ENHANCED DATA SECURITY ALGORITHM USING MODIFIED IMAGE RANDOMIZED LEAST SIGNIFICANT BIT AND EXCLUSIVE-OR ENCRYPTION

BY

OWOIDIGHEABASI MICHAEL UBON

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCE,

AHMADU BELLO UNIVERSITY,

ZARIA, NIGERIA

JANUARY, 2017

ENHANCED DATA SECURITY ALGORITHM USING MODIFIED IMAGE RANDOMIZED LEAST SIGNIFICANT BIT AND EXCLUSIVE-OR ENCRYPTION

BY

UBON, OWOIDIGHEABASI MICHAEL

P13SCMT8011

A DISSERTATION SUBMITTED TO SCHOOL OF POSTGRADUATE STUDIES

AHMADU BELLO UNIVERSITY, ZARIA IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD OF A MASTER DEGREE IN

COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCE,

AHMADU BELLO UNIVERSITY,

ZARIA, NIGERIA

JANUARY, 2017

DECLARATION

I hereby declare that the work in this dissertation entitled “Enhanced Data Security Algorithm using Modified Image Randomized Least Significant Bit and Exclusive-OR Encryption” has been carried out by me in the Department of Computer Science, under the supervision of Dr. A. A. Obiniyi and Dr. B. I. Ahmad

The information derived from the literatures has been duly acknowledged in the text and a list of references provided. No part of this work has been presented for another degree or diploma at any institution.

UBON, OWOIDIGHEABASI MICHAEL		
NAME OF STUDENT	SIGNATURE	DATE

CERTIFICATION

This dissertation entitled “An Enhanced Data Security Algorithm Using Modified Image Randomized Least Significant Bit and Exclusive-OR Encryption” meets the regulation governing the award of the degree of M. Sc. Computer Science of Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Dr. A. A. Obiniyi	Signature	______________________
Chairman, Supervisory Committee		
	Date	______________________
Dr. B. I. Ahmad	Signature	______________________
Member, Supervisory		
	Date	______________________
______________________	Signature	______________________
Member, Supervisory Committee		
	Date	______________________
______________________	Signature	______________________
Head of Department		
	Date	______________________
______________________	Signature	______________________
Dean, School of Postgraduate Studies		
	Date	______________________

DEDICATION

This dissertation is dedicated to my beloved parents Mr. Michael and Mrs. Theresa Ubon and my siblings.

ACKNOWLEDGEMENT

All praise, honour and adoration belong to God Almighty for His manifold blessings, wisdom, sustenance and protection upon my life especially during my Master's pursuit.

Special gratitude goes to my supervisors Dr. A. A. Obiniyi and Dr. B. I. Ahmad for their time and invaluable contributions toward the successful completion of this work. Worthy of my appreciation are all the instructors – Prof. S. B. Junaidu, Prof. O. Adewale, Dr. S. E. Abdulahi, who took me through the course work and prepared me for this dissertation. May God bless you all with more knowledge, strength and longevity.

My special thanks also go to my parents Mr. Michael and Mrs. Theresa Ubon for their love, prayers, financial and moral support. To my siblings Glory, NseAbasi, Ndifreke, Aniekan, Idongesit and Uduak and my brother – in- law Engr. Ye-Obong Udoka you all made the task easier. May God bless and keep you safe to enjoy from your labour. I love you all.

I use this rare opportunity to appreciate my colleagues of P13 class for their contributions in one way or the other to my study especially Fatima Atta you inspires me greatly, Gideon, Mark Eche, Ibrahim etc meeting and knowing all of you have brought invaluable blessings to my life.

To my Church members (Church of Christ, Hayin Dogo Samaru) I say thank you for adding value to my spiritual life and your constant prayers towards my success. Indeed education without God is worthless. I would not forget to mention my beloved friend EtieneAbasi Udofot who has been there through thick and thin to encourage, support and pray for my success. Thank you dear.

ABSTRACT

Security of data over unsecured channel is a global issue. Image steganography and Exclusive-OR (XOR) encryption are techniques employed to ensure data security. This dissertation develops and implements an algorithm that further improves the traditional image Least Significant Bit algorithm. Encryption of secret message was performed using 64 bits key and plaintext to obtain a cipher text. Each bits of the cipher text were hidden randomly using specific condition in red and blue planes of a selected cover 24 bits colour image. Extraction of the embedded cipher text was done as well as decryption of cipher text back to plain text. Experimental results showed that the proposed algorithm hides significant amount of data and extract it successfully without any loss at a very good Peak Signal to Noise Ratio (PNSR) of above 60dB. The statistical analysis of cover and stego image showed no much difference which further proved the proposed technique is more secured and efficient for private communication. Steganalytic result showed that the algorithm survives steganalysis test. Based on these results, the algorithm out performs other algorithm considered in the study.

TABLE OF CONTENTS

COVER PAGE... i
FLY LEAF ii
TITLE PAGE... iii
DECLARATION.. iii
CERTIFICATION iv
DEDICATION.. v
ACKNOWLEDGEMENT... vi
ABSTRACT.. vii
TABLE OF CONTENTS viii
LIST OF FIGURES xi
CHAPTER ONE 1
INTRODUCTION... 1
1.1 Background of the Study 1
1.2 Problem Statement... 2
1.3 Motivation of the Study 3
1.4 Aim and Objectives of the Study 3
1.5 Research Methodology.. 4
1.6 Scope of the Study 4
1.7 Dissertation Organization 4
CHAPTER TWO 6
LITERATURE REVIEW 6
2.1 Information security 6
2.1.1 Security attacks 7
2.2 Digital watermarking.. 7
2.3 Cryptography 7

2.3.1 Types of encryption 8

2.3.2 Some encryption techniques 9

2.4 Steganography 10

2.4.1 Types of steganography 11

2.5 Image Steganography 12

2.5.1 Digital Image 13

Digital Image Compression 14

2.5.2 Requirements for Image Steganography 15

2.6 Image Steganographic Techniques 17

2.6.1 Transform Domain Technique 17

2.6.2 Spatial domain technique 19

2.7 Image Least Significant Bit Method 19

2.7.1 Advantages and Limitations of image LSB method 21

2.8 Image Steganalysis 22

2.8.1 Steganalytic method 23

2.8.2 Steganalysis tools 24

2.9 Comparison of Steganography and Cryptography 24

2.10 Related works 25

2.11 Literature gap 35

CHAPTER THREE 36

RANDOM BYTE POSITION IMAGE LSB CRYPTO-STEGANOGRAPIC ALGORITHMS 36

3.1 The Proposed Framework 36

3.2 XOR Encryption and Decryption process 38

3.2.1 Encryption Algorithm 38

3.2.2 Decryption Algorithm 39

3.3.1 Image LSB technique algorithm 40

3.3.2 Proposed embedding algorithm 41

3.3.3 Proposed extraction algorithm 43

CHAPTER FOUR 45

RESULT ANALYSIS AND EVALUATION 45

4.1 Implementation details 45

4.1.1 Java Programming Language 45

4.1.2 StegExpose 45

4.1.3 Data 46

4.2 System Specification 48

4.3 Implementation 48

4.4 Performance Metrics 50

4.4.1 Payload capacity 50

4.4.2 Imperceptibility test using PSNR 51

4.4.3 Additional security 54

4.4.4 Statistical Analysis using Histogram 55

4.4.5 Steganalytic Test 58

CHAPTER 5 60

SUMMARY, CONCLUSION AND RECOMMENDATION 60

5.1 Summary 60

5.2 Conclusion 60

5.3 Contribution to knowledge 61

5.4 Recommendation 61

REFERENCE 62

LIST OF FIGURES

Figure 2.1: Traditional Cryptography System 8
Figure 2.2: Image Steganographic System) 12
Figure 2.3: RGB color space 13
Figure 2.4 : Pixels in image 14
Figure 2.5: Accessing bits of colour RGB image 20
Figure 2.6: Embedding Flowchart 32
figure 2.7: Extraction Flowchart 33
Figure 3.1: Proposed Architecture 37
Figure 3.2: Encryption Flowchart 40
Figure 4.1: Clover.bmp 46
Figure 4.2: Tulip.bmp 46
Figure 4.3: Flower.bmp 47
Figure 4.4: Lena.png 47
Figure 4.5: Peppers.png 47
Figure 4.6: Colors.png 48
Figure 4.7: System GUI 49
Figure 4.8: Graph of Payload Capacity 51
Figure 4.9: Graph of PSNR against 6 different images 54
Figure 4.10: Histogram of Clover.png 55
Figure 4.11: Histogram of tulip.bmp 56
Figure 4.12: Histogram of flower.bmp 57
Figure 4.13: Histogram of Lena.bmp 57
Figure 4.14: Histogram of Peppers.png 57
Figure 4.15: Histogram of Colors.png 58

LIST OF TABLES

Table 2.1: The Differences between Cryptography and Steganography 25
Table 4.1: Amount of payload capacity hidden in stego images………………………………...50
Table 4.2: PSNR and MSE comparison………………………..…………………………..…52
Table 4.3: Additional Security Support ………………………………………………………53
Table 4.4: Steganalytic Report for Sample Images …………………………………………..59

CHAPTER ONE: INTRODUCTION

1.1 Background of the Study

Humans have continually sought for new and efficient means of communicating sensitive information through the web and other communication channels. Seberry and Pierprzyk (1989) explained that “Security of information results from the need for private transmission of both military and diplomatic messages. This need is as old as civilization itself”. Advancements in technology introduced written language, telegraph, radio/television, and most recently electronic mail, telephone, social media chat, and communicating privately through these media is not without security challenges.

Data transmission through Internet has been the fastest, simple and most used means of communication. However, the use of Internet poses serious threat on the security of such data. Therefore it becomes imperative to consider data security as it is one of the most important factor during data transmission process.

Data security basically means the protection of data from unauthorized users or hackers and provision of high security against data modification. This area of study is growing in astronomical rate due to the growing number of Internet users globally. Data security has five main requirements to satisfy: confidentiality, data integrity, non repudiation, availability and authentication (Stallings, 2011).

Cryptography and steganography are two powerful techniques employed in data security. Steganography is the art and science of communicating in a way which hides the existence of the communication (Narayana and Prasad, 2010). The idea of modern steganography was first described in Simmons (1983) while describing communication between two prisoners. Cryptography is the study of encryption principles/methods. The process of converting from plaintext to ciphertext is known as enciphering or encryption; restoring the plaintext from the ciphertext is deciphering or decryption (Stallings, 2011).

The general objective of encryption is not to protect communications as such but rather the content (Becket, 1988). This is due to it being extremely difficult to stop communication interceptions. By using encryption, even if there is a transmission interception, the content of the communication is kept secret. This is extremely important in communication channels such as the Internet where many networks are connected together using broadcast mediums (such as Ethernet) and so data interception is real and active problem. Cryptographic techniques could be vulnerable due to improvement in the computers' capabilities in terms of speed of processing. However, this vulnerability can be significantly curbed by combining it with steganography. Steganography can be thought of as a higher version of cryptography and hence its usage can further increase the integrity and security of data. Traditionally in steganography, the goal is to hide a secret message, which is transmitted between two or more communication partners in a way to ensure confidentiality and integrity of data.

Successful embedding depends on the selection of the cover image and the method involved. It is also important to note that steganographic technique not only involves in embedding information inside digital media but also the ability to successfully retrieve the information from the media.

1.2 Problem Statement

There are many conventional approaches to image least significant bit (LSB) steganography algorithms to guarantee secure data communication constraints but none is absolutely ideal. Each steganography algorithm has its own strength and weakness in terms of security and complexity. Some algorithm cannot retrieve hidden bits without loss of some part of it. The work therefore seeks to modify existing LSB algorithm and combine cryptography for the security of embedded data while maintaining high imperceptibility of the cover media with no loss of data during extraction process.

1.3 Motivation of the Study

More Internet users are using photos and videos as a social currency, about 54% of internet users have posted original photos or videos to websites and 47% share photos or videos they found elsewhere online (Duggan, 2013). Young adults and women lead the way in each of these activities. Cell phones and smartphones have given rise to photos and video sharing applications. Therefore, harnessing the influx of multimedia on the Internet for covert communications has become a new area of research to proffer solution to transmitted or stored sensitive information. Image steganography technique is the most suited technique since image files contains high rate of redundancy. Hence, the study of combining cryptography with image randomization steganographic technique to enhanced data security of sensitive information.

1.4 Aim and Objectives of the Study

The aim of this dissertation is to develop an improved randomized image LSB steganographic and encryption system to guarantee confidentiality and security of data sent across the Internet. The objectives of the study are to:

a. develop a steganographic algorithm by modifying the existing steganographic algorithm (LSB) which is achieved through randomizing the bit and byte positions of red and blue pixel of a coloured image
b. implement the algorithm in (a)

c. analyze and compare the result with the work of Singh *et al.,* (2014) in terms of 3 metrics – payload, imperceptibility and security.

1.5 Research Methodology

Least Significant Bit (LSB) technique is a method used in image steganography to address the issue of confidentiality of data sent across unsecure channels. The steps used are:

a. Survey of encryption and steganography concept and techniques.
b. Encryption of secret message using XOR encryption algorithm to obtain a cipher text.
c. Hiding the cipher text randomly in red and blue component of a coloured image using proposed algorithm.
d. Extraction of the hidden cipher text
e. Decryption of the cipher text to obtain the plain text *i.e* the secret message
f. The Prototype algorithm was implemented using Java programming language.
g. Evaluation and comparison of the payload, imperceptibility and security of data with Singh *et al.,* (2014)

1.6 Scope of the Study

This study focused only on spatial domain. It is therefore limited and susceptible to cropping, rotation, low bit reverse attack due to LSB weakness and compression techniques.

1.7 Dissertation Organization

The organization of the rest of this dissertation and a brief outline of the chapters are as follow.

Chapter 2 discusses the concept of information security, tools used for information security which includes watermarking, cryptography and steganography. The chapter also discussed related works in the area of image steganographic techniques. In Chapter 3, the proposed algorithm is presented. Chapter 4 involves the implementation of the algorithm proposed in chapter 3 as well as evaluation of the results obtained. The conclusion, summary and recommendation are captured in chapter 5.

CHAPTER TWO

LITERATURE REVIEW

2.1 Information security

The idea of communication is only for the intended receiver to gain access to information without the interference of a third party. In the information super high way “Internet” this aim is always defeated, likewise in personal computers without adequate security. Cryptography and steganography become the two major techniques to overcome these hurdles. In general, security denotes the state of being secure or free from danger (Stallings, 2011). Computer security is the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (such as hardware, software, firmware, information/data, and telecommunications) (NIST, 1995). Security is divided into different layers according to the type of content intended to be secured as summarized by Deepesh and Bhandari (2013) in a to e:

a. Physical security: it defines the required issues needed to protect physical data or object from unauthorized intrusion.
b. Personal security is the security of the individuals who are officially authorized to access the information about a company and its operations.
c. Operational Security: it is mainly concern with the protection of information about a particular operations or chain of activities.
d. Communication security: it encompasses the security issues regarding the organization’s communication media, technology and content.

e. Network security: it is concerned with safeguarding the information regarding the network components, connections and contents.

Information security is defined as the measures to safeguard attack on stored and transmitted data (Stallings, 2011).

2.1.1 Security attacks

Normal flow of data is from the source to authorized destination. However, a hacker or man in the middle may intrude and interfere with this flow, access and modify the information. This process is refers to security attack. It can be classified into passive – accessing without modification of information and active attack. To curb security attack on data, approaches such as cryptography, steganography and digital watermarking are adopted.

2.2 Digital watermarking

Watermarking is a data hiding technique that protects digital documents, files, or images against removal of copyright information (Almohammed, 2010). Watermarking aims at protecting the rights of the owners of digital media such as images, music, video and software. Even if people copy or make minor modification to the watermarked file, the owner can still prove it is his or her file. Here the major requirement of the system is robustness. The robustness feature describes the ability of watermarking to be impossible to remove in case of an attack. Watermarking is a branch of data security that is classified into information hiding method like steganography.

2.3 Cryptography

Cryptography is the study of encryption principles and methods. Encryption is a mathematical scheme that employs algorithms to actual data and transforms it into unreadable data or text. The process of converting information (plain text) by transforming it into unreadable format (ciphertext) is known as encryption (Philjon and Rao, 2011). The reverse process of obtaining the plaintext from a cipher text is known as decryption.

Cryptography, concerns itself with the projection of trust i.e. with taking trust from where it exists to where it is needed. A strong cryptography might for example, have the potential to make the existing information system more secure. Many cryptographic algorithms are available for securing information. However, such security is dependent on secrecy of the secret key and the key entropy because the larger the size of key, the stronger will be the security. Also, large key size is difficult for the user to remember.

There are many encryption algorithms in use today. The algorithm can be classified based on the number of keys used for encryption and decryption processes. Figure 2.1 explains the traditional process of cryptography. It shows that cryptography is generally of two phases namely the encryption and the decryption phase.

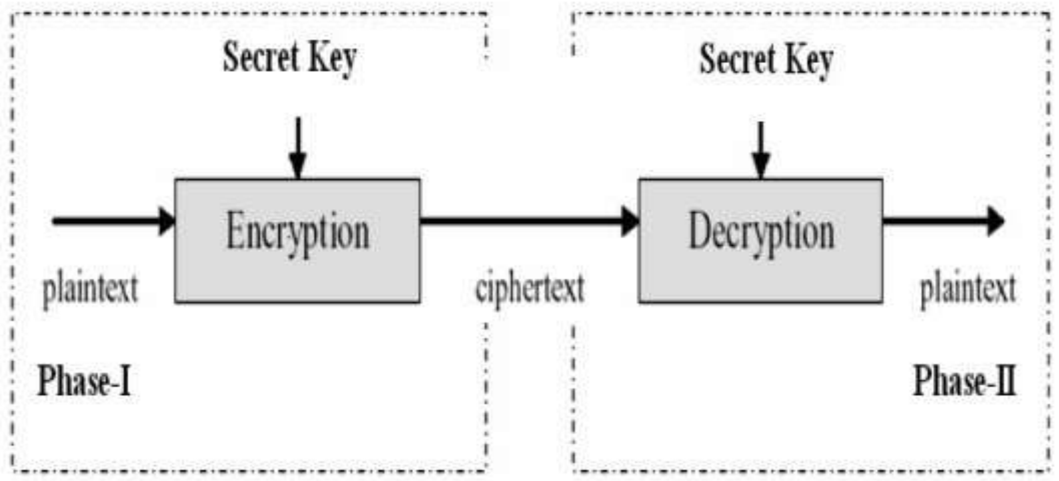


Figure 2.1: Traditional Cryptography System (Dagaar, 2014)

2.3.1 Types of encryption

a. Symmetric encryption: Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption (Stallings, 2011). The sender uses the same key to encrypt the message and at the receiver’s end the receiver uses the same key to decrypt the cipher to get original message. It is the most widely used type for communication between two persons.

b. Asymmetric encryption: Asymmetric Encryption is also called Public-key cryptography. It is a cryptography in which a pair of keys is used to encrypt and decrypt a message so that it is transmitted to the recipient securely. Encryption and decryption are carried out using two different keys. The two key pair is referred to as public key and private key. Predominantly, public key is used for encryption of a message and private key is used to decrypt it.

2.3.2 Some encryption techniques

Many encryption techniques are available for securing data today. Some of the widely used techniques include:

a. Data Encryption Standard (DES): It was designed by IBM and published first in 1975, adopted by National Institute for Standards and Technology (NIST) for commercial and unclassified government applications. It is the most commonly used symmetric cryptography. Triple-DES (3DES) and DESX are the two variant of DES (Venkatraman and Paprzycki, 2004). In the 1990s it became clear that the DES algorithm was no longer secure enough, as attacks using the methods of differential and linear cryptanalysis were used to successfully break DES keys. So it was been deprecated and replaced by the Advanced Encryption Standard (AES).

b. Advanced Encryption Standard (AES): it is also known as Rijndael. NIST selected the Rijndael algorithm for AES because it offers a combination of Security, performance, efficiency, ease of implementation, and flexibility. AES has become one of the popular encryption algorithms that use symmetric keys for encryption and decryption process (Stallings, 2011). Rijndael is similar to DES in that there is a series of rounds which twist a plaintext block into a cipher text block, and a key expansion algorithm that takes the key and massages it into a bunch of round keys.

c. Transposition cipher: the plaintext letters are shifted to form a cryptogram. This can be done in many ways. In some systems whole words are transposed, rather than individual letters. Single column, double column, Grills are the most types of transposition cipher.

d. XOR Encryption: Exclusive – OR encryption is not a public key system and it is almost unbreakable through brute force methods. It requires that both sender and receiver have access to the encryption key. It is simple to implement but nearly unbreakable. XOR encryption works by using the Boolean algebra function exclusive-OR (XOR). The idea behind this encryption is that it is impossible to reverse the operation without knowing the initial value of one of the two arguments. When a key is random and is at least as long as the message, the XOR cipher is much more secure than when there is key repetition within a message (Churchhouse, 2002). Example if the key is A and the message to be encrypted is “I am home now”. The key and message are converted to their ASCII equivalent and binary format before encryption using XOR Boolean function.

2.4 Steganography

Nivedhitha and Meyyappan (2012), defines Steganography as the art and science of communicating in a way which hides the existence of the communication. The steganography word is classified into two parts: Steganos means “secret or covered” (where you want to hide the secret messages) and graphic means “writing” (Text) combining the two work steganography means cover writing (Amin *et al.,* 2003). In modern times steganographic technologies have been an important part of the future of security and privacy on open systems such as internet (Singh *et al.*, 2014).

2.4.1 Types of steganography

Steganography is basically classified according to the container used in hiding information. All forms of digital files which contain some degree of redundancy are said to be useful for steganographic purposes however the following are widely used:

a. Image: Image steganography is the process in which data is hidden within an image so that there is no any perceivable change in the original image. The conventional image steganography algorithm is LSB embedding algorithm (Suri *et al.*, 2014).

b. Audio: The method of hiding secret information in an audio is known as audio steganography. Sound has characters such as frequency, phase angle and speech cadence. Slight changes in these values are not recognizable by human ears. It is an advantage for hiding message in these values. There are various methods for hiding secret data in an audio such as LSB, Phase Coding etc.

c. Video: The method of hiding secret information in a video is known as video steganography. Video consists of images as well as audio. Hence, both images and audio steganography can be used for video steganography.

d. Text file: The method of hiding secret information in a text is known as text steganography. Text steganography requires less memory as it can only store text files. It provides quick communication or transfer of files from one computer to another. Text steganography is not commonly used as text files does not contain large amount of redundant data.

e. Network protocol: The term network protocol steganography refers to the technique of embedding information within messages and network control protocols used in network transmission. There are covert channels in the layers of the Open System Interconnect (OSI) network model where steganography can be used. Here the header of IPv4 is used to covertly transmit data (Shukla *et al*., 2014).

However, this dissertation will only be concerned with image steganography.

2.5 Image Steganography

Image steganography uses image as the carrier for another data which could be image or text. The image steganography scenario can be described diagrammatically as shown in figure 2.2.

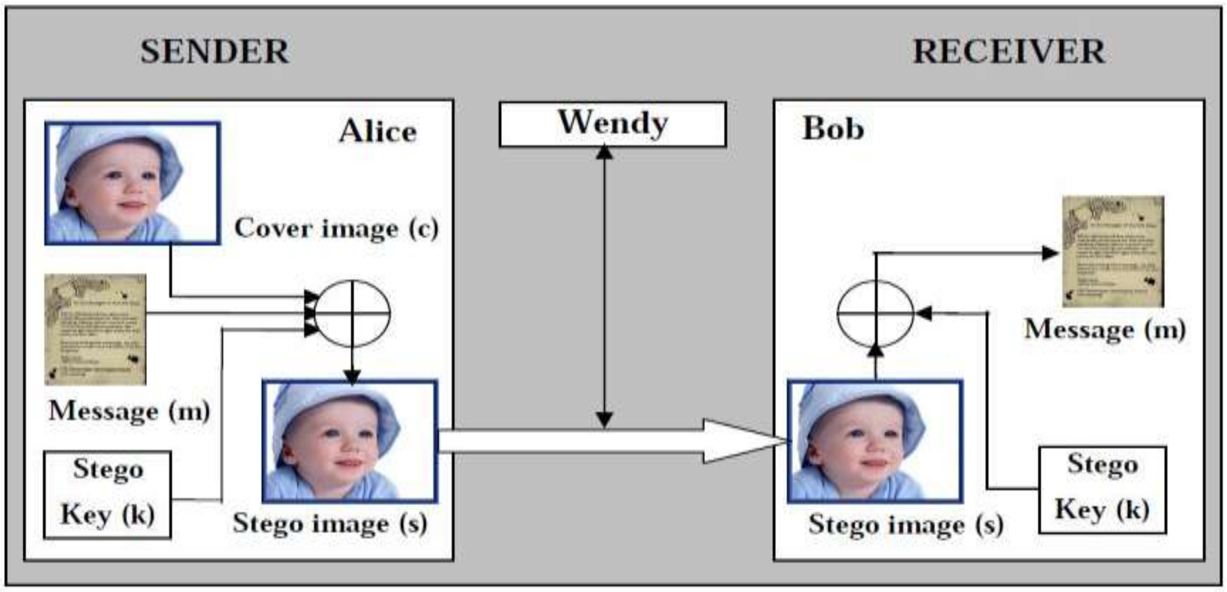


Figure 2.2: Image Steganographic System (Attalah, 2012)

In image steganography for hiding secret message, the physical location of a pixel is considered and then the binary format of that pixels value is used to hide the secret message. Since image is the carrier object, understanding the concept of image is very important.

2.5.1 Digital Image

An image file is a binary file containing a binary representation of the color or light intensity of each picture element (pixel) comprising the image. Digital images can be expressed as a collection of pixel values at each pixel position (Gonzalez and Woods, 2002). It is divided into three types: binary (Black- White), Gray scale and Red-Green-Blue (RGB) images (Atallah, 2012). The binary image has one bit value per pixel represented by 0 for black and 1 for white pixels. Gray scale image has 8 bits value per pixel represent by 00000000 for black and 11111111 for white pixels.

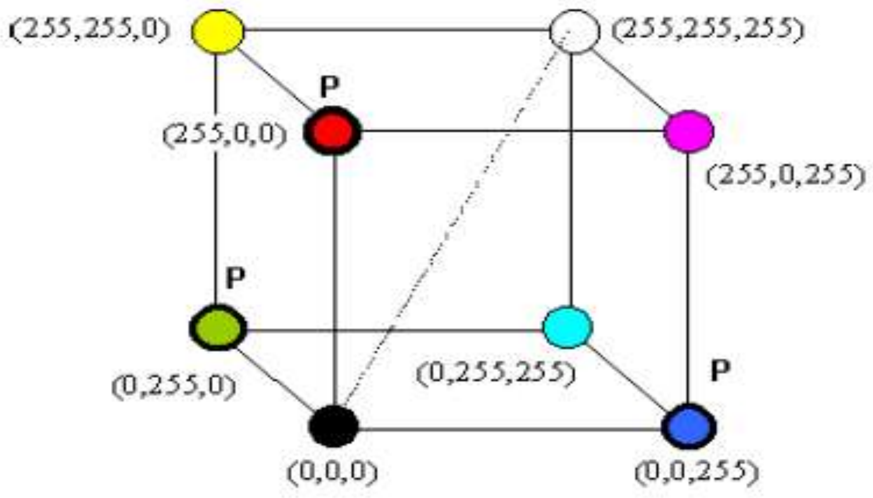


Figure 2.3: RGB color space (Laskar and Hemachandran, 2013)

The RGB image has 24 bits value per pixel represented by (00000000, 00000000 and 00000000) for black and (11111111, 1111111 and 11111111) for white pixels. 24 bit color image is best define by RGB color model in which each color appears in its primary spectral component of red, green and blue as shown on figure 2.3. The RGB image is the most suitable form of image for steganography because it contains a lot of information that helps in hiding the secret information with a bit change in the image resolution which does not affect its quality and make the message more secure.

Hypertext Markup Language (HTML) format for indicating colors in a web page often use a 24-bit format employing six hexadecimal digits, each pair representing the amount of red, blue, and green, respectively (Anshit, 2014). The color orange, for example, would be displayed with red set to 100% (decimal 255, hex FF), green set to 50% (decimal 127, hex 7F), and no blue (0), so the number "#FF7F00" would indicate Orange color in the HTML code.

The size of an image file is directly related to the number of pixels and the granularity of the color definition. A typical 640 X 480 pixel image using a palette of 256 colors would require a file of about 307 KB in size (640 X 480 bytes), whereas a 1024 X 768 pixel high-resolution 24-bit color image would result in a 2.36 MB file (1024 X 768 X 3 bytes) (Davison, 2009). The computer’s memory visualizes an image as making up of tiny dots (pixels) as illustrated in figure 2.4.

Figure 2.4: Pixels in image

Digital Image Compression

Image files can be quite large, and larger file types mean more disk usage and slower download. Compression is a term used to describe ways of cutting the size of the file. To avoid sending files of enormous size, a number of compression schemes have been developed over time for Bitmap Image (BMP), Graphic Interchange Format (GIF) and Joint Photographic Experts Group (JPEG) file types. A lossless algorithm will look for a recurring pattern in the file, and replace it with a short abbreviation. By doing this file size is reduced. In contrast, a lossy algorithm might store color information at a lower resolution than the image itself, since the eye is not so sensitive to changes in color of a small distance (Bakar, 2011). From steganographic point of view, all the file types mentioned above are not equally suited.

GIF and 8-bit BMP files employ *lossless* compression, this scheme allows the software to exactly reconstruct the original image. JPEG, on the other hand, uses lossy compression, which means that the expanded image is very nearly the same as the original but not an exact duplicate. While both methods allow computers to save storage space, lossless compression is much better suited to applications where the integrity of the original information must be maintained, such as steganography.

2.5.2 Requirements for Image Steganography

Various techniques for image steganography were proposed for different purposes therefore the need to measure their efficiency. The efficiency depends on several standards which is called requirements. There is set of certain and essential requirements to be fulfilled in most of algorithms. These are transparency, capacity, robustness and complexity of algorithm (Deepesh and Bhandari, 2013). The hiding capacity is the most significant characteristic in the steganography followed by the transparency and security while robustness has more important role in watermarking. These are related to each other. For instance, increasing capacity will diminish the transparency and vice versa. A good steganographic system has trade-off between these requirements and which is hard to achieve in one algorithm (Johnson and Jajodia, 1999).

a. Transparency: Most of the data hiding techniques have to insert data as much as possible without affecting the perceptual degradation in quality of the host file. It is one of the most important factors in designing algorithms for hiding data. The fidelity of the steganography algorithm is usually known as a perceptual resemblance between the cover file and stego-cover. However, the differences should be with minimal levels. The evaluation of imperceptibility is usually based on an objective measure of quality or subjective test (Amin *et al.*, 2003). Some image steganography techniques can be categorized as methods that have high transparency

b. Payload capacity: The amount of information that information hiding scheme can successfully hide without introducing any perceptual distortion is the capacity. It represents the number of hidden bits according to the size of host cover. The difficulty lies in how to embed secret data as much as possible while preserving the quality of the host cover. It is measured in bits per pixel for images steganography and bits per second for audio steganography.

c. Robustness and security: Robustness is defined as how hidden message should not be prone to elimination or modification while security prevents unauthorized person from extracting the hidden data. There are two kind of attacks that may have effect on the stego-cover: unintentional attack that try to modify or destroy the stego-cover (such as compression, rotation, blurring, noising and other filtering techniques) and intentional attack that try to reveal the stego-cover and extract the hidden information (Al-Othmani *et al.*, 2012). Usually there is a trade-off between robustness and capacity that can hardly be fulfilled together in the same steganographic system. The robustness is an important factor for copyright protection and watermarking applications, while imperceptibility and high hiding capacity is more significant for steganography applications because the goal is to hide as large amount of data while preserving the quality of the cover file.

d. Complexity: Hiding algorithms consumes time in performing embed/extract process and this depends on the complexities of the algorithm. Secret data should be rapidly embedded/extracted from the host file, so that streaming data hiding real time can be delivered over the network (Delforouzi and Pooyan, 2009).

2.6 Image Steganographic Techniques

Image steganography is a technique which is used to hide secret message within an image. The binary bits of secret of message are hidden in the binary of image and this slightly affects the intensities of colour or brightness which is not detectable by naked human eyes. The image steganography algorithm is classified into two broad categories namely – transform and spatial domain techniques.

2.6.1 Transform Domain Technique

In the transform domain, the cover image first undergoes a transformation into its frequency domain and then its transformed coefficients are altered to embed the secret information. It first decorrelate the image by scrambling the pixels randomly, which in effect whitens the frequency domain of the image and increases the number of transform coefficients in the frequency domain thus increasing the embedding capacity. Transform domain techniques have an advantage over spatial domain techniques as they hide information in areas of the image that are less exposed to compression, cropping, and image processing however the approach hides small amount of data compared to the latter (Kaur and Kumar, 2011). The approaches are broadly classified into:

a. Discrete Cosine Transform

Discrete Cosine Transform (DCT) is used extensively with video and image compression e.g. JPEG lossy compression. For each color component, the JPEG image format uses a discrete cosine transform (DCT) to transform successive 8.8 pixel blocks of the image into 64 DCT coefficients each (Walia *et al.*, 2010).

b. Discreet Wavelength Transform

The wavelet transform is a transformation to basic functions that are localized in frequency. The wavelet compression methods are better at representing transients, such as an image of stars on a night sky. This means that elements of some data signal that are transient can be represented by a smaller amount of information than would be the case if some other transform, such as the more widespread discrete cosine transform, had been used. Wavelet compressions are good for transient signal characteristics but not for smooth, periodic signals. The steps are to take the DCT or wavelet transform of the cover image and find the coefficients below a specific threshold. Replace these bits with bits to be hidden (for example, use LSB insertion) and then take the inverse transform and store it as a regular image.

c. Discreet Fourier Transform Technique (DFT)

The Fourier Transform decomposes an image into its sine and cosine components when processing (Kaushal and Chaudhary, 2013). The output of the processed image is in Fourier domain while the input is in spatial domain. Each point in Fourier domain Image represents a particular frequency contained in the spatial domain image. This method is useful in many areas such image analysis, image reconstruction and image filtering.

2.6.2 Spatial domain technique

Spatial domain approach involves direct manipulation of the cover image pixels to hide bits of the secret data. These technique are classified into

a. Image Least Significant Bit method (LSB)
b. Pixel value differencing (PVD)
c. Edges based data embedding method (EBE)
d. Random pixel embedding method (RPE)
e. Mapping pixel to hidden data method

Among these techniques, Image Least Significant Bit (LSB) method will be used in the dissertation due to its simplicity in implementation and the ability to extract hidden bits without any loss depending on the algorithm used.

2.7 Image Least Significant Bit Method

A steganography technique based on modifying the least significant bit layer of images is known as the LSB technique. In the LSB technique, the least significant bits of the pixels is replaced by the message and bits are permuted before embedding. In some cases LSB of pixels visited in random or in certain areas of image and sometimes increment or decrement the pixel value.

Digital images are more attractive for steganographic purpose since it contains a significant amount of data and can be modified slightly without leading to visible artifacts (Eggers *et al.*, 2002). A simple image steganographic model contains an original image, called cover image (I) in which secret component secret message/image (M) is hidden and a stego key (K) which is used to hide the information as well as to extract. In this method, one can take the binary representation of the hidden data and overwrite the LSB of each byte within the cover image. In

24-bit color, the amount of change will be minimal and indiscernible to the human eye. Figure 2.5 illustrates how RGB color image are accessed in image steganography.

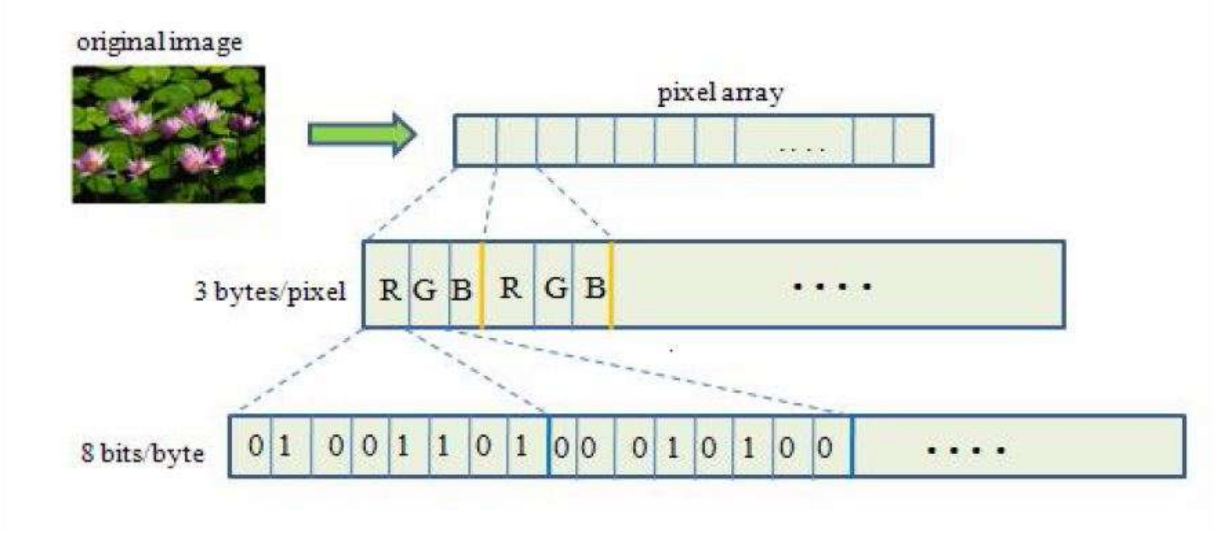


Figure 2.5: Accessing bits of colour RGB image (Davison, 2009)

As an example, suppose that there are three adjacent pixels (nine bytes) with the following RGB encoding:

11110101 11001100 10101001

10100110 11001110 11001010

10101111 00010011 11001000

Supposed we want hide letter B of which ASCII is 66 and binary equivalent is 1000010 in the above pixels using LSB the result will be

1111010**1** 1100110**0** 1010100**0**

1010011**0** 1100111**0** 1100101**1**

1010111**0** 00010011 11001000

The result shows 7 pixels are used with 3 changes.

2.7.1 Advantages and Limitations of image LSB method

Suri *et al.,* (2014) identified the following advantages of LSB technique:

a. There is less chance for degradation of the original image. If message bit is same as the pixel's least significant bit then no change at all is required for that pixel value
b. More information can be stored in an image at minimal degradation. If pixel value is different from message bit then effective change in pixel value is either +1 or –1. The +1 or –1 change in pixel value is invisible to human eye.

Some limitations of this method are discussed below:

a. The message can be easily removed by unauthorized person (intruder) as message is in least significant bit.
b. Intruder can modify the least significant bits of all the image pixels since messages are hidden there. In this way the hidden message can be destroyed, but the change in image quality is in the range of +1 to –1 at each pixel position, which is negligible to human eye.
c. Due to imaging system the least significant bit is easily corrupted by hardware imperfections or quantization noise so message can be distorted in this way also. (Chander Kant *et al.,* 2008)

To overcome disadvantages a and b, the message is encrypted and distributed at random locations which is the focused point of this dissertation. Whereas, c is overcome using transform domain technique which is beyond the scope of this study.

Application of Image Steganography

Some applications areas of image steganography are:

a. Key management: Key management is a major problem in cryptography, with steganography key can be sent secretly between communicating parties.
b. Intelligence agencies: Image steganographic technique is used for communication between staff.
c. Medical imagery: it is considered necessary for confidentiality between patients' image data or DNA sequences and their captions, e.g., physician, patient's name, address and other particulars (Cheddad, 2010). Patients x-ray is sent to patient covertly by hiding within another image. Thus, embedding the patient's information in the image could be a useful safety measure and helps in solving such problems.
d. E-commerce: It is used in preventing e-document forgery. For instance, a customer financial transaction details can be hidden from an attacker (Deepesh and Bhandari, 2013).

2.8 Image Steganalysis

Steganalysis is the breaking of steganography and it is the science of detecting hidden information (Amin *et al.,* 2003). The main objective of steganalysis is to break steganography and the detection of stego image. In the process of steganalysis, the cover file is tested for its file properties and verified for any pattern in the file content. Almost all steganalysis algorithms depend on steganographic algorithms introducing statistical differences between cover and stego image.

Steganalysis are of three different types:

a. Visual attacks: it discovered the hidden information, which helps to separate the image into bit planes for further analysis.

b. Statistical attacks: statistical attacks may be passive or active. Passive attacks involve identifying presence or absence of a secret message or embedding algorithm used. Active attack is used to investigate embedded message length or hidden message location or secret key used in embedding.
c. Structural attacks: The format of the data files changes as the data to be hidden is embedded, identifying this characteristic structure changes can help in finding the presence of image/text file.

2.8.1 Steganalytic method

Different methods are employed by steganalyst to detect the presence of hidden message within an image. The steps that ensure stego image survives steganalysis are measured through:

a. Visual inspection: here the steganalyst look for suspicious artifacts using simple visual inspection after preprocessing the stego-image. If, at the same time, the message is embedded sequentially, one can have a convincing argument for the presence of steganographic messages in an image (Fridrich and Goljan, 2012). An algorithm that is randomized with insignificant degradation on cover image makes it harder to detect any changes between the stego and cover image.
b. Statistical analysis of pair of values (Histogram analysis): Histogram measures the variance of cover and stego image. A test for the statistical significance of the fact that the occurrences of both values in each pair (stego and cover image pair) are the same is carried out. In a case where the stego technique embeds message bits sequentially into subsequent pixels/indices/coefficients starting in the upper left corner, one will observe an abrupt change in the statistical evidence at the end of message. A steganographic algorithm that preserves the carrier file is difficult to detect the presence of image.

2.8.2 Steganalysis tools

Steganalysis tools are developed to detect the presence of hidden messages within digital files. It is widely deployed by forensic analyst. Just as steganographic algorithms are gaining ground, many steganalysis tools too are being developed. Some of the tools determine the length of the message, steganographic algorithm used or the key. Some available steganographic tools are:

a. Virtual Steganographic Laboratory (VSL): it is a graphical block diagramming tool that allows complex using, testing and adjusting of methods both for image steganography and steganalysis (Fridich, 2012). VSL provides simple GUI along with modular, plug-in architecture. It is written and maintained by Michał Węgrzyn .
b. StegExpose: software designed for use in the real world to analyze images especially those using LSB steganographic approach (Boehm, 2014). It is capable of analyzing bulk of images in a time efficient manner.

2.9 Comparison of Steganography and Cryptography

Steganography and cryptography both have the same aim of providing security to information but using different techniques. Table 2.1 summarizes the differences between the two.

Table 2.1: The Differences between Cryptography and Steganography (Rahmani *et al.*, 2014)

Criterion/Technique	**Steganography**	**Cryptography**
Definition	Steganography means cover writing	Cryptography means secret writing
Objectives	Focuses on keeping existence of a message secret	Focuses on keeping the contents of a message secret
Key	Optional	Necessary
Carrier	Any digital media	Usually text based
Visibility	Never	Always
Security services offered	Confidentiality, authentication	Confidentiality, availability, data integrity, non – repudiation
Attacks	It is broken when attacker detects that steganography has been used known as Steganalysis	It is broken when attacker can read the secret message known as Cryptanalysis
Result	Stego file	Cipher text

2.10 Related works

Marvel and Retter (2000) introduced a new method of image steganography. The method embeds hidden message within White Gaussian Noise (WGN) which is subsequently added to the digital image to form a stego-image. The hidden message is encoded by an error control code before it is embedded into the WGN signal. The WGN signal with the embedded data, the stego-signal, is then added to the image. At the receiver's end, the embedded stego-signal is estimated as the difference between the stego-image and the denoised version of the stego-image. The embedded message is extracted from the estimated stego-signal and any remaining errors are corrected by the error-control decoder. Unfortunately, the estimate of the stego-signal is typically poor because the power of the signal is low compared to the image power, and thus denoising process is not optimal. Consequently, decoding errors are made. In an effort to address this shortcoming without increasing the stego-signal power (and visual detect ability), they found out that the locations of poor signal estimation, and thus decoding errors correlate to the edges within the image. Therefore, the limitation of this approach is its inability to retrieve all the message bits.

Wang *et al.*, (2001) proposed a method using the genetic algorithm to embed secret data into each host pixel and the transformed value is closer to the original host pixel. However, using the genetic algorithm consumes huge computational time and the solution of a mapping function is not optimal.

Koppola (2009) worked on a Master's thesis titled A High Capacity Data-Hiding Scheme In LSB-Based Image Steganography. The technique aim at embedding large amount of data in Red, Green, Blue and Alpha (RGBA) images while keeping the perceptual degradation to a minimum. He used the YIQ color space model to transform RGB value of the secret image pixel into three separate components, which are then embedded into the least significant bits of each color pixel in a cover colour RGB image, as well as in the alpha channel. This technique holds good against visual attacks but the disadvantage is that it is detectable against statistical analysis because it uses 13 LSB bits each of the cover image. Also the retrieved secret image incurred some loss of data due to the transformation performed on the original secret image. This was due to the rounding off the pixel values i.e. like rounding off from 0.5 to 1.0 or 0.1 to 0.

Sujay and Prasad (2010) in their work titled “Two New Approaches For Secured Image Steganography using Cryptographic Techniques and Type Conversions” introduced two new methods wherein cryptography and steganography are combined to encrypt the data as well as to hide the encrypted data in another medium to concealed the fact that a message is being sent. One of the methods shows how to secure the image by converting it into cipher text using S-DES algorithm with a secret key and conceal this text in another image by LSB steganographic method. Another method shows a new way of hiding an image in another image by encrypting the image directly by S-DES algorithm using a key image and the data obtained is concealed in another image. The Method used is that the message bit is Exclusive-ORed with the 2nd least significant bit of the carrier byte and the LSB of carrier medium is replace by the result bit.

Ghosal (2010) worked on a dissertation titled; An Enhanced, Secure and Comprehensive Data Hiding Approach Using 24 Bit Color Images. The proposed work discusses the design of an application by encrypting textual information using a secret key and the same is embedded into a 24 bit colour image by using a novel steganographic method. The image is again encrypted using another secret key. Finally, the encrypted image is embedded within the cover image using the proposed steganographic method. The researcher embedding algorithm is presented thus:

Step 1: Load a 24 bit color image and assume the image consists of a set of 3x3 blocks.

Step2: Take each block, where the block contains 9 pixels and embed the secret data on the blue component of the pixel positions 1,3,5,7 and 9.

Step3: Now we will check the MSB (5th, 6th and 7th bit) bit of the pixel positions 1, 3, 5, 7 and 9 and up to 4 numbers of bits is to be embedded starting from LSB (Least Significant Bit).

To extract the message the reverse of embedding process is done thus:

Step1: Load the stego-image to extract the hidden data and let, we assume, the image consists of a set of 3x3 blocks.

Step2: Take each block, where the block contains 9 pixels and extract the hidden data from the blue component of the pixel positions 1,3,5,7 and 9.

Step3: Now we will check the MSB (5th, 6th and 7th) bit of the pixel positions 1, 3, 5, 7 and 9 and up to 4 numbers of bits is to be extracted starting from LSB (Least Significant Bit). The work compared to the traditional LSB proffered higher hiding capacity, however much degradation is incurred by the stego image produced.

In Challita and Farhat (2011) steganography and cryptography were combined to secure data. They illustrated two different approaches that help achieve a higher level of secrecy and security, together with their limitations. The first method is about combining steganography and cryptography in such a way to make it harder for a steganalyst to retrieve the plaintext of a secret message from a stego-object if cryptanalysis were not used. The second method does not use any cryptographic techniques at all and relies solely on steganographic ones. The procedure involved is divided into two steps. The first step converts the cover image denoted by image1 and the secret message denoted by secret1 into their bit equivalent. The next step encodes the secret message Secret1 based on Image1. The idea is based on the problem of finding the longest common substring of two strings using a generalized suffix tree, which can be done in linear time. The algorithm uses a divide-and-conquer strategy and works as follows.

It starts with the whole bits of Secret1 and tries to find a match of all the bits of Secret1 in Image1. If this is the case, it stores the indexes of the start and end bits of Secret1 that occur within Image1 in an output file Output1. If not, the algorithm recursively tries to find a match of the first and second halves of Secret1 in Image1. It keeps repeating the process until all the bits of Secret1 have been matched with some bits of Image1.

Another study conducted by Nivedhitha and Myeyappan (2012) used DES encryption algorithm and encryption to secure image data. The encrypted image is hidden in another image using LSB techniques, so that the secret's of the image very existence is concealed. The decryption is performed by the same key image using DES algorithm. The pixel value of encrypted image is hidden in the LSB of pixels of carrier image by merging it with the 2nd LSB of carrier pixel. If the size of the encrypted image is mxn, then the size of carrier image must be mxnx8 as each encrypted byte requires 8 bytes (pixels) of carrier image. So if the carrier image size is not eight times the size of the payload, then it has to be resized. Large image files often slowdowns the bandwidth of network as such this approach is not Internet friendly.

Laskar and Hemachandran (2013) proposed a novel approach to increase the security of a hidden data in RGB image. In their work, Data is hidden in the LSB of a particular colour plane (Red plane) of randomly selected pixel in the RGB colour space. The embedded algorithm uses a pseudorandom number generator (PRNG) to choose random pixels in which to embed the message and a stego-key. Their Pseudocode for embedding and extraction is presented below

The embedding algorithm:

a. Read character from text file that is to be hidden and convert the ASCII value of the character into equivalent binary value into an 8 bit integer array.
b. Read the RGB colour image into which the message is to be embedded and Extract the red component of the host image.
c. Read the last bit of red pixel i.e. from RGB (8+8+8) bits.

d. Initialize the key which gives the random position of the red pixel to be processed for embedding.

e. Check the last bit if it is 0 or 1 and present in each pixel and store the cumulative sum into two integer variables suppose x1 and x2 holding total number of 0s & 1s respectively.

f. Pick up the message bit. If the message bit is zero (or one), check if x1 > x2 otherwise swap x1 and x2. Do the reverse operations for the message bit one or zero.

g. If value of flag is 0 then embed the data bit (either 0 or 1) to the last bit of colour red of the pixel otherwise if flag is 1 then inverse the data bit embed it to the last bit of colour red of the pixel.

h. Write the above pixel to Stego Image File.

Extraction algorithm

a. Open the Stego image file in read mode and from the Image file, read the RGB colour of each pixel.

b. Extract the red component of the host image.

c. Read the last bit of each pixel i.e. from RGB (8+8+8) bits.

d. Initialize the key that gives the position of the message bits in the red pixel that are embedded randomly.

e. Read the last bit of pixel in colour red. Based on its value, set integer variable check flag 0 or 1.

f. For decoding, select the pixels using the same pseudo-random sequence. Check if the 2 pixels are within the pre-specified range. If x1>x2, the message bit is zero (one) otherwise the message bit is one (zero).

g. If check flag is 0 then read the last bit of each pixel that is the LSB of colour red and put it directly in an array otherwise take the invert value of the last bit & put it on array.
h. Read each of pixels in this way and convert the content of the array into decimal value that is actually ASCII value of hidden character.
i. If terminating character's ASCII found, print nothing otherwise print the corresponding character of the calculated ASCII value.

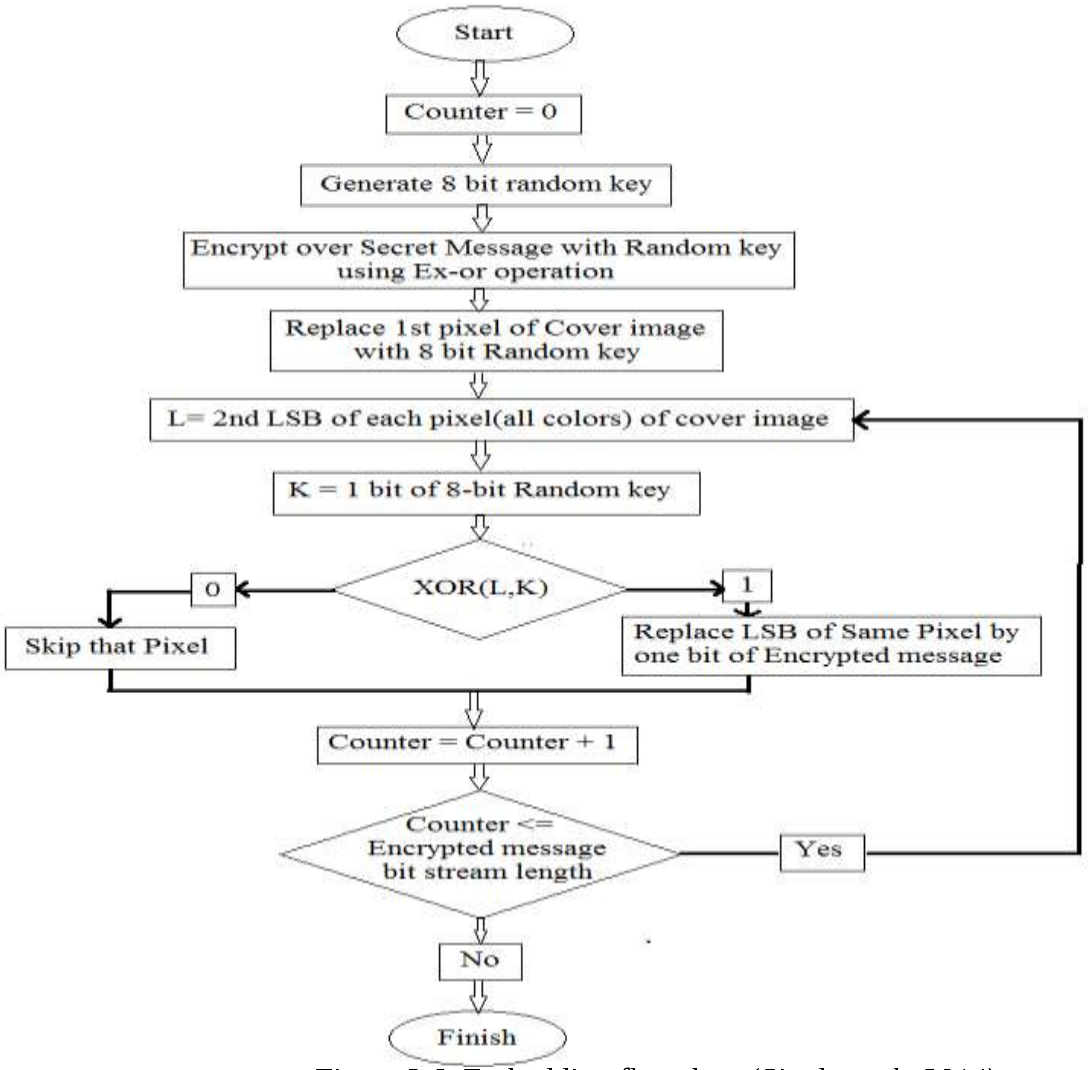
Three images were used for the experimental analysis. They claimed that the difference of the stego-image can hardly be distinguished after embedding process and no loss of data during extraction. The mean square error MSE and peak signal to noise ratio PSNR of the original image and the stego-image were measured for the three image samples. The result indicated that PSNR is about 58 dB and also the size of the image was not changed after embedding. Their approach however hides fewer amounts of data within an image.

However, Tiwari *et al,* (2013) carried out a survey on digital image steganography and steganalysis. They explained the concept of steganography and how it differs from cryptography. Comparison of the effectiveness of common image steganographic algorithms in terms of six parameters namely invisibility, bit rate, robustness against statistical attack, robustness against Image Processing (IP) attacks, independence of file format and use of unsuspicious files was presented in the work. It was noted that an ideal Steganographic algorithm would have a high level in every requirement which is non-existent yet creating a trade-off, depending on which requirements are more important for the specific application. In conclusion, they claimed hybrid Steganography techniques like Patchwork and Spread Spectrum Techniques are efficient enough for enabling covert communication. It recommended combining them with Cryptographic techniques to provide high security. Their work did not carry out any experiment to analyze its claim.

Further research was carried out by Singh *et al*., (2014). Their work takes binary representation of the hidden information and overwrites LSB of selected byte in cover image. 8-bit random key is used to determine whether to hide one bit of secret key into cover image or not. Simple Ex-or operation is used for choosing the hidden method. They used the formula below and the flowchart of embedding and extraction is given in figure 2.6 and figure 2.7 respectively.

Secret plain text + 8-bit random key = encrypted text

Cover image + 8-bit random key + encrypted text = stego image


Start
Counter = 0
Generate 8 bit random key
Encrypt over Secret Message with Random key using Ex-or operation
Replace 1st pixel of Cover image with 8 bit Random key
L= 2nd LSB of each pixel(all colors) of cover image
K = 1 bit of 8-bit Random key
XOR(L,K)
0
Skip that Pixel
1
Replace LSB of Same Pixel by one bit of Encrypted message
Counter = Counter + 1
Counter <= Encrypted message bit stream length
Yes
No
Finish
Figure 2.6: Embedding flowchart (Singh *et al*., 2014)

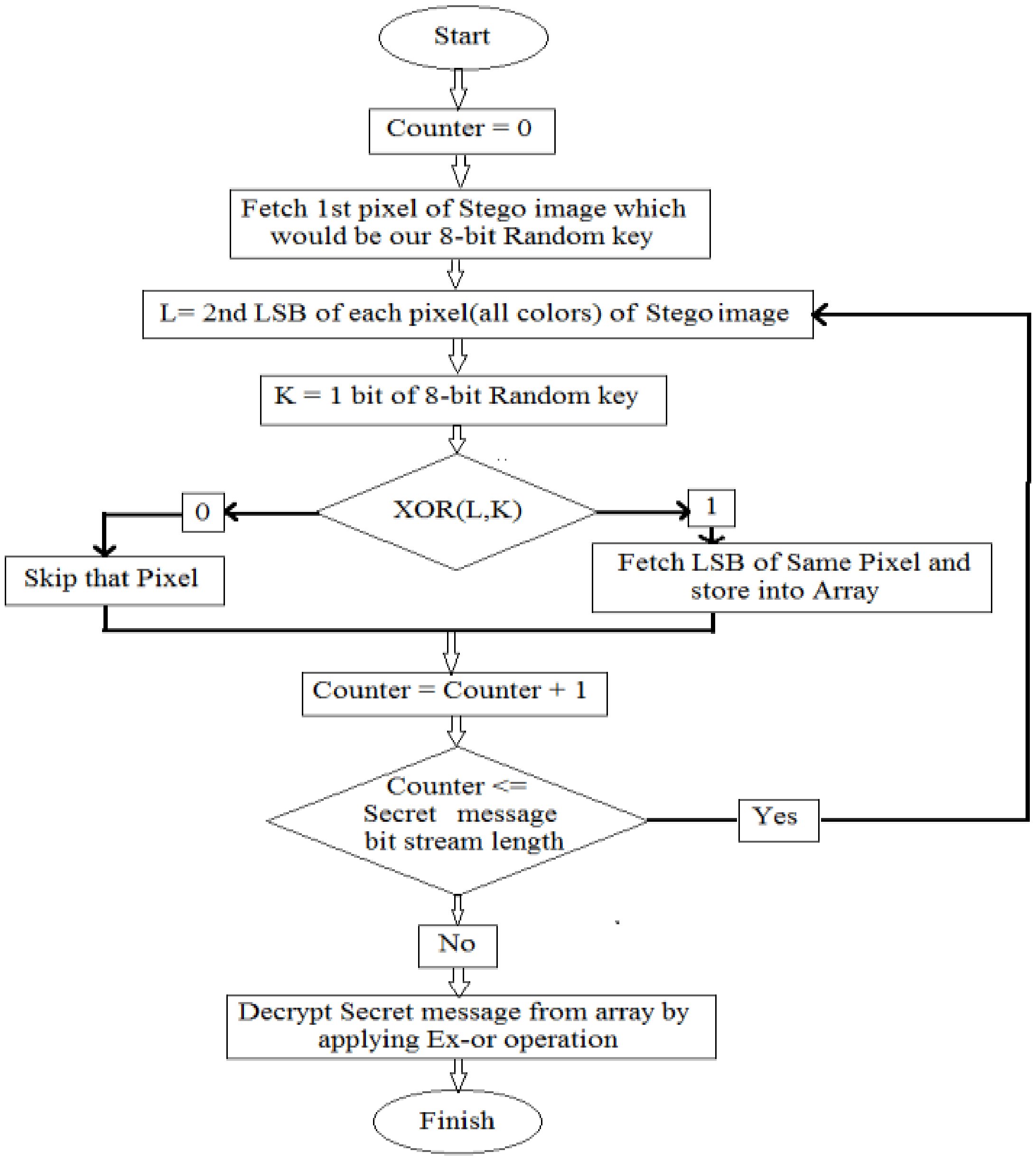

Start
Counter = 0
Fetch 1st pixel of Stego image which would be our 8-bit Random key
L= 2nd LSB of each pixel(all colors) of Stego image
K = 1 bit of 8-bit Random key
XOR(L,K)
0
1
Skip that Pixel
Fetch LSB of Same Pixel and store into Array
Counter = Counter + 1
Counter <= Secret message bit stream length
Yes
No
Decrypt Secret message from array by applying Ex-or operation
Finish

Figure 2.7: Extraction flowchart (Singh *et al.*, 2014)

Moreover, Saini *et al.*, (2014) proposed a new steganographic approach to RGB images using modulus factor to determine the pixel location to hide bits of secret data. To insert message bits, for each pseudorandom pixel location, mod value is calculated by taking modulus of Red, Green and Blue component with the Mod Factor(4). Then comparison is done for the calculated mod value with the message bits to be hidden. For retrieval of message, same concept is used as for insertion but in reverse. Receiver will calculate the mod value of RGB components with the Mod Factor (4) for the same pseudorandom pixels as agreed with the sender. The authors assume the sender and receiver agree on the cover image to be used and the pseudorandom number which serves as the key.

Prashanti and Sandhyarani (2014) proposed a novel steganographic method using LSB in their work titled; Secure Communication using modified DES with Steganography. They first generate a hash value of the message by using SHA-1 hash function. This hash value is hidden in the 4LSB of the red component of the pixel. The second step involved encryption of the secret message using modified simplified DES algorithm in which a # function is introduced instead of XOR function. This encrypted message (cipher text) is then hidden in the 4LSB of the green and blue components of the pixel. At the receiver side, 4LSB of the red component of the pixel are extracted and placed in an array which is hash value that checks the integrity of message. The 4LSB of the green and blue components of the pixel are extracted (cipher text) on which modified simplified DES algorithm is applied which gives the original message (plaintext).

In Al-Shataiwa and Emam (2015), a new algorithm for image steganography to hide a large amount of secret data in a color image was proposed. This algorithm was based on different size image segmentations (DSIS) and modified least significant bits (MLSB), where the DSIS algorithm has been applied to embed a secret image randomly instead of sequentially; this approach is applied before embedding process. The number of bit to be replaced at each byte is non uniform, it bases on byte characteristics by constructing an effective hypothesis. The approach produces a resultant image with very low perceptibility of 35dB.

2.11 Literature gap

Many approaches mentioned in the literature present their own strength and weakness in terms of security and complexity. Some of the weaknesses include loss of message bits during extraction process, others tackle payload capacity at a very low PSNR which causes much degradation to resultant image thereby leading to steganalysis detecting the algorithm. The proposed algorithm in chapter three further addressed the challenges identified in the literature reviewed.

CHAPTER THREE

RANDOM BYTE POSITION IMAGE LSB CRYPTO-STEGANOGRAPIC ALGORITHMS

A classical steganographic system's security relies on the encoding system's secrecy. An example of this type of system is a Roman general who shaved a slave's head and tattooed a message on it. After the hair grew back, the slave was sent to deliver the now-hidden message (Johnson and Jajodia, 1999). Although such a system might work for a time, once it is known, it is simple enough to shave the heads of all the people passing by to check for hidden messages ultimately, such a steganographic system fails. Modern steganography attempts to be detectable only if secret information is known. This is similar to Kerckhoffs' Principle in cryptography, which holds that a cryptographic system's security should rely solely on the key material.

3.1 The Proposed Framework

The steganographic framework is a modification of the traditional LSB. The traditional LSB only hides message bits direct into the LSB of each colour component without any encryption. The approach in this dissertation is divided into two phases as illustrated in figure 3.1, encryption of the message and then hiding it in Red and Blue plane of a coloured cover image. The dotted box illustrates the improvement on traditional LSB by the proposed technique.

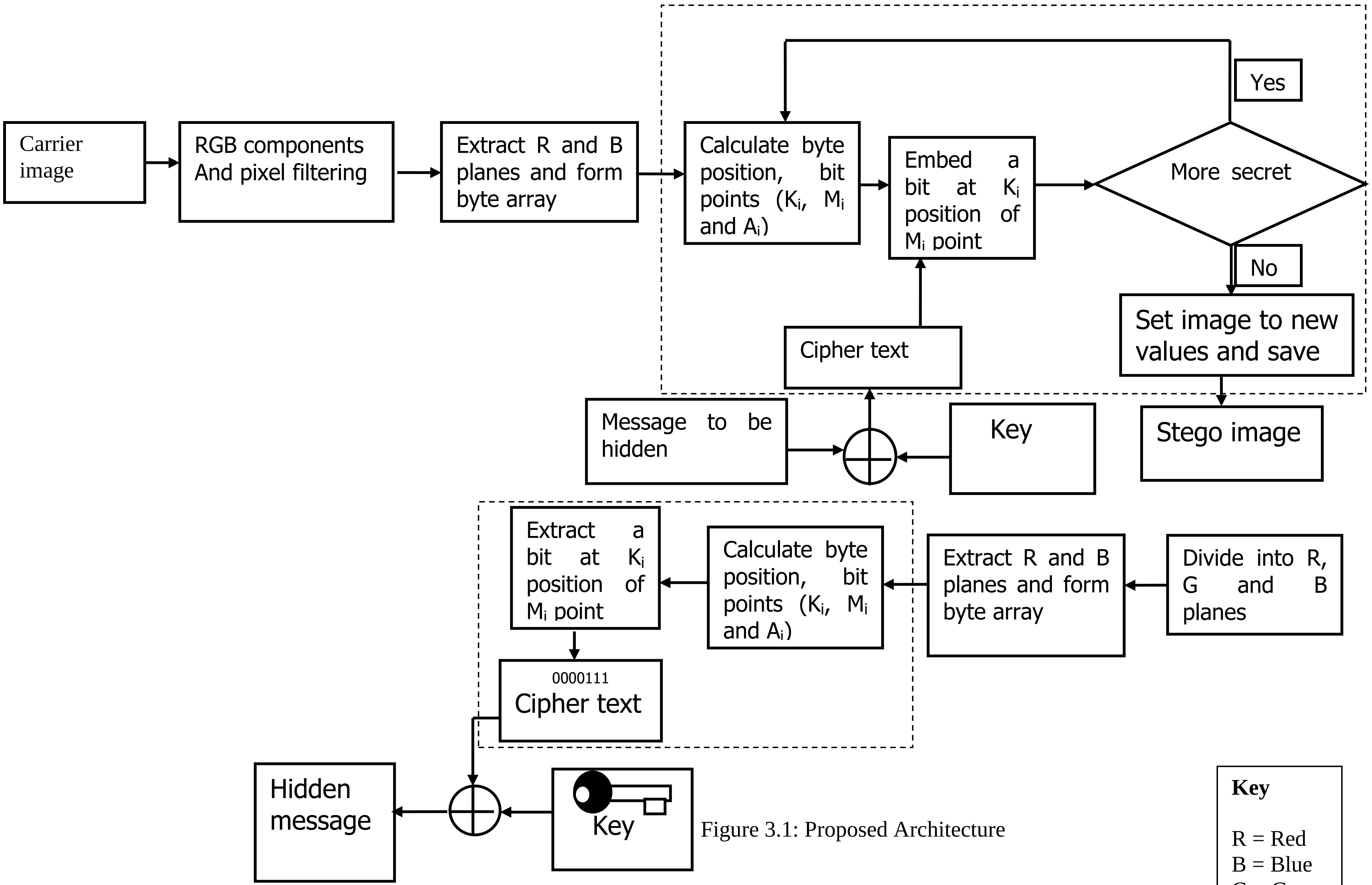

Carrier image
RGB components And pixel filtering
Extract R and B planes and form byte array
Calculate byte position, bit points (K_i, M_i and A_i)
Embed a bit at K_i position of M_i point
More secret
Yes
No
Set image to new values and save
Stego image
Cipher text
Message to be hidden
Key
Divide into R, G and B planes
Extract R and B planes and form byte array
Calculate byte position, bit points (K_i, M_i and A_i)
Extract a bit at K_i position of M_i point
0000111
Cipher text
Hidden message
Key

Figure 3.1: Proposed Architecture

Key

R = Red
B = Blue
G = Green

3.2 XOR Encryption and Decryption process

XOR encryption algorithm is widely used in cryptography, especially in stream ciphers, and has received lots of attention on them (Pantelimon, 2009). The algorithm is simple and takes less space but very efficient. In the proposed technique, the length of the plaintext and encrypted message (ciphertext) is the same. By increasing the key length to 64 bits the complexity of algorithm against brute force and statistical attack is improved.

3.2.1 Encryption Algorithm

The encryption algorithm takes the original message (plaintext); its ASCII representation is converted to its binary equivalent called P and the binary representation of key K. S signifies the key size and r the plain text binary size. B is the quotient obtained from s and r which is used to form blocks for the plain text. XOR operation is performed between binary value of key and binary value of plaintext at each blocks. This step or iteration is performed according to the value b. Finally, each block is combined to form a cipher text ‘C’ and it’s stored in form of a byte array. The process of manipulation is given in algorithm 3.1 and 3.2.

Algorithm 3.1: Encryption Algorithm (P, K)

```
// Input plaintext P and key K
// Output cipher text C
Let P = plaintext, K = key, C = ciphertext
$P_i$: i = (1, 2, …,n)
s ⟵ K          // s is the binary length of the key
r ⟵ P          // r is the binary length of the plaintext
        b ⟵ r/s       //number of blocks
            For i ⟵ 0 to b - 1
                    $b_i \leftarrow P_{i,\ldots,b-1}$
                For j⟵1 to s
                    $C += b_i \leftarrow P_i \oplus K_i$
                    j++
                End-For
              i++
            End-For
Display C
```

3.2.2 Decryption Algorithm

The decryption process uses the same key as encryption. The decryption algorithm is given in algorithm 3.2.

Algorithm3.2: Decryption Algorithm (C, K)

```
// Input Ciphertext C and key K
// Output plaintext P
Let P = plaintext
Let K = key
Let C = ciphertext
$C_i$: i = (1, 2, …,n)
s ⟵ k          // s is the binary length of the key
r ⟵ c          // r is the binary length of the ciphertext
        b ⟵ r/s       //number of blocks
            For i ⟵ 0 to b - 1
                  $b_i \leftarrow C_{i,\ldots,b-1}$
                For j⟵1 to s
                    $P += b_i \leftarrow C_i \oplus K_i$
                  j++
                End-For
              i++
            End-For
Display P
```

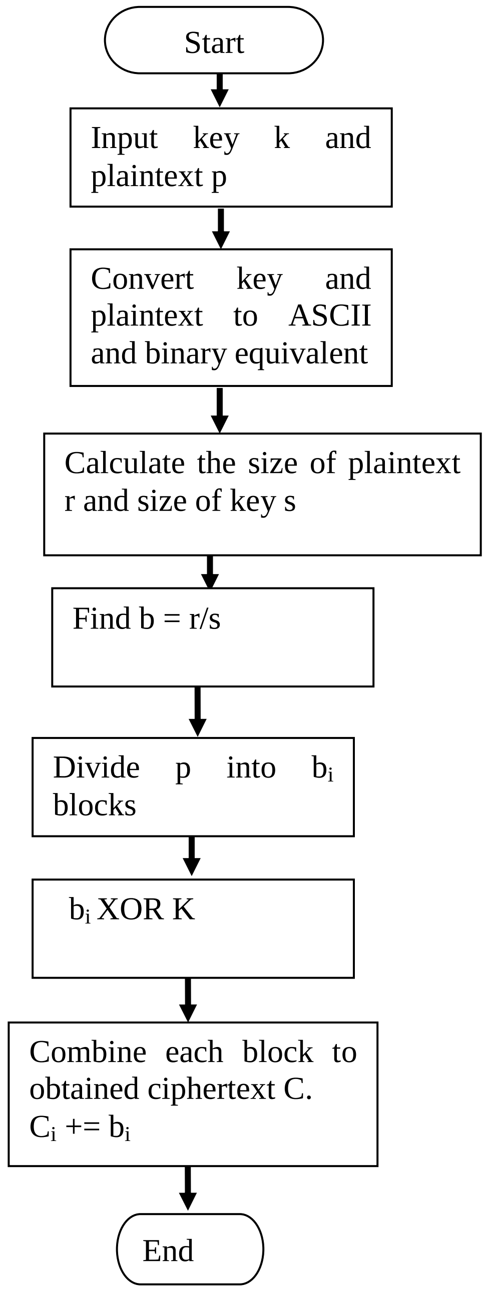

Start
Input key k and plaintext p
Convert key and plaintext to ASCII and binary equivalent
Calculate the size of plaintext r and size of key s
Find b = r/s
Divide p into b_i blocks
b_i XOR K
Combine each block to obtained ciphertext C.
C_i += b_i
End

Figure 3.2: XOR Encryption Flowchart

3.3 Steganographic process

3.3.1 Image LSB technique algorithm

The traditional LSB technique uses sequence-mapping approach to replace last bit of each colour plane of the cover image with the message bit thereby producing a stego image. The message embedding procedure is given in equation 1:

$$I_{s(i,j)} = \begin{cases} I_{(i,j)} - 1 & LSB(I_{(i,j)}) = 1 \; and \; m = 0 \\ I_{(i,j)} & LSB\,(I_{(i,j)}) = m \\ I_{(i,j)} + 1 & LSB(I_{(i,j)}) \neq 0 \; and \; m = 1 \end{cases} \quad \ldots \quad 3.1$$

Where LSB ($I_{(i,\ j)}$) stands for the LSB of cover image at row i and column j and m is the next message bit to be embedded and $I_{s(i,j)}$ is the stego image Laskar and Hemachandran (2013). The LSB algorithm is given in algorithm 3.3. Changes in the image resolution are quite clear which makes it easier for attacker to break the system.

Algorithm 3.3: LSB hiding Algorithm

```
//input cover image, message
// output stego image
Step 1: Input cover image I and message M
Step 2: Encode secret message in binary
Step 3: Calculate size of I and M
Step 4: Scan image row by row and encode it in binary
        Step 4.1: For-each colour
                        Divide I into its component // (RGB)
                        Hide one bit each of m in each part of the pixel in the last LSB
                        Set I with the new values
                End For-each colour
Step 5: Set the image with new value and save it as stego image
```

Extraction process is reverse process of embedding whereby message bits are retrieved directly from the LSB of each plane. This work has been developed to overcome a sequence-mapping problem when using LSB by varying the insertion point and the pixel position as discussed in the next section.

3.3.2 Proposed embedding algorithm

The approach used randomizes the point of embedding of the secret file in the colour image file red and blue sample points. The size of the secret data file is of great significance to the randomization algorithm. The randomization pattern used in this design is twofold the first step finds out the byte numbers that will be the sample point for embedding data and the second step will be the bit place in that byte where the one bit of the secret data will be embedded as presented in algorithm 3.4 and sub algorithm 3.4.1. By this way the imperceptibility and transparency of the steganographic technique is increased.

Some assumptions/notations

P is the plain text (i.e original message)
C is the size of the secret message (Cipher - text)
I is the cover image

$K_i = \boldsymbol{C/8}$	F1
$\mathbf{A_i}$ = additive intermediate result for sum of each digits of $\mathbf{K_i}$	F2
$\mathbf{M}_i = A_i\%8$	F3
$\mathbf{K_{i+1}} = M_{\mathbf{i}} + K_i$	F4
$\mathbf{A_{i+1}}$ = additive intermediate results for sum of each digits of K_{i+1}	F5
$\mathbf{M_{i+1}} = A_{i+1}\%8$	F6

Stego Image S = Cipher + cover image

<u>Algorithm 4: Proposed Embedding Algorithm</u>

```
Step1: Input cover image, message m;
Step2: Apply encryption technique to produce C; // see encryption
Step 3: Calculate the size of C;
Step 4: For-each color in I;  // Color ε $\{R, G, B\}$
            Step 4.1: Call Randomization (.) Algorithm to find the random byte position and
            LSB insertion points; // see Sub-Algorithm 3.1
        End For-each color
Step 5: replace last byte with value of C
Step6: Set the image with the new values and save it as stego image S.
```

Sub-Algorithm 4.1: Embedding points

```
Randomization (BPLSBP) {
//Scanning each byte to locate Byte Positions and Least Significant Bit Point (BPLSBP)
        **For-each** raster R,
                Scan, extract RB and form byte array; // byte array Red and Blue colors only
                        To get first embedded point;
                                Calculate $K_i$;            //Using F1
                                Calculate $A_i$;            // Using F2
                                Calculate $M_i$;            //Using F3
                          Perform 1st embedding// embed first message bit at $K_i$ position and $M_i$
                                point
                        **For-each** remaining message bit;
                                Calculate new $K_i$      // $K_{i+1}$ see F4
                                Calculate new $A_i$      //$A_{i+1}$ see F5
                                Calculate new $M_i$;     //$M_{i+1}$ see 6
                            Perform embedding according to the values; // embed the remaining
                             message bits at each $K_i$ positions of Mi points
                              **If** $M_i$ is 0, 7, 6, 5;
                               Then embed at 1st LSB point
                              **End-If**
                        **End For-each** **//**end of For each remaining message bit
        **End For-each** // end of For each raster
} // End sub-Algorithm 3.1
```

3.3.3 Proposed extraction algorithm

Extraction procedure retrieves the embedded message from image file. It performs the reverse operation as presented in algorithm 3.5.

Algorithm 3. 5: Proposed Extraction Algorithm

```
**Step 1: Input** stego image S
**Step 2: For each** colour in Stego Image S;
**Step 3:**    **For each** raster R;
            **Step 3.1:** Scan, extract RB and form byte array // byte array Red and Blue colors only
            **Step 3.2: Read** last byte of byte array for value of C;
            **Step 3.3: calculate** BPLSBP // F1 - F6
            **Step 3.4: perform** extraction depending on the Ki and Mi values gotten;
                **Step 3.4.1** If $M_i$ is 0, 7, 6, 5, extract bit at position 1;
           **End For-each** // for each raster
        **End For-each**    // end of for each colour
**Step4: Call** algorithm 3.2 to get original message M // see decryption algorithm
```

EXAMPLE

Assuming the message to be hidden is “terrorist attack ongoing more troops needed” then this message can be hidden within an image using the proposed embeddimg algorithm thus.

Size of the message C is 42 x 8 = 336 byte

$K_i = (336)/8 = 42$

$A_i = 0+4+2 = 6$

$M_i = 6\%8 = 6$ The first bit of the message file will be stored into the **42th** byte position of RB extract and at 1st LSB position.

$K_{i+1} = K_i + M_i = 42 + 6= 48$

$A_{i+1} = 0 + 4 + 8 = 12$

$\mathbf{M}_{i+1}= 12\%8 = 4$

The next bit of the message file will be stored at 4th LSB position of the 48**th** Byte position.

Similarly the **3**rd bit will be embedded at 1st LSB position of the 52nd Byte. This will continue till all the bit in the secret message gets embedded into the image file.

CHAPTER FOUR

RESULT ANALYSIS AND EVALUATION

4.1 Implementation details

4.1.1 Java Programming Language

Java is an object-oriented language that was created and developed by software engineers at Sun Microsystems during the early 1990s. Code written in Java is compiled into bytecode which can then be executed on the Java Virtual Machine. In practice this means that programs written in Java can be run on any computer which supports a JVM, regardless of the underlying hardware or operating system, which is a key feature of the language. The proposed algorithm was built using JDK 7.0.400.43 on NetBeans IDE 8.0.1 Platform. The many security plug-ins available in Java makes it a suitable programming language for this dissertation.

4.1.2 StegExpose

It is a computer program which allows users to automate the process of steganalysis (Boehm, 2014). The tool is built to be universal for lossless images such as those stored under the PNG or BMP format. StegExpose belongs to freeware category of software made available for steganalysis purposes. It was designed by Benedikt Boehm in Computing University of Kent, England. StegExpose can be run in the background for analyzing multiple images without human supervision and it will return a detailed steganalytic report once the tool has finished its job.

4.1.3 Data

To test the program, different sizes of 24 bit coloured images of .jpg .png and .bmp format were used to hide text of varying length. However, for evaluation purposes six standard images of .bmp and .png were considered. The images were retrieved from Internet. The following are the images used

Figure 4.1 Clover.png

Figure 4.2 tulips.bmp

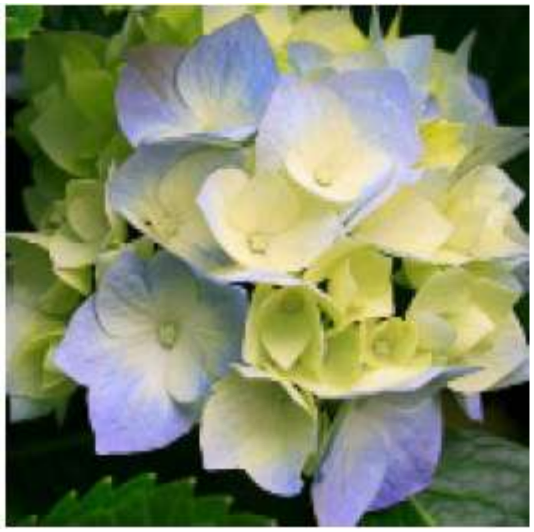


Figure 4.3 flowers.bmp

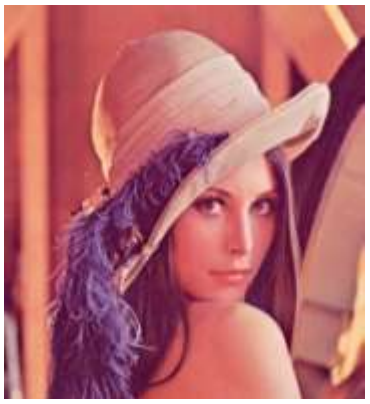


Figure 4.4: lena.bmp

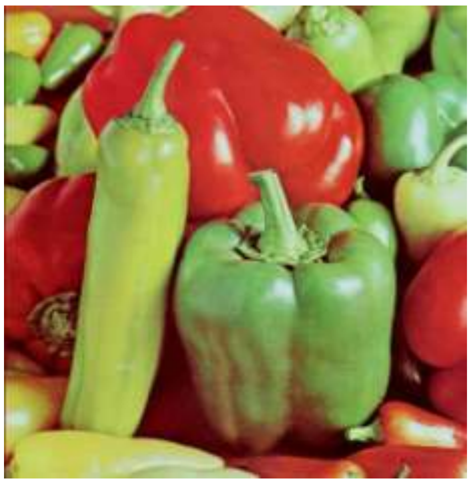


Figure 4.5: peppers.png

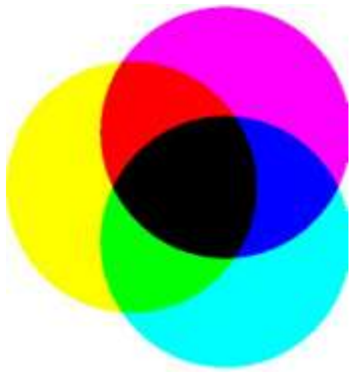


Figure 4.6: colors.png

4.2 System Specification

The hardware and software specification used for the simulation of the algorithms are as follows

Hardware

a Hewlett Packard (HP) laptop with a Intel Duo Core processor running at 2.40GHz

b 4.00GB of RAM

c 500GB of hard disk

Software

a Windows 8.1 operating system

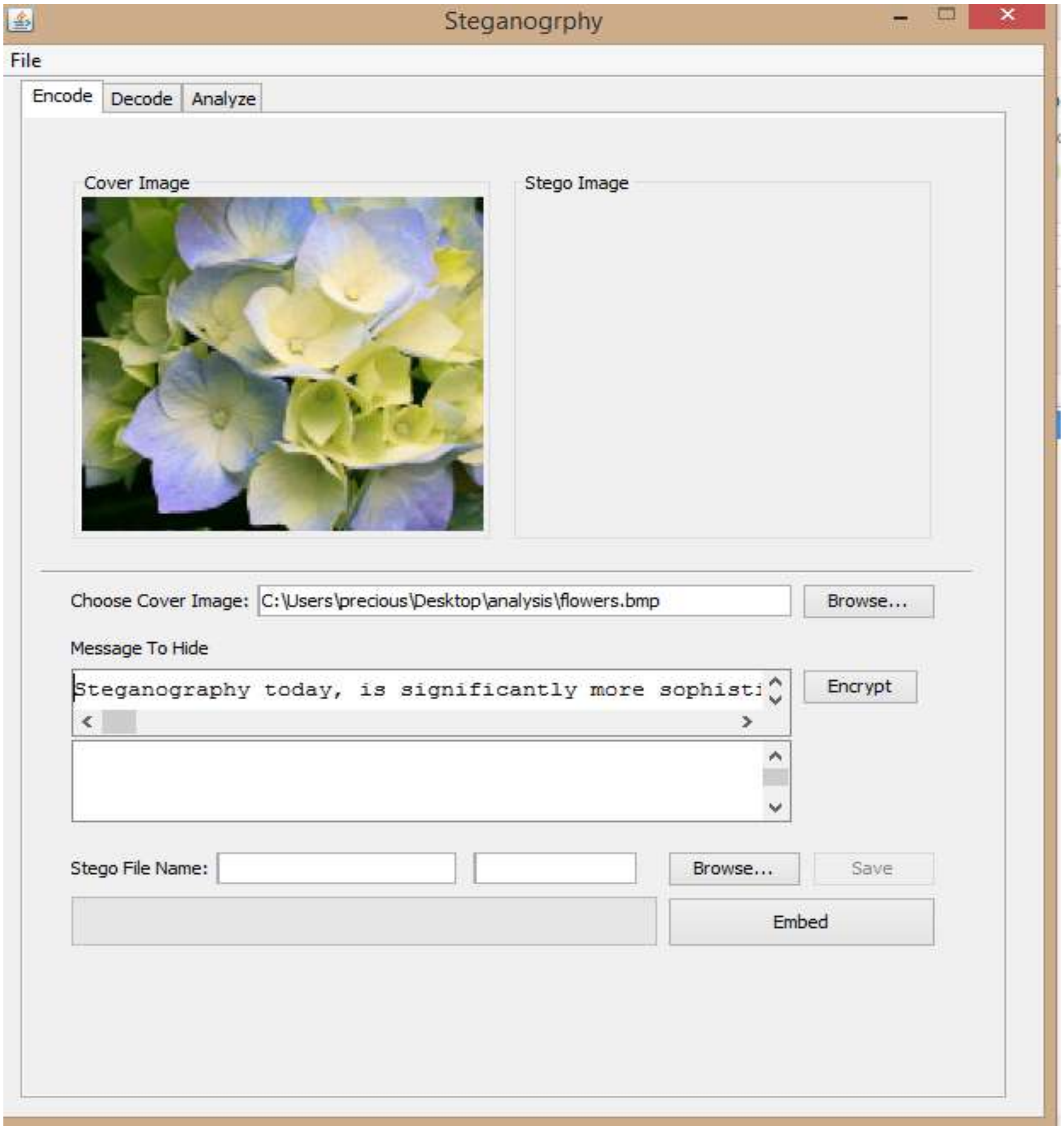
b Netbeans 8.0.1 IDE

c JDK 7.0.400.43

4.3 Implementation

The implementation of the work is made easy by designing a graphical user interface (GUI). With this, the communicating partners can easily perform their task. The snapshot of the GUI is as shown in figure 4.7. In the proposed system, different operations can be performed. Encode interface is where a sender performs embedding and encryption operations. In the decode interface, the receiver extract the hidden message bits by selecting the stego image he want to extract the bits from and providing correct encryption key to decrypt the ciphertext to plaintext.

The analyze interface takes care of analysis of the cover image and the corresponding stego image produced. It calculates the Histogram, PSNR, MSE and message bytes. File on the other hand contains two commands namely reset and exit commands.



C:\Users\precious\Desktop\analysis\flowers.bmp

Steganography today, is significantly more sophisti

Figure 4.7: System GUI

4.4 Performance Metrics

A series of experiments were conducted to show the effectiveness and security of the proposed technique. The efficiency of the proposed technique is measured by five metrics which are

a. Payload capacity.
b. PSNR (Peak Signal-to-Noise Ratio) of cover and stego images.
c. Additional security of payload
d. Statistical analysis using histogram of the cover and stego images
e. Steganalytic test

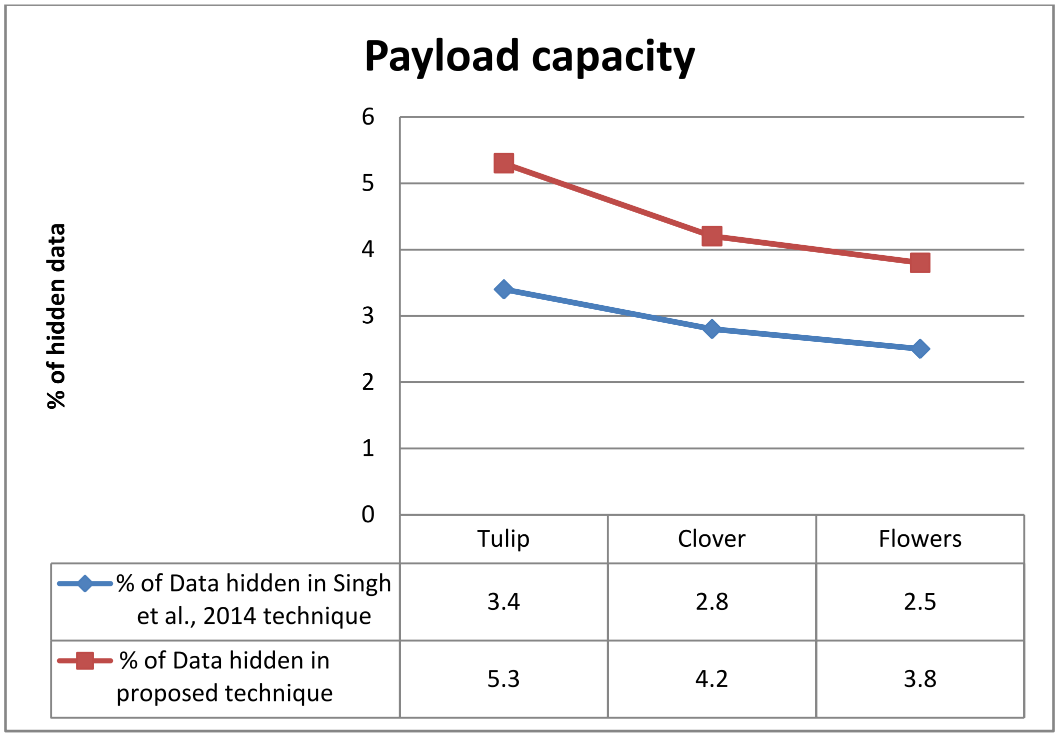
4.4.1 Payload capacity

Payload or embedding capacity refers to the amount of secret information that can be embedded without much degradation to the quality of the cover image. It is important to measure this criterion to avoid exposing the stego image produced to attack.

Table 4.1 shows that the proposed technique has the capability of embedding up to the 5.3% amount of secret data as the cover image. Singh *et al.,* (2014) technique only embed bits of secret message where XOR operation between the key and 2nd LSB is 1 else it skips the pixel, therefore can only hold 3.4% of payload. However, this result can also be affected by the type and size of image used. Three different sizes of images were used to determine the capacity of embedding between the two techniques as illustrated in table 4.1 and corresponding graph in figure 4.8.

Table 4.1: Amount of Data Hidden in the Stego Images

Images	**% of Data hidden in Singh *et al.*, 2014 technique**	**% of Data hidden in proposed technique**	**Size of the Image (Bytes)**
Clover	3.4	5.3	81920
Tulip	2.8	4.2	93184
Flowers	2.5	3.8	70656



	Tulip	Clover	Flowers
% of Data hidden in Singh et al., 2014 technique	3.4	2.8	2.5
% of Data hidden in proposed technique	5.3	4.2	3.8

Figure 4.8: Graph of payload capacity

4.4.2 Imperceptibility test using PSNR

PSNR (Peak Signal to Noise Ratio) is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. The signal in this case is the cover image, and the noise is the error introduced by bits of secret image. The higher the value of PSNR, the more quality is the stego image. This ratio is often used as a quality measurement between the original (cover) image and a stego image as it defines the similarities between cover image and stego image. Higher indicates minor degradation of the stego image. Equations 4.0 and 4.1 show how to calculate PSNR and MSE (Saini *et al.,* 2014).

$$PSNR = 10 \times log\left(\frac{(255)^2}{MSE}\right) \qquad \ldots \qquad 4.0$$

MSE (Mean Square Error) is the average squared difference between a reference image and a distorted image. It is computed pixel-by-pixel by adding up the squared differences of all the pixels and dividing by the total pixel count. It is a measure use to quantify the difference between the cover image I and the Steg image I'. Lower MSE indicates better quality of Stego Image. If the image has a size M*N then:

$$MSE = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I_{(i,j)} - I_{(i,j)}\right]^2 \qquad \ldots \qquad 4.1$$

Where:

I $_{(i,j)}$ is the Intensity of pixels ij in the Cover Image

I'$_{(i,j)}$ is the Intensity of pixels ij in the Stego Image

N is the Number of pixel rows in the Cover Image

M is the Number of pixel columns in the Cover Image

Table 4.2 presents the result of MSE and PSNR gotten using the proposed technique and that of Singh *et al.,* (2014) and figure 4.9 shows the corresponding graph of PSNR of the two algorithms. The proposed algorithm performs well when embedded data is high and when secret data is small it performs much better. The reason being the encryption key has no effect on steganographic algorithm. Singh *el al.,* (2014) approach performs better with small amount of hidden message.

Table 4.2: PSNR and MSE Comparison

Original Image (KB)	**Stego-Image (KB)**	**Proposed MSE**	**Singh *et al.*, 2014 MSE**	**Proposed PSNR (dB)**	**Singh *et al.*, 2014 PSNR (dB)**	**Data Embedded (in bytes)**	**Data Extracted (in bytes)**
Peppers.png	Peppers_Stegos.png	0.0039	0.0030	70.2	72.35	24	24
Colors.png	Colors_Stegos.png	0.0046	0.0036	71.2	71.7	30	30
Lena.bmp	Lena_Stego.bmp	0.0031	0.0037	70.8	70.5	64	64
Flowers.bmp	Flowers_Stego.bmp	0.0256	0.0889	64.1	48.17	1300	1300
Clovers.png	Clovers_Stego.png	0..0304	0.0804	61.3	41.81	1548	1548
Tulips.bmp	Tulips_Stego.bmp	0.0406	0.0947	60.2	40.39	2002	2002

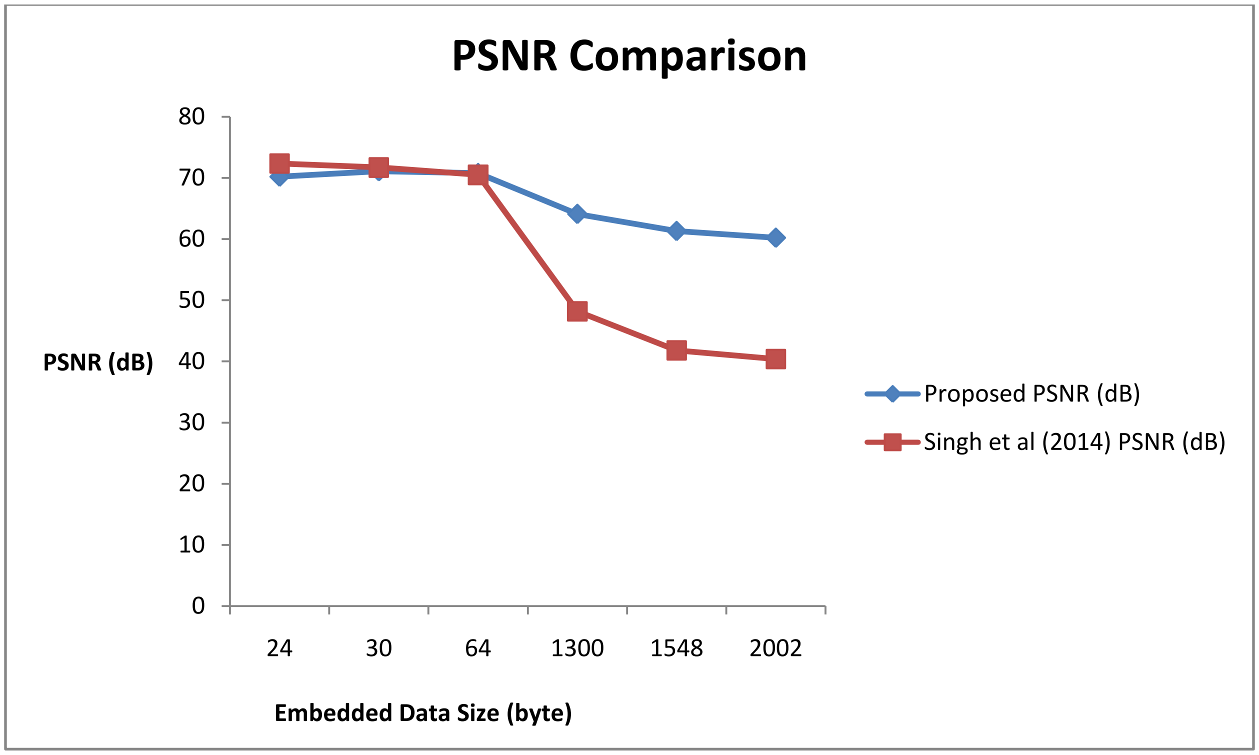


Figure 4.9: Graph of PSNR against 6 different images

4.4.3 Additional security

Image steganographic algorithm can be further enhanced by encryption and randomizing positions of embedding. Table 4.3 compares both algorithms in this regard.

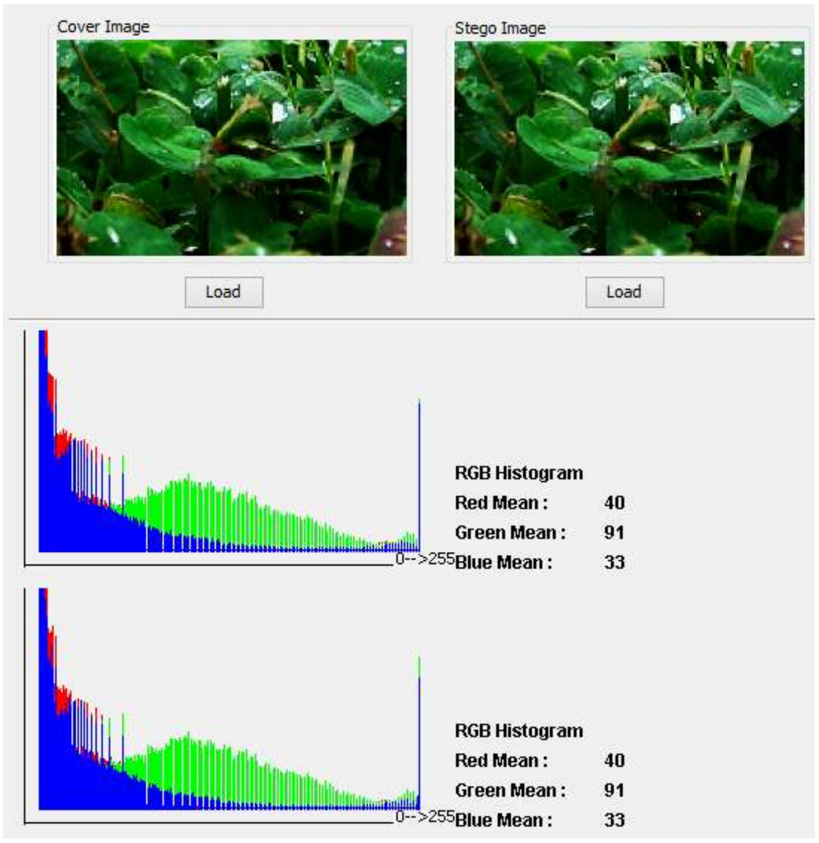
Table 4.3: Additional Security Support

Payload	**Proposed Technique**	**Singh *et al.*, 2014 Technique**
Encryption	Yes	Yes
Randomized	Yes	No

From the table, it can be deduced that proposed approach supports both encryption and randomization whereas Singh *et al.*, (2014) only supports encryption.

4.4.4 Statistical Analysis using Histogram

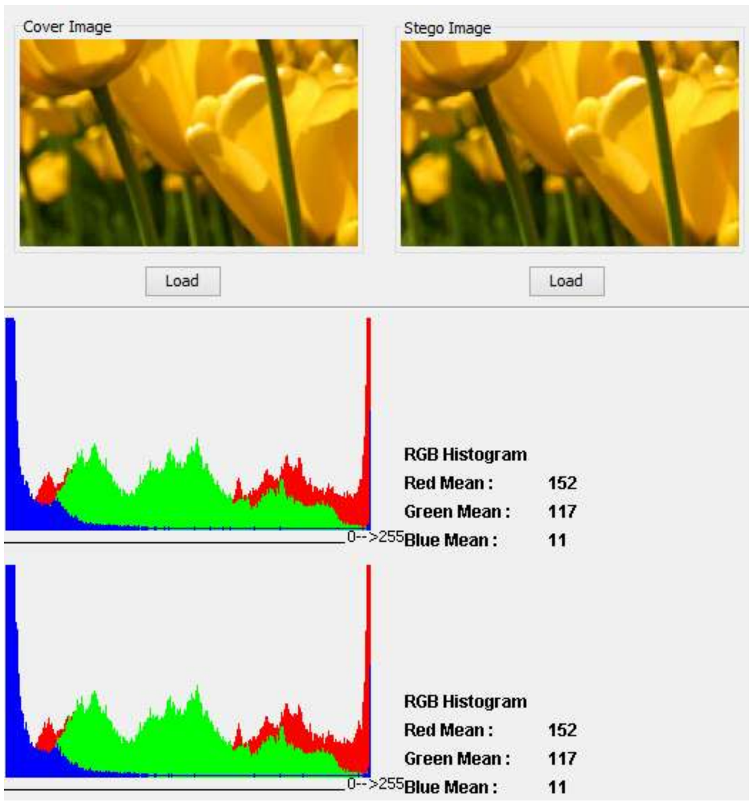
One of the best ways of finding out a good steganography technique is by analyzing the histogram of all stego images and comparing them with original one. Histogram represents the number of pixels that have colors in each of a fixed list of color ranges that span the image's color space, the set of all possible color mixture. Comparing the histogram of the original channels, before and after embedding can give a clear idea of the security of steganographic algorithm used. Figures 4.10 - Figure 4.15 show snapshots of the Histogram of both cover and stego images. From the figures, it could be observed that difference between the cover and stego images histogram are negligible.



RGB Histogram
Red Mean : 40
Green Mean : 91
Blue Mean : 33

RGB Histogram
Red Mean : 40
Green Mean : 91
Blue Mean : 33

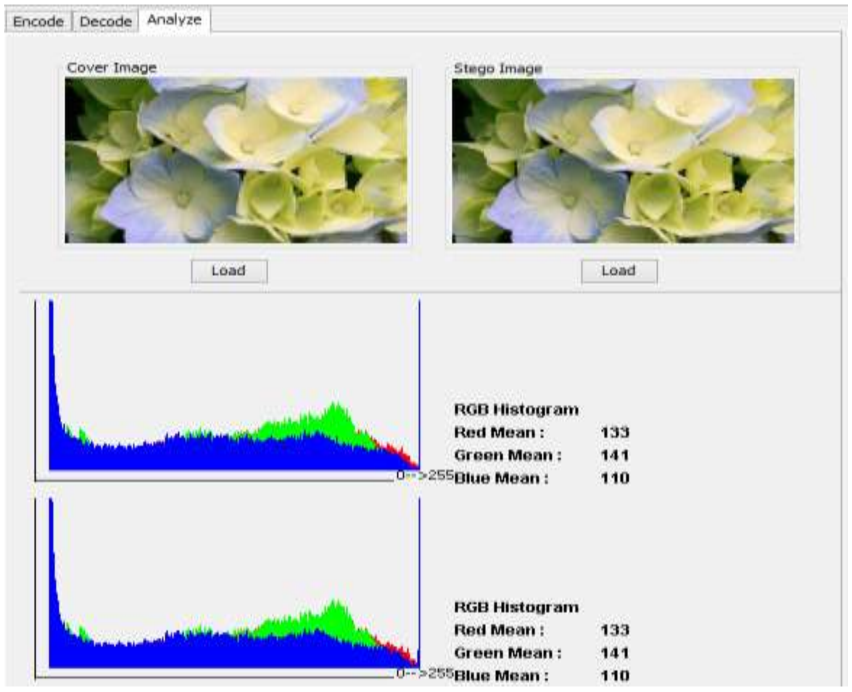
Figure 4.10: Histogram of Clover.bmp



RGB Histogram
Red Mean : 152
Green Mean : 117
Blue Mean : 11

RGB Histogram
Red Mean : 152
Green Mean : 117
Blue Mean : 11

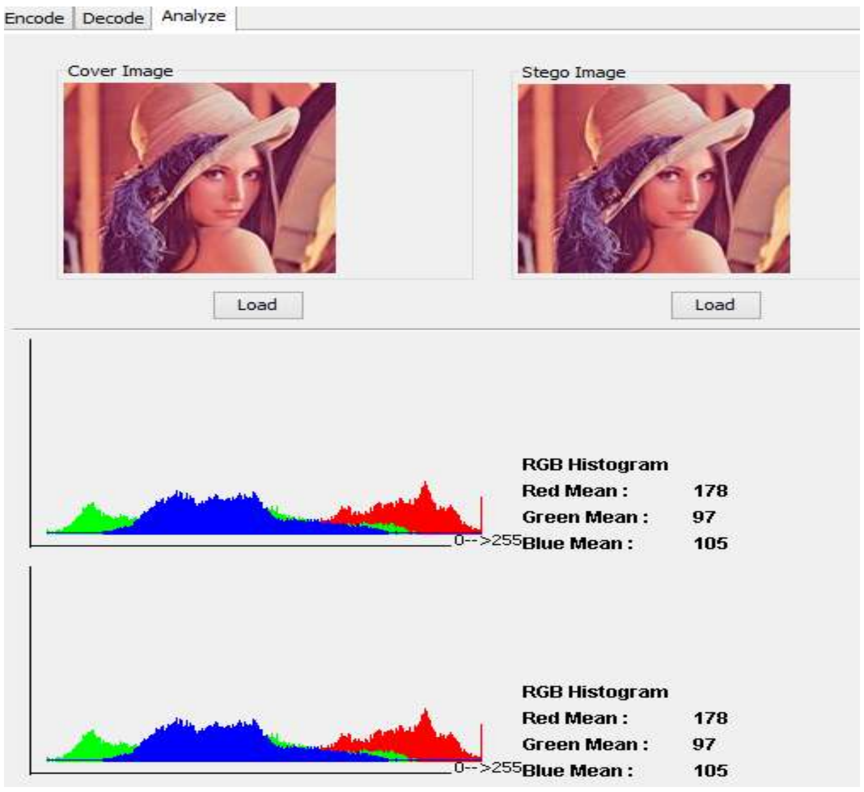
Figure 4.11: Histogram of tulip.bmp



RGB Histogram
Red Mean : 133
Green Mean : 141
Blue Mean : 110

RGB Histogram
Red Mean : 133
Green Mean : 141
Blue Mean : 110

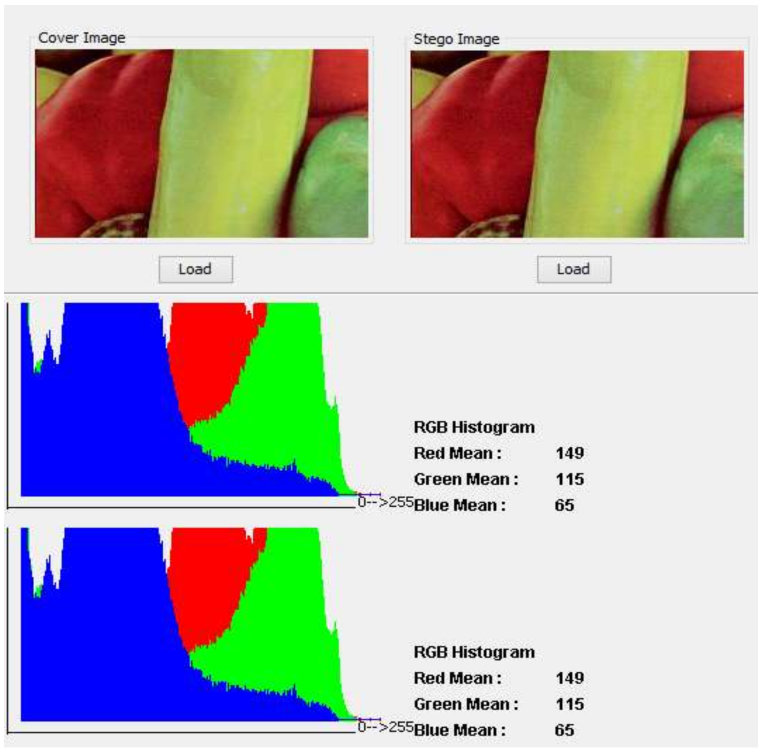
Figure 4.12: Histogram of flower.bmp



RGB Histogram
Red Mean : 178
Green Mean : 97
Blue Mean : 105

RGB Histogram
Red Mean : 178
Green Mean : 97
Blue Mean : 105

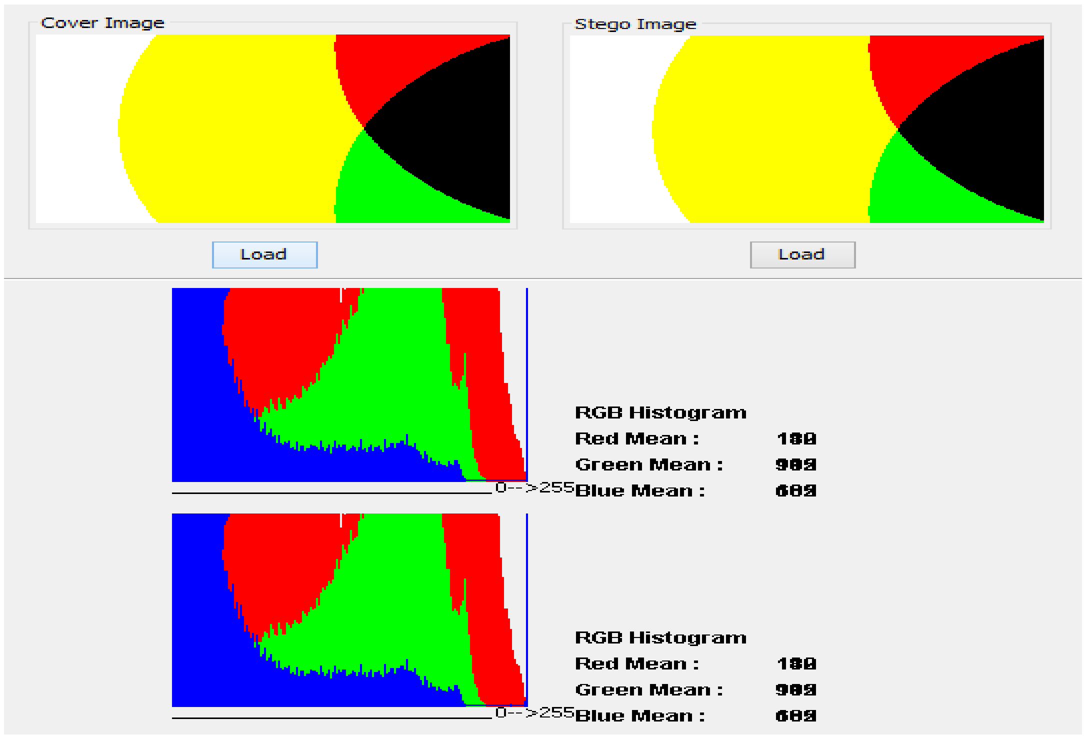
Figure 4.13: Histogram of Lena.png



RGB Histogram
Red Mean : 149
Green Mean : 115
Blue Mean : 65

RGB Histogram
Red Mean : 149
Green Mean : 115
Blue Mean : 65

Figure 4.14: Histogram of peppers.png



```
RGB Histogram
Red Mean :      180
Green Mean :    982
Blue Mean :     682

RGB Histogram
Red Mean :      180
Green Mean :    982
Blue Mean :     682
```

Figure 4.15: Histogram of colors.png

4.4.5 Steganalytic Test

The accuracy and security of proposed technique has been tested on an image pool of 12 lossless images, where 6 of them were stego images (images with hidden data) and other 6 are cover images (images without hidden data) using stegExpose tool. Cover images and stego images are differentiated with the file names, images with _stego are the ones with hidden message. The tool deploys four different steganalysis algorithms namely Primary sets, Chi Square, Sample Pairs and RS analysis and takes the (Fusion) mean of results of image file. A mean of 0.25 is assume as the threshold value to determine if an image is clean (false) or has secret message (true). The message size is ignored in cases of clean files. The result presented in table 4.4 shows all the images return false. This implies that the information leakage in the proposed steganographic process is negligible and the algorithm is secure against the steganalysis attack.

Table 4.4: Steganalytic report for sample images

File name	Above stego threshold ?	Secret message size (ignore for clean files)	Primary Sets	Chi Square	Sample Pairs	RS analysis	Fusion (mean)
clovers.png	FALSE	4092	0.049642	Null	Null	0.077664	0.063653
clovers_stego.png	FALSE	4092	0.049494	Null	Null	0.07683	0.060662
colors.png	FALSE	2	0.002442	0	0.002521	0.002521	0.001871
Colors_stegos.png	FALSE	2	0.002442	0	0.002529	0.002676	0.001909
flowers.bmp	FALSE	5649	0.066009	Null	0.01656	0.08448	0.079016
Flower_stego.bmp	FALSE	5650	0.066004	Null	0.01765	0.080651	0.079217
lena.bmp	FALSE	8163	0.076535	Null	0.011556	0.007296	0.051681
Lena_stegos.bmp	FALSE	8163	0.076352	Null	0.015487	0.008062	0.051685
peppes.png	FALSE	4967	0	Null	0.022764	0.075472	0.027659
Pepper_stegos.png	FALSE	4967	0	Null	0.022727	0.075472	0.027659
tulips.bmp	FALSE	5848	0.0586	Null	Null	0.113284	0.085942
Tulip_stego.bmp	FALSE	5848	0.0586	Null	Null	0.113283	0.085943

CHAPTER 5

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

This dissertation presented a Crypto-Steganographic technique that adequately enhanced interpersonal private communication by modifying the known LSB image steganographic technique. The approach encrypts message before applying the proposed LSB algorithm to ensure data is adequately protected against attack. Six different types of lossless image files ranging from 69KB to 90KB were studied under message of varying length. The metrics use to analyze the images showed security of the system is guaranteed. Experimental results shows that the proposed technique is able to improved communication of sensitive information at minimum distortion of the cover image therefore more secured than other work in the literature considered. The results show our algorithm out performs Singh *et al.,* (2014) algorithm in some performance metric. Steganalysis tools could not identify the presence of image in the stego images also.

5.2 Conclusion

The dissertation introduced the concept of combination of cryptography and steganography. It also proposed a new algorithm to overcome steganalysis. The proposed method provided a higher similarity between the cover and stego image that also yields a better imperceptibility. As per the results obtained, steganography when combined with encryption provides a secured means of secret communication between two parties.

5.3 Contribution to knowledge

This study contributes new knowledge regarding Random Image Steganographic LSB and the application of the technique to secure communication. This research will promote the use of public images for secure interpersonal communication even when using an unsecure channel like the Internet.

5.4 Recommendation

The system should be adopted by intelligence unit of the Nigerian arm force for effective communication to fight insurgencies. Future research should focus on investigating the algorithm under different domain like in transform domain. This however, could further provide improve robustness and tamper resistance to the stego image.

REFERENCES

Almohammad, A. (2010). Steganography – Based Reliable and Secret Communication: Improving Steganographic Capacity and Imperceptibility. (PHD Thesis) Brunel University, 1 - 135.

Al-Othmani, A., Manaf, A.A. and Zeki, A.M., (2012). A Survey on Steganography Techniques in Real Time Audio Signals and Evaluation. *International Journal of Computer Science 9*(1), 55 – 63.

Al-Shatanawi, O. M. and Emam, N. N. (2015). A New Image Steganography Algorithm Based on MLSB Method with Random Pixels Selection, *International Journal of Network Security & Its Applications (IJNSA)*, *7*(2), 37 – 53.

Anshit, A. (2014). Stretching the Limits of Image Steganography. *International Journal of Scientific & Engineering Research*, *5*(2), 1253 – 1256.

Amin, M. M, Salleh, S. I., Katmin, M. R. and Shamsuddin, M. Z. (2003). Information Hiding using Steganography. *Proceedings on National Conference of Telecommunication* Technology (pp. 234 – 238), Shah Alam, Malaysia, IEEE ISBN 0-7803-7773.

Atallah, A. M. (2012). A New Method in Image Steganography with Improved Image Quality. *Journal of Applied Mathematical Sciences*, *3*(5), 3907 - 3915.

Becket, B. (1988). *Introduction to Cryptology*. (pp 55 - 60) Blackwell Scientific publications, London.

Boehm, Benedikt (2014) StegExpose: A Tool for Detecting LSB Steganography (Master Thesis). School of Computing, University of Kent, England, 1 - 17.

Challita, K. and Farhat, H. (2011). Combining Steganography and Cryptography: New Directions. *International Journal on New Computer Architectures and Their Applications (IJNCAA) , 1*(1), 199-208.

Chander K., Kaur, P and Chanda A. (2008). Biometric Security using Steganography. *International Journal of Security, 2*(1), 11 - 18.

Chandramouli, R., Kharrazi, M. and Memon, N. (2003). Image Steganography and Steganalysis: Concepts and Practice, *Proceedings of the 2nd International Workshop on Digital Watermarkin*g, (pp. 99 - 106) Malashiya.

Cheddad, A. Condell, K. Curran, K. and Mc Kevitt, P. (2010). Digital Image Steganography: Survey and Analyses of Current Methods. *Signal Processing Journal, 90(3),* 727 - 752

Churchhouse, R. (2002). *Codes and Ciphers:* Julius Ceasar, the Enigma and the Internet. Cambridge: Cambridge University Press (pp 88).

Dagar, S. (2014). Highly Randomized Image Steganography using Secret Keys. *IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE) New Jersey*, pp 5 - 15.

Davison, A. (2009). *Java Programming Techniques for Games. Java Art*. National Academy Press, Newyork, pp. 99.

Delforouzi, A. and Pooyan, M. (2009). Adaptive and Efficient Image Data Hiding Method in Temporal Domain. Proceeding of the 7th *International Conference on Information, Communications and Signal Processing (ICICS).* New Orlean, pp 2 - 67.

Deepesh, R. and Bhandari, V. (2013). A Steganography Technique for Hiding Image in an Image using LSB Method for 24 Bit Color Image. *International Journal of Computer Applications* 64(20), 15 - 19.

Duggan, M. (2013). Photo and Video Sharing Grow Online, Retrieved January 16, 2016, from http://www.pewinternet.org.

Eggers, E., Salter, J and Shieneir, W. (2002). A Communication Approach to Image Steganography, Security and Watermarking of Multimedia Contents IV. San Jose, California: *Proceedings of SPIE* , 22 - 30.

Forczmański, P., and Węgrzyn, M. (2008). Virtual Steganographic Laboratory for Digital Images. *In Information Systems Architecture and Technology*: Information Systems and Computer Communication Networks, pp. 163–174.

Fridrich, J. and Goljan, M. (2012). Practical Steganalysis of Digital Images – State of the Art. http://www.ssie.binghamton.edu/fridrich. pp 1 -13.

Ghosal, S. K. (2010). An Enhanced, Secure and Comprehensive Data Hiding Approach Using 24 Bit Color Images. (Master's dissertation), Jadavpur University, pp. 1 - 120.

Gonzalez, R. C and Woods, R. E. (2002). *Digital Image Processing.* (2nd Ed.) Prentice Hall, INC London, pp. 340

Isbell, R. A (2012). Steganograpy: Hidden menace or Hidden Saviour. *White Paper on Computer Security Mechanism*, Indiana press, India, pp 20 - 30.

Johnson, N. F. and Jajodia, S. (1999). Exploring Steganography: Seeing the Unseen. In *Computer Science and application Jounal, 3*(1), 26-34.

Kaur, J. and Kumar, S (2011).”Study and Analysis of Various Image Steganography Techniques” *Proc. International Journal of Computer Science and Technology 2*(3), pp. 55 – 64.

Kaushal, A. and Chaudhary, V. (2013): Secured Image Steganography using Different Transform Domain. *International Journal of Computer Applications*. *7*(2), pp. 24 -28

Koppola, R. R. (2009). A high capacity data-hiding scheme in LSB - based image Steganography. (Master's thesis) University of Akron, Ohio, USA. retrieved from http://www.scribd.com/doc/48764974/Steganography-Data-hiding-using-LSB-algorithm

Laskar, S. A. and Hemachandran, K. (2013). Steganography Based on Random Pixel Selection for Effiecient Data Hiding. *International Journal of Computer Engineering and Technology (IJECET)* , *4*(2), pp 31-44.

Marvel K. and Retter, F. (2000). Retrieved October 13, 2015 from U.S. Army Research Laboratory, http://www.eecis.udel.edu/m˜arvel/marvel isita2000.pdf.

Mobile Statistics, Retrieved December 30, 2015 from http://www.mobilestatistics.com

National Institute of Standards and Technology (1995) *An Introduction to Computer Security*: The Computer Security Handbook, (Sp. publication, pp. 800 -812) London.

Nivedhitha R. and Meyyappan T. (2012). Image Security using Steganography and Cryptographic Techniques. *International Journal Engineering Trends and Technology*, *3*(3), 366 - 371.

Philjon, L. T. and Rao, V. (2011). Metamorphic Cryptography A Paradox between Cryptography and Steganography Using Dynamic Encryption. *International Electrical and Electronic Engineering -International Conference on Recent Trends in Information Technology, ICRTIT, 9*(2), 80 - 87.

Prashanti, G. and Sandhyarani, K. (2014). Secure Communication using Modified DES with Steganography. *International Journal of Advanced Trends in Computer Science and Engineering*. *3*(5), 515 – 518.

Rababah, A. and Abdulgader, U. 2011. New technique for hiding data in audio file. *International Journal of Computer Science Network Security*, *11(4),* 12 – 19.

Rahmani, K. I. Arora, K. and Pal, N. (2014). A Crypto-Steganography: A Survey. *International Journal of Advanced Computer Science and Applications, (IJACSA) , 5(3),* 149-155.

Saini, R., Chugh, G and Yadav, R. (2014). A New Image Steganographic Approach Based on Mod Factor for RGB Images. *International Journal of Signal Processing, Image Processing and Pattern Recognition, 7*(3), 27-44.

Seberry, J . and Pierprzyk, J. (1989). *"An Introduction to Computer Security".* Prentice Hall-Advances in Computer Science Series, pp 123 - 134.

Selvaraj, D. (2014). *Development of a Secure Communication System based on Steganongraphy for Mobile Devices.* (Master's thesis) Frankfurt University of Applied Sciences Retrieved December 13, 2015 from http://baunvorlesungen.appspot.com/Abschlussarbeiten/Dasarathan_Selvaraj_Masterthesis_2014.pdf

Shukla, C. P., Chadha, R. S. and Kumar, A. (2014). Enhance Security in Steganography with cryptography. *International Journal of Advanced Research in Computer and Communication Engineering, 3*(2), pp. 5696 – 5699.

Singh, B., Kataria, S., Kumar T., and Shekhawat, N. S. (2014). A Steganography Algorithm for Hiding Secret Message inside Image using Random Key. *International Journal of Engineering Research & Technology (IJERT*), *3*(12), 902 – 906.

Simmons, G. J. (1983). The Prisoners' Problem and the Subliminal Channel. *Proceedings of Crypto'83,* Plenum Press, pp 51 - 57.

Stallings, W. (2011). *Cryptography and Network Security: Principles and Practice* (5th ed.), New York: Prentice Hall, 79.

Sujay, N. and Prasad, G. (2010). Two New Approaches For Secured Image Steganography using Cryptographic Techniques and Type Conversions. *Signal and Image Processing: An International Journal (SIPIJ)* , *1*(2), 60-73.

Suri, S., Joshi, H., Mincoha, V. and Tyagi, A. (2014). Comparative Analysis of Steganography for Coloured Images. *International Journal of Computer Sciences and Engineering, 2(4),* 180 – 184. www.ijcaonline.org

Tiwari, G., Yadav, K. and Mishra, M. (2013). A Survey on Digital Image Steganography and Steganalysis. *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)* 2278-2834, *8*(1), 56-60 www.iosrjournals.org

Venkatraman, A. and Paprzycki, M. (2004). Significance of Steganography on Data Security. *Proceedings of International Conference on Information Technology: Coding and Computing. ITCC,* 347 - 351.

Walia, E., Jain, P. and Navdeep, A. (2010). An Analysis of LSB and DCT based Steganograghy, *Global Journal of Computer Science and Technology*, *10*(1), 4 - 8.

Wang R. Z, Lin, Chi-Fang, S. and Lin, J. (2001). Image Hiding by Optimal LSB Substitution and Genetic Algorithm. *Pattern Recognition Society* , *34*(3), 671 - 683.

JAVA IMPLEMENTATION

Encryption

```
package steganography.lib;

import java.awt.image.BufferedImage;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.Base64.Decoder;
import java.util.Base64.Encoder;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

/**
 *
 * @author OwoidigheAbasi
 */
public class Encryption {


   public static String Encrypt(String message, BufferedImage coverImage) throws Exception
   {
      if(message.length() == 0 || coverImage == null) throw new Exception("Empty or null
argument passed");
      byte[] messageByte = Operation.ConvertMessageToByte(message);
      int[] greenPixels = Operation.getGreenComponent(coverImage,messageByte.length);
      byte[] cipherByte = new byte[messageByte.length];
      for(int i = 0; i < messageByte.length; i++)
      {
         if(greenPixels[i] > 127) {
            continue;
         }
         cipherByte[i] = (byte)(messageByte[i] ^ greenPixels[i % greenPixels.length]);
      }
     return new String(cipherByte);
   }

   public static String Encrypt1(String message) throws Exception
   {
      String salt = "StegoSalt";
      String password = "password";
      message = salt+message;
      SecretKeySpec key = new SecretKeySpec(Operation.padPassword(password), "AES");
```

```
        Cipher c = Cipher.getInstance("AES");
        c.init(Cipher.ENCRYPT_MODE, key);
        byte[] encVal = c.doFinal(message.getBytes());
        Encoder encoder = Base64.getEncoder();
        String encrypted=encoder.encodeToString(encVal);

        encoder.encodeToString(encVal);
        return encrypted;
    }

    public static String Decrypt(String message) throws Exception
    {
        String password = "password";
        String salt;
        Cipher c;
        String decryptedValue = null;
        try {
        c = Cipher.getInstance("AES");
        SecretKeySpec key = new SecretKeySpec(Operation.padPassword(password), "AES");
        c.init(Cipher.DECRYPT_MODE, key);
        Decoder decoder = Base64.getDecoder();
        byte[] decordedValue = decoder.decode(message.getBytes());
        byte[] decValue;
        decValue = c.doFinal(decordedValue);
        decryptedValue = new String(decValue);
        salt = decryptedValue.substring(0, 9);
        if(salt.equals("StegoSalt"))
        {
        decryptedValue = decryptedValue.substring(9);
        }else{
        throw new Exception("Wrong Passoword");
        } }
        catch (Exception e) {e.printStackTrace();
        throw new Exception("Wrong Passoword");
        }
        return decryptedValue;
            }

    private Encryption() {
    }

}
```
Embedding
```
public class Embeding {

    public static BufferedImage Embed(String cipher, BufferedImage coverImage){
        int[] redAndBlueComponents = Operation.getRedAndBlueComponent(coverImage);
        byte[] messagebyte = cipher.getBytes();
        int Ki = quotientOfCipherAndEight(cipher.length());
        int Ai = SumOfKiDigits(Ki);
        int Mi = AiModuloEight(Ai);
        for(int i = 0; i < messagebyte.length; i++){
            int add = messagebyte[i];
            for(int bit=7; bit>=0; --bit){
              // System.out.print(redAndBlueComponents[Ki]);
                int currentBit = (add >>> bit) & 1;
                if(currentBit == 1) {
                    redAndBlueComponents[Ki] = Operation.SetBit(redAndBlueComponents[Ki], Mi);
                } else {
                    redAndBlueComponents[Ki]   =   Operation.UnSetBit(redAndBlueComponents[Ki],
Mi);
                }

                Ki += Mi;
                Ai = SumOfKiDigits(Ki);
                Mi = AiModuloEight(Ai);
            }
        }
        System.out.println(redAndBlueComponents[redAndBlueComponents.length - 1]);
        redAndBlueComponents[redAndBlueComponents.length - 1] = messagebyte.length;
        System.out.println(redAndBlueComponents[redAndBlueComponents.length - 1]);
        coverImage = Operation.WriteCoverImage(coverImage,redAndBlueComponents);
        System.out.println(redAndBlueComponents[redAndBlueComponents.length - 1]);
        return coverImage;
    }

    public static String Extract(BufferedImage coverImage){
        int[] redAndBlueComponents = Operation.getRedAndBlueComponent(coverImage);
        int messageSize = redAndBlueComponents[redAndBlueComponents.length - 1];
        System.out.println(messageSize); //debugging statement
        byte[] secretMessage = new byte[messageSize];
        int Ki = quotientOfCipherAndEight(messageSize);
        int Ai = SumOfKiDigits(Ki);
        int Mi = AiModuloEight(Ai);
        for(int i = 0; i < messageSize; i++)
        {
            for(int j=0; j<8; ++j)
            {
              int position = getBitPosition(Mi);
              //System.out.print(redAndBlueComponents[Ki] + " ");
              secretMessage[i]  =  (byte)((secretMessage[i]  <<  1)  |  (redAndBlueComponents[Ki]  >>
position) & 1);
```

```
            Ki += Mi;
            Ai = SumOfKiDigits(Ki);
            Mi = AiModuloEight(Ai);
           }

           //System.out.println();
        }
        return new String(secretMessage);
    }
    private static int quotientOfCipherAndEight(int length) {
        return length/8;
    }

    public static int SumOfKiDigits(int Ki) {
       String[] digits = Integer.toString(Ki).split("(?<=.)");
       int sum = 0;
       for(String str : digits)
           sum += Integer.parseInt(str);

       return sum;
    }

    private static int AiModuloEight(int Ai) {
        int remainder =  Ai%8;
        return remainder == 0 ? 7:remainder;
    }

    private static int getBitPosition(int Mi) {
        switch(Mi)
        {
          case 0:
          case 7:
          case 6:
          case 5:
          case 4:
               return 0;
          default:
              return Mi;
        }
    }

    private Embeding() {
    }
}
```
<u>Histogram determination</u>

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;


public class Histogram {
    BufferedImage image;
    int[] RedFrequency;
    int[] GreenFrequency;
    int[] BlueFrequency;
    int redColor;
    int greenColor;
    int blueColor;
    int rgb;
    long sumOfRedFrequency;
    long sumOfBlueFrequency;
    long sumOfGreenFrequency;
    int w;
    int h;
    public Histogram(BufferedImage image,int h, int w){
        this.h = h;
        this.w = w;
        this.image = image;
    }
    public void paint(Graphics g) {
        Graphics2D g2D=(Graphics2D)g;
        RedFrequency=new int[256];
        GreenFrequency=new int[256];
        BlueFrequency=new int[256];
        int width = image.getWidth();
        int height = image.getHeight();
        for(int i=0;i<width;i++) {
            for(int j=0;j<height;j++) {

                rgb =image.getRGB(i,j);
                redColor = (rgb >> 16) & 0xff;
                greenColor = (rgb >> 8) & 0xff;
                blueColor = (rgb) & 0xff;
                RedFrequency[redColor]++;
                GreenFrequency[greenColor]++;
                BlueFrequency[blueColor]++;
            }
        }

        int iw=w/2;
        int ih=h/2;



       for(int i=0;i<256;i++) {

            g2D.setColor(Color.RED);
    g2D.drawLine(20+i,h-20,20+i,h-(RedFrequency[i]/10)-20 );

             g2D.setColor(Color.GREEN);
            g2D.drawLine(20+i,h-20,20+i,h-(GreenFrequency[i]/10)-20);
            g2D.setColor(Color.BLUE);
            g2D.drawLine(20+i,h-20,20+i,h-(BlueFrequency[i]/10)-20);

            sumOfRedFrequency+=RedFrequency[i]*i;
            sumOfGreenFrequency+=GreenFrequency[i]*i;
            sumOfBlueFrequency+=BlueFrequency[i]*i;

        }
         g.setColor(Color.BLACK);
        g2D.drawLine(10,h-10,10,h-256);
        g2D.drawLine(10,h-10,256,h-10);
        g2D.drawString("0-->255",260,h-10);
        g2D.setFont(new Font("Aril",Font.BOLD,12));

        g2D.drawString("RGB Histogram",300,100);
        g2D.drawString("Red Mean :",300,120);
        g2D.drawString(Long.toString( sumOfRedFrequency/(height*width)),400,120);
        g2D.drawString("Green Mean :",300,140);
        g2D.drawString(Long.toString(( sumOfGreenFrequency/(height*width))),400,140);
        g2D.drawString("Blue Mean :",300,160);
        g2D.drawString(Long.toString(( sumOfBlueFrequency/(height*width))),400,160);
    }


}
```

Performance matrix

```
import java.awt.image.BufferedImage;
import java.awt.image.Raster;

/**
 *
 */
```

```
public class PerformanceMatrix {
   private final BufferedImage coverImage;
   private final BufferedImage stegoImage;
   private double mse;
   private double psnr;
   public PerformanceMatrix(BufferedImage coverImage, BufferedImage stegoImage)throws
Exception{
       this.coverImage = coverImage;
       this.stegoImage = stegoImage;
       if(!CheckImage()) throw new Exception("Error: Cover image and stego image not the
same.");
   }

   private boolean CheckImage()
   {
       return (coverImage.getHeight() == stegoImage.getHeight()) && (coverImage.getWidth()
== stegoImage.getWidth());
   }

  public double CalculateMSE() {
      int sumOfSquares = 0;
      int height = coverImage.getHeight();
      int width = coverImage.getWidth();

      for(int row = 0; row < height; row++){
          for(int col = 0; col < width; col++){
              int coverImagePixel = coverImage.getRGB(col, row);
              int stegoImagePixel = stegoImage.getRGB(col, row);
              int pixelDiff = stegoImagePixel - coverImagePixel;
              sumOfSquares += (pixelDiff * pixelDiff);
          }
      }
      mse = sumOfSquares/(double)(height*width);
      return  mse;
  }

  public double calculatePSNR(){
      double x = Math.pow(255, 2)/mse;
      psnr = 10.0 * Math.log10(x);
      return psnr;
  }

          public static double printPSNR(BufferedImage im1, BufferedImage im2) {
                    assert(
                              im1.getType() == im2.getType()
                                        && im1.getHeight() == im2.getHeight()
```

```
                        && im1.getWidth() == im2.getWidth());

                double mse = 0;
                int width = im1.getWidth();
                int height = im1.getHeight();
                Raster r1 = im1.getRaster();
                Raster r2 = im2.getRaster();
                for (int j = 0; j < height; j++)
                        for (int i = 0; i < width; i++)
                                mse
                                        += Math.pow(r1.getSample(i, j, 0) - r2.getSample(i, j, 0),
2);
                mse /= (double) (width * height);
                System.err.println("MSE = " + mse);
                int maxVal = 255;
                double x = Math.pow(maxVal, 2) / mse;
                double psnr = 10.0 * logbase10(x);
                System.err.println("PSNR = " + psnr);
                return psnr;
        }
     public static double logbase10(double x) {
                return Math.log(x) / Math.log(10);
        }
}
```