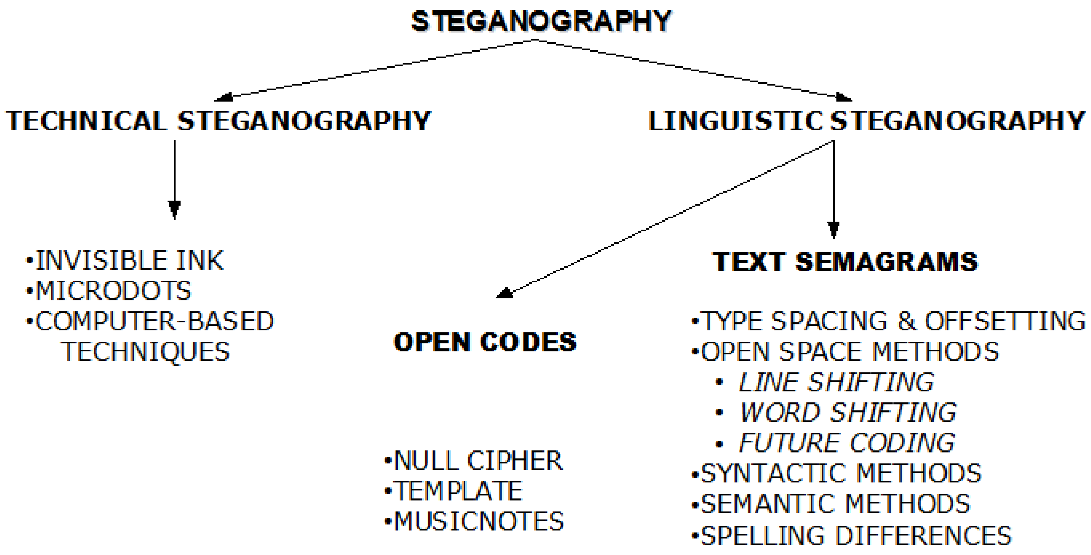


Figure 2.2 Forms of Steganography

2.3.1 Linguistic Steganography

2.3.1.1 Text Semagrams

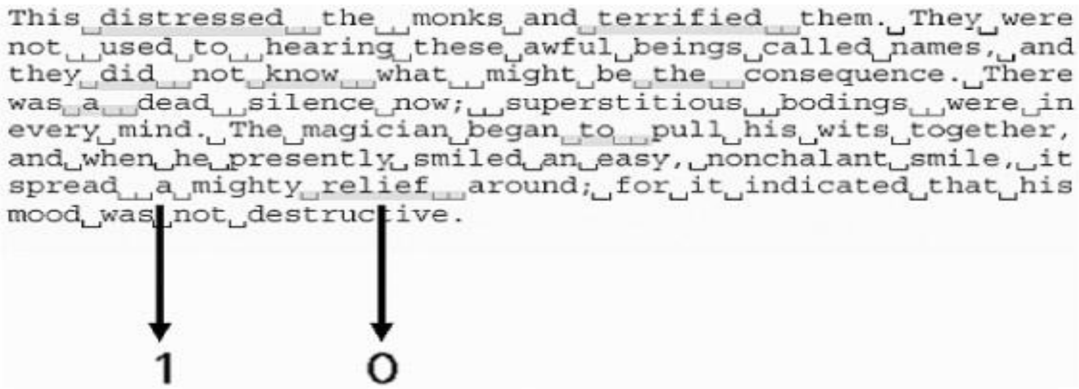
Text files are another cover media to hide secret message. For open space methods both using a software and handwriting are possible. But while writing to hide message more attention is needed. Handwriting and software can be used for hiding message in text using semantic, syntactic and spelling methods. All three of these text based encoding methods require either the original file or the format knowledge of the original files to be able to decode the secret message [7] [4].

a. Type Spacing and Offset

Type spacing or type offsetting is a way of hiding message in a text. The first reason to use type spacing method was to discourage illegal copying of textual material. Although the purpose of this method to add a watermark, type spacing can also be used to hide a secret message in text. To encode a secret message using type spacing all one would have to do is adjusting specific letters ever so slightly from their normal position. The letters that are out of position indicate the secret message [4].

This method changes the distance between the letters or words adding extra white spaces or changing words position from original by slightly shifting. Figure 2.3 shows an example of non digital form of type shifting method. Secret words are underlined. They have 1 white space at left and 2 white spaces at right.

It is also possible to define this method not only for words but also characters. For example, if the decoding definition of this method is to take the character after extra white space. As soon that our secret message: tmtuhdssbwpaa. Writing secret words cannot be possible for all kinds of texts. So, encoding letters using type spacing method can be more useful.

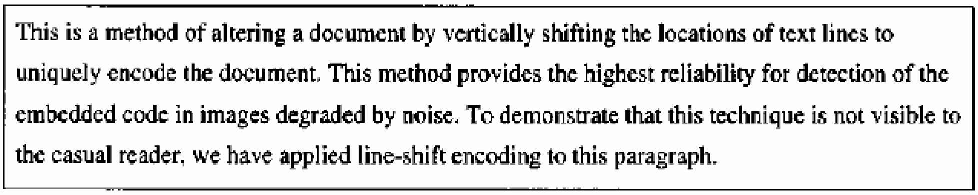


```
This␣distressed␣␣the␣␣monks␣and␣terrified␣␣them.␣They␣were
not␣␣used␣to␣␣hearing␣these␣awful␣beings␣called␣names,␣and
they␣did␣␣not␣know␣␣what␣␣might␣be␣the␣␣consequence.␣There
was␣a␣␣dead␣␣silence␣now;␣␣superstitious␣␣bodings␣␣were␣in
every␣mind.␣The␣magician␣began␣to␣␣pull␣his␣wits␣together,
and␣when␣he␣presently␣smiled␣an␣easy,␣nonchalant␣smile,␣it
spread␣␣a␣mighty␣relief␣␣around;␣for␣it␣indicated␣that␣his
mood␣was␣not␣destructive.
```

Figure 2.3 Type Spacing and Offsetting Method for Non-Digital Text

b. Line Shifting Coding Protocol

Line-shift coding involves actually shifting each line of text vertically up or down by a small fraction (such as 1/300th of an inch) according to the coding parameters. HVS cannot detect the shifting lines, but this small fraction is detectable using computer. It is possible to encode data in text defining lines shifted up or down as 1 or 0. So, the capacity to hide message depends on line counts in text. Figure 2.4 shows an example of line shifting coding [63].



This is a method of altering a document by vertically shifting the locations of text lines to uniquely encode the document. This method provides the highest reliability for detection of the embedded code in images degraded by noise. To demonstrate that this technique is not visible to the casual reader, we have applied line-shift encoding to this paragraph.

Figure 2.4 Line Shift Coding Example

c. Word Shift Coding Protocol

Word-shift encoding works in much the same way that line-shift encoding works; only we use the horizontal spaces between words to equate a value for the hidden message. This method of encoding is less visible than line-shift encoding but requires that the text format support variable spacing. Coding technique is same with line shifting. 1 or 0 represents if a word is shifted left or right measuring the spaces between each word. Figure 2.5 and Figure 2.6 show an example of word-shift encoding method [63] [4].

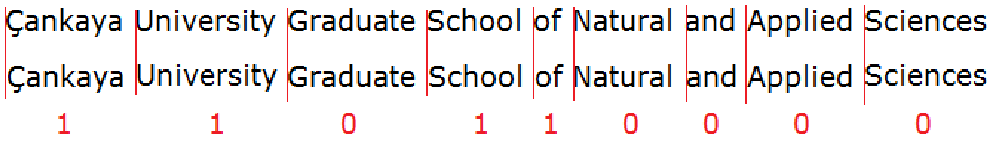

Çankaya University Graduate School of Natural and Applied Sciences
Çankaya University Graduate School of Natural and Applied Sciences
1 1 0 1 1 0 0 0 0

Figure 2.5 Word Shift Coding Example

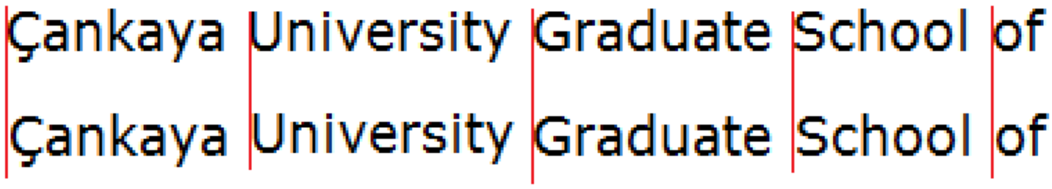
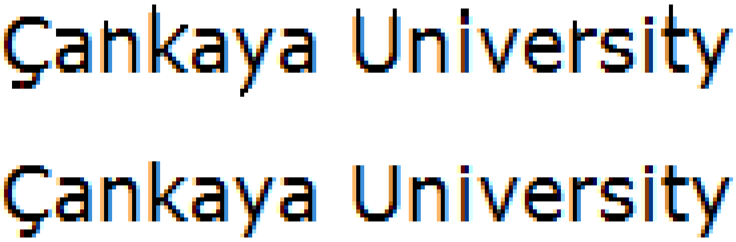

Çankaya University Graduate School of
Çankaya University Graduate School of

Figure 2.6 Zoom to Word Shift Coding Example

In this example the first line uses normal spacing while the second has some words shifted to right. If word is shifted right, it is represented by 1. If there is no shifting to right or left, it is represented by 0. Also without having the original for comparison it is possible to read secret data from text. The distance between two starting words can give a threshold value, and then by comparing the distance between other words using the threshold value message can be extracted.

d. Feature Coding Protocol

Feature Coding Protocol uses characters attributes such as vertical/horizontal length of letters such as b,d,T,L,N,P or the dots above i and j to encode secret message into the text [4.7] [8]. Although, these small changes do not take an attention of HVS, coding whole document can be detectable. So, defining coding letters, which can changeable horizontal, as if there is a manipulation on right direction, it is represented by 1, left direction represents 0. If there is no change, it means that letter does not carry any secret. This method can be defined for letters which can manipulate by changing vertical attributes or having dot. Figure 2.7 shows an example of feature coding. All these three methods line-shifting, word-shifting and feature coding can be usable together to increase capacity to hide a secret message.



Çankaya University

Çankaya University

Figure 2.7 Feature Coding Example

e. Syntactic Method

Syntactic method uses the manipulation of punctuation to hide information [63]. Defining ',' as 1 or 0 where it does not change the meaning of the sentences coding text is possible. Also apostrophe ' can be used to encode data.

For example:

bread, cereal, and milk

bread, cereal and milk

f. Semantic Method

Another data hiding method in text involves changing the words themselves. This method assigns two synonyms primary or secondary value. For example, the word "big" can be considered primary and "large" secondary. Receiver must have the same table whether a word has primary or secondary for decoding. Also this table assigns a value for each word as 0 or 1 to help encode secret message [9]. Some synonyms:

student and pupil (noun)
petty crime and misdemeanor (noun)
buy and purchase (verb)

sick and ill (adjective)
quickly and speedily (adverb)
on and upon (preposition)

g. American / British English Differences

There are differences between more than 1700 words spelling in British and American English. Differences can be categorized into eleven types [10]. Steganography takes advantage of these differences for hiding message choosing British or American English as 1 or 0.

British vs. American	British	American
-ise vs. –ize	analyse	analyze
-l- vs. –ll	enrolment	enrollment
final -l vs. –ll	appal	appal
-ae- vs. -e-	esthetic	aesthetic
-oe- vs. -e-	diarrhea	diarrhea
-our vs. –or	honour	honor
-tre vs. –ter	centre	center
-gramme vs. –gram	programme	program
-logue vs. –log	catalogue	catalog
-ence vs. –ense	defence	defense
Miscellaneous	aeroplane	airplane

2.3.1.2 Open Codes

a. Null Cipher

Null Cipher is a form of steganography using fixed positions of each word to hide message in the cover message [6]. Example:

Fishing freshwater bends and saltwater coasts rewards anyone feeling stressed. Resourceful anglers usually find masterful leapers fun and admit swordfish rank and overwhelming any day.

The following message emerges taking the third letter in every word: *Send lawyers guns and money.*

b. Template

Template, which uses a template as piece of paper with holes cut in or predefined locations on the page to hide a message, is another form of steganography. Obviously, sender and receiver must have the same template to hide and read message [5]. Example:

THE MOST COMMON WORK ANIMAL IS THE HORSE. THEY CAN
BE USED
TO FERRY EQUIPMENT TO AND FROM WORKERS OR TO PULL
A PLOW.
BE CAREFUL, THOUGH, BECAUSE SOME HAVE SANK UP TO
THEIR
KNEES IN MUD OR SAND, SUCH AS AN INCIDENT AT THE
BURLINGTON
FACTORY LAST YEAR. BUT HORSES REMAIN A SIGNIFICANT
FIND. ON
A FARM, AN ALTERNATE WORK ANIMAL MIGHT BE A BURRO
BUT THEY
ARE NOT AS COMFORTABLE AS A TRANSPORT ANIMAL.

The receiver can read the hidden message as follows, using a template to the text above.

HORSE
FERRY
SANK
IN
BURLINGTON
FIND
ALTERNATE
TRANSPORT

c. Music Notes

Matching letters to specific music notes to hide message into notes, described in Gaspar Schott's book Schola Steganographica [4].

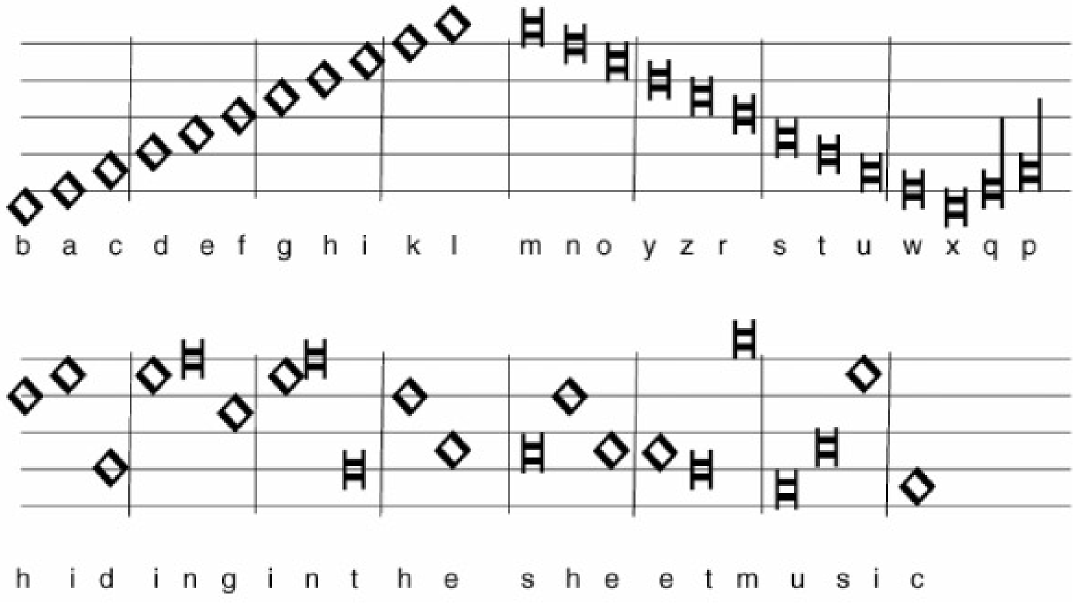

b a c d e f g h i k l m n o y z r s t u w x q p
h i d i n g i n t h e s h e e t m u s i c

Figure 2.8 A Codebook and an Example for Coding Using Music Notes

2.3.2 Technical Steganography

2.3.2.1 Invisible Ink

Invisible inks, colorless liquids that require light, heat or special chemicals to make them visible, are one form of steganography. Acid based colorless liquids can applied to paper and dried to hide a message [4]. To make it visible and change the color, pH indicator is used to reacts with the acid properties. Milk, vinegar, lemon juice are some types of liquids, have been used throughout history.

Most color laser printers add an identifying code on printed page carrying information about serial number of printer, date, and time [11]. This code is microscopic yellow dots normally invisible, HVS cannot detect, but it can be visible using a blue LED light.

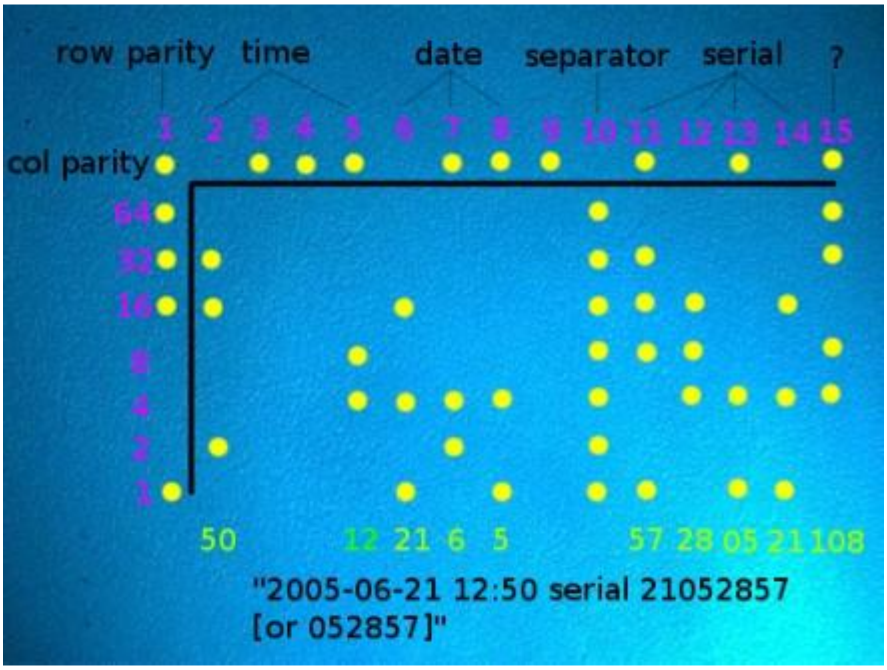


Figure 2.9 An Invisible Ink Example on Laser Printers

2.3.2.2 Microdots

The microdots are a form of page-size photograph reduced to 1 mm in diameter. The microdots used as a form of steganography firstly in World War II. Size of taken photograph of hidden message reduces to the size of postage stamp. Using reverse microscope image size brings down to 1 mm [12].

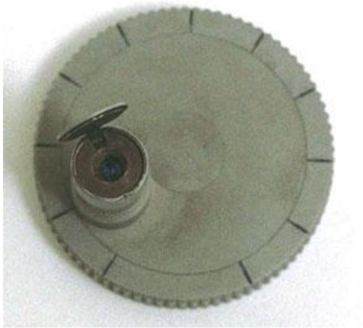


Figure 2.10 Sample Microdot

2.3.2.3 Computer Based Techniques

a. Hiding in Audio Files

The techniques that are used to hide information inside Audio files may be listed as follows [4]:

- ***Low bit encoding***, which is somewhat similar to LSB that is generally used in images.

○ The problem with low bit encoding is that it is usually noticeable to the human ear, so it is a rather risky method for someone to use if they are trying to mask information inside of an audio file.

- ***Spread Spectrum*** is another method which adds random noises to the signal the information and spreads across the frequency spectrum.
- ***Echo data hiding*** which uses the echoes in audio files and. adds extra sound to an echo
- ***Differential phase variation*** in which the file is divided into blocks and using the embedded message, block's initial phase is transformed.

b. Hiding in Image Files

Image steganography techniques can be divided into two groups: those in the Image (Spatial) Domain and those in the Transform (Frequency) Domain. Image domain techniques embed messages in the intensity of the pixels directly. Frequency domain techniques first transform the image and then embed the message in more significant areas of cover image. Most suitable image types are uncompressed and lossless compressed images for spatial domain steganography methods.

In Chapter 4 steganography algorithms will be explained in categories according to image file formats and the domain in which they are performed.

c. Hiding in Video Files

Hiding a message in video files such as .avi and .mpeg is similar to hiding in images. Mostly, the program use Discrete Cosine Transform (DCT) method to hide message rounding a value in a part of frame [17].

CHAPTER 3

IMAGE CODING and COMPRESSION

An image is a collection of numbers that constitute different light intensities in different areas of the image [18]. The smallest component of a digital image is called as pixel. Each pixel has own coordinates. Pixels are displayed horizontally row by row. Number of row and columns gives the dimension of an image. The number of the bits per pixel (bpp) represents the color of a pixel, which describes the color depth or bit depth of an image. If each pixel represented by 1-bpp, called as monochrome. There are 2 colors for 1-bpp to represent pixels. 8-bpp generally uses for grayscale images to display 256 different shades of gray, or uses for color images. 16-bpp color depth images are called as “High color” images, 5 bits represents red color channel, 5 bits for blue color channel, and 6 bits for green color channels. 24-bpp is called as “True color”. These images can show almost 16.8 million different colors. Each primary color (red, green, and blue) are represented by 8-bits.

Digital images can be stored using different techniques. Some of them compresses image to reduce data size but it defects the quality of an image, called as lossy compression. Some of these techniques reduce data size, but not as much as lossy compression methods, by not changing pixel values, called as lossless compression.

To compress data different compression and coding algorithms are used. Lossless compression methods Lempel Ziv 77 (LZ77), Lempel Ziv 78 (LZ78), Lempel Ziv Welch (LZW) and Huffman Coding algorithm are described in this chapter. After compression or manipulation on an image, there can be significant changes. Some image analysis methods, Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Weighted Peak Signal-to-Noise Ratio (WPNR), Histogram Analysis, are described to calculate differences between original and manipulated image.

3.1 Image Types

3.1.1 Grayscale Images

Grayscale images carry on a series of shades from white to black at each pixel. Generally, each pixel stores 8-bit integer, giving 256 different grayscale intensities. For medical imaging use, more details are needed. So, some image formats as JPEG200, GIF, and PNG support 16-bpp (65,536 tones) grayscale images. Binary representation of color black is 0 for different pixel depths and white is the maximum value. White is 255 for 8-bit pixel depth and 65,535 for 16-bit pixel depth. Figure 3.1 shows 8 and 16-bit shades of gray.

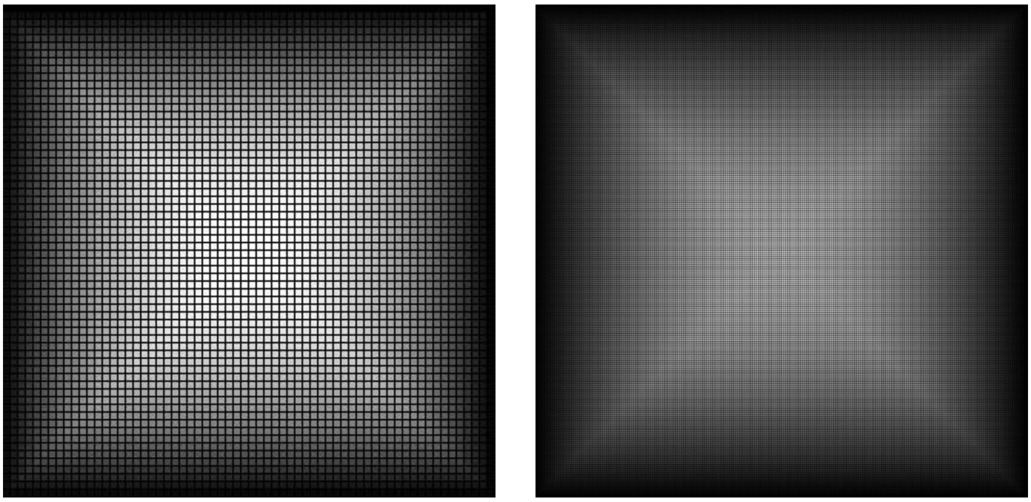


Figure 3.1 (a) 8-bit Shades of Gray (b) 16-bit Shades of Gray

To convert any color to grayscale representation, firstly the color is converted to RGB format if it is not. Then, typically adding %30 of the red value, %59 of the green value, and %11 of the blue value give the grayscale tone of the color [19].

3.1.2 RGB Color Space

The RGB color model is a mixture of different tones of red, green, and blue colors, and light spectra to produce tones of other colors. Zero intensity gives the black color, and full intensity of each color spaces gives the white color [20]. If the tones of color channels are close to each other, mixture of them gives the shade of gray depending on the intensity. If a color channel has the strongest value and there is a big difference between other color channels, the color is close to this primary color. If two color channels have the strongest value, the color is close to mixture of these two primary colors (cyan: green and blue, magenta: red and blue, yellow: red and green). Figure 3.2 shows the RGB color space.

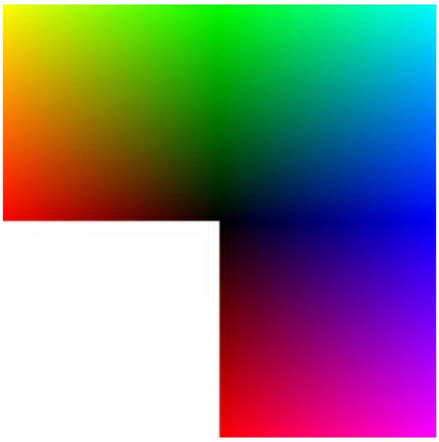


Figure 3.2 Color Space

3.1.3 Palette Images

These images are coded using one number to represent a pixel color. Each image file contains its own palette. This palette is the list of used colors in image. For example, if a color image dimensions are 10x10, an uncompressed image carries 24 bit per pixel and totally 2400 bits (300 byte). If the palette image represents each pixel by 8 bits, this file contains 800 bits (100 byte) for pixels and 768 bits (256x3) for palette. These 768 bits (96 byte) define the RGB values of used 256 colors. While this uncompressed image needs 300 bytes, palette image needs 196 bytes. Limited number of distinct colors is the disadvantage of palette images. But it can be useful for images or drawings which do not need more color than the number of colors supported by palettes. [21]

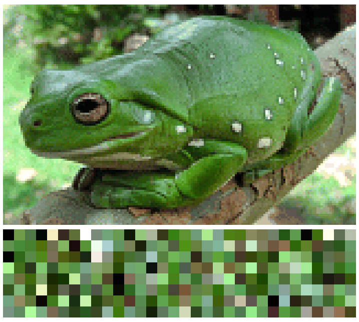


Figure 3.3 Sample Palette Image

3.2 Image Formats

Table 3.1 shows properties of different image formats. Image file may be uncompressed in this case file format has the exact value of each pixel. If it is lossless compressed, the file stores less information than uncompressed files. This type of format may give exact value of each pixel after some operations. Lossy compressed files, store less information than uncompressed files, after some calculations, it may give closer values to each pixel. Natural images are kind of photos taken by cameras which have smooth variation of tone and color. In Table 3.1 “line drawing row” represents images, have hard edges, like textures, iconic graphics. ‘S’ represents “Suggested” and “ Not S” represents “Not Suggested” to use image formats to encode natural or line drawing images.

Transparency is represented by 1 bit. A pixel can be %100 transparent or not. PNG and BMP image formats have Alpha channel. This channel carries 8-bit or 16-bit transparency tones for each pixel. But systems may output maximum 255 different tones for each pixel. PNG, TIFF and GIF support palette-indexed color for 1, 2, 4, and 8-bit. But JPEG and JPEG2000 do not support palette-indexed color images. Supported color depths and file extensions are given in Table 3.1 [22] [23] [24]

Table 3.1 Comparison of Image Formats

Image Formats Properties	BMP	JPEG	JPEG2000	GIF	PNG	TIFF
Uncompressed	Y	N	N	N	N	N
Lossy Compressed	N	Y (DCT)	Y (DWT)	N	N	N
Lossless Compressed	N	Y	Y	Y (LZW)	Y (LZ77 & HUFFMAN)	Y (LZW)
Natural Images	S	S	S	Not S	S	S
Line Drawings	S	Not S	Not S	S	S	S
Transparency	N	N	Y	Y	N	Y
Alpha Channel	Y	N	N	N	Y	N
Palette-Indexed	Y	N	N	Y	Y	Y
Animations	N	N	N	Y	N	N
Color Depth	1,4,8, 16, 32-bit	12, 24-bit	Up to 48-bit	1,2,3,4,5 ,6,7,8-bit	1,2,4,8,16,24, 32,48,64-bit	1,2,4,8,16, 24,32-bit
Grayscale Depth	8-bit	8-bit	8, 16-bit	8-bit	8, 16-bit	8, 16-bit
File Extensions	.bmp .dib .rle .2dp	.jpg .jpeg .jpe	.jp2 .j2c .jpc .j2k	.gif .giff .gfo	.png	.tif .tiff

3.3 Image Analysis

Differences between two images can be calculated using MSE, PSNR, and WPSNR formulas to have an idea if these images are identical or manipulated. Histograms show the frequency distribution on color or grayscale images. Comparing histograms of two same looking images can give an idea if there is any modification on image.

3.3.1 Mean Squared Error (MSE)

MSE is the ratio of sum of the square of the differences in the pixel values between the corresponding pixels of the two images over total pixel number. MSE can be calculated if two images' dimensions are equal. If two images are identical MSE value is 0. Formula 3.1 shows how to calculate MSE value. I1 and I2 are images with same dimensions. M and N are the dimensions of images. [25]

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \tag{3.1}$$

3.3.2 Peak Signal to Noise Ratio (PSNR)

The peak signal-to-noise ratio (PSNR) is the ratio between maximum power of a signal and the power of the signal's noise. After lossy compressing an image to reduce data size or changing quality of image, pixel values changes. Calculating PSNR value defines the changes on image. PSNR is usually expressed in decibels. MSE and PSNR values can be calculated separately for each color channel. If two images are identical the PSNR value is infinite. [26]

$$PSNR = 10 \log_{10}\left(\frac{R^2}{MSE}\right) \tag{3.2}$$

Formula 3.2 shows how to calculate PSNR value. R is the maximum pixel value for the image. 2^n-1 gives the maximum pixel value for an image. R value is 2^8-1=255 if pixels are represented by 8-bits for image.

3.3.3 Weighted Peak Signal to Noise Ratio (WPSNR)

Quality measurement function Weighted Peak Signal to Noise Ratio uses different approach than PSNR. WPSNR calculates different weights for different blocks of image while PSNR uses same weight for all images. Using different weights gives better results about changes in the perceptual quality more accurately than PSNR. HVS is less sensitive for to changes in highly textured areas. So, WPSNR uses Noise Visibility Function (NVF), which uses a Gaussian model to estimate how much texture exists at different blocks of the image. NORM is a normalization function and δ is the luminance variance of block at formula 3.4. [27]

$$WPSNR = 20\log_{10}\left(\frac{\mathrm{R}}{\sqrt{MSE}\ x\ NVF}\right) \tag{3.3}$$

$$NVF = NORM\left\{\frac{1}{1+\delta_{block}^{2}}\right\} \tag{3.4}$$

An example of WPSNR, PSNR and MSE values for a compressed image with different qualities are given at Appendix A..

3.3.4 Histogram

An image histogram is a graphical display of the tonal distribution in an image, shown as bars. Let k is the maximum pixel value in an image, m is the pixel value, and n is total repetition of n in image. [28]

$$n_i = \sum_{i=1}^{k} m_i \tag{3.5}$$

Example:

Figure 3.5.a, b show the color distribution on 3 channel (red, green, blue) of Lena.bmp (Figure 3.4.a) and compressed with quality 80 Lena.jpeg (Figure 3.4.b) respectively. ImageJ [29] was used for histogram analysis of pictures.

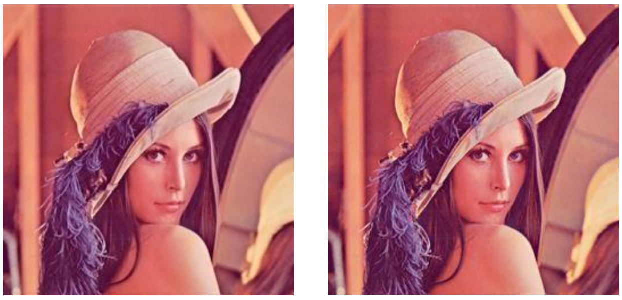
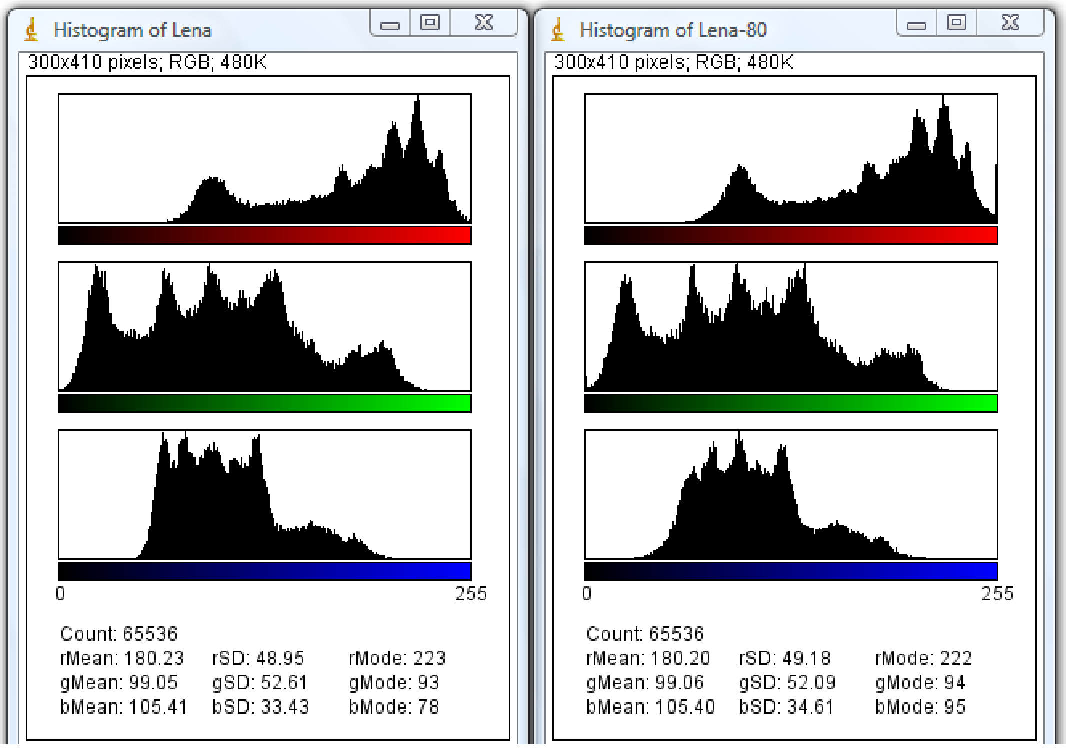


Figure 3.4 (a) Original Image (b) %80 Quality Compressed Image



300x410 pixels; RGB; 480K

Count: 65536
rMean: 180.23 rSD: 48.95 rMode: 223
gMean: 99.05 gSD: 52.61 gMode: 93
bMean: 105.41 bSD: 33.43 bMode: 78

300x410 pixels; RGB; 480K

Count: 65536
rMean: 180.20 rSD: 49.18 rMode: 222
gMean: 99.06 gSD: 52.09 gMode: 94
bMean: 105.40 bSD: 34.61 bMode: 95

(a) Color Histogram Lena.bmp, (b) Color Histogram Lena.jpg

Figure 3.5 Histogram of Lena .bmp and .jpeg Images

3.4 Compression and Coding Methods

Compression methods are used to reduce the data size. Image formats use them to encode image files. PNG image format uses combination of LZ77 lossless compression algorithm and Huffman coding method called as DEFLATE [30]. GIF image format uses LZW lossless compression algorithm, modified version of LZ78. Compression ratio of .gif files are between 4:1 and 10:1. TIFF image files also use LZW algorithm to reduce data size. JPEG image format applies DCT to image than uses Huffman coding method. Compression ratio of .jpeg image changes between 10.1 and 100.1 depending on the quality of image. PNG image format compresses better than .gif files, between %10 and %30 [31]. PNG gives the best quality/compression ratio for all kind of images. In this part, lossless compression algorithms LZ77, LZ78, LZW and Huffman coding method are explained.

3.4.1 Lempel Ziv 77 (LZ77)

LZ77 is a lossless data compression algorithm defined by Abraham Lempel and Jacob Ziv in 1977. LZ77 algorithm searches back to find longest sequences to represent next characters. This algorithm carries 3 characters for each step at coding table. First one defines how character long to go back from current position, second character defines how long character will be copied from that position and the last character is a character to add end of that sequence. [32]

Assume that the character sequence is S = (A,B,B,C,D,A,B,D,B,C,C) to be compressed.

Table 3.2 Encoding Process of LZ77

	Back to i.th character	Get L character	Add to end of sequence	Sequence
Step 1	0	0	A	A
Step 2	0	0	B	AB
Step 3	2	1	C	ABBC
Step 4	0	0	D	ABBCD
Step 5	1	2	D	ABBCDABD
Step 6	3	2	C	ABBCDABDBCCC

3.4.2 Lempel Ziv 78 (LZ78)

LZ78 is a lossless data compression algorithm defined by Abraham Lempel and Jacob Ziv in 1978. This method is similar to LZ77. Advantage of this method is to store 2 characters for each step. LZ78 stores the result of each step in a dictionary. When new characters will be added, checks the previous steps to find the longest matching step. [33]

Assume that the character sequence is S = (A,A,B,A,C,A,B,A,A,C,C) to be compressed.

Table 3.3 Encoding Process of LZ78

	Dictionary	Back to dictionary	Add to end of sequence	Sequence
Step 1	A	0	A	A
Step 2	AB	1	B	AAB
Step 3	AC	1	C	AABAC
Step 4	ABA	2	A	AABACABA
Step 5	ACC	2	C	AABACABAACC

3.4.3 Lempel Ziv Welch (LZW)

LZW is an improved implementation of LZ78 algorithm, published by Welch in 1984. This algorithm defines a code book and which encoder and decoder know. LZW stores new codes generated by encoder. Decoder uses the coded sequence to analyze the meaning of new codes. [34]

Assume that the character sequence S = (A,B,C,A,C,D,B,C,A,B,C,A)

Codebook have this representations for these characters, A=1, B=2, C=3, D=4

Table 3.4 Encoding Process of LZW

	New Value	Codeword
Step 1	5	AB
Step 2	6	BC
Step 3	7	CA
Step 4	8	AC
Step 5	9	CD
Step 6	10	DB
Step 7	11	BCA
Step 8	12	ABC

Compressed sequence of S is (1,2,3,1,3,4,6,5,7)

Decoding process:

Decoder knows the meaning of 1,2,3,4. So, decoder generates same table. Till it comes to new codes '6','5', and '7' codebook has values of these codes

3.4.4 Huffman Coding

The entropy encoding algorithm is developed by David A. Huffman in 1952, used for lossless data compression. This method creates binary tree using frequencies of characters. [35]

Assume that frequencies of 7 characters are:

A= 1 B= 3 C= 7 D= 14 E= 20 F= 25 G= 27

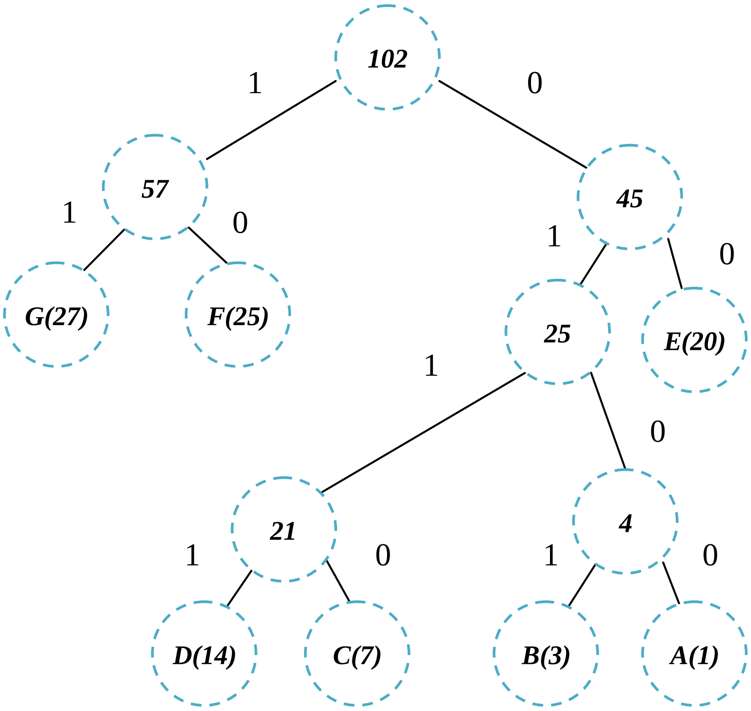


Figure 3.6 Huffman Tree

Using Hamming algorithm each character now has different represented value. Higher frequency characters have smaller codeword length so this is the advantage of Hamming algorithm to encode data using fewer characters for the most repeated characters.

A= (0100) B=(0101) C=(0110) D=(0111) E=(00) F=(10) G=(11)

CHAPTER 4

STEGANOGRAPHY METHODS ON STILL IMAGES

Image steganography techniques deal with two subjects as Frequency / Transform Domain and Image / Spatial Domain [36]. Spatial Domain techniques embed secret data pixel directly into cover image. The most popular data hiding method is, changing pixels' left most digits or last two, known as (Least Significant Bit) LSB. Lossless image formats like .bmp, .png, and 8-bit gray-scale .gif are usable for LSB methods. LSB embedding method is not usable for palette images because of changing just last bit of a pixel causes a big difference on image. After embedding all of secret message to palette image, HVS can detect manipulations on image. There are some techniques to hide data in palette images like changing color orders in palette or adding new colors to palette. Transform Domain techniques, firstly transform image then embed data in it. These techniques are usable for .jpeg images (See in Chapter 3.1). One byte data can be hidden in each 8x8 block using the DCT for lossy compression.

In this study;

MxN	: image size
y_i	: i.th position of cover image
$\hat{y}_i$	: i.th position of stego image

m_i : i.th bit of the message

$y_{i,j}$: j.th bit of i.th pixel of cover image

If there is no additional explanation, cover image is grayscale and when talking about pixel, it means only one color channel is mentioned.

4.1 Least Significant Bit (LSB) Replacement

LSB insertion is the most preferable method to embed data in cover image. The last bit of each pixel swaps with secret message's bit [37]. For example, if cover media is color image and secret data is 'a':

ASCII value of 'a' is 97 = 01100001

Color image pixels:

10101111 00011000 11000010	(175, 24, 194)
10110000 00010110 11001000	(176, 22, 200)
10110100 00011000 11000100	(180, 24, 196)

After making changes on LSB of pixels, new pixel values are;

1010111**0** 0001100**1** 1100001**1**	(174, 25, 195)
10110000 00010110 11001000	(176, 22, 200)
10110100 0001100**1** 11000100	(180, 25, 196)

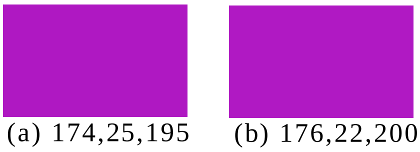
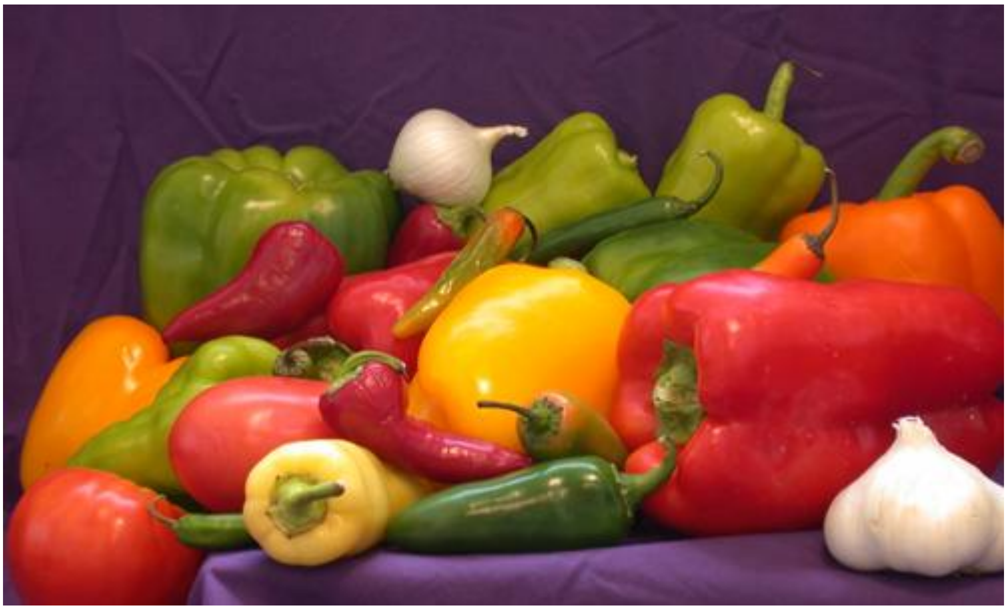

(a) 174,25,195 (b) 176,22,200

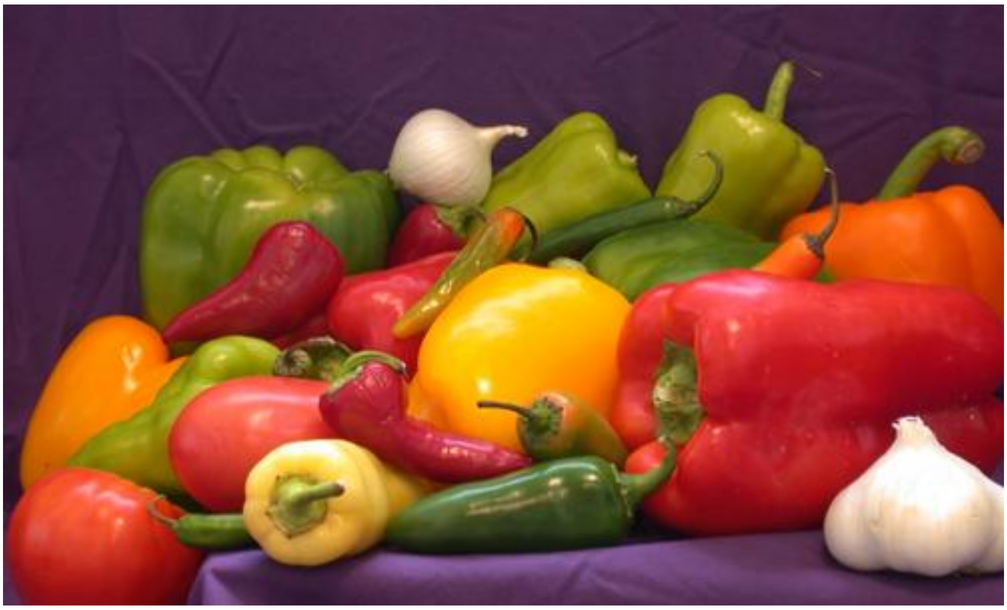
Figure 4.1 Color Representations on RGB

When LSB insertion is applied to an image, generally as much as half of the last bits changes. For a 240x320 color image, the total number of pixels is 240*320*3, which is equal to 230400 bits, 28800 bytes or 28,125 kb. Thus, 28,125 kb secret message may be embedded in the color image. If a method with last two bit insertion is used, the capacity gets doubled.

For an instance, a color image, size of image is 500x300x3, is used to hide 10240 bytes data in it. Maximum capacity of image is 56250 bytes for LSB insertion. Figure 4.2 (a) is the cover image and (b) is the stego image. Figure 4.3 (a) is the color histogram of cover image and (b) is the histogram of stego image.



(a)Cover Image

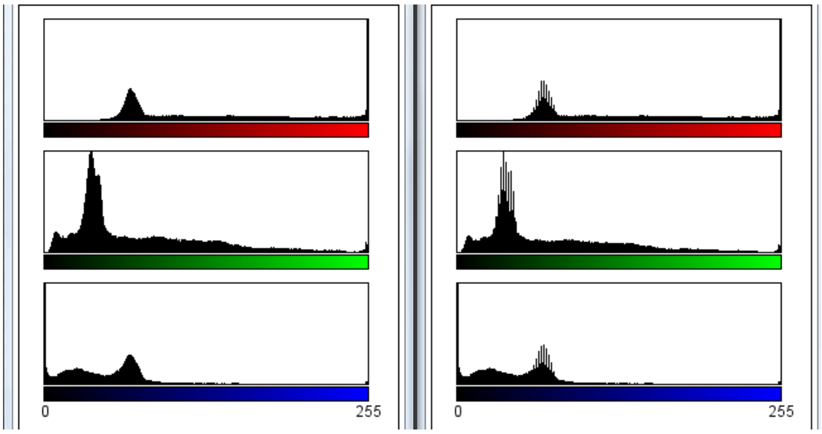


(b) Stego Image

Figure 4.2 (a) Cover Image (b) Stego Image

Table 4.1 Effects of Information Hiding on Image

Embedded Bits	Modified Bits	MSE (Red)	MSE (Green)	MSE (Blue)	PSNR (Red)	PSNR (Green)	PSNR (Blue)
81920	41042	0.0906	0.0916	0.0914	58.5595	58.51	58.5226



(a)Cover image histogram (b) Stego image histogram

Figure 4.3 Histograms of Cover and Stego Images

There are some methods to hide data using up to last 4 bits of grayscale pictures. As mentioned at Chapter 3.1.1 8-bit represents grayscale colors between 0 (black) and 255 (white). At worst case, changing last 4 bit cause a change of maximum 15 colors change in palette. If the pixels last 4 bits is 0000 and secret message bits to hide there are 1111. Figure 4.4 shows changing 15 colors in grayscale palette. So, depending on picture and coding technique, last 4 bits can be used to encode data in 8-bit grayscale images. Although steganalyze techniques can detect changes in image easily, HVS cannot detect the difference between cover image and stego image.

(a)Grayscale value 0 (b)Grayscale value 15

Figure 4.4 Color Representations

4.2 Hiding Function Using Modulus Function

High-hiding capacity method based on modulus function is proposed by Thien and Lin in 1993 [38]. This method hides k bits in 1 pixel and causes less modification than hiding k bits per pixel using in LSB Replacement method.

How this method works:

Let k be the number of host bits of a pixel to embed data
m is the secret data can be represented by k bits
y is the cover pixel value
d is the difference

$$d = m - y\ mod 2^{k} \tag{4.1}$$

d' is the minimal difference value between the original value y and the modified value $\hat{y}$.

$$\begin{aligned} d' &= d\ if\ \left(-\left|\frac{2^{k}-1}{2}\right|\right) \leq d \leq \left\lceil -\left|\frac{2^{k}-1}{2}\right|\right\rceil \\ d' &= d + 2^{k}\ if\ (-2^{k}+1) \leq d \leq \left(-\left|\frac{2^{k}-1}{2}\right|\right) \\ d' &= d - 2^{k}\ if\ \left\lceil -\left|\frac{2^{k}-1}{2}\right|\right\rceil \leq d \leq 2^{k} \end{aligned} \tag{4.2}$$

$\hat{y}$ is, the modified pixel value, the sum of d' and y.

$$\hat{y} = d' + y \tag{4.3}$$

For an instance, let

Cover pixel is y_1 = 184 $(10111000)_2$

If last two bits of cover pixel is used to hide data

If m_1 and m_2 = (11)

d = 3 - (184 mod 2^2) = 3

d' = 3-4 = -1 (equation 4.2. condition 3)

$\hat{y}$ = -1 + 184 = 183

Table 4.2 shows all combinations for hidden bits for cover pixel 184.

Table 4.2 Results of Modulus and Replacement Methods on Cover Pixels

S. Bits	L5 bits*	Modulus	L5 bits	Distance	Replacement	L5 bits	Distance
00	11000	184	11000	0	184	11000	0
01	11000	185	11001	+1	185	11001	+1
10	11000	182	10110	-2	186	11010	+2
11	11000	183	10111	-1	187	11011	+3

*L5 bits are the last five bits of color channel

As seen in Table 4.2 Modulus method affects more than k bits to minimize the difference between cover pixel and stego pixel. So, this approach gives better MSE, and PSNR values according to LSB replacement method.

Extraction of secret bits:

Secret Bits = ($\hat{y}$ *mod* 2^k)

If Stego pixel is 186 and k is 2,

Secret bits = (186 mod 4) = 2 = $(10)_2$

4.3 LSB Matching Method

LSB Matching method is described by J.Mielikainen in 2006 [39]. The purpose of this method is to embed data in image, by changing fewer bits than classical LSB

Replacement method. LSB Matching method has the same capacity with LSB Replacement method. LSB Matching method groups secret bits by two and groups the pixels by two, depending on cover bits and secret data bits, changes the LSB of cover image.

If the 2 message bits are different from the LSB of cover pixels, LSB Replacement method changes both two pixels' LSB. But LSB Matching uses a different approach. This method does not change the second pixels value. Increase or decrease the first pixels value using these properties. After changing two cover bits, LSB of the first pixel is the first message bit, and the second message bit can be found using formula (4.4).

$$\begin{aligned} &f(l,n) = LSB\left(\left\lfloor\frac{l}{n}\right\rfloor + n\right) \\ &\text{property 1:} f(l-1,n) \neq f(l,n+1) \\ &\text{property 2:} f(l,n) \neq f(l,n+1) \end{aligned} \tag{4.4}$$

How this method works:

1. Check if the LSB of first cover pixel ($y_{1,1}$) is equal to 1st message bit (m_1)
    a. If they are equal, first stego pixel is equal to first cover pixel ($\hat{y}_1=y_1$)
        i. If the LSB of sum of the last 2nd bit of first cover pixel ($y_{i,2}$) and LSB of second cover pixel ($y_{2,1}$) is equal to 2nd message bit, LSB second of stego pixel is equal to second cover pixel ($\hat{y}_2=y_2$).
        ii. Else, stego pixel 2 can be 1 more or 1 less ($\hat{y}_2\pm1$)
    b. Else, stego pixel 2 is equal to cover pixel 2 ($\hat{y}_2=y_2$)
        i. If the LSB of sum of the last 2nd bit of first cover pixel minus 1 (y_1-$1,_2$) and LSB of second cover pixel ($y_{2,1}$) is equal to 2nd

message bit (m_2), first stego pixel is first cover pixel minus 1 ($\hat{y}_1$-1).

ii. Else, first stego pixel is first cover pixel add 1 ($\hat{y}_1$+1).

Table 4.3 Results of Matching and Replacement Methods on Cover Pixels

Cover Pixels		Message Bits		LSB Matching		LSB Replacement	
6	7	0	0	6	7	6	6 or 8
6	7	0	1	6	6	6	7
6	7	1	0	7	7	5 or 7	6 or 8
6	7	1	1	5	7	5 or 7	7

The LSB replacement and matching algorithm were tested for 5 grayscale images. Results are in Table 4.4.

Table 4.4 Results of Matching and Replacement Methods on Images

Image	Lena	Cameraman	Goldhill	Baboon	Boat
Image Size	128x128	256x256	256x256	111x111	256x256
LSB Capacity (bits)	16384	65536	65536	12321	65536
Test File Size (bits)	14825	33200	33200	11696	44912
Replacement MSE	0.4433	0.2554	0.2503	0.1632	0.0859
Replacement PSNR	51.6638	54.0580	54.1461	56.0031	58.7926
Replacement Changed Bits	7263	16740	16404	5948	22509
Matching MSE	0.3347	0.1917	0.1897	0.1517	0.0645
Matching PSNR	52.8848	55.3057	55.3498	56.1682	60.0366
Matching Changed Bits	5483	12560	12433	4405	16903
If cover and hidden bits are different *[1]	1713	4216	4262	1457	5679
Ratio of Changed bits Rep/Match *[2]	24.5078	24.9701	24.2075	25.9415	24.9056
Less Changes *[3]	12.0068	12.5904	11.9608	13.1915	12.4822

*[1] For example, two pixel last bits of cover pixels are [0 0] and secret bits group [1 1].

*[2] For Lena, Ratio = (7263-5483)*100/7263

*[3] For Lena, (7263*100/14825) – (5483*100/14825)

As seen in Table 4.3, LSB Matching method modifies about %12.5 less than LSB Replacement method.

4.4 Hiding Information Using Chan's Method

The purpose of this method is to reduce modified bit numbers on stego image while having the same payload ($\alpha = 1/1$) with Mielikainen's method. Chan's method does not group cover pixels and message bits by 2. This method uses XOR function to decide how to embed message bit. If the result of LSB of cover pixel at position i ($y_{i,1}$) and last 2nd bit of cover pixel at position i-1 ($y_{i-1,2}$) is equal to message bit m_i, no need to modify cover pixel. If the result is not equal Chan uses a formula which is modified version of Mielikainen's method [40]. Chan's method's efficiency increases if the message length increases, the other algorithms have the static embedding efficiency.

Chan's formula

$$F(y_i) = y_{i-1,2} \oplus y_{i,1} \tag{4.5}$$

How this method works:

1. Firstly, XOR the last bit of pixel at position i, and last 2nd bit of pixel at position i-1.
    a. Compare the result of 1. With the stego bit for position i. If they are equal, no need to change.
    b. If they are not equal use F function. XOR the last bit of pixel at position i and last 2nd bit of pixel at position i+1. Compare the result with stego bit at for position i+1.
        i. If they are not equal, decrease the pixel value 1 at position i, get the last 2nd bit, and compare it with the last 2nd bit of the original value at position i.
            1. If they are equal decrease pixel value by 1 at position i.
            2. If they are not equal increase the pixel value by 1.

ii. If the result of step 3 is equal, subtract the last bit of pixel value and add the complement of last bit to the pixel value.

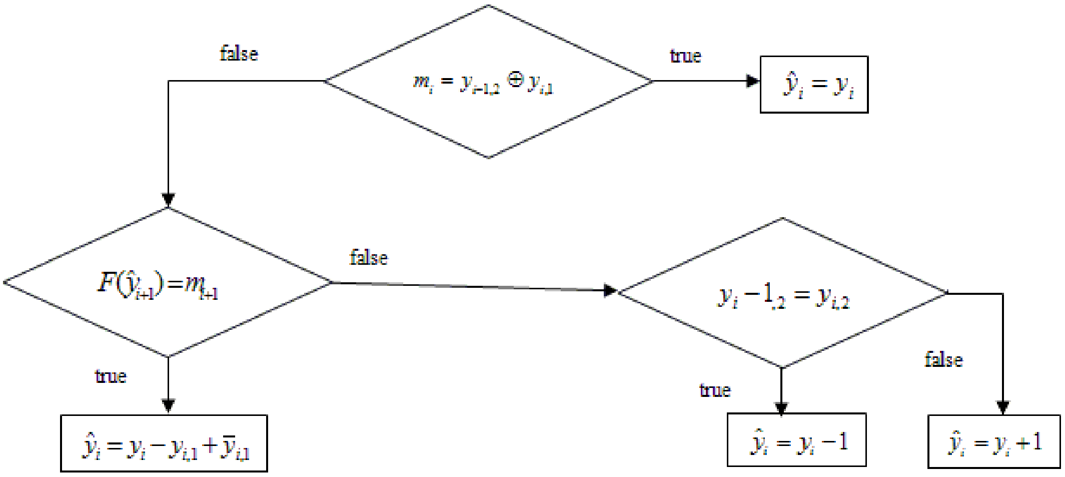


Figure 4.5 Flowchart of Chan's Method

Example:

If the cover pixel is 2 to hide 0, and the previous cover pixel number is 0;

	y_{i-1}		y_i
	0	↘	1
			0
XOR			0
Stego Bit			0
Result	1. step, no need to changes		

Result of XOR function is equal to message bit, so any modification is not needed.

If cover pixels are y_1=2, y_2=2 and stego bits are m_1=1 ,m_2=1;

			Before					After		
	0	↘	1	↘	1	0	↘	1	↘	1
			0		0			1		0
XOR			0		1			1		1
Stego Bit			1		1			1		1
Result	a. and ii. step, need to increase first pixel by 1.									

Result of XOR function is not equal to message bit, so result of XOR function for next pixel is checked if it is equal to message bit for that position. Since the result is equal, stego pixel is equal to cover pixel plus 1.

If cover pixels are y_1=2, y_2=2 and stego bits are m_1=1 ,m_2=0;

			Before					After		
	0	↘	1	↘	1	0	↘	0	↘	1
			0		0			1		0
XOR			0		1			1		0
Stego Bit			1		0			1		0
Result	i. and 1. step, need to decrease first pixel by 1.									

Result of XOR function is not equal to message bit, so the next cover bit is checked using XOR function if the next message bit is equal to result of second XOR function. It is not equal, so change first cover pixel's last 2 two bits by subtracting 1. As a result of this change, also next message bit is embedded.

The advantage of Chan's method increases depending on message length. So, the example in Table 4.3, message bits, grouped by 2 for replacement and matching methods, is grouped by 8 for this method and Table 4.5 shows the results.

Table 4.5 Hiding Information Using Chan's Method

Cover Image Pixels	66776776	66776776	66776776	66776776
Message Bits	00000000	01010101	10101010	11111111
Stego Image	67777777	66786676	56676786	76876876

Table 4.6 shows the results of changes on 256x256 grayscale image after embedding a file 1850 byte using different methods with payload α = 1/1.

Table 4.6 Changes on Images Information Hiding Methods α = 1/1

File Size (byte)	Method	Modified Bits	Embedding Efficiency	PSNR	MSE
100	Mielikainen	286	2.7972	71.7319	0.0044
	Chan	255	3.1373	72.2302	0.0039
	Replacement	368	2.1739	70.6371	0.0056
250	Mielikainen	730	2.7397	67.6624	0.0110
	Chan	658	3.0395	68.1133	0.0100
	Replacement	941	2.1254	66.5597	0.0144
500	Mielikainen	1502	2.6631	64.5289	0.0229
	Chan	1321	3.0280	65.0866	0.0202
	Replacement	1952	2.0492	63.3908	0.0298
1024	Mielikainen	3118	2.6273	61.3568	0.0476
	Chan	2729	3.0311	61.9356	0.0416
	Replacement	4152	1.9730	60.1130	0.0643
1850	Mielikainen	5589	2.6480	58.8223	0.0853
	Chan	4926	3.0045	59.3707	0.0752
	Replacement	7391	2.0024	57.6086	0.1128

4.5 2/3 Embedding Efficiency Method

This method uses 3 pixels to hide 2 bits and if the modification is necessary changes only 1 bit of them by increasing or decreasing. The other bit, which is not hiding 2 message bits, defines how to decode message bits [41]. If that bit is 0, two bits are message bits, if that is 1, complement of two bits are message bits. Table 4.7 shows how to encode 2 message bits in 3 bits for all conditions and Table 4.8 shows an example for all conditions to hide 2 message bits (0, 0).

Table 4.7 Coding using 2/3 Efficient Embedding Method

m_i, m_{i+1}	00	01	10	11
$\hat{y}_{i,1}, \hat{y}_{i+1,1}, \hat{y}_{i+2,1}$	000	001	010	011
	111	110	101	100

Table 4.8 Example for 2/3 Efficient Embedding Method for m_i, m_{i+1} = 00

$y_{i,1}, y_{i+1,1}, y_{i+2,1}$	000	001	010	011	100	101	110	111
$\hat{y}_{i,1}, \hat{y}_{i+1,1}, \hat{y}_{i+2,1}$	000	00**0**	0**0**0	**1**11	**0**00	1**1**1	11**1**	111

Payload (α): 2/3

Changes: 1/4 → no change

3/4 → 1 change

Embedding Efficiency (ε): 3/8

Although this method has the same embedding efficiency with Mielikainen's method, the disadvantage of this method is %33 less embedding capacity.

4.6 Matrix Embedding Using Hamming Codes

Hamming codes mostly is used in transmission, can detect 2 bits fault or detect 1 bit fault and correct it. It is possible to use 1 bit detection and correction property of Hamming codes for steganography [42][43]. This method splits the message into p-length groups and using hamming codes changes 1 bit.

The embedding efficiency of this method:

$$\varepsilon = \frac{p}{1-2^{-p}} \tag{4.6}$$

Payload:

$$\alpha = \frac{p}{2^p - 1} \tag{4.7}$$

How this method works:

1. Split message *p*-length groups
2. Get a group length 2^p-1 bits from cover pixels
3. Construct a hamming matrix `p x 2^p-1`
4. Calculate the multiply of Hamming matrix and transpose of grouped cover pixels
    a. If the result of multiplication is equal to p-length stego bit group no need to change
    b. Else,
        i. XOR the result of multiplication and p-length stego bit group
        ii. Change a bit from cover pixel group which is the result of 4.b.i

Let message group length *p*=3, cover pixel length to hide 3 bits is 2^p-1 = 7. Let the message group is m= (110) and cover pixels bits are y= (1001000). Hamming matrix for *p*=3 and the process to embed message are in Figure 4.6.

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$H.y^T = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \neq \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Figure 4.6 Matrix Embedding Calculations Using Hamming Codes

As a result, 3rd bit of cover pixels group's should change. After embedding that 3 message bits in 7 pixels, the LSB of these pixels are ŷ= (1011000). To decode the *p*-length secret message, multiply 2^p-1 cover pixels group and 2^p-1 length Hamming codes.

Table 4.9 shows results of needed pixel number and modification number to hide 1024 message bits for splitting different p-length groups.

Table 4.9 Matrix Embedding Using Hamming Codes with Different P-lengths

P :	1	2	3	4	5	6	7	8	9	10
N. Pixel *	1024	1536	2389	3840	6348	10752	18578	32640	58140	104760
Modified	512	384	298.66	240	198.40	168	145.14	127.50	113.55	102.30
Ratio **	0.50	0.25	0.125	0.0625	0.0313	0.0156	0.0078	0.0039	0.0020	0.0009

*Needed pixel size

**Need pixel size / Modified Pixels

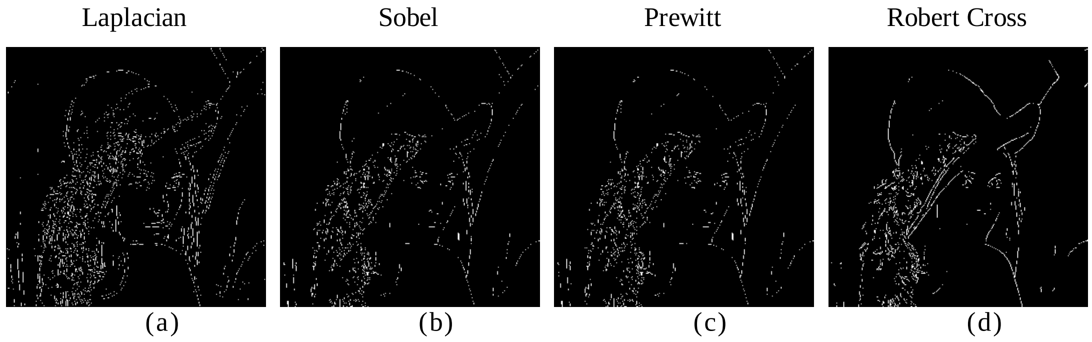
4.7 Hiding Information Randomly Pixels' LSB

The basic method to hide data using LSB insertion method sequentially is not very effective and not secure and not resistant to visual attacks on some images. So, spreading the secret bits on image using pseudo random number generator is advisable, using a method to select a position once in a time [44]. If the attacker has the stego image, and know the distribution technique or, has the software, attacker can easily extract the hidden message from stego image. Using stego key for distribution bits in cover image randomly makes it harder to extract message.

4.8 Hiding Information into Edges of Images

Edge detection is the process to find where the image brightness changes sharply such as corners, thin lines, textures, and edges. There are different filters to find edges like Laplacian, Sobel, Prewitt, Robert Cross, and Canny methods.

These methods have different matrices to find the edges. Figure 4.7 shows the results of the filters to an image.



Laplacian

Sobel

Prewitt

Robert Cross

(a)

(b)

(c)

(d)

Figure 4.7 Edge Filter Results

Kh. Manglem Signh *et al.* [45] have described a method, which does not need to have cover image to extract hidden data, to hide data on the edges of image using Roberts cross operator in 2007. Because of the pixels on edges appear to be much brighter or dimmer than neighbors, the suspicion of the present of message bits on edges will be less for an attacker. Edges can be found easily using detection filters. Defining the threshold value is the most important part of this method. Because after applying edge filter to cover image and embedding secret data into edges, for stego image edges may be changes. So, the right threshold value makes it possible to have the same edges on stego image, after changing the edge bits on cover image.

x_1	x_2
x_3	x_4

Figure 4.8 Robert Cross Convolution Mask

$$D = |G_x| + |G_y| \text{ where } G_x = x_3 - x_2,\ G_y = x_1 - x_4 \tag{4.8}$$

$$x_1 = \begin{cases} x_1 + 1, D \geq \theta \ \&\ G_x \geq x_1 \ is\ even \ \&\ MB = 1 \\ x_1 - 1, D \geq \theta \ \&\ G_x < x_1 \ is\ even \ \&\ MB = 1 \\ x_1 + 1, D \geq \theta \ \&\ G_x \geq x_1 \ is\ odd \ \&\ MB = 0 \\ x_1 - 1, D \geq \theta \ \&\ G_x < x_1 \ is\ odd \ \&\ MB = 0 \\ x_1, otherwise \end{cases} \tag{4.9}$$

$$x_2 = \begin{cases} x_2 - 1, D \geq \theta \ \&\ G_y \geq x_2 \ is\ even \ \&\ MB = 1 \\ x_2 + 1, D \geq \theta \ \&\ G_y < x_2 \ is\ even \ \&\ MB = 1 \\ x_2 - 1, D \geq \theta \ \&\ G_y \geq x_2 \ is\ odd \ \&\ MB = 0 \\ x_2 + 1, D \geq \theta \ \&\ G_y < x_2 \ is\ odd \ \&\ MB = 0 \\ x_2, otherwise \end{cases} \tag{4.10}$$

x_3 can be calculated with the same formula to calculate $x_{1.}$

x_4 can be calculated with the same formula to calculate $x_{2.}$

4.9 DCT Based Information Hiding on JPeG Images

The .jpeg image format uses the discrete cosine transform to reduce data size. It causes the loss of the image quality. But it helps to hide data without big significant changes on image depending on the chosen quality level. As described before at Appendix A. in detail, the steps of .jpeg compression and embedding data [46]:

1. Convert image from RGB format to YUV format
2. Split image into 8x8 pixel blocks
3. Calculate the DCT coefficients for each block
4. Use quantization matrix to reduce the data size
5. Replace the lowest order coefficient bits all greater than zero with hidden bits
6. Send the modified coefficients to the Huffman coder.

Only hiding 1 byte data is possible into 8x8 blocks using DCT compression.

4.10 Patchwork: A Statistical Approach

The patchwork algorithm is a statistical approach to invisibly embed data in a cover image that has a Gaussian distribution using pseudo random generator given by Bender *et al* [47]. After two patches are chosen pseudo randomly in cover image as the first A, and the second B, patch A is lightened and patch B is darkened. In other words, while intensities of the pixels in the one patch are decreased with a value between 1 and 5, the pixels of the other patch are increased by the same value. The average luminosity does not change and the contrast changes in that patch are unnoticeable. The disadvantage is low embedded data rate one-bit signature per image so it is more useful for low bit-rate applications as the digital watermark.

How this algorithm works:

Take any two points, A and B, chosen at random in an image. The brightness at point A is A_i and at B is B_i.

Increase the brightness of pixels in patch A_i by an amount ∂ between 1 and 5.

Decrease the brightness of pixels in patch B_i by an amount ∂ between 1 and 5.

After the repeating n times this algorithm to embed data, S = 0.

$$S_n = \sum_{i=1}^{n} S_i = \sum_{i=1}^{n} a_i - b_i \tag{4.11}$$

The expected value for S_n is:

$$S_n = n \times S = n \times 0 = 0 \tag{4.12}$$

The patch shape is an important part of this algorithm. If the patch will place in higher frequencies the Figure 4.9 (a) is suitable with sharp edges. Using that kind of patch will make the distortion hard to detect. If the cover image format is JPEG, placing a patch's energy in low frequencies is better and the patch shown on Figure 4.9 (b) is suitable for this situation. Figure 4.9 (c) shows a sharp-edged patch, spreading the patch energy around the frequency spectrum.

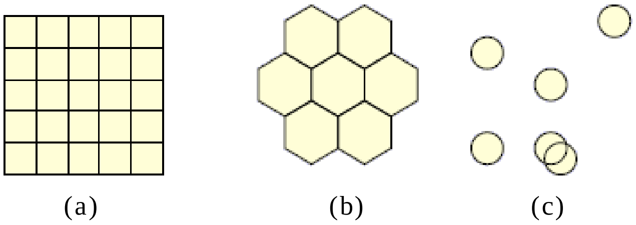


Figure 4.9 Sample Patches

4.11 Bit Plane Complexity Segmentation (BPCS) Steganography

BPCS-Steganography technique is defined by Eiji Kawaguchi and Richard O. Eason in 1997 to embed secret data in a 24-bit BMP format image and in an 8-bit indexed color image. The most important advantage of this technique is the high capacity to embed data. This method can hide information around %50 of true color image. The bit-planes of an image are used for data embedding [48].

Assume that the cover image format is 24-bit bmp. Red, Green, and Blue channels are divided into bit-planes as,

P= (PR1, PR2, PR3, PR4, PR5, PR6, PR7, PR8,...
(PG1, PG2, PG3, PG4, PG5, PG6, PG7, PG8,...
(PB1, PB2, PB3, PB4, PB5, PB6, PB7, PB8)

PR8, PG8, PB8 are the LSB (Least Significant Bit)

PR1, PG1, PB1 are the MSB (Most Significant Bit).

A set of 24 binary pictures is created for each bit plane in same color channel.

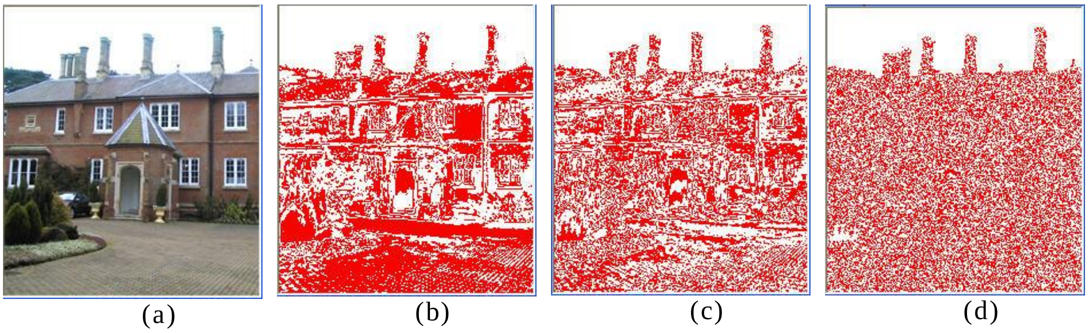
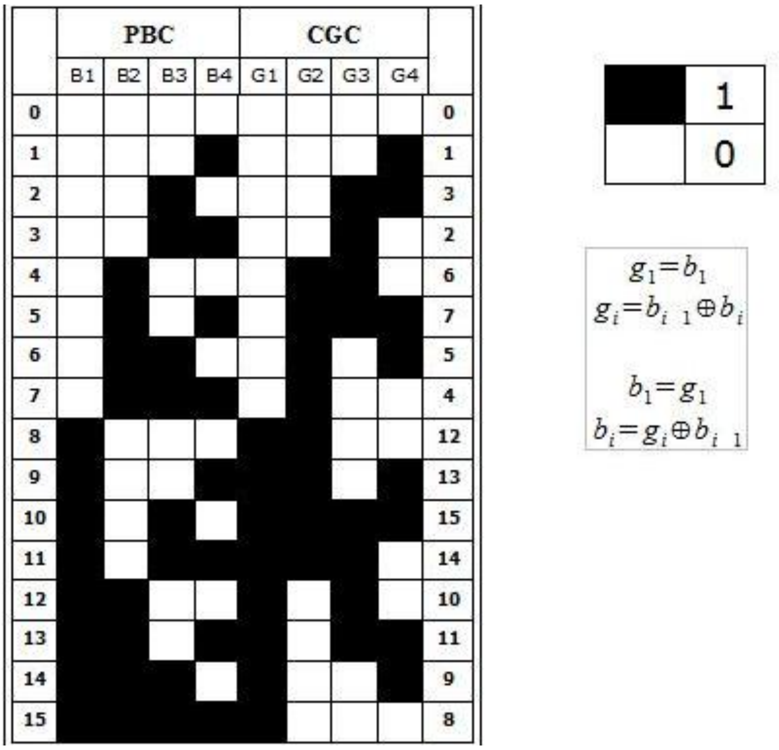


Figure 4.10 BPCS on Different Bit Channels

(a) Original picture, (b) Red plane Third Bits, (c) Red Plane Fourth Bits, (d) Red Plane Sixth Bits

Complexity of each bit-plane pattern increases from the MSB to the LSB. Then these bit-planes are converted from PBC (Pure Binary Code) to CGC (Canonical Gray Code). BPCS Steganography method works after converting bit-planes from PBC to CGC. Figure 4.11 shows how to convert PBC to CGC.



	PBC				CGC				
	B1	B2	B3	B4	G1	G2	G3	G4	
0	□	□	□	□	□	□	□	□	0
1	□	□	□	■	□	□	□	■	1
2	□	□	■	□	□	□	■	■	3
3	□	□	■	■	□	□	■	□	2
4	□	■	□	□	□	■	■	□	6
5	□	■	□	■	□	■	■	■	7
6	□	■	■	□	□	■	□	■	5
7	□	■	■	■	□	■	□	□	4
8	■	□	□	□	■	■	□	□	12
9	■	□	□	■	■	■	□	■	13
10	■	□	■	□	■	■	■	■	15
11	■	□	■	■	■	■	■	□	14
12	■	■	□	□	■	□	■	□	10
13	■	■	□	■	■	□	■	■	11
14	■	■	■	□	■	□	□	■	9
15	■	■	■	■	■	□	□	□	8

■	1
□	0

$$g_1=b_1$$
$$g_i=b_{i\ 1}\oplus b_i$$

$$b_1=g_1$$
$$b_i=g_i\oplus b_{i\ 1}$$

Figure 4.11 Converting Between PBC and CGC

If a bit in the 3rd least plane is changed from 0 to 1 in PBC

As in Figure 4.11:

0 (000) → 4 (100), 1 (001) → 5 (101), 2 (010) → 6 (110), 3 (011) → 7 (111)

Changing 3rd bit in PBC increases the all integer value by 4 and it will cause a blocking effect.

Converting from PBC to CGC, changing 3rd bit 0 to 1, and converting back to PBC will cause of the average change 4. This will not cause a blocking effect remarkably.

How the algorithm works to encode data is described as follows [49]:

1. First step is to covert cover image format from PBC to CGC.
2. Split each bit-plane using a threshold value, default is 0.3, into noise-like and informative regions.
3. Group the secret file as secret blocks.

4. If a block of cover image is less than the threshold, conjugate it to have a more complex block.
5. Embed or replace each secret block into the noise-like regions of the bit-planes. If the block is conjugated, then record this fact in a “conjugation map.”
6. Embed the conjugation map.
7. Convert the stego image from CGC back to PBC

CHAPTER 5

STEGANALYSIS

Steganalysis is the art and science to detect whether a given medium has a hidden message in it and to judge the performance of steganographic techniques. The wide variation of data embedding algorithms makes steganalysis a tough mission for images. Cover image and its stego image always differ from each other, because data embedding period makes changes on original images. Steganalysis techniques can be divided into two parts. First one is, if analyst does not have the original image and does not know which specific data hiding algorithm is used, called as blind steganalysis. If the analyst has cover media or knows the embedding method is the second situation. Analysts often use statistical digital signal processing to detect data within images. On the steganography side, it is important to improve algorithms and find new undetectable methods using steganalysis techniques.

Steganalysis is especially also important in the security aspect, namely monitoring a user's communication with the outside world. In the age of Internet, images are sent via email or by posting on websites. Detecting whether or not data is hidden in the images will allow furthering analyzing the suspicious images in order to find what the hidden message is.

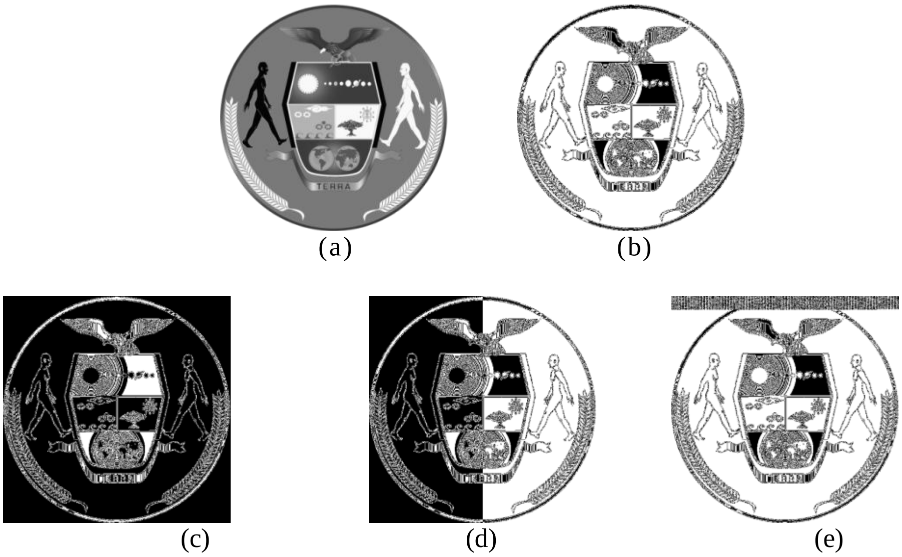
In this part, some important blind steganalysis techniques are explained. If analyst has the original image, it is easy to detect if there is a manipulation on image calculating MSE, PSNR values or checking histograms.

Firstly, visual attack is described. Visual attack is mostly usable for grayscale images. HVS can detect if there is a hidden message in image using visual attack [50] [51]. Other methods use statistical properties to detect stego images.

Secondly, statistical attack using chi-square analysis is described. This analysis technique, which is successful for sequentially embedding secret data to LSB of channels, is defined by Westfeld [51]. Another steganalysis method developed by Fridrich et al. [53] for detection of LSB embedding in 24-bit color images, called as the Raw Quick Pairs (RQP) method. Fridrich also proposed RS (Regular and Singular groups) method, which is suitable for color and grayscale images if the hidden data is embedded randomly [54]. There are more steganalysis methods such as Sample Pair Analysis and Difference Image Histogram those are not discussed in this thesis. Sample Pair Analysis (SPA) is defined by Dumitrescu S. et al. which works well if the embedding ratio is more than %3 and can give an estimated length of hidden message [55][58]. The idea of SPA method is based on finite-state machine theory; these states are selected as multi sets of sample pairs. There are some inherent relations between sample pairs. After embedding message to LSB of pixels, statistic relations of multi sets will change. This method uses these statistics of multi sets to estimate hidden message length. DIH (Difference Image Histogram) steganalysis method is proposed by Zhang T. and Ping X. [59]. This method can detect the existence of message and estimate the length of message if the message is embedded using LSB Replacement method sequentially or randomly in images. DIH gives better results than RS if the embedding ratio is more than %50.

5.1 Visual Attack

This is very simple method to discover if there is a hidden message in image or not. The purpose of visual attack is to detect whether there is a potential message or not in image by HVS [50]. Effectively this method is usable with grayscale images or color images which have white or black area. This method eliminates all 7 high level bits for each pixel and uses LSB to create a new picture. If LSB of pixel is 1 it becomes maximum value, (i.e. for 8-bit pictures it is 255). If LSB of pixel is 0, this value stays same for new picture [51]. The LSB of the original image becomes visible, good enough for a visual check. Figure 5.1 shows an example of visual attack. Picture (a) is the cover image, (b) is the form of after visual attack to original picture, (c) is the worst case, if all LSB of pixels change, (d) is the half of c, and (e) is the form of which has 2kb hidden message.

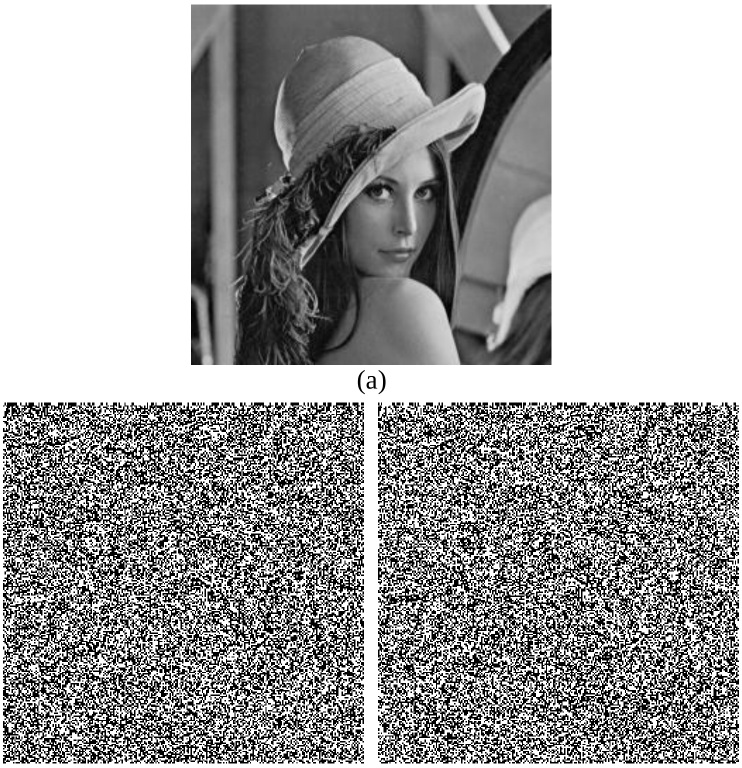


(a) (b)

(c) (d) (e)

Figure 5.1 Effects of Visual Attack on Images

Using visual attack it is possible to detect whether there is a hidden message in picture or not. However, by changing all LSB, visual attack does not give any clue in same grayscale pictures. As seen in Figure 5.2, it has different 230 gray tones, and it does not contain any area that is full black or white color. After, creating binary image depending on last bits of pixels, if the image contains hidden data and if there is a block on image has the same color; new image on that block will be noisy. Because, statistical analysis shows that after embedding data into last bits about half of the last bits changes. If there is no hidden data that block will be completely black or white. Figure 5.2 (b) is the form original picture after visual attack and (c) is the form of original picture that all LSB are swapped.



(a)

(b) (c)

Figure 5.2 Visual Attack to Grayscale Image

(a) Cover image, (b) visual attack to original image, (c) visual attack to stego image

It is possible to detect if the image contains hidden data or not for color images using filter. Figure 5.3 shows an example for color images.

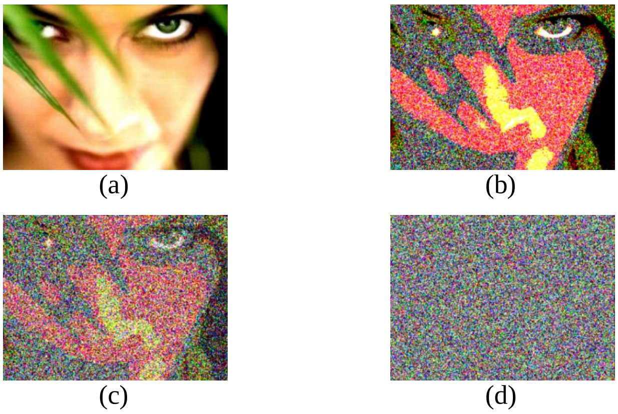


Figure 5.3 Visual Attack to Color Image

(a) Cover image, (b) Filtered of a, (c) Filtered image, contains data %50 of capacity, (d) Filtered image, contains data %100 of capacity

5.2 Statistical Attack: Chi-Square Analysis

This statistical attack was published in 2000 by Andreas Westfeld [51]. Changing least significant bits in cover image creates new value which is closest to original one. If the channel value for original image is 2, after hiding a bit in that channel value stays same or becomes 3. These pairs are called PoV. The frequencies of both values of each PoV become equal, if the bits used for hiding data at the least significant bits are equally distributed. The purpose of the statistical attack using chi-square analysis is to compare the distribution of PoV in stego image with the theoretically expected frequency in original image.

Let we have k categories which are all palette indices. Each observation falls in only one category. The minimum expected frequency of odd values of PoV must be greater than 4.

After embedding an equally distributed message, the expected frequency in i is;

$$n_i^* = \frac{\left|\left\{\text{colour}\left|sortedIndexOf\left(colour\right) \in \left\{2i, 2i+1\right\}\right.\right\}\right|}{2} \tag{5.1}$$

The measured frequency is in a random sample

$$n_i = |\{color|sortedIndexOf(color) = 2i\}| \tag{5.2}$$

The X^2 statistic is given as

$$X_{k-1}^2 = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^*} \text{ with } k-1 \text{ degrees of freedom} \tag{5.3}$$

Formula to calculate the probability p if the distributions of n_i and n_i^* are equal.

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma\left(\frac{k-1}{2}\right)} \int_0^{X_{k-1}^2} e^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} dx \tag{5.4}$$

How does this attack works?

Assuming that the part of picture where the secret message will be hidden has the same pixel values as 15. When secret message is embedded there, about half of the LSB will change. So, about %50 of pixels stay as 15 and the other half of pixels are 14. 15 and 14 are Pairs of Values (PoV). Table 5.1 shows the pair of values for last 3 bits. If last 3 bits are used to encode, there are 4 PoV. If 8-bits represent a pixel and LSB is used to encode data there will be 128 Pair of Values.

Table 5.1 Bit Representation of PoV

000 001	First Pair of Values.
010 011	Second Pair of Values.
100 101	Third Pair of Values.
110 111	Fourth Pair of Values.

This method assumes PoV almost have the same frequency. Firstly, calculates total of PoV and constructs a table. Then constructs a second table which consists of PoV values how to be and compares these tables using chi-square test to find an answer whether this table is significantly different or not. Table 5.2 first column shows a sample PoV table of original image and Table 5.2 second column shows a table of stego image. Figure 5.4 used for this

Table 5.2 Original PoV and Stego PoV.

…	…
246 : 439	**246 : 543**
247 : 668	**247 : 564**
248 : 799	248 : 5381
249 : 19113	249 : 14531
250 : 3471	**250 : 2827**
251 : 2017	**251 : 2661**
252 : 496	252 : 353
253 : 129	253 : 272
254 : 92	**254 : 38760**
255 : 78478	**255 : 39810**

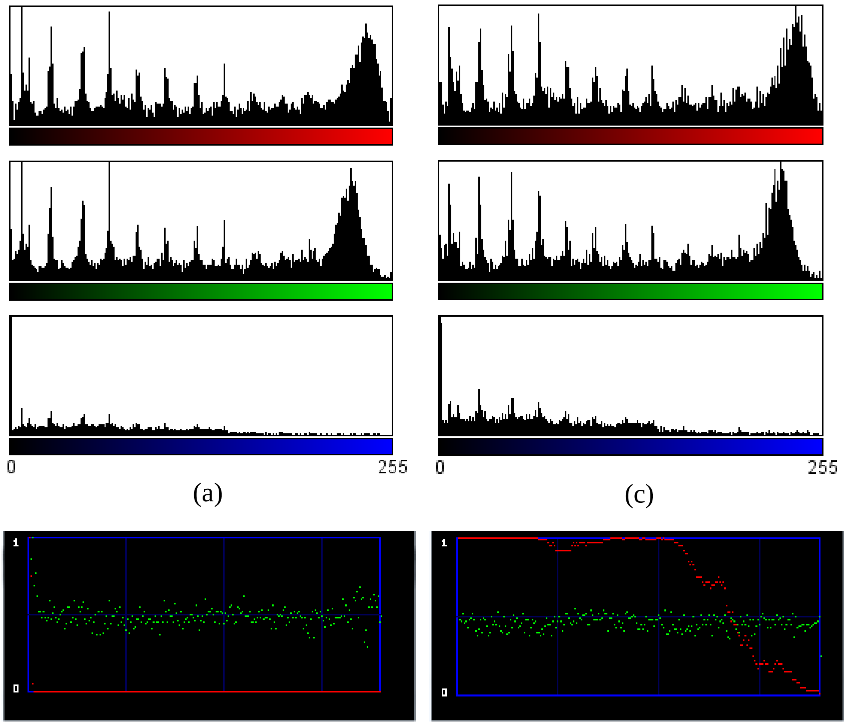
Table 5.2 first column shows the distribution of pixel values in original grayscale image and second shows the stego image which hide the data about half of the capacity of cover image. As seen in above frequency of PoV gets closer after encoding.

Software is used to test and see the results of chi-square attack to stego image written by Guillermito El Loco [52]. This program reads each time 126 bytes of data and makes calculation then plots the results on a graph. First reads bytes from 0 to 128 then reads 128 bytes more and tests from bytes 0 to 256. The red curve is the result of the chi-square test. The probability for a random embedded message is high, if it is close to one. The second output is the average value of the LSB on the current block of 128 bytes. So, this green curve will stay around 0.5, if there is a random message embedded. Every vertical blue line represents 1024 byte of embedded data block on the graph.

Figure 5.4 Çankaya University Logo

For instance, in Figure 5.4, File Size is 100x98. Capacity of this image for LS2B (Least Significant 2 bits) embedding is 100x98x3=29400 bits (3,58kb), 29400 bits x 2=58800 (7,17kb) bits for last 2 bits.

Figure 5.5 shows the histograms of original image and stego image after embedding 4.05kb data, and results of chi-square test for both images.



(a) (c)

(b) (d)

Figure 5.5 Histograms and Chi-Square Analysis

(a) Color Histogram of cover image, (b) Graph of chi-square analysis to cover image, (c) Color Histogram of stego image, (d) Graph of chi-square analysis to Stego image

5.3 Raw Quick Pairs (RQP) Steganalysis

RQP is, based on analysis of close pairs of colors in 24 bit color images, developed by Fridrich. If the secret message is embedded to LSB, it works well as long as the number of unique colors in the cover image is less than %30 of the number of pixels. Unless the number of unique colors exceeds about %50 percent of the number of pixels, RQP method can provide a rough estimate of the secret message size [53].

Suppose that the number of unique colors in an image is U.

P is the number of close color pairs in the image palette for U.

Two neighbor pixels (R1, G1, B1) and (R2, G2, B2) are close if;

$$\begin{aligned}&|R_1 - R_2| \leq 1 \wedge |G_1 - G_2| \leq 1 \wedge |B_1 - B_2| \leq 1\\&\text{or}\\&(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2 \leq 3\end{aligned} \tag{5.5}$$

The number of all pairs of colors is greatest than the number of closest pairs of color:

$$\binom{U}{2} \geq P \tag{5.6}$$

The ratio R, between these values is;

$$R = \frac{P}{\binom{U}{2}} \tag{5.7}$$

The idea of this method is, the number of unique color increase after embedding a message to picture. If there is no hidden message, the number of close pairs relative to the number of all possible pairs of colors gets smaller than an image that has an embedded message. Embedding a message to an image does not change the ratio R significantly, if stego image has a large image. On the other hand, the ratio R increases significantly, if the image does not contain a secret message.

How the Detection Algorithm Works:

1. Using equation (5.7), calculate the ratio *R* between the number of all pairs of close colors *P* and the number of all color pairs.
2. Embed randomly a test message in pixels changing LSB. Test message length is α3MxN.
3. Calculate *U'*, *P'*, and *R'* for new image.
4. Take the ratio of *R'* / *R*

If there is no hidden message or it is too short, calculated two ratios will be very close to each other.

If there is a large hidden message the two ratios will be sufficiently different.

5.4 Regular and Singular Groups (RS) Steganalysis

Fridrich *et al.* also proposed a powerful RS (regular and singular groups) method [54]. This method uses the statistics of the alterations of regular groups and singular groups in the image to estimate the embedded length accurately. It is suitable for color or gray-scale images if the secret message is embedded randomly.

Let the cover image size MxN

P is the set of pixel values.

For an 8-bit grayscale image; P = {0, 1, 2,…, 255 }

Using a discrimination function $f(x_1,...,x_n) \in R$, to a group of pixels $G = (x_1,...,x_n)$ capture the spatial correlations.

$$f(x_1, x_{2,}, \dots, x_n) = \sum_{i=1}^{n-1} |x_{i+1} - x_i| \tag{5.8}$$

$$F_1(x) = \begin{cases} x+1, x\ even \\ x-1, x\ odd \end{cases} \\ \ni x \in \{0, \dots, 255\} \tag{5.9}$$

$$F_{-1}(x) = \begin{cases} x+1, x\ odd \\ x-1, x\ even \end{cases} \\ \ni x \in \{-1, \dots, 256\} \tag{5.10}$$

$$F_0(x) = \{x, for\ all\ x \tag{5.11}$$

Three types of pixel groups:

If *f*(F(G)) > *f*(G) => G is Regular (R)

If *f*(F(G)) < *f*(G) => G is Singular (S)

If *f*(F(G)) > *f*(G) => G is Unchanged (U)

A mask is defined as n-tuple which consists of M (-1,0,1)

Firstly, divide image to n groups then divide each group n into k-tuples and define the mask ,M, such that $M_1, M_{2,\dots}, M_k$ Є {-1,0,1} to calculate to as $F_{M1}, F_{M2,\dots}, F_{Mk}$. Then compare *f*($F_M$(G)) and *f*(G) for all groups to compute the number of R,S,U groups. For –M (1,0,-1) calculate *f*(F_{-M}(G)), and find the numbers of R,S,U groups. Statistically, number of regular groups is larger than singular group for cover images. After applying mask M to cover image, which has no secret message, R_M and R_{-M}, S_M and S_{-M} are approximately equal. If the image hides a message, R_M and S_M get closer to each other. On contrary, the difference between R_{-M} and S_{-M} increases, R_M and S_M is approximately equal after flipping the half of pixels' LSB [57]. The expected values of R_M and S_M after all possible LSB randomizations are given in [17] as shown in Figure 5.6. *p* is the message length (in percents of pixels), assume that LSB Replacement method is used to embed message. So, almost half of the message bits are changed (Chapter 4.1).

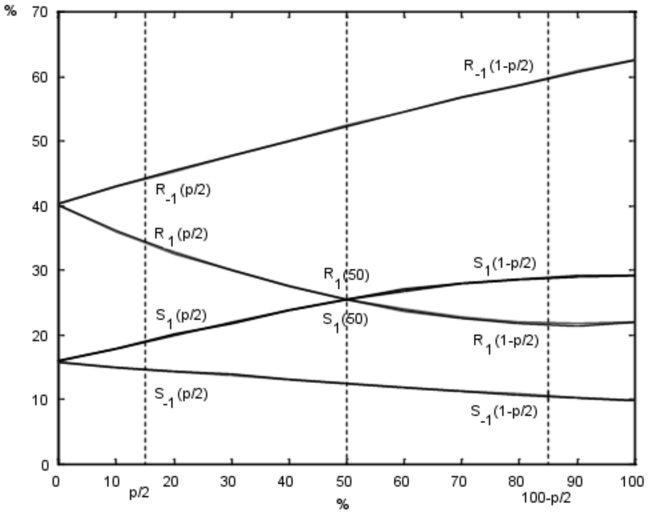


Figure 5.6 RS Diagram

The x-axis is the percentage of pixels with flipped LSBs, y-axis is the relative number of R and S groups with masks M & -M, M=[0 1 1 0]

How this method estimates a message length:

1. Find the number of R,S,U groups for M and –M

   (R_1(p/2), S_1(p/2), R-$_1$(p/2), S-$_1$(p/2))

2. Flip all the LSB of pixels and recalculate for M and –M

   (R_1(1-p/2) S_1(1-p/2) R-$_1$(1-p/2) S-$_1$(1-p/2)

3. Randomly change all the LSB of pixels and recalculate for M and –M

   (R_1(50), S_1(50)).

4. Solve this equations to find message length

$d_0=R_M(p/2)-S_M(p/2)$, $d_1= R_M(1-p/2)-S_M(1-p/2)$,

$d_{-0}= R_{-M}(p/2)-S_{-M}(p/2)$, $d_{-1}= R_{-M}(1-p/2)-S_{-M}(1-p/2)$

$2(d_1+d_0)z^2+(d_{-0}-d_{-1}-d_1-3d_0)z+d_0-d_{-0}=0$

p=z/(z-1/2) message length

Manoharan S. tested RS steganalysis method for natural images how it is efficient to guess hidden message lengths [56]. LSB Replacement, LSB Random Replacement, LSB Matching, LSB Random Matching methods were used to embed data. Results show that RS steganalysis method is mostly effective for LSB Replacement method. LSB Random Matching method is the most resistant method to RS steganalysis. This steganalysis method can guess about %50 length of hidden message with large message sizes, encoded using LSB Random Matching method.

CHAPTER 6

EXPERIMENTAL RESULTS

In this chapter, the effect of encoding and compression methods for hidden message, and LSB embedding methods are tested.

6.1 Coding and Compression Algorithms

S-Tool4 [60], most known steganography tool, uses a compression algorithm.

S-Tool4 is referred as the best algorithm (when compared to TurkSteg, InfoStego, Stego_LSB, Hermetic_Stego, F5, Hide and Seek) in Şahin's thesis [63]. In this part we compared S-Tool4 with popular compression software WinRAR (uses LZMA2 algorithm), encoding algorithm Huffman.

Sample text file size is = 2768 bytes, Table 6.1 shows some different techniques to reduce the data size.

Table 6.1 Rate of Gain Using Compression Algorithms

Method	Compressed Size	Rate of gain
S-Tool4 highest compression	1429 bytes	%48.37
Winrar [62] highest compression	1308 bytes	%52.78
Huffman encoding	1604 bytes	%41.90
Huffman encoding + Winrar	659 bytes	%76.20

Using algorithms above ten random text files are compressed. Figure 6.1 shows the compressed file sizes. Table 6.2 summarizes gains of algorithms.

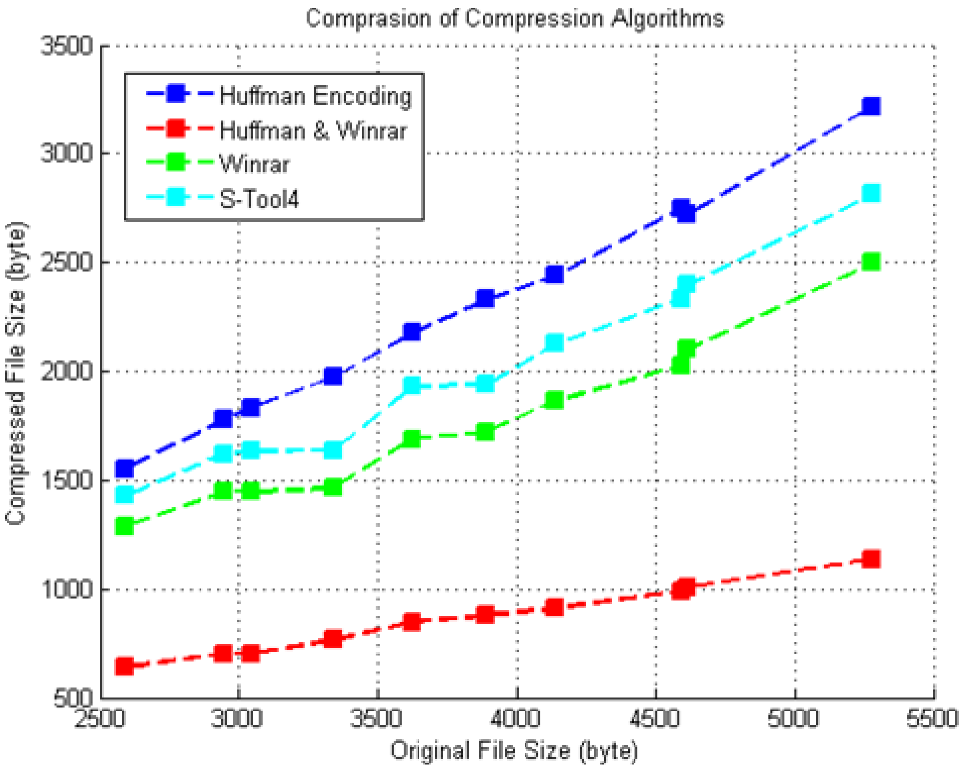


Figure 6.1 Comparisons of Compression Algorithms

Table 6.2 Gain Results of Compression Algorithms

Total Size of Original Files	38046
Total Size of files after Huffman Encoding	22763
Gain of Huffman Encoding	%40.1698
Total Size of files after S-Tool4	19852
Gain of S-Tool4	%47.8211
Total Size of files after Winrar	17541
Gain of Winrar	%53.8953
Total Size of files after Huffman Encoding & Winrar	8573
Gain of Huffman Encoding & Winrar	77.4668

6.2 Effects of Coding and Compression Algorithms with Different Embedding Methods

This section, involves PSNR and MSE values for different LSB embedding methods. As described in Chapter 3, higher PSNR values show that the difference between stego image and cover image is less. So, Chan's method is the best way to embed bits if only the LSB of pixel will be used to hide bit.

Table 6.3 PSNR and MSE Values for Different LSB Methods

Method	File Size	MSE (R)	MSE (G)	MSE (B)	PSNR (R)	PSNR (G)	PSNR (B)
LS2B*[1] Replacement	2768	0.2640	0.2710	0.2646	53.9155	53.8013	53.9041
LSB Replacement	2768	0.1178	0.1165	0.1125	57.4181	57.4670	57.6194
LSB *[2] Replacement	1308	0.0629	0.0646	0.0634	60.1473	60.0296	60.1127
STool4	1429	0.0612	0.0635	0.0615	60.2660	60.1041	60.2393
LSB*[3] Matching	659	0.0270	0.0263	0.268	64.6785	65.4094	63.9923
Chan*[3]	659	0.0239	0.0240	0.251	70.2866	72.3412	70.4148

*[1]LS2B, the last 2 bits of a pixel is used to hide bits.
*[2] 1308 bytes size of the same data after compression.
*[3] 659 bytes, after Huffman encoding and compression

As seen in Table 6.3 to minimize the data size, using Huffman encoding method and LZMA2 algorithm for compression gives better results. LSB Matching method to LSB of cover image's pixels causes the minimum changes on images.

Then, the question is where to embed data, randomly in all image or the edges? The answer of this question depends on message size and cover image type. If the message size is enough for edge embedding method, edge embedding can be used; edge embedding method size also depends on used filter to find edges. How much the distance between edges are closer each other, finding a threshold value and hiding there, message will be harder. If the message size is high for edge embedding, randomly embedding method can be used to get a higher capacity with edge embedding method.

After deciding where to hide data, image statistics must be recorded. While embedding data in cover image, unused bits can be changed to be successful against statistical analysis.

6.3 Embedding Efficiency of LSB Methods

Figure 6.1 shows the effects of modified pixel numbers and needed pixel numbers for different LSB methods. Message length starts from 256 bits up to 4096 bits increasing by 256 bits [16].

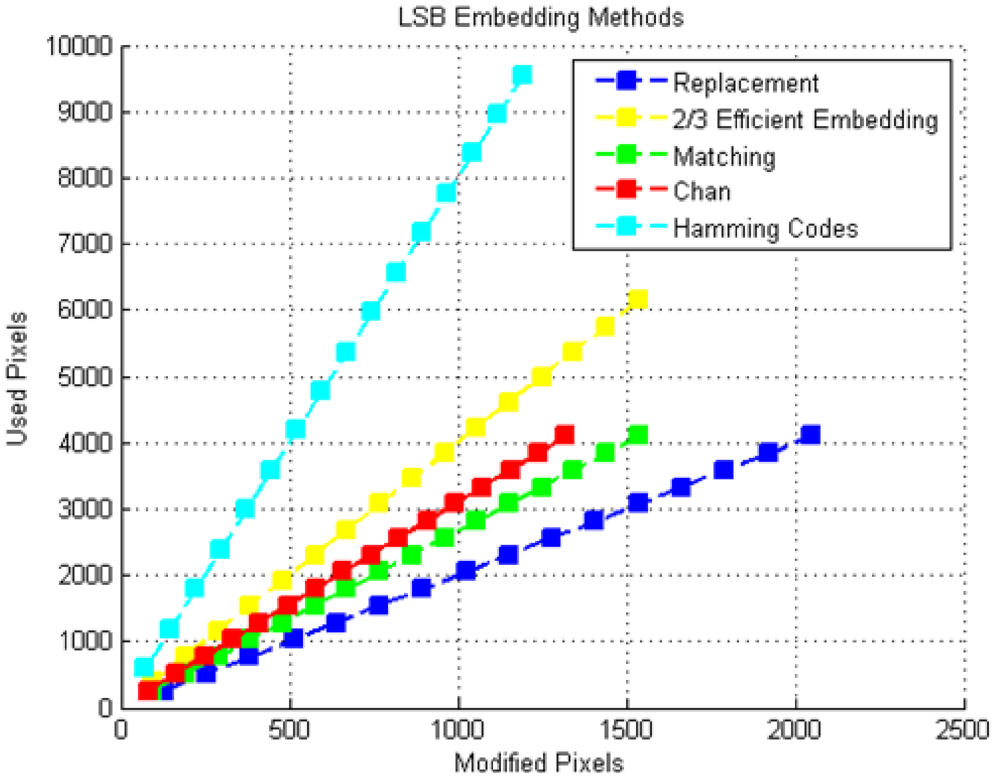


Figure 6.2 Modified Bits / Used Pixels LSB Embedding Methods

Steganalysis methods can detect stego images if the image has at least 0.002 changes over total pixel numbers [61]. So that, Table 6.3 shows needed total pixel number to hide 1024 bits data using different LSB embedding methods.

Table 6.4 Needed Pixel Number to Hide 1024 Bits with LSB Methods

Method	Replacement	Matching	Chan*	2/3 Efficient Embedding	Hamming**
Needed Pixel	256000	192000	170500	288000	58140
Modified Pixel	512	384	341	384	113.5

*Embedding Efficiency for Chan's method is 3/1

**Used data in Table 4.8 for Matrix Embedding using Hamming Codes

CHAPTER 7

CONCLUSION

This thesis explains some steganography methods for still images and their analysis. There are different techniques to hide data in still images depending on cover image format. Mostly, LSB embedding methods for uncompressed or lossless compressed image formats are explained. Remembering the purpose of steganography is to cause minimum changes in cover image to hide maximum data. These LSB embedding methods are analyzed. Also, location of embedded bits is important to robust steganalysis techniques. So, random embedding and edge embedding methods are described. How statistical analysis methods can be successful to find out stego images are explained.

There are several techniques to defeat steganalysis methods. But it is hard to be successful for all known attacks. For example, a developed method may give good results after RS analysis; however it may be detectable for SPA or etc. method. For a good steganography technique, first step is to select right cover image. Minimizing changes in cover image is another important part. So, before hiding message in cover image secret message should be compressed.

The best algorithm that is mentioned in this thesis is the Chan's Method when the size of message is large and the transmission channel is secure. If the transmission channel is not secure to avoid steganalysis methods, Matrix embedding using Hamming code may be preferred.

REFERENCES

[1] **Petitcolas, F. A. P., Anderson, J. R., Kuhn, G. M.** (1999), “*Information Hiding A Survey*”, Proceedings of the IEEE, special issue on protection of multimedia content, Vol.87 No.7 1062-1078.

[2] **Simmons, G.** (1984), “*The Prisoners' Problem and the Subliminal Channel*", CRYPTO83 Advances in Cryptology, 51-67.

[3] **Kharrazi, M., Sencar, H., Memon, N.** (2004), “*Image Steganography: Concepts and Practice*”, Lecture Notes Series, National University of Singapore.

[4] **Kipper, G.** (2004), “*Investigator's Guide to Steganography*”, Auerbach Publications.

[5] **Kessler, G. C.** (2002), “*Steganography: Hiding Data within Data*”, Windows & .NET Magazine.

[6] http://www.jjtc.com/stegdoc/sec202.html

[7] **Kessler, G. C.** (2004), “*An Overview of Steganography for the Computer Forensics Examiner*”, Forensic Science Communication, Vol.6 No.3.

[8] **Diskin, P., et. al.** (2004), “*Steganography and Digital Watermarking*”, The University of Birmingham.

[9] **Sui, X., Luo, H., Zhu, Z. L.** (2006), “*A Steganalysis Method Based on the Distribution of First Letters of Words*”, International Conference on Intelligent Information Hiding and Multimedia Signal Processing.

[10] http://www2.gsu.edu/~wwwesl/egw/jones/differences.htm

[11] http://w2.eff.org/Privacy/printers/docucolor

[12] http://en.wikipedia.org/wiki/Microdot

[13] http://en.wikipedia.org/wiki/Steganography

[14] http://thomas.gloeckler-ulm.de/fhu-old/www/stego.html

[15] **Westfeld, A., Wolf, G.** (1998), "*Steganography in a Video Conferencing System*", Proceedings of the Second International Information Hiding Workshop, Vol.1525, 32-47.

[16] **Olcay, C., Saran, N.** (2010), "*İmge İçine Bilgi Gizlemede Kullanılan LSB Yöntemlerinin Karşılaştırması*", 3. Mühendislik ve Teknoloji Sempozyumu.

[17] **Fridrich, J., Goljan, M., Du, R.** (2001), " *Reliable Detection of LSB Steganography in Color and Grayscale Images*", Proceedings of the 2001 workshop on Multimedia and security: new challenges, 27-30.

[18] **Morkel, T., Eloff, J., Olivier, M.** (2005), "*An Overview of Image Steganography*", Proceedings of the Fifth Annual Information Security South Africa Conference.

[19] http://en.wikipedia.org/wiki/Grayscale

[20] http://local.wasp.uwa.edu.au/~pbourke/texture_colour/colourspace

[21] http://www.scantips.com/palettes.html

[22] http://www.jpeg.org/jpeg2000

[23] http://www.scantips.com/basics09.html

[24] http://www.w3.org/TR/PNG

[25] **Wang, Z., Bovik, A.** (2009), "*Mean Squared Error: Love it or Leave it? A New Look at Signal Fidelity Measures*", IEEE Signal Processing Magazine Vol.26, 98-117.

[26] **Gupta, G., Aggarwal, H.** (2009), "*Digital image Watermarking Using Two Dimensional Discrete Wavelet Transform, Discrete Cosine Transform and Fast Fourier Transform*", International Journal of Recent Trends in Engineering, Vol.1, No.1.

[27] **Parthasarathy, A.** (2007), "*Improved Content Based Image Watermarking*", IEEE Broadcast Technology Society Vol.53, 468-479.

[28] http://www.dpreview.com/learn/?/Glossary/Digital_Imaging/Histogram_01.htm

[29] http://rsbweb.nih.gov/ij

[30] http://www.zlib.net/feldspar.html

[31] http://www.ou.edu/class/digitalmedia/articles/CompressionMethods_Gif_Jpeg_PNG.html

[32] http://oldwww.rasip.fer.hr/research/compress/algorithms/fund/lz/lz77.html

[33] http://oldwww.rasip.fer.hr/research/compress/algorithms/fund/lz/lz78.html

[34] http://marknelson.us/1989/10/01/lzw-data-compression

[35] **Huffman, D.** (1952), **"***A Method for the Construction of Minimum Redundancy Codes",* Proceedings of IRE, Vol.40, No.9, 1098-1101.

[36] **Wu, D. C., Tsai, W. H.** (2003), "*A Steganographic Method for Images by Pixel Value Differencing*", Pattern Recognition Letters, Vol.24, 1613–1626.

[37] **Johnson, N. F., Katzenbeisser, S.** (2000), ”*A Survey of Steganographic Techniques*”, Information Hiding Techniques for Steganography and Digital Watermarking, ISBN-13: 978-1580530354, 43-78.

[38] **Chan, C.S., Chang, C.C.** (2007), "*A Survey of Information Hiding Schemes for Digital Images*", IJCSES International Journal of Computer Sciences and Engineering Systems, Vol.1, No.3.

[39] **Mielikainen, J.** (2006), “*LSB Matching Revisited*”, IEEE Signal Processing Letters, Vol.13, No.5, 285-287.

[40] **Chan, C.S.** (2009), ”*On Using LSB Matching Function for Data Hiding in Pixels*”, Fundamenta Informaticae, Vol.96, 49–59.

[41] www.sti.uniurb.it/events/fosad08/slides/ker-slides-part2.pdf

[42] **Fridrich, J., Soukal, D.** (2006), ”Matrix *Embedding for Large Payloads*”, Security, Steganography, and Watermarking of Multimedia Contents VIII. Proceedings of the SPIE, Vol.6072, 727-738.

[43] **Fridrich, J., Lisonek, P., Soukal, D.** (2007), “*On Steganographic Embedding Efficiency*”, Proceedings of the 8th Information Hiding Conference, Lecture Notes in Computer Science Vol.4437, 282-296.

[44] **Amin, M.M., et. al.** (2003), “*Steganography: Random LSB Insertion Using Discrete Logarithm*”, Proceedings of the 3rd International Conference on Information Technology in Asia, 234–238.

[45] **Signh, M.K., et. al.** (2007), "*Hiding Encrypted Message in the Features of Images*", IJCSNS International Journal of Computer Science and Network Security, Vol.7, No.4.

[46] **Westfeld, A.** (2001) “*F5-a Steganographic Algorithm: High Capacity Despite Better Steganalysis*”, Proceedings of the 4th International Workshop on Information Hiding.

[47] **Bender, W., et. al.** (1996), "*Techniques for Data Hiding*", IBM Systems Journal, Vol.35, No.3-4, 313-336.

[48] **Kawaguchi, E., Eason, O.** (1999), "*Principle and Applications of BPCS-Steganography*", Proceedings of SPIE The International Society Vol.3528, 464-473.

[49] **Beaullieu, S., Crissey, J., Smith, I.** "*BPCS Steganography*", University of Texas.

[50] **Martin, F., Stripf, H.S.** (2005), "*Visual Steganalysis of LSB-Encoded Natural Images*", Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA'05) Vol.2, 746-751.

[51] **Westfeld, A., Pfitzmann, A.** (2000), "*Attacks on Steganographic Systems: Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools– and Some Lessons Learned*", 3rd International Workshop on Information Hiding.

[52] http://www.guillermito2.net/stegano/tools/index.html

[53] **Fridrich, J., Du, R., Meng, L.** (2000), "*Steganalysis of LSB Encoding in Color Images*", Proceedings IEEE International Conference on Multimedia and Expo, New York City, NY.

[54] **Fridrich, J., Goljan, M., Du, R.** (2001), "*Reliable Detection of LSB Steganography in Grayscale and Color Images*", Proceedings. of the ACM Workshop on Multimedia and Security, Ottawa, Canada, October, 27-30.

[55] **Dumitrescu, S., Wu X., Wang, Z.** (2003), "*Detection of LSB Steganography via Sample Pair Analysis*", IEEE Transactions on Signal Processing, Vol.51, No.7, 1995-2007.

[56] **Manoharan, S.** (2008), "*An Empirical Analysis of RS Steganalysis*", the 3rd International Conference on Internet Monitoring and Protection, 172-177.

[57] **Fridrich, J., Goljan, M.** (2002), "*Practical Steganalysis-State of the Art*", Proceedings of the SPIE Photonics West, Vol.4675, 1-13.

[58] **Luo, X., Liu, F.** (2007), “*A LSB Steganography Approach Against Pixel Sample Pair Steganalysis*”, International Journal of Innovative Computing, Information and Control, Vol.3, No.3, 575-588.

[59] **Zhang, T., Ping, X.** (2003), “*Reliable detection of LSB steganography based on the difference image histogram”*, Proceedings of the IEEE ICSAAP, Part III, 545-548.

[60] ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip

[61] **Ker, A.D.** (2004), “*Quantitative Evaluation of Pairs and RS Steganalysis*” Security, Steganography, and Watermarking of Multimedia Contents VII. Proceedings of the SPIE, Vol.5306, 83-95.

[62] http://www.win-rar.com

[63] **Şahin, A.** (2007), “*Görüntü Steganografide Kullanılan Yeni Metodlar ve Bu Metodların Güvenilirlikleri*”, Ph.D. Thesis, Edirne University.

[64] http://www.cs.auckland.ac.nz/compsci708s1c/lectures/jpeg_mpeg/jpeg.html

[65] http://www.opennet.ru/docs/formats/jpeg.txt

APPENDIX A

JPEG Lossless and Lossy Compression

1.1 Lossless Compression

Lossless encoding, used for JPEG, gives around 2:1 compression on image. Each color channel is encoded separately. Three pixel values and selection values give the prediction value. Difference of actual value and prediction value, and selection value is stored using Huffman encoding or arithmetic operations to encode pixel value of X [64].

Table A.1 Lossless Compression

Pixels

		C	B	
		A	X	

Selection Value	Prediction
0	No Prediction
1	A
2	B
3	C
4	A+B-C
5	A+((B-C)/2)
6	B+((A-C)/2)
7	(A+B)/2

1.2 Lossy Compression

JPEG image format uses DCT method to compress the image and to reduce the data size. Firstly, if an image format is RGB, it is converted to YCbCr color space, consists of two chroma components (Cb: Blue/Yellow and Cr: Red/Green), and one luma channel which represents brightness of image (Y). Figure A.2 shows Y, Cb, and Cr color spaces of an image and Figure A.1 shows the RGB color cube and YCbCr color space. The transformation between YCbCr and RGB is based on the following equations.

RGB to YUV Conversion

$Y = +0.299R + 0.587G + 0.114B$

$Cb = +0.492(B\ Y) = -0.147R - 0.289G + 0.436B$

$Cr = +0.877(R\ Y) = +0.615R - 0.515G - 0.100B$

YUV to RGB Conversion

$B = 1.164(Y - 16) + 2.018(U - 128)$

$G = 1.164(Y - 16) - 0.813(V - 128) - 0.391(U - 128)$

$R = 1.164(Y - 16) + 1.596(V - 128)$

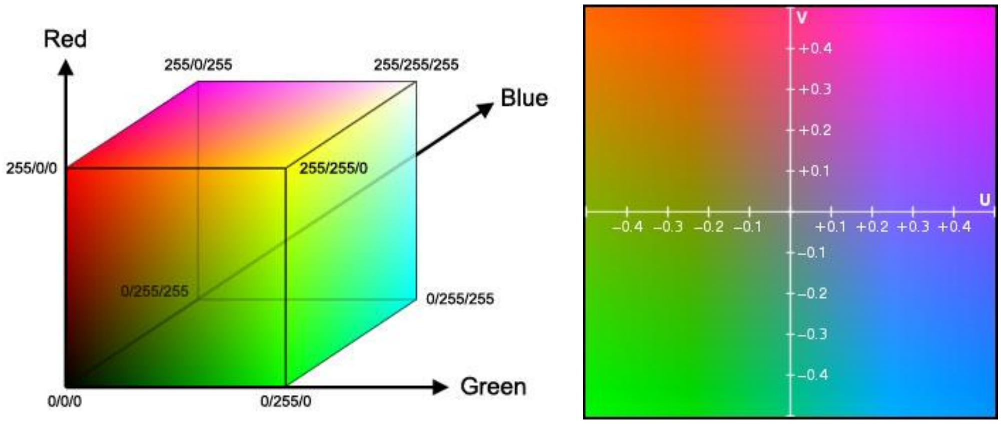


Figure A.1 (a) RGB Color Cube (b) YUV Color Space

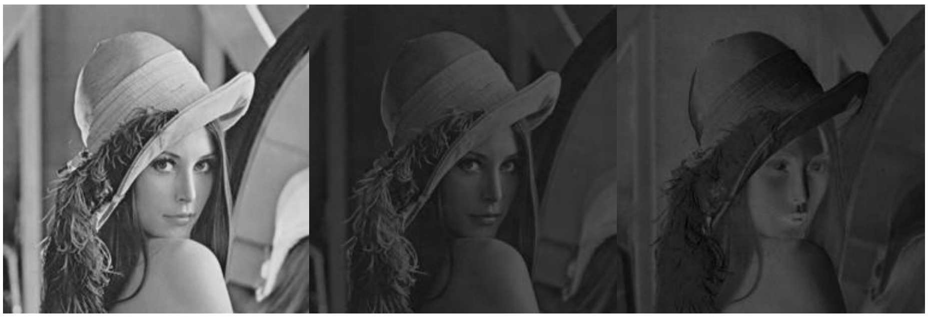


Figure A.2 Y,Cb, and Cr Color spaces of Image

Second step is to divide the image to 8x8 pixel blocks to apply DCT to each of the three color spaces.

$$f(u,v)=\frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right]$$

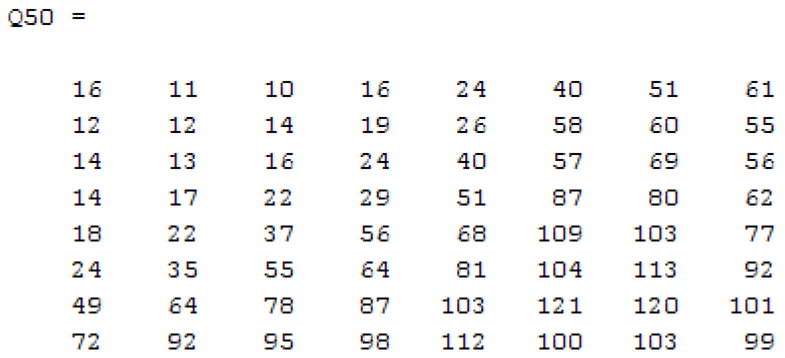
$$\text{for } u=0,\ldots,7 \text{ and } v=0,\ldots,7$$

$$\text{where } C(k)=\begin{cases}1/\sqrt{2} \text{ for } k=0\\ 1 \text{ otherwise}\end{cases} \qquad \text{(A.1)}$$

The transformed 8x8 block has 64 DCT coefficients. The first coefficient F(0,0) is the DC component which is the sum of other 64 coefficients multiplied by 1/8 from equation (A.1) for u=0, v=0. Other 63 coefficients are AC component. To quantize the transformed coefficients is the next step. Each transformed coefficients is divided by the corresponding quantization parameter and rounds to nearest integer.

$$F_q(u,v) = Round\left(\frac{F(u,v)}{Q(u,v)}\right) \qquad \text{(A.2)}$$

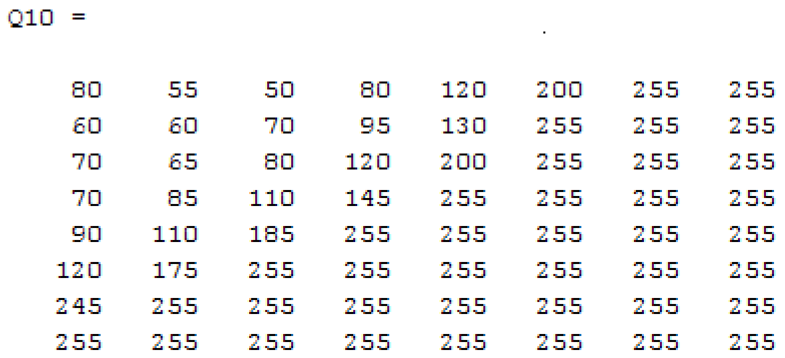
Different quantization matrixes can be selected to compress image depending on wanted quality of image. Figure A.3 and Figure A.4 are the quantization tables for quality 50 and 10. To have another table with different quality, multiply matrices, for Quality50, 50/quality level, then reconstruct matrices between 1 and 255 for out of range values.



```
Q50 =

    16    11    10    16    24    40    51    61
    12    12    14    19    26    58    60    55
    14    13    16    24    40    57    69    56
    14    17    22    29    51    87    80    62
    18    22    37    56    68   109   103    77
    24    35    55    64    81   104   113    92
    49    64    78    87   103   121   120   101
    72    92    95    98   112   100   103    99
```

Figure A.3 Quantization Table Quality %50



```
Q10 =

    80    55    50    80   120   200   255   255
    60    60    70    95   130   255   255   255
    70    65    80   120   200   255   255   255
    70    85   110   145   255   255   255   255
    90   110   185   255   255   255   255   255
   120   175   255   255   255   255   255   255
   245   255   255   255   255   255   255   255
   255   255   255   255   255   255   255   255
```

Figure A.4 Quantization Table Quality %10

This process removes the high frequencies in the original image. The HVS is much more sensitive to lower spatial frequencies than higher frequencies. Next step is to use zigzag sorting (Figure A.5). This process traverses the 8x8 DCT coefficients in order of increasing the spatial frequencies and gives lots of consecutive zeroes.

0	**1**	**5**	**6**	**14**	**15**	**27**	**28**
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure A.5 Zigzag Ordering

Zero Run Length Coding of AC coefficient, encodes quantized vector with consecutive zeroes.

If the quantized vector is;

13, 7, 0, 0, 0, 4, 0, 18, 0, 0, 0, 0, 3, 54, 0, …, 0

RLC Encoded;

(0,13); (0,7); (3,4); (1,18); (4,3); (0,54);…

First number represents the number of 0 before second number.

DC coefficients are usually larger value than AC coefficients and they have close connection between adjacent blocks. So, Jpeg stores the differences between the DC coefficients of block *i* and DC coefficient of block *i-1*. Both of differences of DC coefficients and RLC encoded AC coefficients are encoded using Huffman encoding. After adding quantization factor, scale factors and Huffman tables to header information, compression is completed. Below, decompression procedure is described.

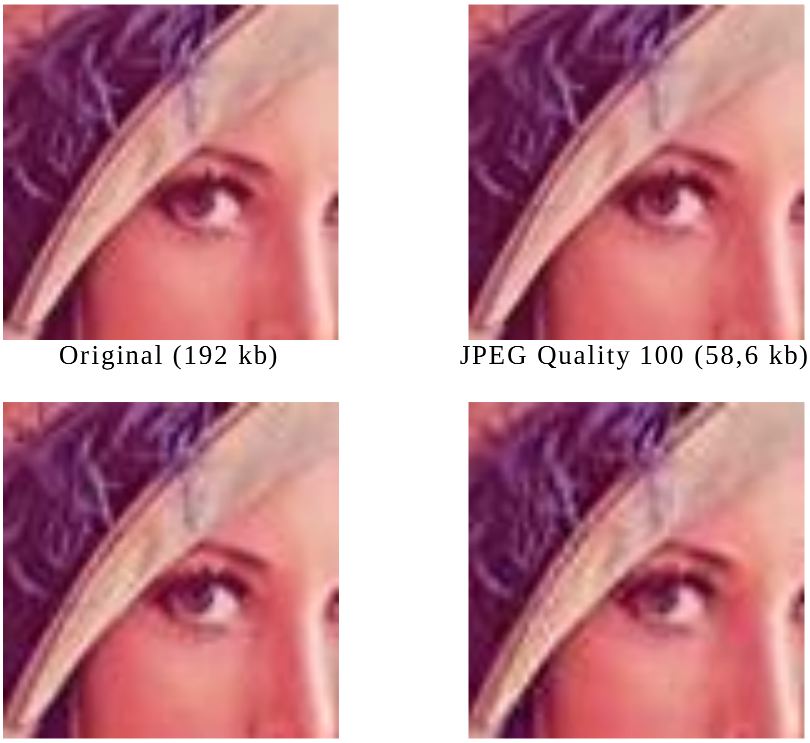
How to decode JPEG images:

1. Remove header info and quantization factors.
2. Extract data from Huffman encode bit stream.
3. Scale each coefficient by the inverse 'quantization' factors.
4. Prepare the coefficients for IDCT in 8x8 blocks.
5. IDCT each coefficient block.
6. Put the 8x8 pixel blocks into the image buffer.
7. Scale up the CbCr components.
8. Convert the YCbCr components into an RGB image.

$$F(u,v) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} F(x,y)\, C(u)\, C(v) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

$$\begin{aligned} \text{where } C(u), C(v) &= \frac{1}{\sqrt{N}} \quad \text{for } u, v = 0 \\ C(u), C(v) &= 1 \quad \text{otherwise} \end{aligned}$$

Figure A.6 IDCT Formula



Original (192 kb)

JPEG Quality 100 (58,6 kb)

JPEG Quality 80 (13,7 kb)

JPEG Quality 40 (7,34 kb)

Figure A.7 Compressed JPEG Images

Table A.2 MSE, PSNR, WPSNR Values of JPEG Images

	MSE (red)	MSE (green)	MSE (blue)	PSNR (red)	PSNR (green)	PSNR (blue)	PSNR (grayscale)	WPSNR (grayscale)
b/c	18,6239	13,3884	24,2249	35,4301	36,8635	34,2882	37,78	58,4624
b/d	47,8736	37,4377	61,1845	31,3326	32,3977	30,2644	33,11	49,8402

APPENDIX B

Curriculum Vitae

Personal Information

Surname, Name	: OLCAY, Cem
Nationality	: Turkish
Date of Birth	: 29/11/1983
Birth Place	: İzmir (Turkey)
Marital Status	: Single
Phone	: +90 532 374 5132
E-Mail	: cem_olcay@hotmail.com

Education

2008 - 2010	:	M.Sc., Computer Engineering, Çankaya University
2001 – 2006	:	B.Sc., Computer Engineering, Çankaya University

Work Experience

2007 -	: Olsea Security and Automation
2005 August – September	: MMC Engineering (Internship)
2004 August – September	: MMC Engineering (Internship)

Languages

Turkish	: Native
English	: Advanced

Hobbies

Basketball, Archery, Rafting