

**Image Steganography Technique Based on
Predetermined Pattern and Histogram Analysis**

تقنية لإخفاء البيانات في صورة بناءً على نمط محدد مسبقاً وتحليل
مخطط تكرار الألوان

By: Haya Mohammad Al Haj

Supervisor: Dr. Mohammed A. Fadhil Al Husainy

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

Faculty of Information Technology

Middle East University

Amman, Jordan

May, 2015

Authorization Statement

I, Haya Mohammad Al Haj, Authorize Middle East University to supply hard and electronic copies of my thesis to libraries, establishments, bodies and institutions concerned with research and scientific studies upon request, according to university regulations.

Name: Haya Mohammad Al Haj

Date: May, 30th 2015

Signature: 

أنا هيا محمد عبد العزيز الحاج، أفوض جامعة الشرق الأوسط للدراسات العليا بتزويد نسخ من رسالتي ورقياً وإلكترونياً للمكتبات، أو المنظمات، أو الهيئات والمؤسسات المعنية بالأبحاث والدراسات العلمية عند طلبها.

الاسم: هيا محمد الحاج

التاريخ: 2015/5/30

التوقيع: 

Committee Decision

This thesis "Image steganography technique based on predetermined pattern and histogram analysis" was discussed and certified on May 30th, 2015.

Thesis committee

1. Dr. Oleg Victorove
Associate Professor
Middle East University

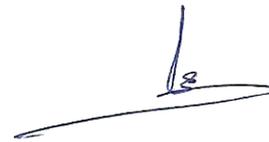
Chairman

Signature



2. Dr. Mohammad A. Fadhil Al Husainy
Associate Professor
Middle East University

Supervisor
and
member



3. Prof. Musbah J. Aqel
Professor
Applied Science University

Member



Acknowledgment

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, all praises to Allah for the good health, protection and ability to complete this thesis. I would like to express my sincerest appreciation to my supervisor Dr. Mohammad A. Fadhil Al Husainy whose thoughtful consideration and guidance has been invaluable. I hope that one day I would become as good an advisor to my students as Dr. Mohammad has been to me. I am also grateful to Dr. Oleg Victorove and Prof. Musbah Aqel for spending time read this thesis and providing useful suggestions about it.

I owe more than thanks to one of the best teachers that I have had in my life, Dr. Heba Nasir Al-Din, who has been an inspiration to me during my master journey, I am so grateful for her continuous encouragement. Many sincere thanks also go to Information Technology Faculty members at Middle East University for their insightful conversations and hard questions, thank you for teaching me how to be a dedicated researcher. Million thanks go to my fellow colleagues for their support and encouragement, and kindness during my study.

I am forever indebted to my family who supported me during this academic journey and my whole life, they had more faith in me than could ever be justified by logical argument. To those who indirectly contributed in this research, your kindness means a lot to me. Thank you very much.

Haya Al-Haj, May 2015

Dedication

{ وَقُلْ رَبِّ زِدْنِي عِلْمًا } [طه: 114]

To the one, who always encourages me to follow my dreams, Thanks for believing in me and for listening to my wild ideas for countless hours. Without your constant love and support, this work could not have happened. You are the love of my life, thank you to the moon and back.

And to my future kids. May you dream big and never let go, and never stop wondering!

List of Contents

Authorization Statement	II
Committee Decision	III
Acknowledgment.....	IV
Dedication	V
List of Contents	VI
List of Tables	VIII
List of Figures	IX
List of Abbreviations	X
Abstract.....	XI
المخلص.....	XII
Chapter 1 Introduction.....	1
1.1 Principles of Steganography.....	2
1.2 Steganography Models.....	3
1.3 Classification of Steganographic Categories	5
1.4 Why Image Steganography?	5
1.5 Image Steganography Categories	7
1.6 Image Steganography Properties	8
1.7 Attacks on Image Steganography Techniques	9
1.8 Image Histogram	9
1.9 Quality Measurements Used in Image Steganography.....	11
1.10 Problem Statement.....	12
1.11 Problem Significance and Motivation	12
1.12 Objectives.....	13
1.13 Thesis Outline.....	14
Chapter 2 Literature Survey.....	15
2.1 Using LSB to Embed Secret Text	15
2.2 Using of Randomization and a Predetermined Pattern.....	16
2.3 Using of Histogram Analysis	19
Chapter 3 The Proposed Image Steganography Technique.....	21
3.1 The Stego-key.....	22
3.1.1 The Predetermined Pattern	23
3.1.2 Image Segmentation.....	24
3.1.3 Histogram Analysis.....	26

3.1.4 Choosing the Appropriate Pixels to Embed the Text.....	28
3.1.5 Matching the Appropriate Pixels with the Pattern.....	31
3.2 Embedding and Extracting Steps.....	34
3.3 Embedding Phase Pseudo Code	37
3.4 Extracting Phase Pseudo Code	38
3.5 Sample of the Proposed Technique	39
Chapter 4 Experimental Results.....	44
4.1 Implementation.....	44
4.2 Implementation Results	45
4.2.1 Capacity, MSE and PSNR.....	46
4.2.2 Undetectability.....	50
4.2.3 Robustness and Level of Security.....	53
4.3 Comparison with Other Steganography Techniques	55
4.4 Analysis and Discussion:	57
4.4.1 Relation between Block Size and Embedding Capacity	57
4.4.2 Relation between the Secret Pattern and Embedding Capacity.....	60
Chapter 5 Conclusion and Future Work.....	62
5.1 Conclusion.....	62
5.2 Limitations	63
5.3 Future Work	63
References:.....	64
Appendices.....	69
Appendix A: shifting Image code.....	70
Appendix B: set colors ranges.....	71
Appendix C: Selecting the appropriate pixels code	72
Appendix D: calculating MSE and PSNR	75

List of Tables

Table 2.1 Meaning of indicator values for pixel indicator technique	17
Table 3.1 Colors values distribution	23
Table 3.2 Generated Patterns	23
Table 3.3 Applying Logical Right Shift to some colors values	27
Table 3.4 Appropriate pixels at red channel	31
Table 3.5 Secret patterns	31
Table 3.6 Matching process	32
Table 3.7 Pixel to embed data at red channel	32
Table 3.8 Color values for some pixels	40
Table 3.9 Colors after applying right logical bit shift	40
Table 3.10 The appropriate pixels table	41
Table 3.11 Pixels values at the secret pattern	41
Table 3.12 Three color ranges generated from the secret pattern	41
Table 3.13 Matched addresses	42
Table 3.14 Appropriate pixels order to embed the secret text	42
Table 4.1 Embedding capacity, MSE and PSNR of one block (512×512 pixels)	46
Table 4.2 Embedding capacity, MSE and PSNR of two segments (512×256 pixels) ..	46
Table 4.3 Embedding capacity, MSE and PSNR of two segments (256×512 pixels) ..	47
Table 4.4 Embedding capacity, MSE and PSNR of four segments (256×256 pixels) ..	47
Table 4.5 Embedding capacity, MSE and PSNR of nine segments (170×170 pixels) ..	47
Table 4.6 Embedding capacity, MSE and PSNR of 16 segments (128×128 pixels)	48
Table 4.7 Embedding capacity, MSE and PSNR of 100 segments (50×50 pixels)	48
Table 4.8 Comparison between cover and stego image using the proposed technique ..	51
Table 4.9 RGB channels histograms for the cover and stego image (Lenna)	52
Table 4.10 The histogram of RGB channels of the cover and stego image (baboon)	52
Table 4.11 The histogram of RGB channels of the cover and stego image (pepper)	53
Table 4.12 Order of the addresses of pixels to embed the text in Red channel	54
Table 4.13 Order of the addresses of pixels to embed the text in Blue channel	54
Table 4.14 Order of the addresses of pixels to embed the text in Green channel	55
Table 4.15 Comparison between the proposed technique and histogram modification method	56
Table 4.16 Comparison between the proposed technique and n-queen technique	56

List of Figures

Figure 1.1 The basic structure of steganography	2
Figure 1.2 Steganography models.....	3
Figure 1.3 pixels having very similar colors with different values	6
Figure 1.4 Generalized image steganography framework.....	6
Figure 1.5 Image steganography categories	7
Figure 1.6 Tradeoff between image steganography properties.....	8
Figure 1.7 An example of image histogram	10
Figure 1.8 Red, Green and Blue Colors Histogram, (a) cover image (b) stego image ...	11
Figure 2.1 No. of N-queen solutions	18
Figure 3.1 General block diagram of the proposed technique	22
Figure 3.2 Examples of images that can be used as secret pattern	24
Figure 3.3 Cover image partitioned into four segments	25
Figure 3.4 Cover image partitioned into 16 segments.....	25
Figure 3.5 Logical right shift one bit.....	26
Figure 3.6 Color histogram 0 - 255 (a), Color histogram 0 - 127 (b)	27
Figure 3.7 example of some shapes in the colors histogram	28
Figure 3.8 flow chart for choosing the appropriate pixels.....	30
Figure 3.9 Flowchart of the embedding phase.....	35
Figure 3.10 Flowchart of the extracting phase.....	36
Figure 3.11 (200 × 200) BMP image	39
Figure 3.12 Partition cover into for blocks	39
Figure 3.13 Secret pattern.....	39
Figure 3.14 128 color histogram for R, G and B	40
Figure 4.1 C# application interface of the proposed technique	44
Figure 4.2 Cover images used to test the proposed technique	45
Figure 4.3 Secret patterns used to test the proposed technique	45
Figure 4.4 MSE and PSNR based block size (Lenna).....	49
Figure 4.5 MSE and PSNR based block size (Baboon)	49
Figure 4.6 MSE and PSNR based block size (Peppers)	50
Figure 4.7 Relation between capacity and block size	58
Figure 4.8 Two different block sizes for one image	59
Figure 4.9 Red histogram of three different segments (128 × 128 pixel)	60

List of Abbreviations

IT	Information Technology
LSB	Least Significant Bit
PSNR	Peak Signal to Noise Ratio
MSE	Mean Square Error
BMP	Microsoft Windows BitMaP
RGB	Red, Green and Blue
HVS	Human Visual System
MSB	Most Significant Bit
bpp	bits per pixel
bpc	Bits per channel
HAS	Human Auditory System
P	Peak
LN1	First Left Neighbor
LN2	Second Left Neighbor
RN1	First Right Neighbor
RN2	Second Right Neighbor
D	Difference
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
BV	Brightness Value
IN	Iteration Number
LLV	Lower Limit Value
ULV	Upper Limit Value

Image Steganography Technique Based on Predetermined Pattern and Histogram Analysis

By: Haya Mohammad Al Haj

Supervisor: Dr. Mohammed A. Fadhil Al Husainy

Abstract

This study presents a steganography technique to hide secret text inside color image, by using the principle of LSB, where the secret text is hidden in the least significant bits of the pixels, with more randomization in selection of the pixels used. Two kinds of stego key were presented, the first part is a predetermined pattern which can be generated from an image, and the second is the color histogram of 128 color. The image will be partitioned into segments of a predefined size, then the color histogram will be calculated for each segment separately. Both keys are used to control the embedding and extracting processes. This technique is applied to RGB images of BMP format.

The advantages of the proposed technique can be summarized as follows, perceptibility on stego image and its histogram. High statistical image quality measurements, i.e. higher Peak Signal to Noise Ratio (PSNR) and lower Mean Square errors (MSE) prove that the proposed technique has good quality of the stego images. Also it is highly acceptable by the human visual system HVS. Relatively high data hiding capacity, comparing to some other random selection based techniques. The stego key space size is very large. Which is hard to detect by brute force attack, thus the level of security for the proposed technique will be increased.

Keywords: Steganography, information security, image steganography, histogram, stego key.

تقنية إخفاء البيانات في صورة بناءً على نمط محدد مسبقاً وتحليل مخطط

تكرار الألوان

اعداد: هيا محمد الحاج

اشراف: د. محمد عباس فاضل الحسيني

الملخص

تقدم هذه الدراسة تقنية لإخفاء نص سري داخل صورة ملونة. بالاعتماد على مبدأ Least Significant Bit (LSB)، حيث يتم استبدال بعض القيم الأقل أهمية في الصورة (يتم اختيارها عشوائياً) مع البيانات المراد إخفائها من النص؛ باستخدام مفتاح خاص مكون من جزئين (نمط محدد مسبقاً سيتم استخراجها من صورة، وتحليل الرسم البياني لمخطط تكرار الألوان لـ 128 لون، سيتم تقسيم الصورة إلى أجزاء متساوية في الحجم وسيكون هذا الحجم سري وبعد ذلك سيستخرج مخطط تكرار الألوان لكل جزء على حدى. سيتحكم المفتاح من خلال جزئيه بعملية إخفاء النص واستخراجها. حيث أنّ عملية إخفاء النص السري لن تعتمد على أحد أجزاء المفتاح مباشرة، وإنما سيتم تحديد أماكن من الصورة للإخفاء أو استبعادها بناءً على التطابق بينهما.

يمكن تلخيص مزايا هذه التقنية على النحو التالي: سعة استيعاب بيانات عالية نسبياً بالمقارنة مع بعض التقنيات الأخرى التي تعتمد مبدأ الاختيار العشوائي لأماكن إخفاء البيانات، بالإضافة إلى أنه لا يمكن الكشف عن الاختلافات بين الصورة قبل وبعد ادخال النص من خلال النظام البصري للإنسان (العين المجردة)، أو من خلال تحليل مخطط تكرار الألوان للصورة الأصلية، والصورة بعد ادخال النص؛ حيث أنّهما متقاربتان بدرجة عالية جداً، وقد أثبتت التقنية نتائج جيدة جداً من حيث المقاييس الإحصائية (العددية) المعتمدة لفحص مدى تأثير إخفاء البيانات في الصورة، وقد طبقت هذه التقنية على مجموعة صور متعارف عليها لإثبات ذلك. إن عدد الطرق الممكنة لاكتشاف المفتاح المستخدم لإخفاء النص كبير جداً، فمن الصعب اكتشافه من خلال تجربة جميع الاحتمالات الممكنة له، مما يعزز أمان هذه التقنية.

الكلمات المفتاحية: إخفاء البيانات في صورة، أمن وحماية المعلومات، مخطط تكرار الألوان.

Chapter 1 Introduction

In today's Information Technology (IT) environment, the significance of sharing and securing the important resource of information against unauthorized access or modification became very important, as everyone rely on IT to store, process or send information, therefore, it is essential to maintain information security. The importance of information security led IT experts to develop innovative methods of protecting the information.

Whatever is the process that has been chosen to secure the information, the main concern is the level of security it has, and for a higher level of security it is better if data are hidden in a such way that no one can suspect its existence, detect or retrieve it except the authorized persons, this is the concept behind *Steganography* where it deals with the ways of hiding the existence of information, not altering the structure of them. In simple words it is the process of hiding information into other information. (Purohit & Sridhar 2014).

Today's steganography systems use multimedia objects like image, audio, video etc. as cover media because people often transmit digital media over email or share them through other Internet communication applications (Rana & Singh, 2010).

In image steganography, secret communication is achieved by embedding data into cover image (used as the carrier to embed data into) and generate a stego image (generated image which is carrying a hidden data). Any image steganography system will necessarily cause some distortion or modification of the cover image. The key to successful steganography is to insure that the distortion caused by the hiding process is undetectable, where the cover and the stego-images should be similar, and to guarantee

that no one apart from the sender and the intended recipient even realize there is hidden data and could extract them.

1.1 Principles of Steganography

The process that involves hiding information in an appropriate carrier without any visual evidence of information exchange is called as steganography (Bhagat & Dhembhare, 2015). The word steganography is obtained from the Greek words stegos (στεγανός) meaning “covered or protected” and grafia (γράφη) meaning “writing”, which is literally means “cover writing” (Subhedar & Mankar, 2014) in other words, the goal of steganography is to hide secret information into other information in such a way that no one apart from the authorized persons even observes that there is secret information.

The basic structure of steganography, Figure 1.1, is made up of three components: The cover medium, the hidden data, and the secret key (Singh, Dhanda & Kaur, 2014). The cover medium is the object that will carry the hidden data, referred to as a cover-text, or cover-image or cover-audio as appropriate, and only the one who has the secret key can access the hidden data (Vyas & Pal 2014).

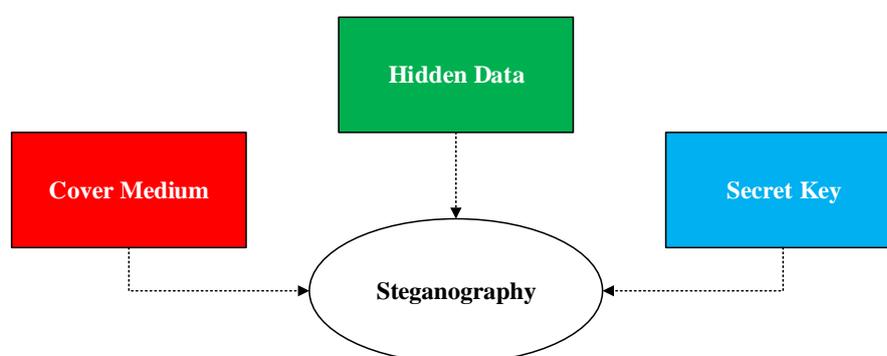


Figure 1.1 The basic structure of steganography

1.2 Steganography Models

Almost all digital media files can be used for steganography, but the files that are more suitable are the ones having high degree of redundancy, which can be defined as the amount of bits of an object that provide high accuracy than the necessary for using and displaying it.

Depending on the type of the cover object there are many suitable steganographic techniques which are followed in order to obtain security, as shown in Figure 1.2. (Bahirat & Kolhe, 2014).

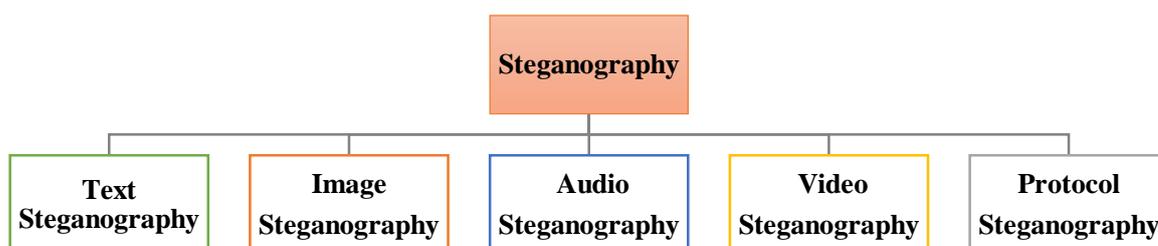


Figure 1.2 Steganography models

1. Text Steganography Model

In text steganography data are hidden in a text file. Many text steganography techniques involve the modification of the text layout, rule like using every n^{th} character or altering the amount of white space after lines or between words. (Vyas & Pal, 2014).

2. Image Steganography Model

Using the cover object as an image in steganography is known as image steganography. Generally, in this technique pixel intensities are used to hide the data, and the most common method used in this model is the Least Significant Bit insertion (LSB).

On which the least significant bit of the image pixel will represent a bit of information. (Kaur, Kaur & Singh, 2014).

3. Audio Steganography Model

In this type, data are embedded in audio files. The methods in which data is hidden in sound files using properties of the Human Auditory System (HAS). Hiding more information into audio files is a more difficult task than that of images, due to superiority of the HAS over human visual system HVS (Bahirat & Kolhe, 2014).

4. Video Steganography Model

Video files are generally a collection of images and sounds, so most of the existing techniques on images can be applied to video files too. Out of the above mentioned steganographic technique the embedding capacity of the secret data increases in video steganography model. As it enables to hide data in image as well as in audio and generate stego video. (Kaur, Kaur & Singh, 2014).

5. Protocol Steganography Model

When taking cover object as network protocol, such as Transmission Control Protocol TCP, User Datagram Protocol UDP, Internet Control Message Protocol ICMP, Internet Protocol IP etc., is known as network protocol steganography. (Gawade, Shetye, Bhosale & Sawantdesai, 2014).

1.3 Classification of Steganographic Categories

Steganography is classified into 3 categories (Sumathi, Santanam & Umamaheswari, 2013):

- **Pure Steganography:** where there is no stego key or there is no need to exchange a stego key. It is based on the assumption that no other party is aware of the communication.
- **Secret Key Steganography:** where the stego key has to be exchanged prior to communication. This is most susceptible to interception. This study focuses on this category.
- **Public Key Steganography:** where a public key and a private key is used for secure communication.

1.4 Why Image Steganography?

Images are preferred medium for the current steganography techniques. Because their content adaptation, visual resilience and the small size make them a good carriers to transmit secret data over the internet (Nilizadeh, 2013) and because of the limited power of the human visual system (HVS), as the human eye does a poor job of distinguishing the differences between two very similar colors. (Garg & Wasson, 2014). For example, the two yellow colored square shown in Figure 1.3. The left square is shaded RGB (247, 235, 107) and the right one is shaded RGB (248,236,106). There exist a subtle change in the red, green and blue values by just one unit. Despite that, they are looking the same for the human visual system. This fact initialize the idea behind image steganography model.



Figure 1.3 pixels having very similar colors with different values

The basic image steganography model consists of Cover Image, Secret Data and Key, the image with embedded data is called as stego image, and the key used to embed data is called as stego key (Subhedar & Mankar, 2014). Figure 1.4 shows a generalized image steganography framework (Sharma, Mohd & Sharma, 2014).

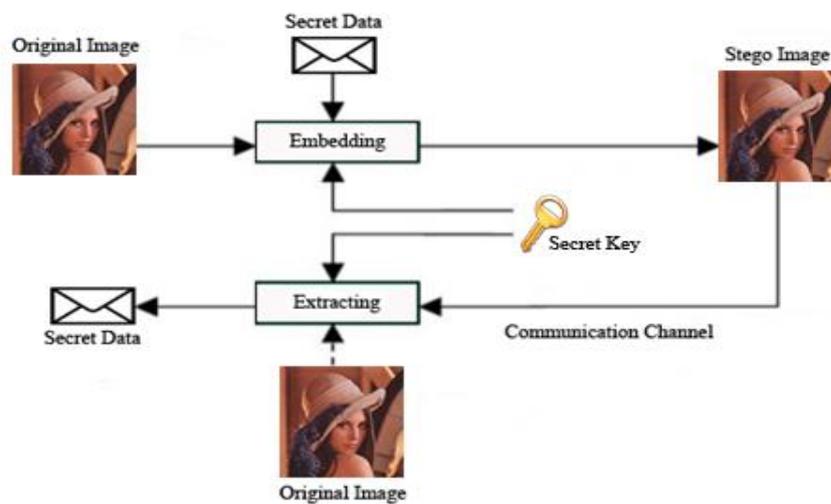


Figure 1.4 Generalized image steganography framework

1.5 Image Steganography Categories

The image steganography model can be categorized into two categories, namely, spatial domain and frequency domain (Subhedar & Mankar, 2014). Shown in Figure 1.5.

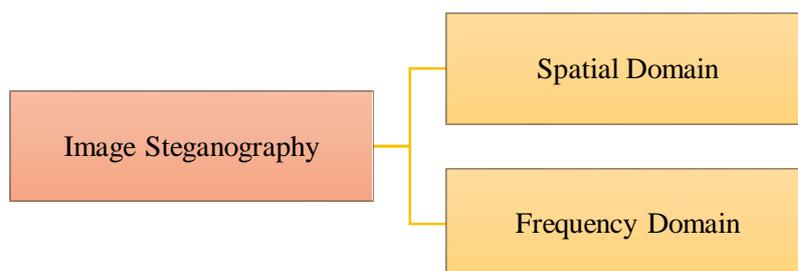


Figure 1.5 Image steganography categories

This study is in the spatial domain, where the pixels intensities are used to hide the data directly. Straight message insertion may encode every bit of information in the image or selectively embed the message in “noisy” areas, which draw less attention – those areas where there is a great deal of natural color attention. The data may also spread randomly throughout the image (Bahirat & Kolhe, 2014). While in transform domain, images are first transformed and then the message is embedded in the image. The image formats that are most suitable for spatial domain steganography are lossless, like BMP format. And the image formats that are suitable for the transform domain are lossy, like JPEG format. (Hamid, Yahya, Ahmad & Al-Qershi 2012).

1.6 Image Steganography Properties

A few properties characterize the strength and weaknesses of any image steganography technique, some of the major properties are *capacity*, *robustness* and *undetectability* (Tiwari & Shandilya, 2010). A tradeoff between those properties is shown in Figure 1.6.

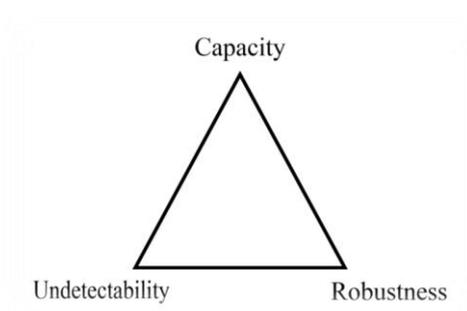


Figure 1.6 Tradeoff between image steganography properties

- **Capacity:** It refers to the amount of data that can be embedded into the cover image. It can be represented in terms of bits per pixel (bpp) which specify the number of bits used.
- **Robustness:** The degree of difficulty required to tear down embedded information without destroying the cover image itself.
- **Undetectability:** The ability to determine whether or not a cover image contains embedded information using visual or statistical means. (Hamid, Yahya , Ahmad & Al-Qershi 2012) (Bashardoost, Sulong & Gerami 2013).

If we increase the capacity of any cover image to store more data than a practical possible amount, then its undetectability will be affected and vice versa. (Gutub 2010).

1.7 Attacks on Image Steganography Techniques

During the last decade, many image steganographic algorithms had been proposed, on the other hand many *steganalysis* methods are also proposed. The goal of steganalysis is to analyze the stego image to detect the presence of the hidden text in it, and extract the hidden text if possible. (Desai & Patel 2014) (Kaur, Kaur, & Singh, 2014).

Steganography technique can be considered secure if it is impossible for attackers to detect the presence of hidden text in the stego image by using any accessible means (Gutub, 2010). This can be obtained by using a strong stego-key. In addition to that the hidden text must be invisible both perceptually and statistically in order to avoid any suspicion of attackers, where by doing a simple check on the stego image histogram, one can basically assume that the image had been processed for a reason such as conveying a secret text. Hence, selecting a suitable image area to embed the secret text is a major task (Nilizadeh & Nilchi, 2013).

1.8 Image Histogram

There are different color spaces that present different forms for storing images. The most common color space is RGB (Red, Green, and Blue). Each pixel in this space is described by 3 channels of 8 bits (3 bytes), where each channel contains the value of individual red, green and blue colors. Another color space is 8-bit grayscale, where the pixel is described by 8 bits (1 byte) and it has a value from (0 – 255). (Gupta & Sandhu, 2013).

An image histogram is a graphical representation of the distribution of colors redundancy in the image's pixel. The histogram plays an important role to distinguish stego images from natural images. The change in the image histograms is called as “comb effect” in literature (Yalman & Erturk, 2009). It points out the unbalanced color value distribution and may easily lead to the detection of the secret text. Figure 1.7 shows an example of color histogram.

Although, the HVS is unable to detect the distortion added by steganography techniques, for example: HVS cannot sense the differences between images presented in Figure 1.8 the cover image (a) and the stego image (b), but it can easily recognize the difference between their histograms. Where not only are the image histogram is different but also the frequency of occurrence of the color values are different. As a result one can basically assume that the image had been processed for a reason such as protecting a secret message by looking at its histogram.

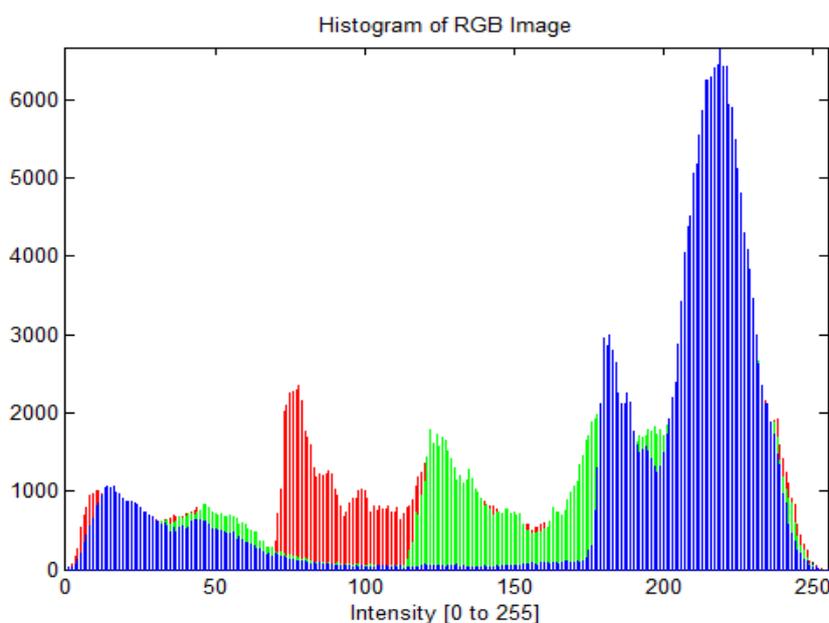


Figure 1.7 An example of image histogram

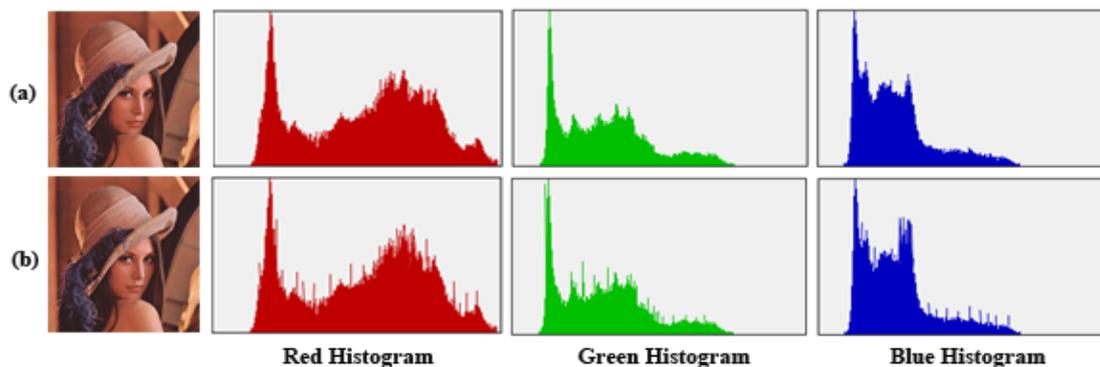


Figure 1.8 Red, Green and Blue Colors Histogram, (a) cover image (b) stego image

1.9 Quality Measurements Used in Image Steganography

Peak Signal to Noise Ratio (PSNR) and Mean Square Error (MSE) parameters are considered for statistical analysis of the steganography methods. The MSE should be computed first as given in the first equation, then the PSNR can be derived, where “I” and “K” are the original and stego image pixel values respectively to be compared, and the image size is “m × n”. Note that, for color images, the size should be multiplied by three (i.e., for Red, Green and Blue colors). (Singh, Dhanda & Kaur 2014).

$$MSE = \frac{1}{m n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2 \quad \dots\dots\dots (1)$$

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) \quad \dots\dots\dots (2)$$

As the value of PSNR increases, the quality of the image improves and as the PSNR value decreases the quality of the image decreases. As the value of MSE increases, the quality of the image decreases, and as the value decreases the quality of the image improves. (Singh, Dhanda & Kaur 2014).

1.10 Problem Statement

The task of designing secure steganography technique is day by day becoming critical. To consider an image steganography technique secure it should be hard to detect the presence of hidden text both perceptually and statistically, by selecting a suitable image area to embed the text. And if the stego image had been detected as hiding secret text it should be hard to extract the text from the stego image. Hence, the need for a strong stego key is a major task.

There are many questions that need to be answered. Initially, how can the proposed technique increase the amount of text to be hidden? How the proposed technique deal with? How much is it hard for the steganalysis to find the presence of the hidden text using common steganalysis methods? Is it possible to extract the hidden text without knowing the stego key? How much is it hard to find the stego key?

All these questions will be discuss in-depth in this study by suggesting a solution to the work of hiding text in image using predetermined pattern and histogram analysis.

1.11 Problem Significance and Motivation

Steganography became more important in a number of application areas from Military Intelligence Agencies, Cloud Security, Online Elections, Internet Banking, Medical-Imaging, and so on. These variety of applications make steganography a hot topic for study.

In the current situation digital images are the most popular cover mediums that can be used to transmit secret text, and with the several advances in attacking algorithms

over image steganography techniques, the task of designing most secure steganography technique is day by day becoming critical.

The motivation of this study is providing secure hiding technique for text inside an image using random pixels of the cover image, in order to develop applications that help users to efficiently hide secret text and use it in different areas.

1.12 Objectives

The main objective of this study is to design a secure steganography technique that satisfy the following:

- Security and secrecy: By increasing randomness to make it hard to extract the secret text, and by strengthen the stego key which consists of two parts (Pre-determined pattern and color Histogram), therefore, it will be hard to detect the pixels used to embed the secret text.
- Undetectability: By using histogram analysis to improve the quality of the stego image by providing less distortion while embedding the secret text.
- Capacity: to higher the quantity of hidden data without affecting the quality of the image.

1.13 Thesis Outline

Chapter one discusses the basic principles regarding steganography in general and image steganography technique, the problem statement, motivation, objectives and limitations of this thesis.

Chapter two reviews previous studies and existing techniques related to the topic of the thesis and other related topics.

Chapter three explains the proposed methodology in details.

Chapter four presents the experimental settings and results, and also makes comparisons between previous techniques of image steganography and the proposed one.

Chapter five summarizes the conclusion of the thesis and proposes future work.

Chapter 2 Literature Survey

Steganography has been an active research topic for decades and there exists a large variety of steganography techniques, some are more complex than others, and all of them have respective strong and weak points. Some techniques related to the proposed one are mentioned as below:

2.1 Using LSB to Embed Secret Text

Least significant bit (LSB) insertion is a common and easy approach for embedding information in a cover image. The least significant bit of the bytes inside an image is changed to a bit of the data. While using a 24-bit RGB image, and since the three colors are each represented by a byte, one can store 3 bits in each pixel, a bit of each of the red, green and blue colors. This will only change the integer value of the byte by one. (Pavani, Naganjaneyulu, Nagaraju 2013).

In some cases LSB of pixels are arranged in random or in certain areas of image, and sometimes increment or decrement the pixel value. And other cases hide the message in the least as well as second, third and even the fourth to least significant bit. Increasing number of bits used to hide the secret data will sure increase the amount of data to be hidden but this will cause more distortion and the quality of the stego image will decreased (Tiwari, Sandilya & Chawla, 2014).

2.2 Using of Randomization and a Predetermined Pattern

Venkata, et.al. (2009) proposed a randomization technique that used RGB values of color images to enhance imperceptibility. The LSB of any one of the 3 channels is used as a pointer to decide embedding capacity in the other two channels. If the last two bits of the channel are 00 there is no hidden data, if it is 01 data is embedded only in channel 2, if it is 10 data is embedded in channel 1 and if it is 11 data is embedded in both the channels. Three methodologies are used. They are:

1. Red is used as default pointer.
2. User selects any channel as pointer.
3. Pointers are chosen based on a cyclic sequence and data is embedded.

Based on the histogram study and the values of MSE and PSNR (Mean Square Error and Peak Signal to Noise Ratio) the randomized method has better secrecy and performance than the classical LSB method.

Gutub (2010) has proposed pixel indicator technique to hide text inside image. Which uses the least two significant bits of one of the color channels RGB, to indicate existence of data in the other two channels. Table 2.1 shows the meaning of indicator values for pixel indicator technique.

Table 2.1 Meaning of indicator values for pixel indicator technique

Indicator Channel	Channel-1	Channel-2
00	No Hidden data	No Hidden data
01	No Hidden data	2 Bits of Hidden Data
10	2 Bits of Hidden Data	No Hidden data
11	2 Bits of Hidden Data	2 Bits of Hidden Data

Thiyagarajan, et.al. (2010) have proposed a dynamic pattern based image steganography. Which aims at strengthen the security by generating dynamic pattern in selection of indicator sequence. The idea is that significant color in a pixel should not suffer from data embedding while the insignificant color channel can be used for data embedding.

Rana & Singh (2010) proposed a steganography technique using LSB with pre-determined random pixel and segmentation of image. It encrypts the secret message, using data encrypted standard. The message is divided into four blocks, with each block a pixel is selected using a predetermined method, which then become the stego key. The method uses a combination of odd and even rows and columns respectively and has three levels of security.

Nilizadeh & Nilchi (2013) proposed a novel method for image steganography in RGB format, where a secret text is embedded in the blue layer of certain blocks. In this technique, each block first chooses a unique $t1 \times t2$ matrix of pixels as a “matrix pattern” for each keyboard character, using the bit difference of neighborhood pixels. Next, a secret message is embedded in the remaining part of the block, those without any role in

the “matrix pattern” selection procedure. In this technique, each pattern sums up with the blue layer of the image. And for increasing the security, blocks are chosen randomly using a random generator.

Singh, Dhanda & Kaur (2014) proposed a method of image steganography that hides the secret message using the n-Queen matrix (pattern) as the stego key. For n-Queen puzzle the numbers of solutions are increasing with ‘n’ increase, as shown in figure 2.1. The level of security is directly proportional to n because the probability of selecting a solution is decreasing with increasing in value of n. The proposed system can be used in the fields where more priority is given to security instead of amount of data shared.

n	1 2 3 4 5	6 7	8	11 12 14	... 25	26
Unique Solution	1 0 0 1 2	1 6	1	34 1,78 45,7	... 275,986,683,74	2,789,712,466,5
			2	1 7 52	.. 3434	
Distinct Solution	1 0 0 2 1	4 4	9	2,6 14,2 365,5	... 2,207,893,435	22,317,699,616,
			2	80 00 96	.. 808,352	364,044

Figure 2.1 No. of N-queen solutions

All the above mentioned steganography techniques having less capacity than the classic LSB techniques as they use LSB in a random selection based, but they provide more security as the classical LSB can be easily detected by the sequential attack, while the random selected based methods are not.

2.3 Using of Histogram Analysis

Ni et al. (2006) initially introduced a histogram based data hiding technique where the crucial information is embedded into the image histogram. Pairs of peak points and zero points are used. This technique has low embedding distortion with respect to low data hiding capacity.

A new steganography method for digital images has been proposed based on a histogram modification approach by (Yalman & Erturk, 2009). It utilizes the fundamentals of the digital image and the idea of modifying brightness values histogram of the digital image. Consequently, it is highly resistant to main geometrical attacks like rotation, warping and scattered tiles. Another advantage of the proposed data hiding method is that it can be applied to very small images. The algorithm principally modifies a cover image's histogram for data hiding where neither the resulting new stego image nor its histogram is noticeably different from the original. Therefore, they are both perceived exactly same as the original ones by the Human Visual System (HVS). The Algorithm considers the Iteration Numbers (INs) of the pixel Brightness Values (BVs) of the cover image, and then the data hiding process is realized based upon it. The cover image histogram is created where the lowest and the highest BVs are determined and named as the Lower Limit Value (LLV) and the Upper Limit Value (ULV). These two boundaries indicate where the process of data hiding can be accomplished. The secret text will be embedded based on computing the mod2 value of the Iteration number and compare it with the stego bit, if they are equal keep it as it is, else subtract the IN by one and add one to the next IN.

Krishna, Rahim, Shaik, & Rajan (2010) introduced another histogram based data hiding algorithm. However it is a reversible data hiding technique based on histogram modification using pairs of Peak Points and Zero Points, it has the process of adding '1', if peak of pixel has been encountered. Otherwise, '0' is added, i.e. if zero is detected. From this they can estimate the number of pixels in the image. However, for an unusual image with equal histogram, with this technique minimum points can be embedded. Also the peak and minimum points should be requirement of the receiver for full recovering.

Chaitanaya, Krishna & Anganeyulu (2013) introduce a 3-level secure histogram based image steganography technique. With high level security and data hiding capacity closed to 20% of the cover image where a matched bit replacement method is used based on the sensitivity of the human visual system (HVS) at different intensities. A histogram equalization preprocessing technique has been explored which improve data embedding capacity.

Most of researches use histogram analysis to minimize the *distortion* of the stego image. In this study the histogram will be used to improve the *security* as a part of the stego key, in addition to provide *less distortion* of the stego image. Random appropriate pixels of the image will be determined based on histogram analysis. The secret text will not be embedded directly in those pixels, but the pixels maybe missed or located based on the matching between those pixels and a predetermined pattern, in order to provide security, undetectability and relatively high embedding capacity as a random selection pixels technique.

Chapter 3 The Proposed Image Steganography Technique

The proposed technique is hiding secret text inside color image, by using the principle of LSB, where text's bits are hidden in the least significant bits of the image's pixels, with more randomization in selection of pixels used to hide the secret text, this randomization is expected to increase the security of the technique.

The pixels are selected using a stego key which consists of two parts: external (predetermined pattern) and internal (colors histogram). The cover image where the text hidden will be partitioned into number of segments where segment size = pattern size, and each segment contains part of the secret text. The secret text will not be embedded directly using the predetermined pattern, but the pixels maybe missed or located based on the colors histogram by focusing on imperceptibility and capacity parameters of the cover image in order to produce a relatively identical histogram statistics, for both of the stego and original images. The proposed technique is applied to RGB¹ images of BMP² format where each pixel is independent and represented by three bytes to indicate the intensity of red, green, and blue in that pixel. But it can also be used in other image formats with some restrictions related to their properties.

Generally, the proposed image steganography technique consists of two main phases: the embedding phase and the extracting phase, where both phases use the same

¹ Red, Green and Blue.

² Microsoft Windows BitMaP File

stego key. Figure 3.1 shows the general block diagram of the elements and processes used in the proposed technique.

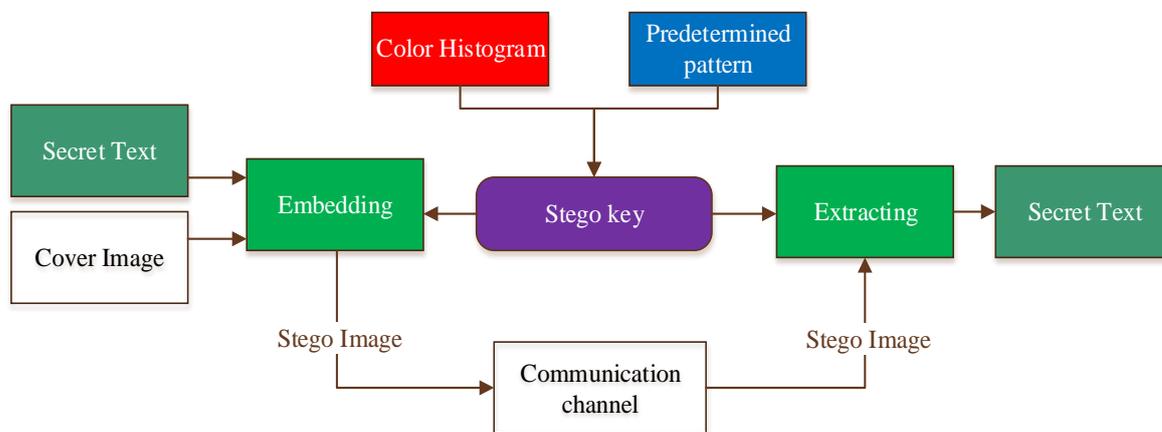


Figure 3.1 General block diagram of the proposed technique

3.1 The Stego-key

The most important requirement for a steganography technique is undetectability: stego images should be statistically indistinguishable from cover images. In other words, there should be no artifacts in the stego image that could be detected by an attacker. The main contribution of this study is to design a strong stego key with large space and large size, where it would be difficult for the attacker to break it and then extract the hidden text.

The stego key that proposed in this technique, consists of two parts: internal and external. The external key is a predetermined pattern, and the internal is the colors histogram. The idea by using two different secret keys is to increase the complexity of the stego key and to make it difficult for an attacker to extract the secret text.

3.1.1 The Predetermined Pattern

The complexity of the stego key is determined by the size of the stego key space and size. As more randomness and more key size implies more security. In the proposed technique the random distribution of colors in an 8-bit gray scale image will be used as a secret pattern, it is to be noted that any RGB image could be converted to 8-bit grayscale and then used as a secret pattern. Thus to break the stego key the attacker needs to apply exhaustive search over an infinite number of images in the Internet, which sounds impossible!

By looking at the random colors distribution at an 8-bit grayscale image, many patterns could be generated based on colors ranges. For example, let G be an 8-bit gray scale image. Based on colors range (0 – 255) three different patterns could be determined, where the first (Pt1) includes all pixels having values (0 – 85), the second (Pt2) includes values (86 – 170) and the third (Pt3) includes values (171 – 255). Table 3.1 shows an example of colors values distribution in an 8-bit grayscale image and Table 3.2 shows the generated patterns.

Table 3.1 Colors values distribution

	0	1	2	3	4	5
0	12	129	15	115	150	224
1	55	117	208	12	117	128
2	224	15	255	0	24	86
3	4	8	175	150	200	214
4	96	215	0	44	215	43
5	100	128	105	179	204	77

Table 3.2 Generated Patterns

	0	1	2	3	4	5
0	Pt1	Pt2	Pt1	Pt2	Pt2	Pt3
1	Pt1	Pt2	Pt3	Pt1	Pt2	Pt2
2	Pt3	Pt1	Pt3	Pt1	Pt1	Pt2
3	Pt1	Pt1	Pt3	Pt2	Pt3	Pt3
4	Pt2	Pt3	Pt1	Pt1	Pt3	Pt1
5	Pt2	Pt2	Pt2	Pt3	Pt3	Pt1

The reason of using a secret pattern is to randomize the selection of pixels that will carry part of the secret text to be hidden which gives more security. Three colors ranges is used in the proposed technique to increase randomness. Figure 3.2 shows some images that could be used in the proposed technique as a secret pattern.



Figure 3.2 Examples of images that can be used as secret pattern

3.1.2 Image Segmentation

In order to strengthen the stego key and to increase the randomness of the proposed technique, the cover image where the text hidden will be partitioned into segments of the same size. And the size of image used as a secret pattern should be equal to the segment size, thus the intruder or the attacker also needs to determine the size of the image used as a secret pattern.

Example: having cover image of 200×200 pixel, it could be divided into four segments each of size 100×100 pixel, or two segments of 200×100 or 100×200 pixels, it can also be divided into 16 segments of 50×50 pixel, and so many different sizes could be obtained. Figures 3.3 shows an example of dividing the original image into four

segments and Figure 3.4 shows another example of dividing the same image into 16 segments.



Figure 3.3 Cover image partitioned into four segments



Figure 3.4 Cover image partitioned into 16 segments

3.1.3 Histogram Analysis

By using histogram analysis the researcher aims to satisfy two features: the first is to improve the *quality* of the stego image by providing less distortion while embedding the secret text, compared with the original image (cover image). The second is to improve the *security* of the proposed technique, as it will be part of the stego key. Where sender and receiver will get the same histogram before and after embedding the text, and then determine the pixels that might hold text. The idea is to discard all the places where changes happened. In this technique changes are in the Least Significant bit of pixel's color value. A logical right shift³ by 1 will be applied to all pixels bits of the cover image in order to discard all LSBs, shown in Figure 3.5.

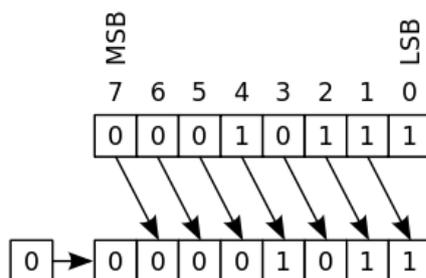


Figure 3.5 Logical right shift one bit

Table 3.3 shows the binary representation for some colors and the new color value after applying logical right shift by 1.

³ It has the effect of dividing each color value by 2

Table 3.3 Applying Logical Right Shift to some colors values

Pixel value	Binary representation	Logical right shift	New value
0	00000000	00000000	0
1	00000001	00000000	0
2	00000010	00000001	1
3	00000011	00000001	1
70	01000110	00100011	35
71	01000111	00100011	35
72	01001000	00100100	36
73	01001001	00100100	36
252	11111100	01111110	126
253	11111101	01111110	126
254	11111110	01111111	127
255	11111111	01111111	127

Applying this process made the cover image appears darker, as shown in figure 3.6 and the colors histogram will be shrink, where pixels having two consecutive color values (even then odd) will have the same new color value thus the range of colors will be decreased from (0 – 255) to (0-127) color. Taking into consideration that this process is done to get the new (0-127) colors histogram only, and data will be embedded inside the original cover image having color ranges from (0 – 255).

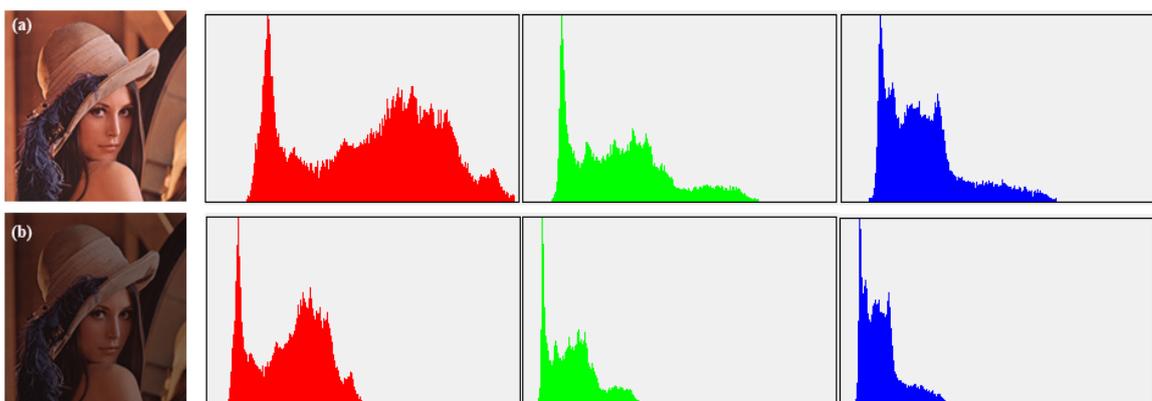


Figure 3.6 Color histogram 0 - 255 (a), Color histogram 0 - 127 (b)

3.1.4 Choosing the Appropriate Pixels to Embed the Text

In order to keep the color histogram for the cover and the stego image relatively identical, it is important to embed the text in appropriate pixels, that don't affect the color histogram shape, before and after embedding. The proposed technique scan the 128 color histogram to determine top or peak points (P) and their neighbors, to determine the shape between them and then choosing the appropriate places to embed the text with less affect to the shape, . Figure 3.7 shows some shapes between Peaks (P) and their neighbors: the first Left Neighbor (LN1), the second Left Neighbor (LN2), the first Right Neighbor (RN1) and the second Right Neighbor (RN2).

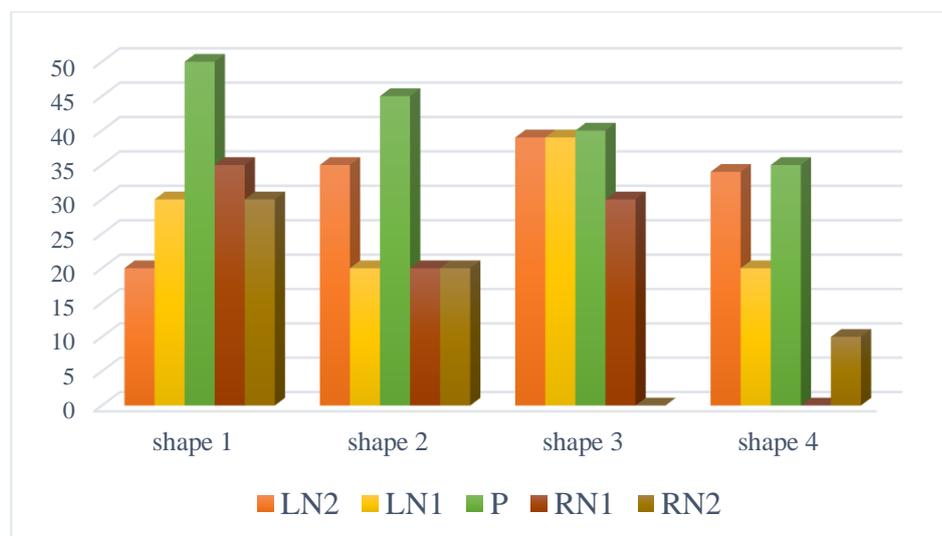


Figure 3.7 example of some shapes in the colors histogram

The differences between Peak and their left and right neighbors will be calculated, where the number of bits that will be used to embed the secret text should not exceed the difference between the peak and its neighbors. In order to prevent flatness and changes in the histogram shape the following condition must be checked:

- Peak must be greater than its neighbors at least by two.
- Neighbors at the same side should not be equal.
- Neighbor's frequency $\neq 0$, as color's frequency = 0 means that this color is not exist in the cover image.

Then, based on the frequency value of peak P and its neighbors, both sides will be checked separately. For the left side, if the first Left Neighbor LN1's frequency is greater than the second Left Neighbor LN2's frequency, the difference D will be taken between both neighbors, $D = |LN1 - LN2|$. Else, LN2's frequency is greater than LN1's, D will be taken between P and LN2, $D = |P - LN2|$. The same conditions will be applied on the right side where if $RN1 > RN2$, $D = |RN1 - RN2|$, Else $D = |P - RN2|$.

If all the condition above satisfied, the first Neighbor from Left side (LN1) or right side (RN1) or both side, will be chosen as pixels that might hold data. The amount of those pixels should not exceed the amount of difference calculated before. Example: assume that Peak was color 55, with a frequency value $f = 510$. The following Neighbors as below:

- LN1 is color 54, $f = 400$
- LN2 is color 43, $f = 0$
- RN1 is color 56, $f = 300$
- RN2 is color 57, $f = 200$

For the Left side at LN2, $f = 0$ then we will discard this side. For the Right side $P > RN1 + 2 > RN2 + 2 > 0$, $RN1 \neq RN2 \neq 0$. Then Difference $D = |300 - 200| = 100$. The color value 56 will be determined as appropriate color to embed text. Amount of pixels having the value 56 to embed text should not exceed 100 and RN1 and RN2 will be marked as used Neighbors. Figure 3.8 shows a block diagram for choosing the appropriate pixels.

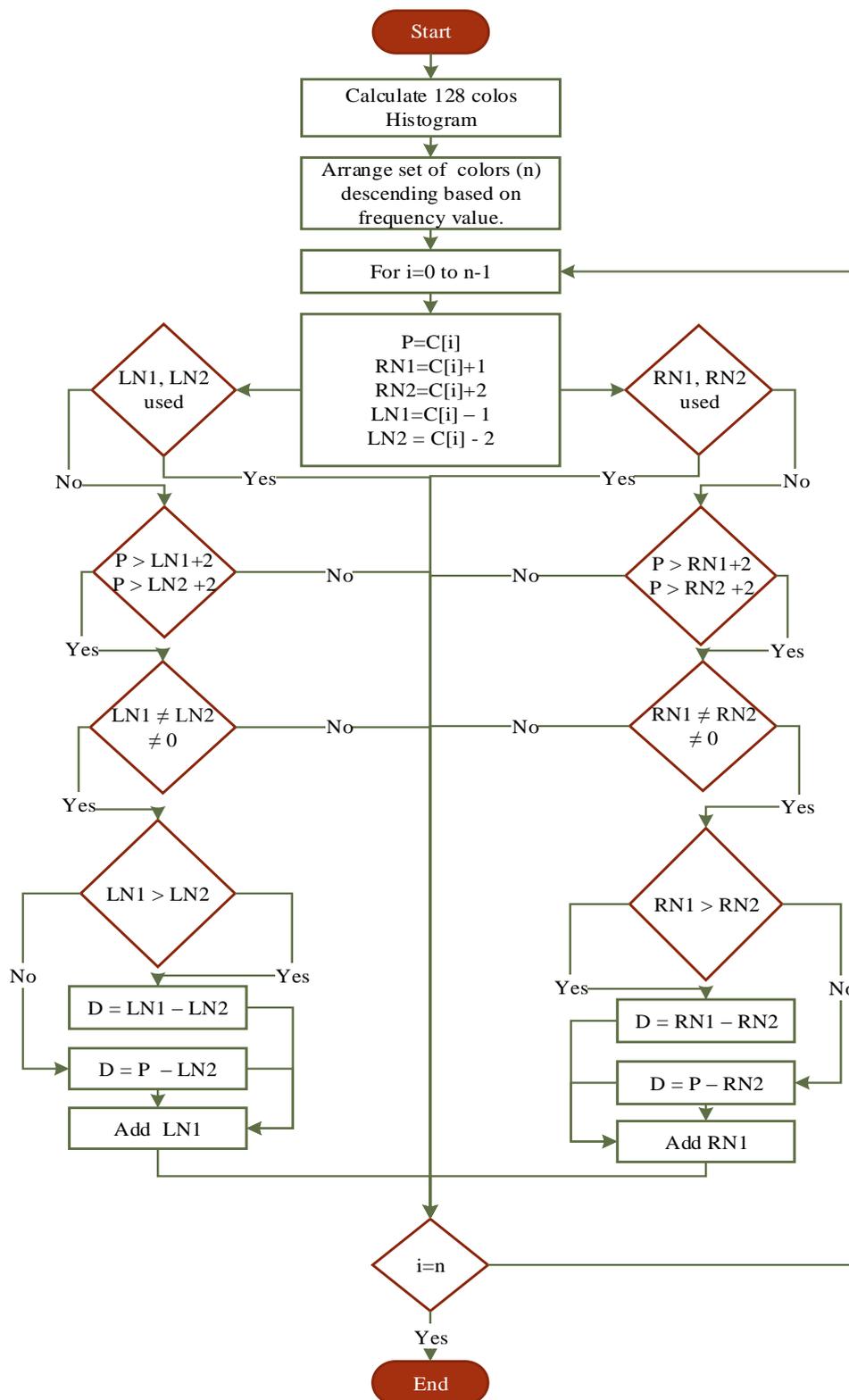


Figure 3.8 flow chart for choosing the appropriate pixels⁴

⁴P: Peak, LN1: first Left Neighbor, LN2: second Left Neighbor, RN1: first Right Neighbor,

3.1.5 Matching the Appropriate Pixels with the Pattern

At this stage the addresses of the appropriate pixels determined from the 128 color histogram, will be matched with the addresses of pixels at the three secret patterns. The pattern with the minimum number of matched pixels will be discarded. For example: consider those are the addresses of the appropriate pixels determined from the 128 red color histogram: (0,0), (0,1), (0,4), (1,2), (1,3), (1,5), (2,5), (3,1), (3,3), (4,0), (4,1), (4,4), (5,1), (5,2), (5, 3), as highlighted in red color at Table 3.4. While Table 3.5 shows the matching process between addresses of the appropriate pixels and the same addresses at the secret pattern.

Table 3.4 Appropriate pixels at red channel

	0	1	2	3	4	5
0	12	129	15	115	150	224
1	55	117	208	12	117	128
2	224	15	255	0	24	86
3	4	8	175	150	200	214
4	96	215	0	44	215	43
5	100	128	65	179	204	77

Table 3.5 Secret patterns

	0	1	2	3	4	5
0	Pt1	Pt2	Pt1	Pt2	Pt2	Pt3
1	Pt1	Pt2	Pt3	Pt1	Pt2	Pt2
2	Pt3	Pt1	Pt3	Pt1	Pt1	Pt2
3	Pt1	Pt1	Pt3	Pt2	Pt3	Pt3
4	Pt2	Pt3	Pt1	Pt1	Pt3	Pt1
5	Pt2	Pt2	Pt2	Pt3	Pt3	Pt1

Table 3.6 Matching process

	0	1	2	3	4	5
0	Pt1	Pt2			Pt2	
1			Pt3	Pt1		Pt2
2						Pt2
3		Pt1		Pt2		
4	Pt2	Pt3			Pt3	
5		Pt2	Pt2	Pt3		

Pt1 = 3
Pt2 = 8
Pt3 = 4
Minimum = Pt1

Table 3.6 shows that number of matched addresses with Pt1 = 3, Pt2 = 8 and Pt3 = 4. As a result, total number of 3 matched address with Pt1 will be discard, and 12 matched addresses with Pt2 and Pt3 will be used to embed the secret text as shown in Table 3.7.

Table 3.7 Pixel to embed data at red channel

	0	1	2	3	4	5
0	12	129	15	115	150	224
1	55	117	208	12	117	128
2	224	15	255	0	24	86
3	4	8	175	150	200	214
4	96	215	0	44	215	43
5	100	128	65	179	204	77

If the three patterns are equal in number of matched addresses, then the first pattern Pt1 and the second pattern Pt2 will be used. If two patterns are equal in number

of matched addresses, then the first pattern will be used, i.e. if $Pt1 = Pt2$ then $Pt2$ will be discarded, If $Pt1 = Pt3$ then $Pt3$ will be discarded and if $Pt2 = Pt3$ then $Pt3$ will be discarded.

Each segment may use different pattern based on number of appropriate pixels generated from the 128 color histogram. This implies more randomness and more security, where different patterns could be used at each segment.

The sequence for embedding the text bits inside the original cover image pixels will be arranged based on the addresses of the matched pixels, starting from the pattern with the maximum number of matched pixels, the embedding at these pixels will be ordered based on the first occurrence using Z shape, then after finishing all the matched pixels with this pattern, the same order will be done to the addresses matched with the next pattern until finishing all the matched pixels

For the above example the order of pixels addresses used to embed the secret text bits will be as follows: (0,1), (0,4), (1,5), (2,5), (3,3), (4,0), (5,1), (5,2), (1,2), (4,1), (4,4), (5,3).

3.2 Embedding and Extracting Steps

The main steps that are required to embed or hide text by using the proposed technique, are as follows:

- Input cover image, and secret text.
- Apply right logical shift by 1 to all pixels in the cover image and keep version of the cover without shifting.
- Partition cover image into segments.
- Input image pattern, where pattern size = Segment size.
- Determine patterns based on colors ranges.
- Find the appropriate pixels based on the difference between Peak and its neighbors.
- Match the appropriate pixels with the pattern ranges.
- Count number of matched pixels with each range.
- Compare between numbers of matched pixels of each range and discard the minimum.
- Replace LSB of Red, Green and Blue of the matched pixels with text bits.

Figure 3.9 shows the general segment diagrams of the embedding phase.

At the receiver side, steps are applied in the same form, where inputs are the stego image and the secret pattern, and the output is the secret text. Figure 3.10 shows the general block diagram for the extracting phase.

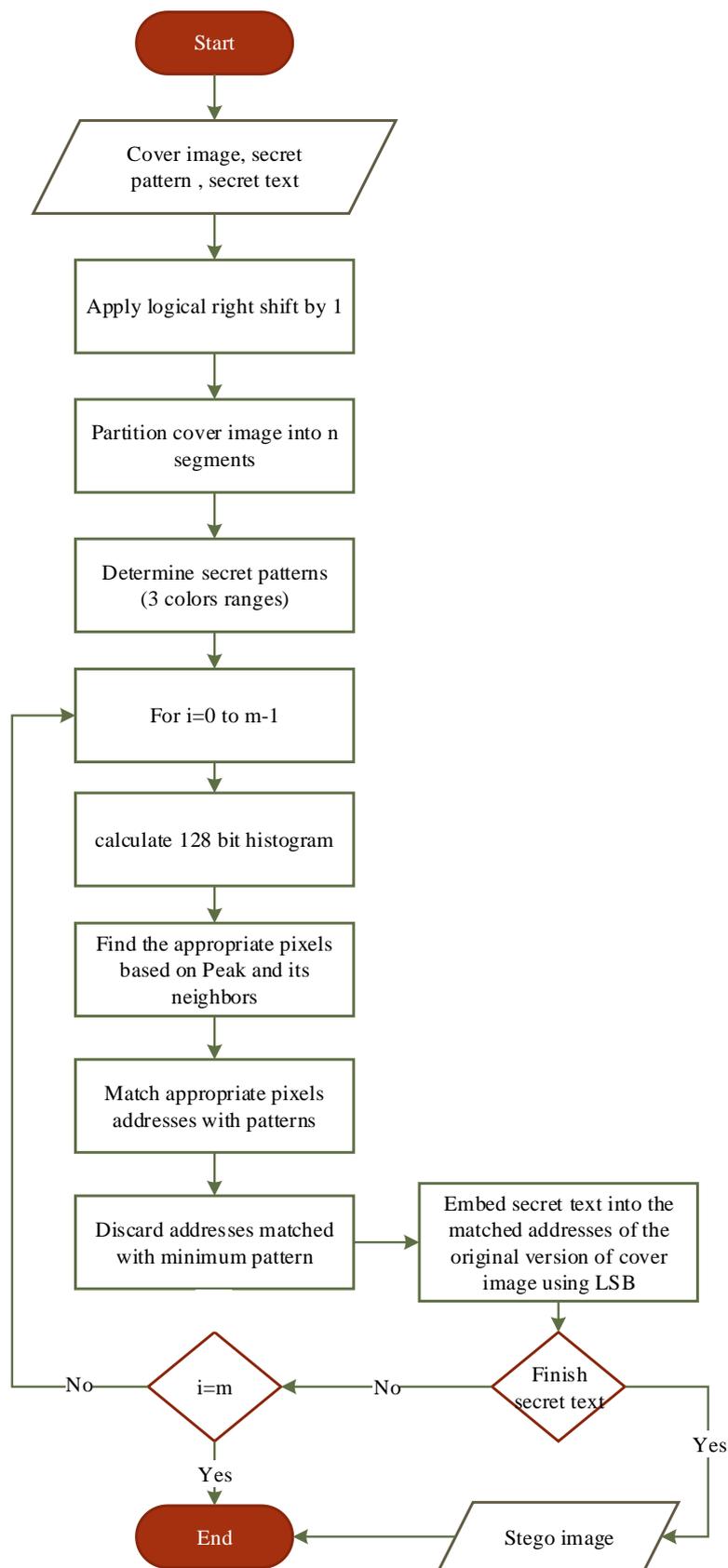


Figure 3.9 Flowchart of the embedding phase

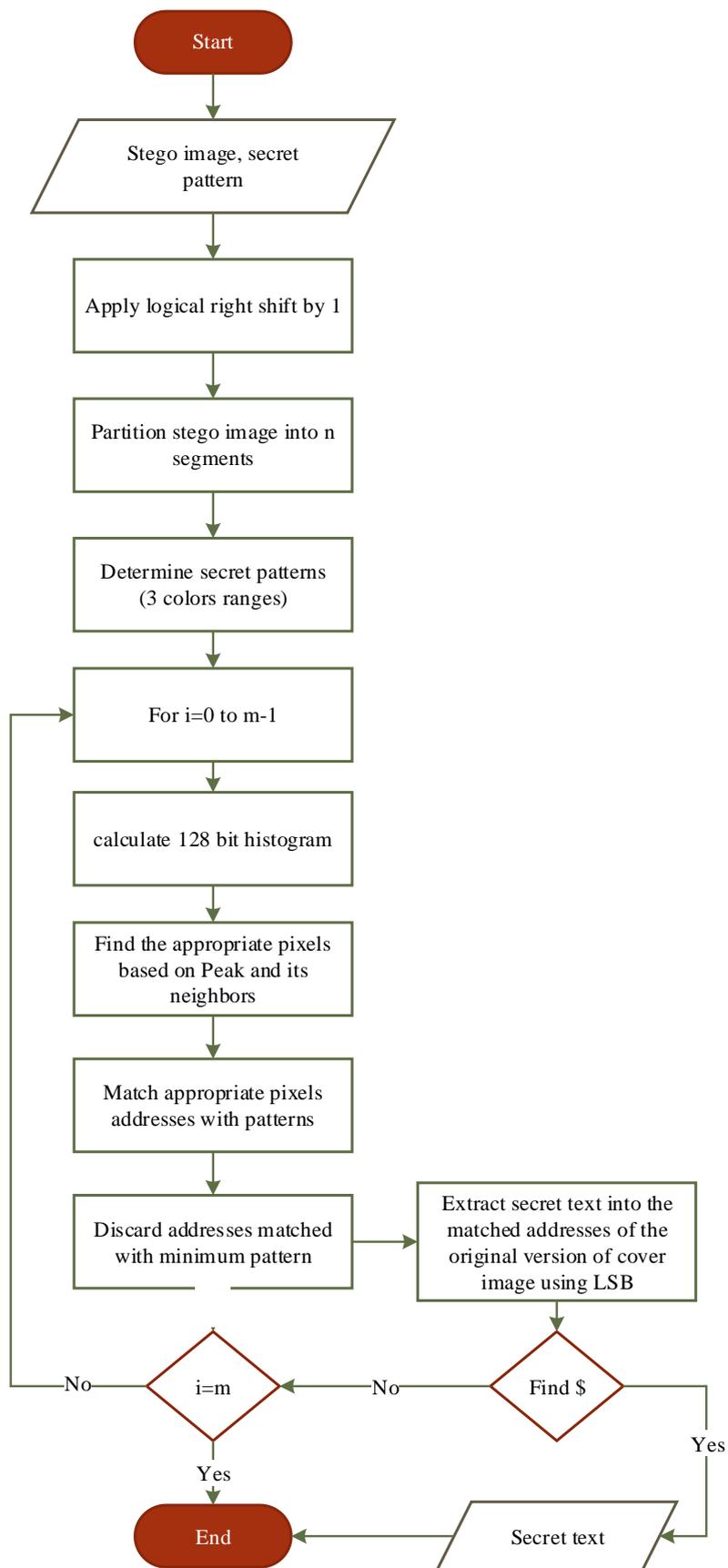


Figure 3.10 Flowchart of the extracting phase

3.3 Embedding Phase Pseudo Code

```

Embedding Input: Cover image C, pattern Pt, secret text
Convert text to binary
Calculate size of C
Partition C into  $n \times m$  segments
For i=0 to n-1
    For j=0 to m-1
        {
            Get pixel value PV from Pt
            Case (0 – 85) add address[i][j] to new Pt1
            Case (86 – 170) add address[i][j] to new Pt2
            Case (171 – 255) add address[i][j] to new Pt3
        }
    For each Block
        Apply right logical bit shift
        Calculate 128 colors histogram
        Arrange frequency values descending
        For i = 0 to 127
            {
                Set Peak P, first left neighbor LN1, second left neighbor LN2, first
                right neighbor RN1, second right neighbor RN2

                If (LN1  $\neq$  LN2  $\neq$  0) AND (both LN1 and LN2 are not used) Then
                    If (LN1 > LN2) Then
                        (D = LN1 – LN2)
                    Else (D = P – LN2)
                        If (LN1 – D)  $\neq$  0 Then
                            Choose D from LN1
                If (RN1  $\neq$  RN2  $\neq$  0) and (both LN1 and LN2 are not used) Then
                    If (RN1 > RN2) Then
                        (D = RN1 – RN2)
                    Else (D = P – RN2)
                        If (RN1 – D)  $\neq$  0 Then
                            Choose D from RN1
                Add LN1 addresses, RN1 addresses to the appropriate pixels table
                Next Peak
            }
        Match appropriate pixels table with secret pattern
        Calculate number of matched pixels at each tables
        Find Min between matched P1, P2 and P3 & Discard all pixels matched with Min
        Arrange the remaining matched addresses for each color channel.
        Replace LSB of the matched Red, Green and Blue with text bits

```

3.4 Extracting Phase Pseudo Code

```

Embedding Input: Stego image S, secret pattern Pt
Calculate size of S
Partition S into  $n \times m$  segments //should be same as the size used in the embedding
phase
For i=0 to n-1
    For j=0 to m-1
        {
            Get pixel value PV from Pt
            Case (0 – 85) add address[i][j] to new Pt1
            Case (86 – 170) add address[i][j] to new Pt2
            Case (171 – 255) add address[i][j] to new Pt3
        }
    For each Block
        Apply right logical bit shift
        Calculate 128 colors histogram
        Arrange frequency values descending
        For i = 0 to 127
            {
                Set Peak P, first left neighbor LN1, second left neighbor LN2, first
                right neighbor RN1, second right neighbor RN2

                If (LN1  $\neq$  LN2  $\neq$  0) AND (both LN1 and LN2 are not used) Then
                    If (LN1 > LN2) Then
                        (D = LN1 – LN2)
                    Else (D = P – LN2)
                        If (LN1 – D)  $\neq$  0 Then
                            Choose D from LN1
                If (RN1  $\neq$  RN2  $\neq$  0) and (both LN1 and LN2 are not used) Then
                    If (RN1 > RN2) Then
                        (D = RN1 – RN2)
                    Else (D = P – RN2)
                        If (RN1 – D)  $\neq$  0 Then
                            Choose D from RN1
                Add LN1 addresses, RN1 addresses to the appropriate pixels table
                Next Peak
            }
        Match appropriate pixels table with secret pattern
        Calculate number of matched pixels at each tables
        Find Min between matched P1, P2 and P3 & Discard all pixels matched with Min
        Arrange the remaining matched addresses for each color channel.
        Extract LSB of the matched Red, Green and Blue and convert them to the text

```

3.5 Sample of the Proposed Technique

Suppose an RGB image of BMP format, figure 3.11, has the pixels values as shown in Table 3.9, and secret text to be inserted in the image is “stg”. To embed the text bits, logical right shift by 1, will be applied to all pixels in the cover image Table 3.8 then the cover will be partitioned into number of segments assume four segments as shown in figure 3.12 the secret pattern will be determined using a secret 8-bit grayscale image, figure 3.13, has the pixels values as shown in Table 3.11, and the color histogram will be calculated for each block figure 3.14.

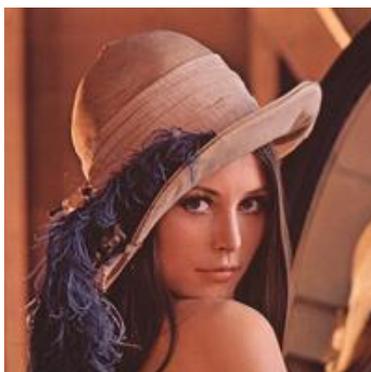


Figure 3.11 (200 × 200) BMP image



Figure 3.12 Partition cover into for blocks



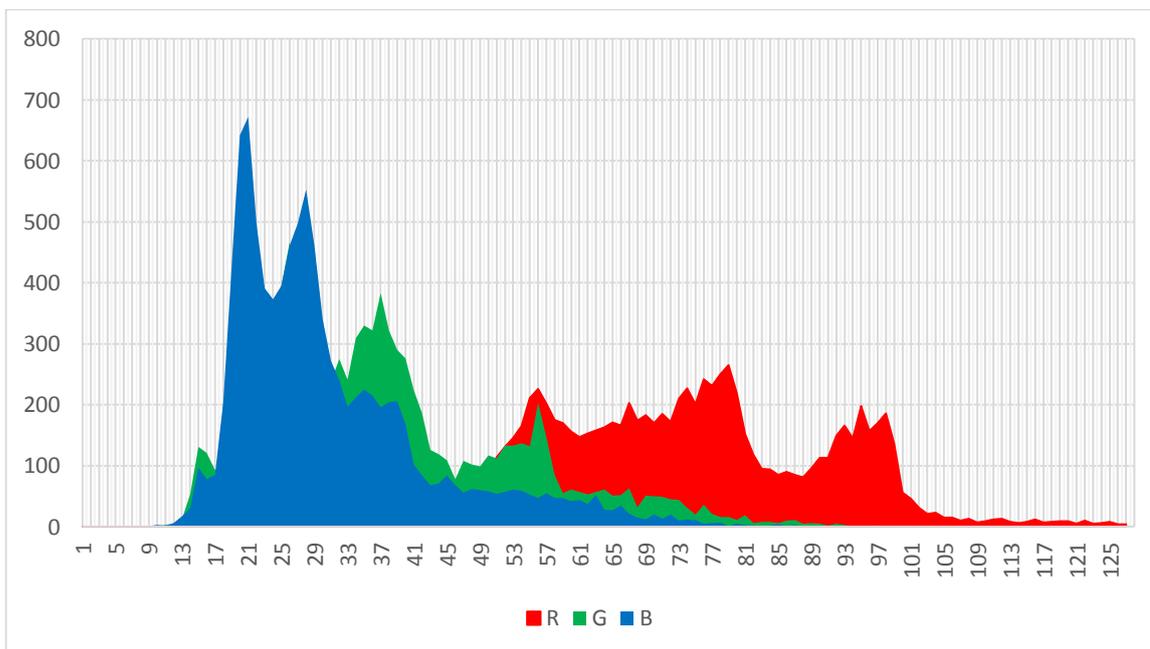
Figure 3.13 Secret pattern

Table 3.8 Color values for some pixels

	0	1	2	3	4
0	196,113,77	198,114,78	196,115,78	192,112,76	192,111,76
1	198,114,78	197,113,78	195,113,77	191,112,75	190,110,77
2	199,113,78	196,112,76	194,112,76	191,112,76	187,108,77
3	199,114,77	195,112,75	191,110,74	188,108,74	184,103,71
4	192,108,71	193,109,73	189,107,71	186,105,72	186,102,71
5	189,103,69	193,108,71	190,106,71	189,105,72	187,102,71

Table 3.9 Colors after applying right logical bit shift

	0	1	2	3	4
0	98,56,38	99,57,39	98,57,39	96,56,38	96,55,38
1	99,57,39	98,56,39	97,56,38	95,56,37	95,55,38
2	99,56,39	98,56,38	97,56,38	95,56,38	93,54,38
3	99,57,38	97,56,37	95,55,37	94,54,37	92,51,35
4	96,54,35	96,54,36	94,53,35	93,52,36	93,51,35
5	94,51,34	96,54,35	95,53,35	94,52,36	93,51,35

**Figure 3.14 128 color histogram for R, G and B**

After applying peak and neighbors algorithm using the Red, Green and blue colors histograms, the following addresses shown in Table 3.10 and highlighted in red, green and blue determined as appropriate pixels.

Table 3.10 The appropriate pixels table

	0	1	2	3	4
0	98,56,38	99,57,39	98,57,39	96,56,38	96,55,38
1	99,57,39	98,56,39	97,56,38	95,56,37	95,55,38
2	99,56,39	98,56,38	97,56,38	95,56,38	93,54,38
3	99,57,38	97,56,37	95,55,37	94,54,37	92,51,35
4	96,54,35	96,54,36	94,53,35	93,52,36	93,51,35
5	94,51,34	96,54,35	95,53,35	94,52,36	93,51,35

Three colors ranges generated from the secret pattern shown at Table 3.12.

Table 3.11 Pixels values at the secret pattern

	0	1	2	3	4
0	160	160	161	163	159
1	157	160	162	162	160
2	158	159	60	63	60
3	158	158	59	63	61
4	257	259	262	263	262
5	258	260	261	261	263

Table 3.12 Three color ranges generated from the secret pattern

	0	1	2	3	4
0	Pt2	Pt2	Pt2	Pt2	Pt2
1	Pt2	Pt2	Pt2	Pt2	Pt2
2	Pt2	Pt2	Pt1	Pt1	Pt1
3	Pt2	Pt2	Pt1	Pt1	Pt1
4	Pt3	Pt3	Pt3	Pt3	Pt3
5	Pt3	Pt3	Pt3	Pt3	Pt3

Number of matched bits for each color channel are as follows:

Red: Pt1 = 0, Pt2 = 7, Pt3 = 3

Green: Pt1 = 1, Pt2 = 4, Pt3 = 3

Blue: Pt1 = 3, Pt2 = 7, Pt3 = 0

Discard the pattern having the minimum number of matched pixels, for Red channel the pattern having minimum number of matched pixels is Pt1 with zero matched address. In the Blue channels the minimum number of matched addresses is 0 which belongs to Pt3, so no pixels will be discarded, but for the Green channel the pattern having minimum number of matched pixels is Pt1 with one address, so the address matched with Pt1 in the green channel and colored in yellow will be discarded. Shown in Table 3.13.

Table 3.13 Matched addresses

	0	1	2	3	4
0	98,56,38	99,57,39	98,57,39	96,56,38	96,55,38
1	99,57,39	98,56,39	97,56,38	95,56,37	95,55,38
2	99,56,39	98,56,38	97,56,38	95,56,38	93,54,38
3	99,57,38	97,56,37	95,55,37	94,54,37	92,51,35
4	96,54,35	96,54,36	94,53,35	93,52,36	93,51,35
5	94,51,34	96,54,35	95,53,35	94,52,36	93,51,35

Then an arrangement to the matched pixels will be applied based on the remaining matched addresses and their frequency, as shown in Table 3.14 below:

Table 3.14 Appropriate pixels order to embed the secret text

Red	0,1	0,2	0,3	0,4	1,0	2,0	3,0	4,0	4,1	5,1
Green	4,4	5,0	5,4	0,1	0,2					
Blue	0,3	0,4	1,2	1,4	2,1	3,0	2,2	2,3	2,4	

The pixels values after embedding the secret text “stg” in R, G and B channels respectively are as bellow:

“stg” in binary is: 01110011 01110100 01100111

Starting from Red channel:

Insert 0 at Pixel [0][1] Red value = 198 = 11000110 >> no changes at LSB >> 11000100
 Insert 1 at Pixel [0][2] Red value = 196 = 11000100 >> convert LSB to 1 >> 11000111
 Insert 1 at Pixel [0][3] Red value = 192 = 11000000 >> convert LSB to 1 >> 11000101
 Insert 1 at Pixel [0][4] Red value = 192 = 11000000 >> convert LSB to 1 >> 11000001
 Insert 0 at Pixel [1][0] Red value = 198 = 11000110 >> no changes at LSB >> 11000000
 Insert 0 at Pixel [2][0] Red value = 199 = 11000111 >> convert LSB to 0 >> 11000110
 Insert 1 at Pixel [3][0] Red value = 199 = 11000111 >> no changes at LSB >> 11000111
 Insert 1 at Pixel [4][1] Red value = 193 = 11000001 >> no changes at LSB >> 11000001
 Insert 0 at Pixel [5][1] Red value = 193 = 11000001 >> convert LSB to 0 >> 11000000

Green channel:

Insert 1 at Pixel [4][4] Green value = 102 = 01100110 >> convert LSB to 1 >> 01100111
 Insert 1 at Pixel [5][0] Green value = 103 = 01100111 >> no changes at LSB >> 01100111
 Insert 1 at Pixel [5][4] Green value = 102 = 01100110 >> convert LSB to 1 >> 01100111
 Insert 0 at Pixel [0][1] Green value = 114 = 01110010 >> no changes at LSB >> 01110010
 Insert 1 at Pixel [0][2] Green value = 115 = 01110011 >> no changes at LSB >> 01110011

Blue channel:

Insert 0 at Pixel [0][3] Blue value = 76 = 01001100 >> no changes at LSB >> 01001100
 Insert 0 at Pixel [0][4] Blue value = 76 = 01001100 >> no changes at LSB >> 01001100
 Insert 0 at Pixel [1][2] Blue value = 77 = 01001101 >> convert LSB to 0 >> 01001100
 Insert 1 at Pixel [1][4] Blue value = 77 = 01001101 >> no changes at LSB >> 01001101
 Insert 1 at Pixel [2][1] Blue value = 76 = 01001100 >> convert LSB to 1 >> 01001101
 Insert 1 at Pixel [3][0] Blue value = 77 = 01001101 >> no changes at LSB >> 01001101

Total of 9 bits are changed. A special character could be added for the receiver to determine end of the secret text.

Chapter 4 Experimental Results

4.1 Implementation

The embedding and extracting algorithms were implemented using C# on visual studio 2010, shown in Figure 4.1. And applied on a bit mapped (BMP) images that have different sizes with 256 colors.

To evaluate the impact of the embedding process on the images, different secret texts have been embedded. It is to be noted that the analysis is carried out three cover images, and they were partitioned into different Blocks and two different images used as secret patterns.

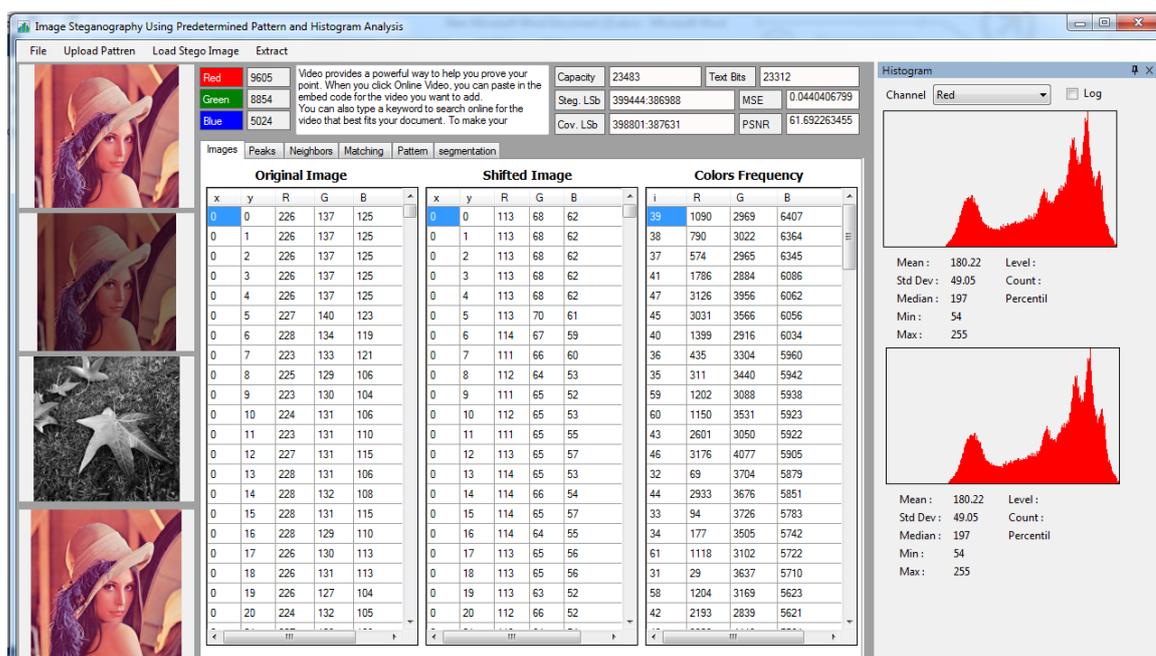


Figure 4.1 C# application interface of the proposed technique

4.2 Implementation Results

An evaluation of the three major properties of any image steganography technique: undetectability, level of security and capacity has been obtained, in order to characterize the strengths and weaknesses of the proposed technique.

The cover images used to test the proposed technique are shown in Figure 4.2, which are Lenna, Baboon and Peppers. And each of size 512×512 .

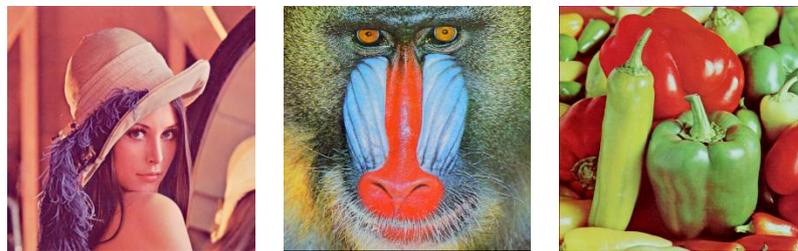


Figure 4.2 Cover images used to test the proposed technique

The secret patterns are as shown in Figure 4.3, which are Leaves and kids. The size of them is related to Block size.



Figure 4.3 Secret patterns used to test the proposed technique

4.2.1 Capacity, MSE and PSNR

In order to evaluate the embedding capacity of the proposed technique different blocks sizes has been used with different secret patterns. And to evaluate the amount of imperceptibility and the quality of the stego image each of MSE and PSNR have been calculated. Tables 4.1- 4.6, give the PSNR and MSE statistics of Lenna, Baboon and pepper images with different block sizes and different embedding capacity.

Table 4.1 Embedding capacity, MSE and PSNR of one block (512 × 512 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	23472	0.04	61.60	14440	0.02	63.74	21008	0.03	62.17
kids	22752	0.04	61.71	14984	0.02	63.61	21104	0.04	62.06

Same number of segments but in different height and width gives different data embedding capacity, MSE, PSNR as shown in Table 4.2 and Table 4.3.

Table 4.2 Embedding capacity, MSE and PSNR of two segments (512 × 256 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	24456	0.04	61.46	15696	0.02	63.38	23648	0.04	61.59
kids	23912	0.04	61.48	16224	0.03	63.23	23672	0.04	61.59

Table 4.3 Embedding capacity, MSE and PSNR of two segments (256 × 512 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	25736	0.04	61.19	15520	0.02	63.42	22912	0.04	61.70
Kids	25952	0.04	61.15	15576	0.02	63.42	23168	0.04	61.69

Decreasing the block size gives better results as shown in Table 4.4, 4.5, 4.6 and 4.7 where number of blocks increases.

Table 4.4 Embedding capacity, MSE and PSNR of four segments (256 × 256 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	29312	0.05	60.65	17984	0.03	62.73	28440	0.05	60.80
Kids	28600	0.05	60.74	17856	0.03	62.80	27240	0.05	61.01

Table 4.5 Embedding capacity, MSE and PSNR of nine segments (170 × 170 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	38480	0.07	59.51	22120	0.04	61.87	34064	0.06	59.95
Kids	36744	0.06	59.68	21800	0.04	61.94	32712	0.06	60.21

Table 4.6 Embedding capacity, MSE and PSNR of 16 segments (128 × 128 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	43944	0.08	58.89	25464	0.04	61.26	38592	0.07	59.47
Kids	41840	0.07	59.10	24720	0.04	61.41	36800	0.06	59.70

Table 4.7 Embedding capacity, MSE and PSNR of 100 segments (50 × 50 pixels)

Secret pattern	Lenna			Baboon			Peppers		
	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR	Capacity bits	MSE	PSNR
Leaves	59248	0.11	57.59	43168	0.08	58.98	52248	0.09	58.16
Kids	54224	0.10	57.99	40624	0.07	59.24	48168	0.09	58.52

The values of PSNR obtained are greater than 50 dB and MSE value is less than 1 i.e. higher Peak Signal to Noise Ratio (PSNR) and lower Mean Square errors (MSE) prove that the proposed technique has good quality of the stego images that is highly acceptable by the human visual system HVS. Figures 4.4, 4.5 and 4.6 shows a flow chart of the MSE and PSNR of the stego images Lenna, Baboon and Peppers using the same pattern, and how MSE increases and PSNR decreases by using larger number of blocks, in other word a smaller block size.

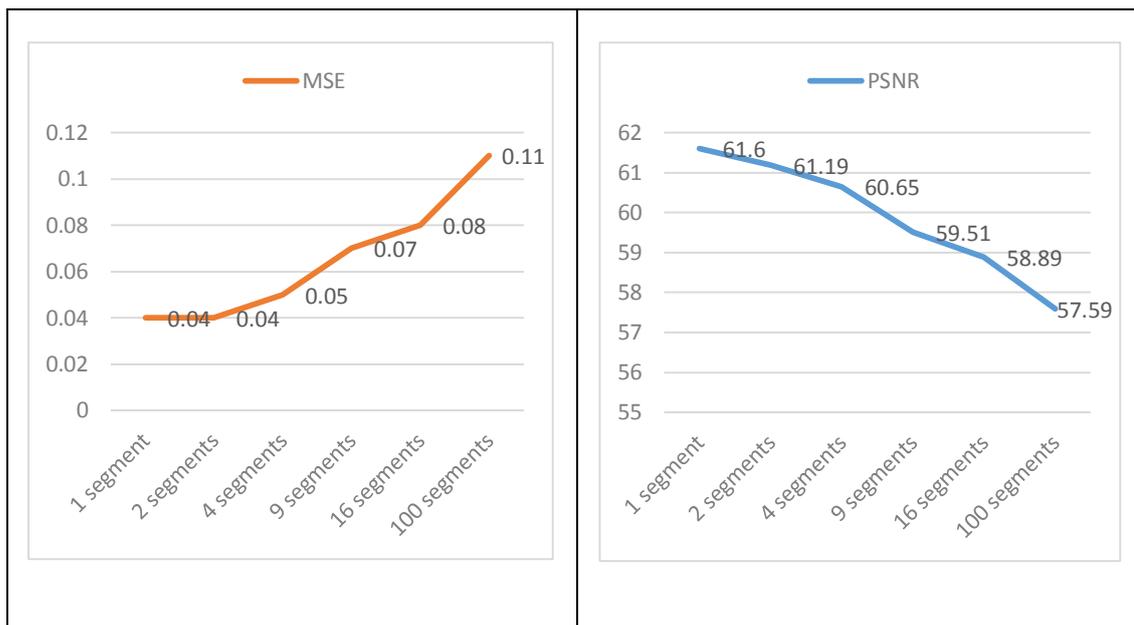


Figure 4.4 MSE and PSNR based block size (Lenna)

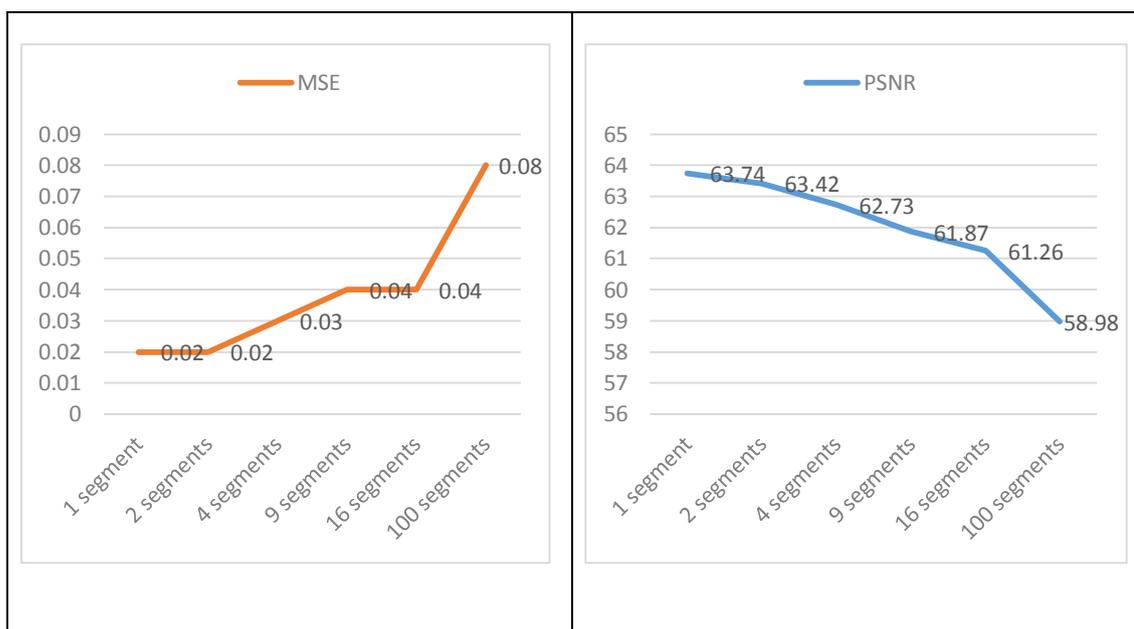


Figure 4.5 MSE and PSNR based block size (Baboon)

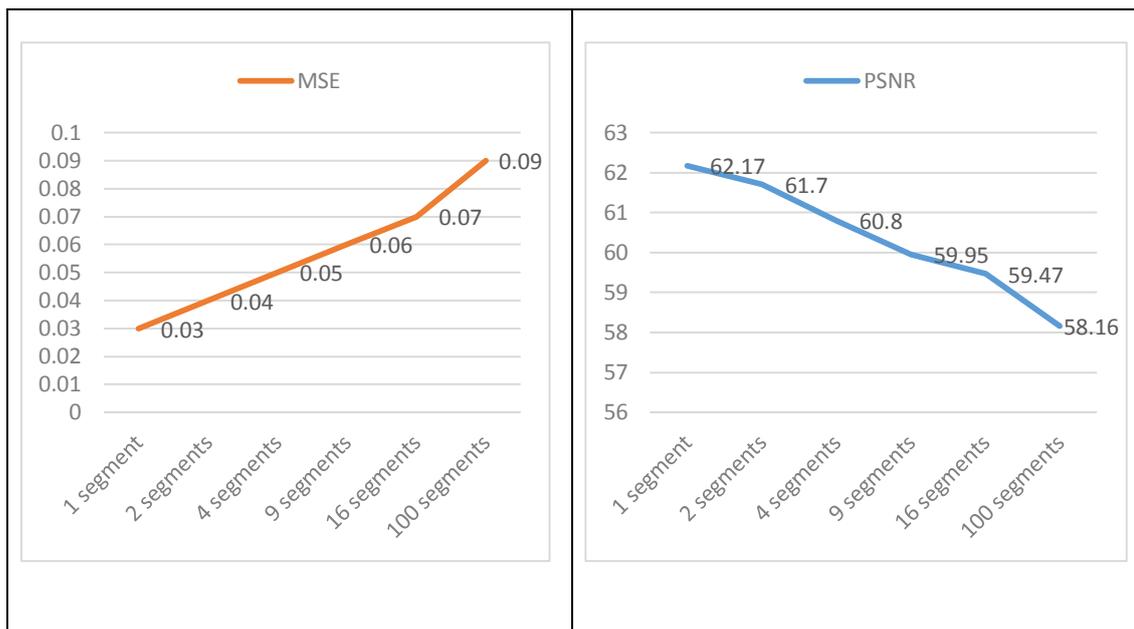


Figure 4.6 MSE and PSNR based block size (Peppers)

4.2.2 Undetectability

Experimental visual results of the proposed technique on the well-known images Lenna, Baboon and Peppers (512×512 pixel) are displayed in Table 4.6. Finding the small color change in the cover and stego image by human eye is so hard. This is the most important of any image steganography technique, well achieved by the proposed technique.

As it is very hard to find the color change in the stego image, a histogram comparison is used to identify the stego image by comparing the histogram of cover image and stego image. Based on the experiments and separate histograms drawn for cover and stego image. It's observed that there is almost no major difference visible while comparing Red, Green and Blue histogram of both cover and stego image histogram, as the shape is remain the same with respect to the maximum secret data embedding

capacity, Tables 4.7, 4.8 and 4.9 show the histogram of RGB channels of cover and stego images (Lenna, Baboon and Pepper) of size (512×512) pixels) using the same secret pattern.

Table 4.8 Comparison between cover and stego image using the proposed technique

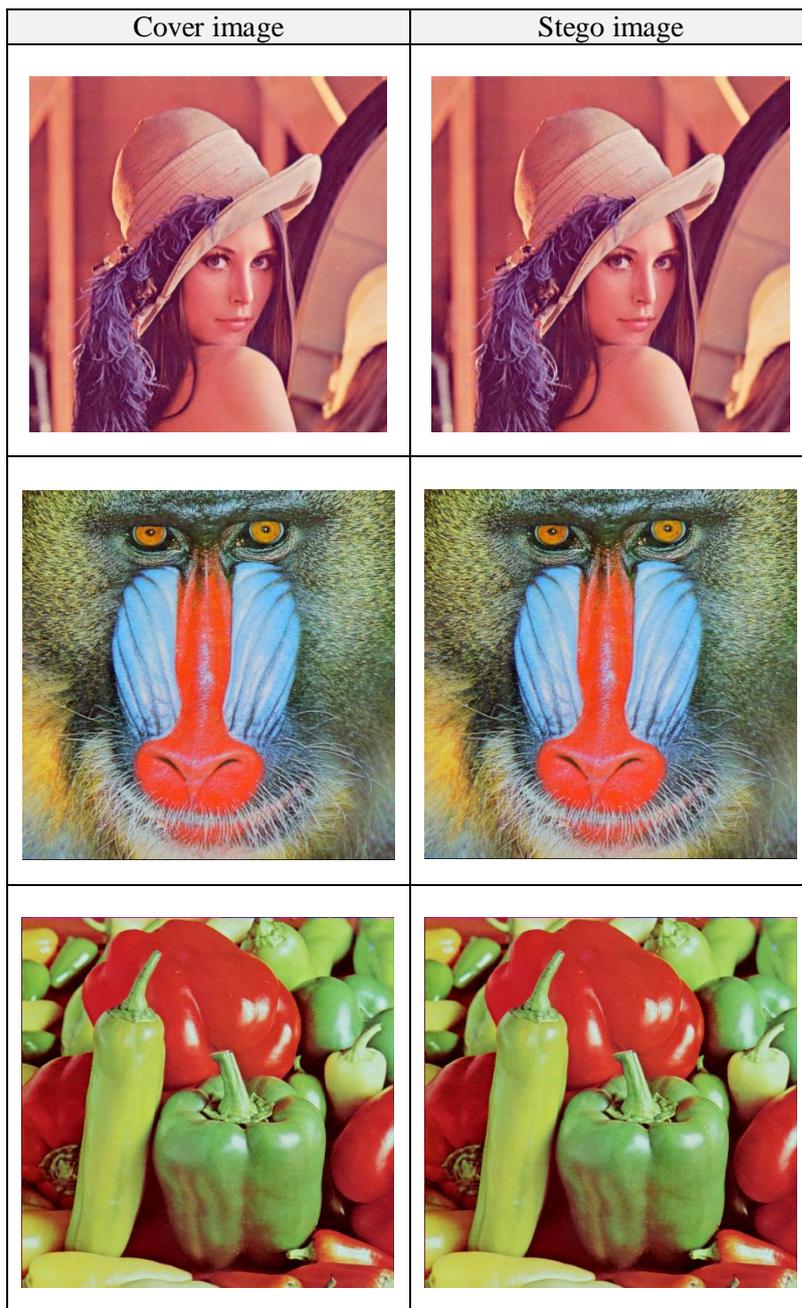


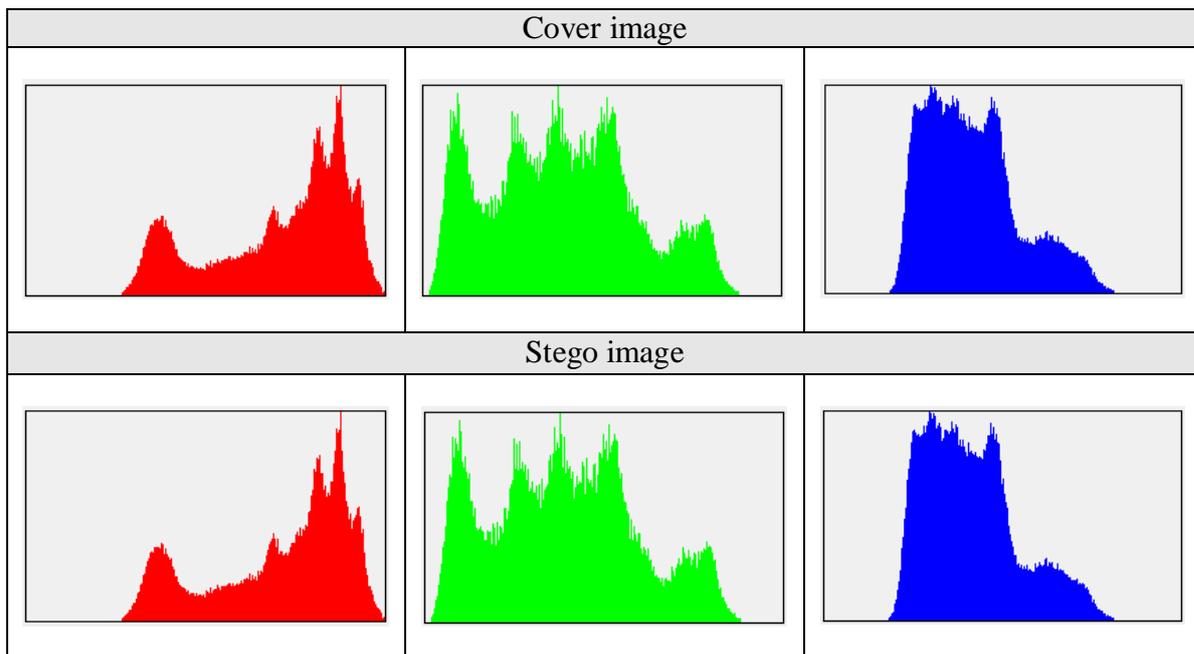
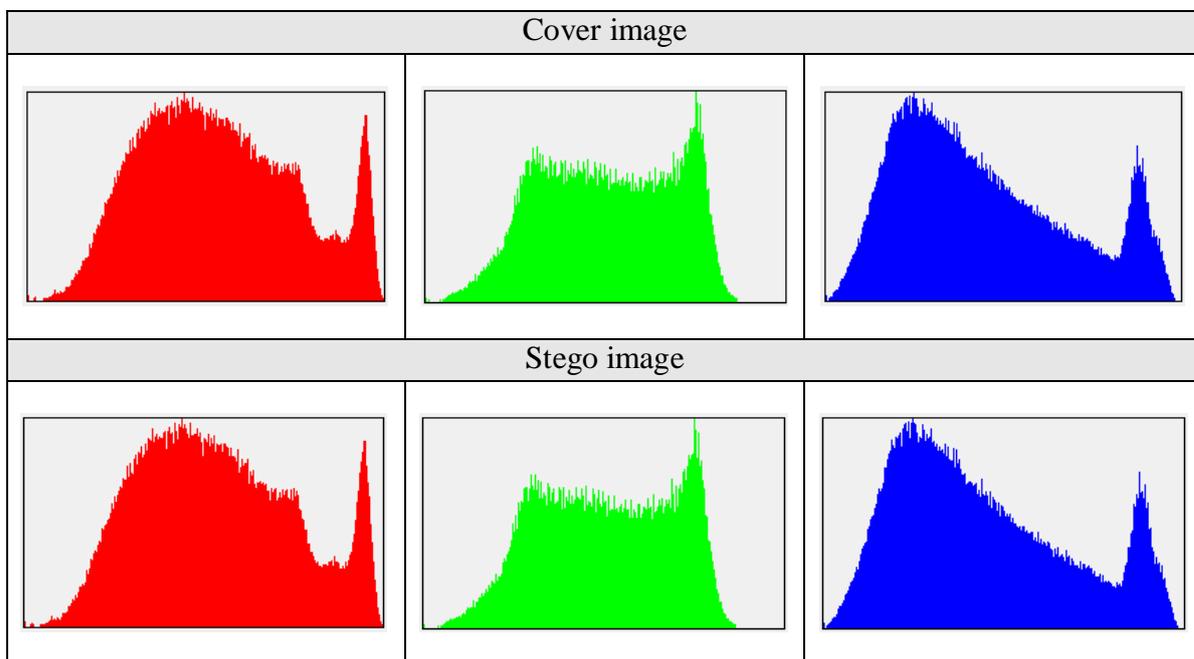
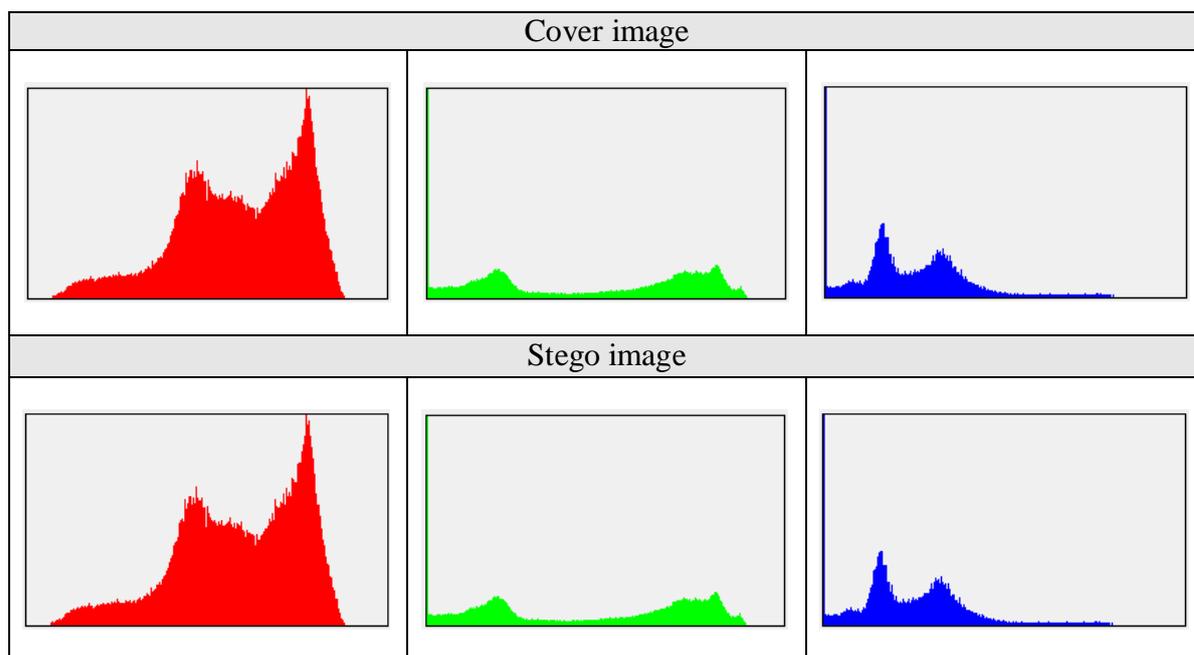
Table 4.9 RGB channels histograms for the cover and stego image (Lenna)**Table 4.10 The histogram of RGB channels of the cover and stego image (baboon)**

Table 4.11 The histogram of RGB channels of the cover and stego image (pepper)

4.2.3 Robustness and Level of Security

One of the common attacks over image steganography techniques is Brute force attack. Which involves in trying an exhaustive search on all possible stego keys until a valid key is found. In the proposed steganography technique the first part of the stego key which is the predetermined pattern, could be chosen from infinite number of images over the Internet, mobile phones, personal computers etc., so the number of distinct secret patterns generated is very huge and it is very hard to find it by brute force attack. In addition to the secret pattern the attacker also needs to know the size of block used to embed the secret text as the secret pattern size is related to the block size, and huge number of block sizes could be obtained. Moreover, based on the distribution of color ranges in the secret pattern and number of matched pixels, the text will be embedded in the random selected matched pixels in a random order, based on colors ranges

distribution. And without using the stego key, it is very hard to extract the hidden text by looking for a defined pattern in the stego image, in which the embedding order is taken place, as the text is hidden at each block in different addresses based on the cover image attributes and the secret pattern used. Table 4.10 shows an example of part of the addresses order, for embedding the text in the Red channel.

Table 4.12 Order of the addresses of pixels to embed the text in Red channel

X	Y	Color
4	61	47
4	62	47
1	51	44
1	52	44
2	56	41
7	65	41
7	67	41
0	60	36
26	91	36
26	98	36
0	56	38

While the addresses for embedding the text in the blue channel is shown at Table 4.11.

Table 4.13 Order of the addresses of pixels to embed the text in Blue channel

X	Y	Color
115	37	109
0	57	5
1	56	5
13	72	5
15	73	5
0	158	79
0	159	79
17	88	79
32	74	79
1	149	81
3	157	81

Table 4.12, shows some of the addresses order for embedding the text in the Blue channel.

Table 4.14 Order of the addresses of pixels to embed the text in Green channel

X	Y	Color
35	39	71
35	148	71
38	34	71
39	11	71
2	26	25
14	72	25
15	73	25
21	9	25
21	91	25
24	101	25
21	126	98

Since the proposed technique does not need to send the cover image to the receiver side, it is a blind technique which makes it more secure. As a result the proposed technique is secure and hence robust enough. But it is not resistant to main geometrical attacks like rotation, warping and scattered tiles.

4.3 Comparison with Other Steganography Techniques

The following Tables 4.13 and 4.14, provide a comparisons between the proposed technique and other random selected based techniques, without any text compression method.

The classical LSB method provides high capacity but less quality and can be detected easily by extracting the bits sequentially. While the proposed technique is more secure as it distribute the text bits randomly using a strong stego key.

Table 4.13 shows a comparison between the proposed technique and histogram modification method.

Table 4.15 Comparison between the proposed technique and histogram modification method

	Histogram Modification		The Proposed technique	
	Capacity	PSNR	Capacity	PSNR
Lenna	665	62.75	23472	61.60
Baboon	747	59.25	14440	63.74
Peppers	700	56.01	21008	62.17

The histogram modification method provides less security than the proposed technique as it is easy to extract the embedded text directly by obtaining the histogram and applying mod 2 to each bin in the histogram. Also it provides less capacity as it uses the histogram bins to hide text, and at the best case without segmentation or histogram equalization process it will have 255 bins, and for 3 color channels it will be $255 \times 3 = 765$ bins hold secret text. But the main advantage of histogram modification method is its resistance to main geometrical attacks like rotation, warping and scattered tiles while the proposed technique is not.

Table 4.14 shows a comparison between n-queen and the proposed technique.

Table 4.16 Comparison between the proposed technique and n-queen technique

N-queen		The Proposed technique	
capacity	PSNR	capacity	PSNR
Up to 15 character	> 50	Greater than 15 character	> 50

N-queen technique provides high security by using a large space of stego key size, but it provides small capacity, as it can hide up to 15 character per image although it uses arithmetic compression technique. The proposed technique is also provides high security using a large space of key size, but it provides more capacity than the N-queen technique, without using any compression techniques.

All of image steganography techniques have respective strong and weak points, the main strong points of the proposed technique can be summarized as follows:

- Providing high security using a large size and large space of stego keys.
- Providing high capacity comparing to other random selection based techniques.
- Providing high stego image quality (visually and statistically).

4.4 Analysis and Discussion:

The experimental results of the proposed image steganography technique, have shown different statistics for the embedding capacity while using different blocks sizes and different secret pattern. The relation between the embedding capacity, block size and the secret pattern will be discussed below.

4.4.1 Relation between Block Size and Embedding Capacity

The embedding capacity of the proposed technique is improved by the block size. Figure 4.7 shows that the embedding capacity of the cover image is increased by decreasing the block size. As much as the cover image size increases as much the number of blocks could be increased, thus the imbedding capacity may also increase.

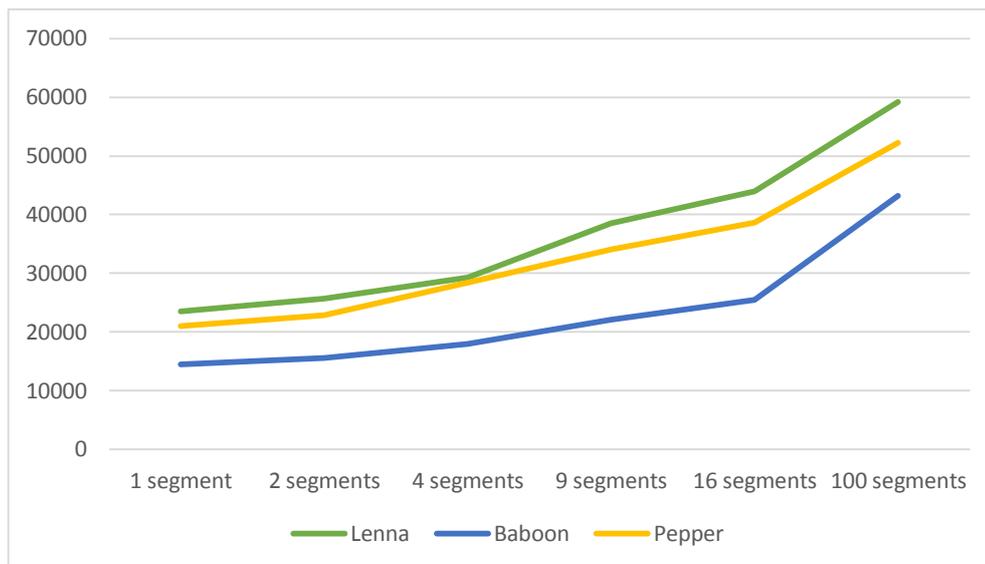


Figure 4.7 Relation between capacity and block size

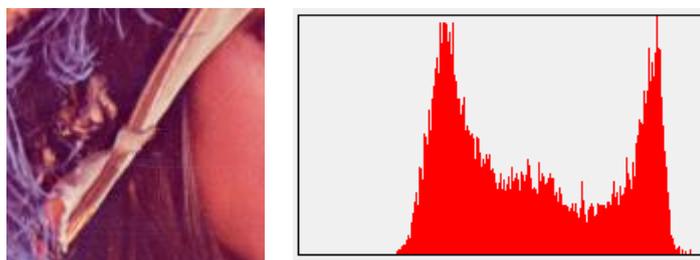
Moreover, blocks with more difference between colors frequency show better embedding capacity than the ones with less difference, also the number of different colors in the block will affect the histogram range, where it will be increased, and this might increase the number of peaks and neighbors and based on difference between colors frequency it might increase number of the appropriate pixels picked from the histogram. As shown in Figure 4.8, the image is partitioned into two segments with the same size but different height and width. Segment (a) is 256×500 . And segment (b) size is 500×256 , the embedding capacity of the image using the same secret pattern is 25664_{bpp} by block (a), and 24500_{bpp} by block (b).



Figure 4.8 Two different block sizes for one image

Figure 4.9 shows the red histogram of two different blocks both of them have the size of 128×128 pixel. The embedding capacity at the red channel, for each block, using the same secret pattern are as follows:

- Block (a) = 654 bpc.
- Block (b) = 685 bpc.
- Block (c) = 1118 bpc.



(a)

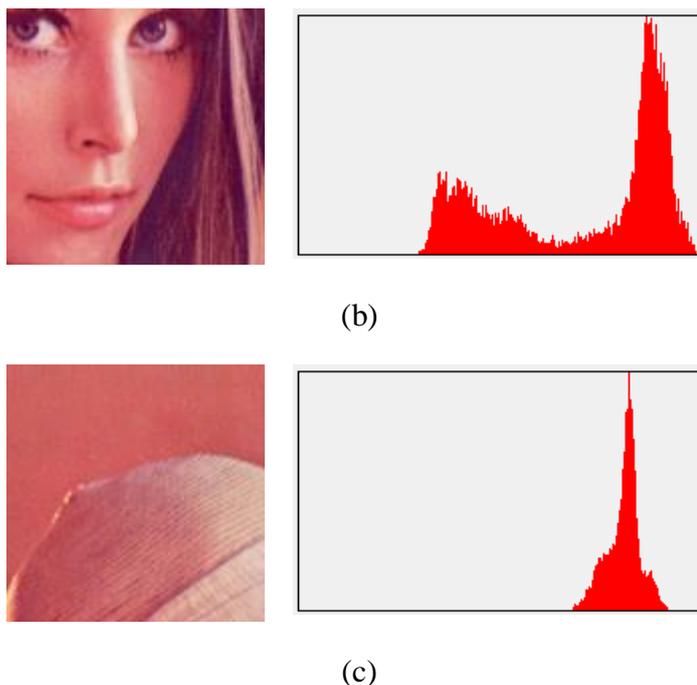


Figure 4.9 Red histogram of three different segments (128×128 pixel)

Block C has the maximum capacity based on the difference between Peaks and neighbors, although it has the smallest colors range histogram.

4.4.2 Relation between the Secret Pattern and Embedding Capacity

The secret pattern affect the embedding capacity by discarding number of pixels, after the matching process between the secret pattern and the appropriate pixels, which picked from the 128 color histogram. The random distribution of colors ranges in the secret pattern does not affect the capacity. For example: having 3 color ranges Pt1, Pt2 and Pt3 each having the following number of colors: 1500, 800 and 200 respectively, If number of matched pixels with Pt1 = 510, Pt2 = 70 and Pt3 = 150 then total number of 660 pixels matched with Pt1 and Pt2 will be used to embed the secret text, and 70 pixels

matched with Pt2 will be discard. Although the number of colors in Pt2 is greater than Pt1 in the secret pattern. Thus no one can guess the number of pixels used to embed the secret text simply by counting number of colors in each range in the secret pattern.

Above all, it is important to point out that image attributes are playing an important role in increasing or decreasing the embedding capacity, as different images with the same size gives different embedding capacity.

Chapter 5 Conclusion and Future Work

5.1 Conclusion

In this thesis a proposed steganography technique to hide text inside image based on predetermined pattern and histogram analysis is presented. The embedding and extracting processes are done using the principle of LSB, where the secret text bits are hidden in the least significant bits of the selected pixels, with more randomization in selection of the number of pixels used. Two parts of stego key were presented. They are used to control the embedding and extracting processes, the first part is the secret pattern, and the second is the color histogram.

The advantages of the proposed technique can be summarized as follows, imperceptibility on stego image and its histogram, high statistical image quality measurements, i.e. higher Peak Signal to Noise Ratio (PSNR) and lower Mean Square errors (MSE) prove that the proposed technique has good quality of the stego images, that is highly acceptable by the human visual system HVS.

Experimental results of the proposed technique for well-known test images Lena, Baboon and Peppers clearly show that the visual differences between the original and the corresponding stego images, with random hidden text bits, cannot be detected by the human visual system. The stego key can be obtain from a huge key space which make it hard to be detected by brute force attack, which increases the level of security for the proposed technique.

The comparison between the proposed technique and the classical LSB shows that the proposed technique has high quality of the stego image and more security as it

distribute the secret text randomly in the cover image. The capacity of the proposed technique shows better results than histogram modification and n-queen technique, also the quality of the stego image still high. All of these results confirm the success of the proposed technique to data hiding over random selection based image steganography methods.

5.2 Limitations

Limitations of this study are:

- Pixels may generally concentrated in one particular frequency region of the color histogram, this limits the embedding capacity.
- The difference between colors frequency may be so small, this will also limits the embedding capacity.
- Size of the cover image.
- Size of the secret text.
- Processing on the stego image will cause secret data loss.

5.3 Future Work

There will be a lot of scope for further enhancement to this technique, some of which are listed below:

1. An image or other digital media file can be embedded in a cover image.
2. Increase the embedding capacity using text compression techniques.
3. Applying the proposed technique in video steganography model.
4. Implement the technique in other color models and image formats.

References:

Bahirat R. & Kolhe A. (2014). Overview of secure data transmission using steganography. *International Journal of Emerging Technology and Advanced Engineering*. Volume (4), issue (3), ISSN: 2250-2459

Bhagat A. R. & Dhembhare A. B. (2015). An efficient and secure data hiding technique – steganography. *International Journal of Innovative Research in Computer and Communication Engineering*. Vol. 3, Issue 2, ISSN (Online): 2320-9801, ISSN (Print): 2320-9798

Bashardoost M., Sulong G. & Gerami P. (2013). Enhanced LSB image steganography method by using knight tour algorithm, Vigenere encryption and LZW compression. *IJCSI International Journal of Computer Science Issues*, Vol. 10, Issue 2, No 1, March 2013 ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784 www.IJCSI.org.

Chaitanaya G., Krishna D. & Anganeyulu (2013) L. A 3-Level secure histogram based image steganography technique. *I.J Image, graphics and signal processing*, 2013, vol. 4, 60-70

Desai M.B. & Patel S.V. (2014). Survey on universal image steganalysis. (*IJCSIT*) *International Journal of Computer Science and Information Technologies*, Vol. 5 (3), 2014, 4752-4759

Gawade R., Shetye P., Bhosale V. & Sawantdesai P N. (2014). Data hiding using steganography for network security. *International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 2.*

Gupta S. & Sandhu P.S. (2013). Exploiting the RGB intensity values to implement a novel dynamic steganography scheme. *International Journal of Research in Engineering and Technology (IJRET)* Vol. 2, No. 4, ISSN 2277 – 4378.

Gutub A.A. (2010). Pixel indicator technique for RGB image steganography. *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no.1

Hamid N., Yahya A., Ahmad R.& Al-Qershi O. (2012) Image Steganography Techniques: An Overview. *International Journal of Computer Science and Security (IJCSS)*, Volume (6): Issue (3)

Kaur S., Kaur A. & Singh K. (2014). A survey of image steganography. *International Journal of Review in Electronics & Communication Engineering (IJRECE)* Volume 2 - Issue 3 p-ISSN 2321-3159

Krishna, S. L. V., Rahim, B. A., Shaik, F. & Rajan, K. S. (2010). Lossless embedding using pixel differences and histogram shifting technique. *IEEE Recent Advances in Space Technology Services and Climate Change (RSTSCC)*, pp. 213 –216, 2010.

Ni, Z. et.al. (2006). *Reversible data hiding*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 3, Mar. 2006, pp. 354–362.

Nilizadeh A.F. & Nilchi A.R. (2013). *Steganography on RGB images based on a “matrix pattern” using random blocks*. IJ.Modern Education and Computer Science, 2013, vol. 4, 8-18 <http://www.mecs-press.org/>

Pavani M., Naganjaneyulu S., Nagaraju C. (2013). A survey on LSB based steganography methods. *International Journal of Engineering and Computer Science*, ISSN: 2319-7242, Volume 2, Page No. 2464-2467

Purohit A., & Sridhar P.S. (2014). Image steganography: a review. (*IJCSIT*) *International Journal of Computer Science and Information Technologies*, Vol. 5(4), 2014, 4891-4896. ISSN: 0975-9646.

Rana R. & Singh D. (2010). Steganography-concealing messages in images using LSB replacement technique with pre-determined random pixel and segmentation of image. *International Journal of Computer Science & Communication* Vol. 1, No. 2, pp. 113-116

Sharma M.K, Mohd N. & Sharma R. (2014). A new steganography technique based on difference scheme of RGB channels and text using histogram analysis. *Int. Journal of Engineering Research and Applications*. www.ijera.com Vol. 4, Issue 5 ISSN: 2248-9622.

Singh A., Dhanda S.K & Kaur R. (2014). *Secure image steganography using n-queen puzzle and its comparison with LSB technique*. International Journal of Innovations and technology (IJIET)

Subhedar M.S. & Mankar V.H. (2014). Current status and key issues in image steganography: a survey. *Computer Science Review*, volume 13-14, <http://dx.doi.org/10.1016/j.cosrev.2014.09.001>

Sumathi C.P., Santanam T. & Umamaheswari G. (2013). A study of various steganographic techniques used for information hiding. *International Journal of Computer Science & Engineering Survey (IJCES)*, Vol.4, No.6

Swain G. & Lenka S.K. (2012). A novel approach to RGB channel based image steganography technique. *International Arab Journal of e-Technology*, Vol.6, No. 4, June 2012

Thiyagarajan P., Aghila G., Venkatesan V. (2010) Dynamic pattern based image steganography. *Journal of computing*, volume 2, issue 8, ISSN 2151-9617.

Tiwari N., Sandilya M. & Chawla M. (2014). Spatial domain image steganography based on security and randomization. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, Vol. 5, No. 1, 2014

Venkata M., et.al (2009), Pixel intensity based steganography with improved randomness, *International Journal of Computer Science and Information Technology*, Vol 2, No 2, pp.169-173.

Vyas K. & Pal B.L. (2014). A proposed method in image steganography to improve image quality with LSB technique. *International Journal of Advanced Research in Computer and Communication Engineering*, Vol.3, Issue 1

Yalman Y. & Erturk I. (2009). A new histogram modification based robust image data hiding technique, *IEEE 24th Int. Symposium on Computer and Information Sciences (ISCIS'09)*, pp.39–43.

Appendices

Appendix A: shifting Image code

The following code describes shifting the least significant bits of image pixels using C#.

```
private void shiftingToolStripMenuItem_Click(object sender, EventArgs e)
{
    Bitmap bm = new
    Bitmap(System.Drawing.Image.FromFile(ImgOpenFileDialog.FileName)),

    // Bitmap newBitmap = null,
    Bitmap newBitmap = new Bitmap(bm),

    for (int i = 0, i < bm.Height , i++)
    {
        for (int j = 0, j < bm.Width , j++)
        {
            //get the pixel from the original image
            Color originalColor = bm.GetPixel(i, j),

            int RT = originalColor.R,
            if (RT % 2 == 1) { RT -= 1, }
            RT = RT / 2,

            int GT = originalColor.G,
            if (GT % 2 == 1) { GT -= 1, }
            GT = GT / 2,

            int BT = originalColor.B,
            if (BT % 2 == 1) { BT -= 1, }
            BT = BT / 2,

            //create the color object
            Color newColor = Color.FromArgb(RT, GT, BT),

            //set the new image's pixel
            bm.SetPixel(i, j, newColor),

        }
    }
}
```

Appendix B: set colors ranges

The following code describes setting colors ranges in the secret pattern using

C#.

```
private void uploadPattrenToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog(),
    open_dialog.Filter = "Image Files (*.bmp)|*.bmp",

    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        Bitmap bm = new Bitmap(open_dialog.FileName),
        PattrenImgPictureBox.Image = bm,

        double rows = (bm.Width), //No of Rows as per Desire
        double columns = (bm.Height), //No of columns as per Desire
        Color pixelColor,

        for (int y = 0, y < columns, y++)
        {
            for (int x = 0, x < rows, x++)
            {
                pixelColor = bm.GetPixel(x, y),
                int grayScale = (int)((pixelColor.R * .3) +
                (pixelColor.G * .59) + (pixelColor.B * .11)),
                if (grayScale <= 85) { DS.Tables["B"].Rows.Add(x, y,
                1), }
                else if (grayScale >= 86 & grayScale <= 170) {
                DS.Tables["B"].Rows.Add(x, y, 2), }
                else if (grayScale >= 171 & grayScale <= 255) {
                DS.Tables["B"].Rows.Add(x, y, 3), }
            }
        }
    }
}
```

Appendix C: Selecting the appropriate pixels code

The following code is for selecting the appropriate pixels from the 128 color histogram using C#

```
private void Calculate_Peaks(string ColorName)
{
    DataView view = DS.Tables["Repetition"].DefaultView,
    view.Sort = ColorName + " DESC",
    DataTable sortedTable = view.ToTable(),

    foreach (DataRow viewRow in sortedTable.Rows)
    {
        int i = Convert.ToInt32(viewRow[0]),

        int NL2 = 0,
        int NL1 = 0,
        int Peak = 0,
        int NR1 = 0,
        int NR2 = 0,
        int D = 0,
        string Condition = "",

        if (i > 0) { NL1 =
Convert.ToInt32(DS.Tables["Repetition"].Rows[i - 1][ColorName]), }
        if (i < 127) { NR1 =
Convert.ToInt32(DS.Tables["Repetition"].Rows[i + 1][ColorName]), }
        if (i > 1) { NL2 =
Convert.ToInt32(DS.Tables["Repetition"].Rows[i - 2][ColorName]), }
        if (i < 126) { NR2 =
Convert.ToInt32(DS.Tables["Repetition"].Rows[i + 2][ColorName]), }
        Peak =
Convert.ToInt32(DS.Tables["Repetition"].Rows[i][ColorName]),

        if (CanAddPeak(ColorName, i))
        {
            AddPeak_And_Neighbors(ColorName, "P", i, Peak, Peak, 0, 0,
0, ""),
            if (Peak > NL1 + 2 & Peak > NL2 + 2 & NL1 != 0 & NL2 != 0 &
NL1 != NL2)
            {
                if (CanAddNeighbor(ColorName, i - 1, i - 2))
                {
                    if (NL1 > NL2)
                    {
                        D = NL1 - NL2,
                        Condition = "NL1 > NL2",
                    }
                    else
                    {
                        D = Peak - NL2,
```

```

        Condition = "NL1 < NL2",
    }
    AddPeak_And_Neighbors(ColorName, "NL1", i - 1,
Peak, NL1, D, NL1 - NL2, 0, Condition),
    AddPeak_And_Neighbors(ColorName, "NL2", i - 2,
Peak, NL2, D, NL1 - NL2, 0, Condition),
    }
}
if (NR1 != 0 & NR2 != 0 & NR1 != NR2)
{
    if (CanAddNeighbor(ColorName, i + 1, i + 2))
    {
        if (NR1 > NR2)
        {
            D = NR1 - NR2,
            Condition = "NR1 > NR2",
        }
        else
        {
            D = Peak - NR2,
            Condition = "NR1 < NR2",
        }
        AddPeak_And_Neighbors(ColorName, "NR1", i + 1,
Peak, NR1, D, 0, NR1 - NR2, Condition),
        AddPeak_And_Neighbors(ColorName, "NR2", i + 2,
Peak, NR2, D, 0, NR1 - NR2, Condition),
    }
}
}
}

private bool CanAddNeighbor(string ColorName, int N1, int N2)
{
    foreach (DataRow Row in DS.Tables[ColorName + "M"].Rows)
    {
        int NC = Convert.ToInt32(Row["C"]),
        if (NC == N1 || NC == N2) { return false, }
    }
    return true,
}

private bool CanAddPeak(string ColorName, int Peak)
{
    foreach (DataRow Row in DS.Tables[ColorName + "M"].Rows)
    {
        int NC = Convert.ToInt32(Row["C"]),
        if (NC == Peak) { return false, }
    }
    return true,
}

private void AddPeak_And_Neighbors(string ColorName, string Type, int
C, int Peak, int F, int D, int LN1_LN2, int RN1_RN2, string Condition)
{
    Boolean has = false,
    foreach (DataRow Row in DS.Tables[ColorName + "M"].Rows)
    {
        int NC = Convert.ToInt32(Row["C"]),

```

```

        if (NC == C)
        {
            has = true,
            break,
        }
    }
    if (has == false)
    {
        int half = D,
        string Odd_Even = "",
        if (C % 2 != 0) { Odd_Even = "Odd", } else { Odd_Even = "Even",
    }

    DS.Tables[ColorName + "M"].Rows.Add(C, F, Type, Peak - F,
LN1_LN2, RN1_RN2, Odd_Even, Condition, half),

    if (Type == "NL1" || Type == "NR1")
    {
        int count = 0,
        foreach (DataRow Row in DS.Tables["ShiftingImage"].Rows)
        {
            int RowG = Convert.ToInt32(Row[ColorName]),
            int x = Convert.ToInt32(Row["x"]),
            int y = Convert.ToInt32(Row["y"]),
            if (RowG == C)
            {
                count++,
                DS.Tables[ColorName + "N"].Rows.Add(D, RowG, Type,
x, y),

                if (count >= half) { break, }
            }
        }
    }
}
}
}
}

```

Appendix D: calculating MSE and PSNR

The following code calculates MSE and PSNR between cover and stego image using C#.

```
private void PSNR_MSE(string test)
{
    Bitmap NImage = new Bitmap(System.Drawing.Image.FromFile(test)),
    Bitmap OImage = new Bitmap(System.Drawing.Image.FromFile(ts)),
    if (OImage == NImage) { MessageBox.Show("Image"), }

    double PSNR = 0,
    double MSE = 0,
    double R = 0,
    double G = 0,
    double B = 0,

    for (int i = 0, i < OImage.Width , i++)
    {
        for (int j = 0, j < OImage.Height, j++)
        {

            Color O = OImage.GetPixel(i, j),
            Color N = NImage.GetPixel(i, j),

            Color O1 = OImage.GetPixel(i, j),
            int Ot = O1.R &
Convert.ToInt32("00000000000000000000000000000001", 2),

            Color N1 = NImage.GetPixel(i, j),
            int Nt = N1.R &
Convert.ToInt32("00000000000000000000000000000001", 2),

            int RO, RN,
            RO = OImage.GetPixel(i, j).R,
            RN = NImage.GetPixel(i, j).R,

            int GO, GN,
            GO = OImage.GetPixel(i, j).G,
            GN = NImage.GetPixel(i, j).G,

            int BO, BN,
            BO = OImage.GetPixel(i, j).B,
            BN = NImage.GetPixel(i, j).B,

            int RS = 0,
            int GS = 0,
            int BS = 0,
```

```
        RS = RO - RN,  
        RS = RS * RS,  
        R += RS,  
  
        GS = GO - GN,  
        GS = GS * GS,  
        G += GS,  
  
        BS = BO - BN,  
        BS = BS * BS,  
        B += BS,  
  
    }  
}  
  
double hw = (OImage.Width * OImage.Height),  
double sum = R + G + B,  
  
MSE = sum / hw,  
  
double S255 = 255 * 255,  
double S255_Mse = S255 / MSE,  
double Lo = Math.Log10(S255_Mse),  
  
PSNR = 10 * Lo,  
  
MSERlabel.Text = MSE.ToString(),  
PSNRRlabel.Text = PSNR.ToString(),  
}
```