Deanship of Graduate Studies

Al-Quds University

Improving LSB Technique for Hiding Information in a Digital Image

Nizar Mohammad Mostafa Shehab

M.Sc. Thesis

Jerusalem – Palestine

1428 / 2007

Improving LSB Technique for Hiding Information in a Digital Image

Prepared By

Nizar Mohammad Mostafa Shehab

B.Sc in Computer Science - Al-Quds University Jerusalem

Supervisor: Dr. Badie’ Sartawi

A Thesis Submitted in Partial Fulfilment of the Requirements

for the Degree of Master of Computer Science from

Department of Computer Science/College of Science

Al-Quds University

1428 / 2007

Al-Quds University
Deanship of Graduate Studies
Master Program / Department of Computer Science

Thesis Approval

Improving LSB Technique for Hiding Information in a Digital Image

Prepared By: Nizar Mohammad Mostafa Shehab
Registration No: 20311840

Supervisor: Dr. Badie' Sartawi

Master thesis submitted and accepted, Date: 29'th of December 2007
The names and signatures of the examining committee members are as follows:

1- Head of Committee: Dr. Badie' Sartawi	Signature:
2- Internal Examiner: Dr. Raed Zaghal	Signature:
3- External Examiner: Dr. Nasser Hamad	Signature:
4- Committee Member:	Signature:

Jerusalem – Palestine

1428 / 2007

Dedication

To my Parents and Family

Nizar Mohammad Mostafa Shehab

Declaration

I Certify that this thesis, submitted for the degree of Master, is the result of my own research, except where otherwise acknowledged, and that this thesis (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed:

Nizar Mohammad Mostafa Shehab

Date:

Acknowledgments

I owe special thanks to my supervisor, Dr. Badie’ Sartawi for his invaluable cooperation, encouragement and guidance. I do appreciate his patience and prompt answers to all my inquiries during the writing of this thesis. Great thanks are also due to Dr. Nasser Hamad for his great assistance in information theory and entropy literature. I appreciate all his great efforts and unlimited cooperation. I should not forget my best friend Mr. Nash'at Abu Ghalieh for his continuous assistance in MATLAB. I also extend my thanks to Al-Quds University, from where I obtained my B.Sc in Computer Science, for giving me the opportunity to pursue my Master’s study. Last, but certainly not least, I thank my parents, wife and family for their continuous support and for tolerating my preoccupations.

Improving LSB Technique for Hiding Information in a Digital Image

Abstract

The study is concerned with hiding information into a digital media. In this study, an English encrypted text is used to be hidden into a digital grey-scale image. The purpose of this study is to embed a maximum text data into the image with lower Bit Error Rates (BER). In order to do that, three algorithms were used. These were: random, odd-even and sequential algorithms.

In this study, we used the hamming distance measurements to calculate the BER. The hamming distance measurement is used for each algorithm and helps us to pick up the least BER of the three algorithms. The study used the Flipping Embedded Text technique (FET) to enhance the LSB technique. FET technique always achieves BER less than 0.5 of the least significant bits of the image pixels.

The study also dealt with the entropy-based measurements, which are used to indicate the best distribution of image bits to be used with a specific text. In the entropy section, the relation between the BER and the entropy difference of both text and image were manipulated.

تحسين تقنية LSB في اخفاء المعلومات في الصورة الرقمية

الملخص

تتناول هذه الدراسة إخفاء معلومات في وسائط رقمية، ولتوضيح ذلك قام الباحث باستخدام نص عادي تم تشفيره وإخفاؤه داخل صورة رمادية (غير ملونة).

تهدف هذه الدراسة إلى تضمين اكبر قدر من المعلومات النصية داخل صورة رمادية مع اقل نسب خطأ بتي. ولتحقيق ذلك استخدمت ثلاث خوارزميات وهي الخوارزمية العشوائية والخوارزمية الفردية الزوجية والخوارزمية التتابعية. كما تم استخدام قياسات البعد الهامي لحساب نسب الخطأ البتي لكل خوارزمية، ليساعدنا على اختيار اقل نسب خطأ بتي لهذه الخوارزميات الثلاث، واعتماد الخوارزمية التي تحقق ذلك.

واستخدمت ايضا تقنية النص المضمن المقلوب لتعزيز وتطوير تقنية LSB، حيث نحصل دائما على خطا نسبي بتي اقل من 50% من البتات الاخيرة الممثلة للنقط الضوئية التي تتكون منها الصورة.
كما تناولت الدراسة القياسات التي تستند الى معدل كمية المعلومات التي تحويها شيفرة او رسالة معينة، وذلك لتحديد افضل توزيع للبتات التي تتكون منها الصورة والتي يمكنها ان تلائم نصا من اجل تضمينه داخل الصورة لنضمن عدد تشويهات اقل للصورة، وفي الجزء الذي يتحدث عن معدل كمية المعلومات تبين انه بالامكان التحكم بالعلاقة بين نسب الخطأ البتي والفرق بين معدل كمية المعلومات للنص والصورة.

Definitions and Abbreviations

MATLAB	Matrix Laboratory
LSB	Least Significant Bit
BER	Bit Error Rate(s)
FET	Flipping Embedded Text
BC	Before Christ
Fig.	Figure
Steganography	The arts and science of hiding information
Stego-Object	Steganography Object
Stego-Key	Secrete Steganography Key
Cryptography	Concealing the content of the message by scrambling
Steganalysis	The process of detecting steganography
WWI	World War One
WWII	World War Two
ASCII	American Standard Code for Information Interchange
HVS	Human Visual System
p.d.f	Probability density function
p.m.f	Probability mass function
DMS	Discrete Memmoryless Source
RGB	Red, Green and Blue
GIF	Graphic Interchange Format
XOR	Logical Operation $\oplus$

BSC	Binary Symmetric Channel
KB	Kilo Byte
Diff.	Difference
Vs.	Versus

Table of Contents

Declaration.......v

Acknowledgments....... vi

Abstract....... vii

الملخص....... viii

Definitions and Abbreviations.......ix

Table of Contents.......xi

List of Figures....... xiii

List of Tables....... xv

Chapter 1.......1

Introduction.......1

1.1 Steganography.......1

1.2 Steganography general model.......2

1.3 Steganalysis.......4

1.4 Research Definition and Objectives.......5

1.5 Synopsis of Thesis.......7

Chapter 2.......9

Background and Literature Review.......9

2.1 Background.......9

2.2 Literature Review Analysis and Comparisons....... 14

2.2.1 Spatial Domain Strategy....... 15

2.2.2 Transform Domain....... 19

Chapter 3....... 22

System Model and Analysis 22

3.1 System Model 22

3.1.1 Text Model 27

3.1.2 Image Model 28

3.2 System Capacity Calculations 29

3.3 System Analysis 31

Chapter 4 40

System Performance Based on Hamming Distance Measurement 40

4.1 Hamming Distance Measurement 40

4.1.1 Sequential Algorithm 42

4.1.2 Odd-Even Algorithm 42

4.1.3 Random Walk Algorithm 43

Chapter 5 45

System Performance Based on Entropy Function Measurement 45

5.1 The Entropy Function 45

Chapter 6 50

Simulation and Experimental Results 50

6.1 Hamming Distance-Based Measurements 50

6.2 Entropy-Based Measurements 55

Chapter 7 62

Conclusions 62

7.1 Concluding Main Results 62

7.2 Open Problems and Future Work 62

Bibliography 64

Appendix A 68

List of Figures

Number *Page*

Figure 1: General model of steganography 3

Figure 2: Cover and stego images seem the same 12

Figure 3: Our proposed model 23

Figure 4: Encryption method diagram. 25

Figure 5: Binary entropy function 28

Figure 6: (a) Vector **t**, (b) Vector **q,** and (c) Vector **s** 32

Figure 7: The p.d.f of random binary bits 33

Figure 8: Entropy diff. L and probability diff. β 33

Figure 9: Case 1 in which E_t and E_q are coincide in the same half 34

Figure 10: Case 2 in which E_t and E_q are uncorrelated 35

Figure 11: Case 4 in which E_t and E_q in different halves and no flip needed 36

Figure 12: Case 5 in which $\beta > 0.5$ and the flip is needed 37

Figure 13: Case 6 in which E_t and E_q in different halves and no need to flip E_t 37

Figure 14: Minimizing β by rotating E_t to $\overline{E}_t$ 39

Figure 15: Lenna image 44

Figure 16: The transition process of **t** into **q** to form **s** 46

Figure 17: Cascaded Binary Symmetric Channel (BSC) 46

Figure 18: Text size Vs. BER for sequential algorithm with flip and no flip 52

Figure 19: Text size with odd-even flip and no flip algorithms 53

Figure 20: Text size with random walk flip and no flip algorithms 54

Figure 21: Text size with all flip and no flip algorithms 55

Figure 22: Text size vs. entropy 57

Figure 23: Text size with entropy difference between vector **t** and vector **q** 58

Figure 24: Entropy diff. with BER 60

Figure 25: Entropy diff. of different images with fixed text size and BER 61

List of Tables

Table1: Text size versus BER of the three algorithms 52

Table 2: Text size with the entropy, entropy difference and BER.. 56

Chapter 1

Introduction

1.1 Steganography

There was a need for communication since man was found on earth. People had to communicate with each other using several ways. Sometimes people had to contact with others confidentially in order to pass on secret information to others. In Ancient times, people were sending messages to their partners by human being holding the message. The secret information was the physical media. For example, in 440 BC the King Darius of Susa shaved the head of one of his prisoners and wrote a secrete message on his scalp. When the prisoner's hair grew back, he was sent through the enemy land to the king partner. The king partner did the same method; he shaved the prisoner's hair a gain and extracted the secret message (Bret, 2002).

Mary Queen of Scots used a combination of cryptography[1] and steganography to hide letters. Her letters were hidden in the bunghole of a beer barrel, which freely passed in and out of her prison (Judge 2001).

After that the steganography was used in different way. Romans used invisible inks, which were based on natural substances such as fruit juices and milk (Bret, 2002). The hidden text was heated and its contents were revealed.

In World War I and II, the security of messages was the most important matter to win the war. At that time, the messages were written by invisible ink and sometimes encrypted by simple and easy algorithms; an example of such an algorithm is, a message sent by a German spy during World War II read as “Apparently neutral’s protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.” By taking the second letter of every word the hidden message "Pershing sails from NY June 1" can be retrieved (Hany and Siwei, 2006).

Due to the breakthrough of advanced technology, hence, digital signal processing techniques including all kinds of digital media and the rapid development of the Internet, a vast field has been opened up for steganographic[2] purposes, wherein, one can see the digitized data (digital media) often includes redundant, unnecessary and unnoticed data spaces that can be manipulated to hide messages (Honeyman and Provos, 2003).

[1] Should be explained later

[2] To be defined latter in this Chapter.

The ideas of hiding the messages and keep their securities was improved rapidly, specifically, the messages (confidential information) were subjected to encryption process, scrambled, and/or hidden in other forms.

Steganography is the art of hiding information to prevent the detection of hidden messages by an unauthorized person (Neil F. and Jajodia S. 1998-a). Moreover, the digital information (represented by bits or symbols) might be hidden in any digital object, either text or image. But more suitable digital objects are those of a high degree of redundancy. The redundant bits of an object are those bits that can be altered without the alteration being detected easily (Morkel, Eloff and Olivier 2005). The digital media such as audio and image files comply with the steganographic requirements because they have a large degree of bits redundancy.

1.2 Steganography general model

Steganography is most widely formulated in terms of the “Prisoners’ Problem” which was proposed by (Simmons, 1983) where he formulated a scenario to describe the (Prisoners' Problem). “Alice and Bob are in jail for some crime they committed and are thrown in two different cells. They wish to develop an escape plan; all their communication must go through a warden named Wendy. Her duty was to prevent any suspicious communication between them which enforce Alice and Bob to settle a subliminal channel between them. If she noticed any unusual message; she will place them in solitary confinement.”

In order to generalize the scenario of (Simmons, 1983), we assume a steganography problem where Alice intends to send a secret message *M* to Bob by hiding *M* into a cover object *C* to obtain the stego-object *S*. The stego-object *S* is, then, sent through a warden Wendy channel. The warden Wendy who is assigned to examine all messages exchanged between Alice and Bob can be passive or active. A passive warden, simply, examines the message and determines whether the stego-object *S*, potentially contains a hidden message or not, if so, warden Wendy takes appropriate action, else she lets the message pass without any alteration. An active warden, on the other hand, can alter messages deliberately even though she does not see any trace of a hidden message, in order to foil any secrete communication that can nevertheless be occurred between Alice and Bob. (Kharrazi, Husrev and Memon 2004).

According to the above scenario the framework of the steganography can be formulated in the following figure.

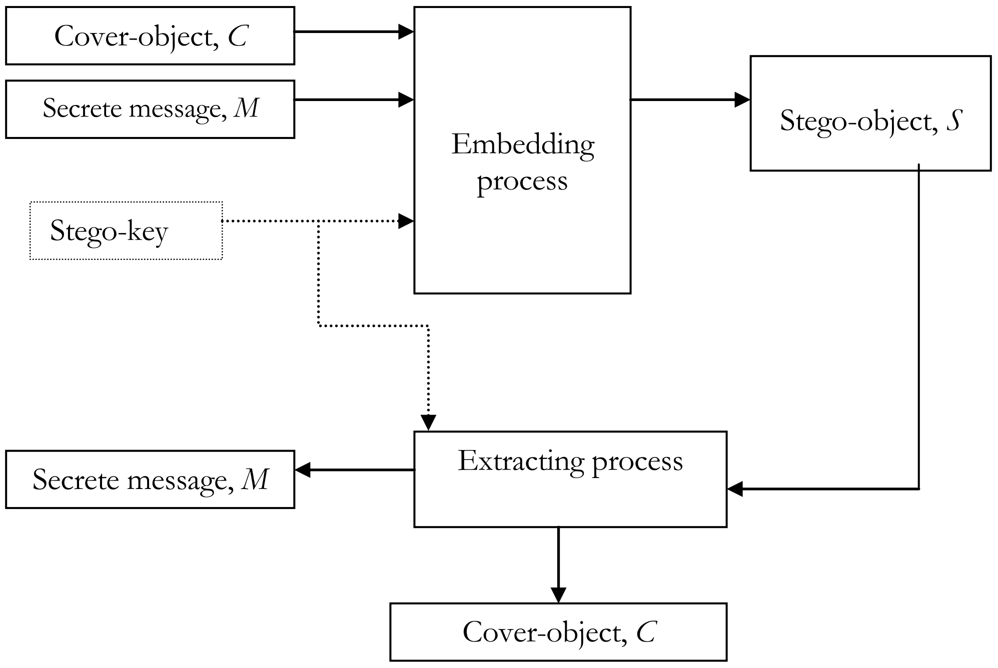

Figure 1: General model of steganography

In our research, the passive warden is considered. That is the stego-object *S* should not contain any suspicious data or should not seem suspicious to the warden Wendy. As a result of generating the stego-object, *S*, the less distortion that the cover object, *C*, might be exposed to, the more increases the security of the stego-system which makes warden Wendy unable to determine whether the cover object contains a secret message or not.

However, the main goal of steganography is to communicate securely in a completely undetectable manner (Neil F. and Jajodia S. 1998-b). Finally, it is worthy to mention that steganography deals with hiding the covert message and not the security of communication link between the two parties (Davidson and Goutam 2004).

In the above scenario, Alice uses some technique to embed a message *M* into a cover object *C*. The technique for embedding the message is unknown to Wendy and shared as secrete between Alice and Bob. The embedding process may depend on a secret key called a stego-key which is used to control the embedding process, namely, where to hide the message *M*, or used as asymmetric key to encrypt the message before it was embedded in the cover object, C.

Wendy should have no knowledge about the stego-key that Alice and Bob share although she is aware of the algorithm that is employed to hide the message.

In our research, the plain text should be encrypted and then embedded into the grey-scale image in a way that didn't increase the suspicious of the attacker, Wendy.

The text that can be read and understood without any special measures is called plaintext or clear text. The method of disguising plaintext in such a way as to hide its substance is called encryption. Encrypting plaintext results in unreadable gibberish called cipher text. One use encryption to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting cipher text to its original plaintext is called decryption (Network Associates, 1990-1998).

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables us to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.

However, steganography is differing from cryptography. The latter attempts to conceal the content of the message by scrambling while the former hides the message rather than change its contents (Vidyasagar, Song and Elizabeth 2005). Even though, the aforementioned techniques can be concatenated together, i.e., it is possible to combine the two techniques using cryptography and then hiding the encrypted message using steganography. The resulting stego-object can be transmitted without revealing that secret information is being exchanged. Furthermore, even if an attacker Wendy were to defeat the steganographic technique and detect the message from the stego-object, she would still require the cryptographic decoding key to decrypt the message. But one should note that in this case the steganographic technique was broken since the duty of an attacker in such technique is to discover a hidden message and not to extract it.

1.3 Steganalysis

The attempt of detecting steganographic messages is known as steganalysis. Theoretically, steganalysis process considered to be success if there is evidence that the stego-object contains a hidden data without actually being able to extract it. Moreover, steganalysis is the practice of attacking steganographic methods by detection, destruction, extraction, or modification of embedded data (Fridrich, Goljan and Soukal 2004).

Referring to the general framework of the steganography, the main goal of steganography is to communicate securely in a completely undetectable manner, i.e., Wendy should not be able to distinguish in any sense between cover-object and stego-object. In this context, steganalysis refers to the body of techniques that are designed to distinguish whether the stego-objects contains a hidden message or not.

As the statistical distribution[1] of the stego-object is closer to the statistical distribution of the cover-object, the warden Wendy cannot decide that the stego-object contains hidden data. In this research, our goal is to decrease the opportunity of Wendy to detect the hidden message *M* embedded in the stego-object *S*. This can be, simply, accomplished by decreasing the changes into the cover-object, *C*, that may occur through steganographic process. We will say that the stego-system is perfectly secure[2] when the stego-object is nearly similar to the cover-object.

1.4 Research Definition and Objectives

Our research is defined by the following problem, we intend to hide a text data into a grey-scale image. The hidden data should be embedded in the image without causing any kind of image degradation. So, the embedded data should be concealed in the Least Significant Bit (LSB) of the image because those bits represent the noisy or redundancy bits in which the changes of any LSB should not harmful the image. The Human Visual System (HVS) can't distinguish between the original image and the stego-image if the LSB of an image is changed.

One should note that as the minimum changes happened in the cover image the stego-system is more secure, and the performance of the stego-system is enhanced. Also, one should note that as the embedded text size goes large the number of image distortions should be increased when such text is embedded into the image. So, our goal is to embed a high capacity of text data into an image with minimum distortion of the image.

Accordingly, the present research introduces a modification on the existing and traditional LSB technique to improve its performance. Hence, it would be possible to hide a maximum capacity of embedded data (approximately 1/8 of image size) with low changing bits, i.e. low BER.

The main goal of our proposed algorithm is to reduce the number of changes happened to the Least Significant Bits of the image binary representation when a maximum text data is being

[1] The concept of statistical distribution is to be clarified in Chapter 2.

[2] Christian Cachin used this term in (Cachin 2005) and he considered that the distribution of cover-object and the stego-object are equal

embedded into the image. Thus, the invisibility or the secure of the system must be increased. As a result, the number of distortions of our proposed algorithm becomes lower than that of the traditional LSB algorithm when we embed a maximum text data in the image, which leads to consider our proposed technique more secure than traditional LSB technique.

Hereinafter, we define the system (algorithm) performance in terms of the Bit Error Rate (BER), where we consider the BER as the average distortion occurred in the image. So our objective is to use the maximum text size that be embedded in the LSB of the grey-scale image in a way that the BER is maintained minimum and always less than BER threshold.

Toward this end, we introduce three algorithms, sequential algorithm in which it is used by the traditional LSB technique, odd-even and random walk algorithms that we propose to obtain the proper results. The principles of hamming distance measurements should be used to calculate the BER of each algorithm, and the minimum one should be adopted. If the BER getting large and exceeds the BER threshold, Flipping Embedded Text (FET) technique should be used to keep the BER as minimum as possible.

As our research objective is to use a maximum text size that should be embedded in an image under the constraint of threshold BER, we first manipulate our problem as a linear programming problem, in which among matrix of choices we picked up the optimum text-image pair that achieves the BER constraint. The picking of such pair forcing us to use the entropy based measurements to measure the amount of information in both the text and the image. The entropy measurements help us as a first indication to detect the appropriate suitable pair, and in which if we do the embed process then we should obtain a BER less than BER threshold.

Accordingly, we simulate our problem as a Binary Symmetric Channel (BSC) in which we have the cover object *C* (the image before embedding) as a transmitter, and stego object *S* (the image after embedding) as a receiver. We formulate the embedding process as a noisy channel which can affect the original image. So our goal is to obtain a reliable communication (error free). In other words, the stego image is noiseless, i.e. with less BER.

In this context, we have a source of information (cover image *C*) which consists of number of bits zeros and ones. This information output from the source should be transmitted to the receiver through a noisy channel (embedding the message *M* which causes bits flipping). In the destination side, the transmission information should be extracted with error free as much as possible. According to the above scenario, we are looking for a reliable channel to transmit the information through, in order to obtain a stego image *S* more closely to the cover image *C* after the embedding of a large text *M*. So, the measurement of entropy should be used to measure the reliability of transmission. In other words, we should employ the principles of information theory to address our problem as a communication problem (Fearghail 2007). The entropy concepts are used to measure the reliability of transmission in order to decrease the number of errors happened through embedding process. So, we expect that there is a relation between the entropy of the source information (cover image *C*) and the entropy of the noisy channel (text message *M*) that help us to determine if the source is suitable for the appropriate channel or not in the sense of errorless transmission (stego image *S* similar to *C*).

It is well known in information theory literature that the entropy function is inversely proportional to probability of occurrence of an event. That is, in a certain image that contains a large number of bits 0's (most likely black), one should expect a smaller BER occurs when using a text of higher distribution of 0's, henceforth. This concept drives us to consider the entropy measurements of both text and image.

The analytical scientific methodology that we used depends on the scientific bases and rules to achieve the optimum results by applying a linear programming problem. We used MATLAB version 6.5 to simulate an ASCII coded text and a pixel coded image, where in this research, an English encrypted text should be hidden in the grey scale image, namely, Lenna 256x256 image pixels. Different sizes of the text should be embed in Lenna image, we should start with 100 bytes of text size, and each time we increment the text size by 100 byte until we reach the maximum text size that the image can be contained. Each time we use the three algorithms to construct the LSB of the image according to the required text. The embedding process is done using the traditional LSB technique and our proposed FET technique. Each time the BER should be calculated using the hamming distance measurements in both techniques. We should show that our FET technique always achieves smaller BER than conventional LSB and the BER should be less than BER threshold. So, the minimum BER should be adopted and chosen, which is used in the stego-system.

We examine our results using the entropy calculations; we should find out that the results of the entropy measurements are coincide with those of the hamming distance measurements.

1.5 Synopsis of Thesis

In Chapter two, we present the background and literature review, a system model was introduced in Chapter three to describe the transmitter that generates the secret message, *M*, the cover object, *C*, and the resultant stego-object, *S*. In Chapter four, we discuss several algorithms to hide the message based on hamming distance measurements.

The algorithms of hiding the messages are discussed in Chapter 5 based on the Entropy measurements. Numerical examples and simulation results are introduced in Chapter 6.

Finally, in Chapter 7, some comparisons, conclusions, and future work problems are opened.

Chapter 2

Background and Literature Review

2.1 Background

The earliest recordings of Steganography were by the Greek historian Herodotus in his chronicles known as "Histories" and dated back to around 440 BC. Herodotus recorded two stories of Steganographic techniques during this time in Greece. The first stated that King Darius of Susa shaved the head of one of his prisoners and wrote a secret message on his scalp. When the prisoner’s hair grew back, he was sent to the Kings son in law Aristogoras in Miletus where he revealed the undetected message.

The second story also came from Herodotus, which claims that a soldier named Demeratus needed to send a message to Sparta that Xerxes intended to invade Greece. Back then, the writing medium was text written on wax-covered tablets. Demeratus removed the wax from the tablet, wrote the secret message on the underlying wood, recovered the tablet with wax to make it appear as a blank tablet and finally sent the document without being detected. (Bret, 2002).

Steganography is the art of sending hidden or invisible messages. The name is coming from the Greek στεγανό-ς, γραφ-ειν meaning “covered writing” (Petitcolas, Anderson and Kuhn 1999).

Steganography is coming from the Greek words stegos, meaning covered and graphia which means writing, is the art and science of hiding a secret message inside another message so that no one can detect the secret message (Neil F. and Jajodia S. 1998-a).

Romans used invisible inks, which were based on natural substances such as fruit juices and milk. This was accomplished by heating the hidden text, thus revealing its contents (Bret, 2002).

During the 15th and 16th centuries, many writers including Johannes Trithemius (author of Steganographia) and Gaspari Schotti (author or Steganographica) wrote on Steganagraphic techniques such as coding techniques for text, invisible inks, and incorporating hidden messages in music (Bret, 2002).

During the American Revolution, invisible ink which would glow over a flame was used by both the British and Americans to communicate secretly (Cummins, Diskin, Lau and Parlett 2004).

During the times of WWI and WWII, significant advances in Steganography took place. Concepts such as null ciphers which taking the 3rd letter from each word in a harmless message to create a hidden message, etc. Microdot (Refers to text or photographic images that are reduced in size to prevent their viewing by unintended recipients), were introduced as great steganographic techniques (Bret, 2002).

For example of such uses, steganography was used in both World Wars. German spies hide text by using invisible ink to print small dots above or below letters and by changing the heights of letter-strokes in cover texts (Cummins, Diskin, Lau and Parlett 2004).

In World War I, a message reading “Father is dead” was modified to read “Father is deceased” and when the reply “Is Father dead or deceased?” came back the censor was alerted to the hidden message (Cummins, Diskin, Lau and Parlett 2004).

A message sent by a German spy during World War II read: “Apparently neutral’s protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.” By taking the second letter of every word the hidden message “Pershing sails for NY June 1” can be retrieved (Hany and Siwei, 2006).

By Steganography we can embed a secret message inside a piece of unsuspicious information and send it without anyone knowing of the existence of the secret message. So, the purpose of steganography is covert communication. It is used to hide the existence of a message from a third party (Kessler G. 2004).

Steganography is an important sub discipline of information hiding; the main goal of steganography is to communicate securely in a completely undetectable manner (Kharrazi, Husrev and Memon 2004). Steganography hides the covert message but not the fact that two parties are communicating with each other. The message is the data that the sender wishes to remain confidential and can be text, image, audio, video, or any other data that can be represented by a stream of bits (Davidson and Goutam 2004).

In the digital world of today, namely 1992 to present, Steganography is being used all over the world on computer systems. Many tools and technologies have been created that take advantage of old steganographic techniques such as null ciphers, coding in images, audio, video and microdot. With the research this topic is now getting developed, we will see a lot of great applications for Steganography in the near future (Bret, 2002).

In the current and previous decades, the modern steganography is intended to use electronic media. The rapid development of the Internet and the digital information revolution caused significant changes in the global society, ranging from the influence on the world economy to the way people nowadays communicate. Versatile and simple-to-use software and decreasing prices of digital devices (e.g. digital photo cameras, camcorders, portable CD and MP3 players, DVD players, CD and DVD recorders, laptops, PDAs) have made it possible for consumers from all over the world to create, edit and exchange multimedia data. Broadband Internet connections and almost an errorless transmission of data facilitate people to distribute large multimedia files and make identical digital copies of them (Cvejic, 2004).

With the spread and extensive use of digital media, the frequent use of this media (image, audio, video, and text format) for the purposes of steganography has become so clear.

Modern steganography is generally understood to deal with electronic media rather than physical objects and texts. This makes sense for a number of reasons.

- The size of the information is generally (necessarily) quite small compared to the size of the data in which it must be hidden (the cover text), electronic media is much easier to manipulate in order to hide data and extract messages.

- Extraction itself can be automated when the data is electronic, since computers can efficiently manipulate the data and execute the algorithms necessary to retrieve the messages.

- Electronic data also often includes redundant, unnecessary, and unnoticed data spaces which can be manipulated in order to hide messages. In a sense, these data spaces provide a sort of hidden pockets into which secret messages can be inserted and sent off to the receiver.

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily (Morkel, Eloff and Olivier 2005). Steganographic information can be hidden in almost anything, and some cover objects are more suitable for information hiding than others. Image and audio files are the most famous types of data that comply with steganographic requirements. But in our research we should use the digital image[1] as a cover

[1] especially grey-scale image

object to hide a text message inside. This type of data was chosen because of the aforementioned reasons.

To a computer, an image is a collection of numbers that constitute different light intensities in different areas of the image (Neil F. and Jajodia S. 1998-a). This numeric representation forms a grid and the individual points are referred to as pixels. So, an image can be defined as a two-dimensional function of two real variables, $f(X,Y)$, where X and Y are spatial coordinates, and the amplitude f at a given pair of coordinates is called the intensity or grey level of the image at that point. When X, Y and the amplitude values of f are all finite, discrete quantities, we call the image a digital image. So, the digital image is composed of a finite number of elements, called pixels, each with a particular location and a finite value determined by X and Y (Gonzalez, Woods and Eddins 2004).

Computer images have been "digitized", a process which converts the real world color picture to numeric computer data consisting of rows and columns of millions of color samples measured from the original picture (Morkel, Eloff and Olivier 2005).

The steganography process generally involves placing a hidden message in some transport medium, called the carrier or cover object. The secret message is embedded in the carrier to form the steganography medium or stego-object.

Once the message is embedded into the cover object by using some technique, the result is a stego-object that contains the hidden message which is very similar to the cover object. The stego-object must remain unchanged or almost unchanged to the naked eye. If the stego-object changes significantly and can be noticed, a third party may see that the information is being hidden and therefore could attempt to extract or destroy it (Fridrich, Goljan and Soukal 2004).

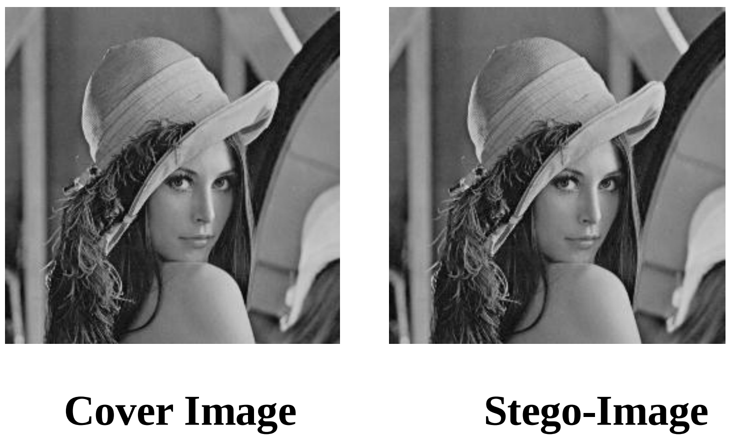

Figure 2: Cover and stego images seem the same

One should note that the cover-medium and stego-object must be of the same data type, but the embedded message may be of another data type (Moskowitz, Longdon and Chang 2000).

The embedding process may depend on a secret stego-key. The stego-key is used to control the embedding process, such as the selection of pixels or coefficients carrying the message (Fridrich, Goljan and Soukal 2004). The stego-key adds more security to the stego-system.

The most important requirement for a steganographic system is the undetectability: Stego-images should be statistically indistinguishable from cover images. In other words, there should be no artefacts in the stego-image that could be detected by an attacker with probability better than random guessing, given the full knowledge of the embedding algorithm, including the statistical properties of the source of cover images, except for the stego-key.

The warden Wendy (attacker) who is free to examine all messages exchanged between Alice and Bob can be passive or active. A passive warden simply examines the message and tries to determine if it potentially contains a hidden message. If it appears that it contains such a message, she suppresses the message and/or takes appropriate action, else she lets the message through without any action. An active warden, on the other hand, can alter messages deliberately, even though she does not see any trace of a hidden message, in order to foil any secret communication that can nevertheless be occurring between Alice and Bob. The amount of change the warden is allowed to make depends on the model being used and the cover-objects being employed. For example, with images, it would make sense that the warden is allowed to make changes as long as she does not alter significantly the subjective visual quality of a suspected stego-image.

We state ourselves in the case of passive warden and not the active warden, so, we should not give the warden any sense that the cover medium contains any hidden data. Since our own work aims to decrease the number of distortions happened to the cover image while embedding the secrete message, it is difficult to the warden to determine that the stego-object is containing a secrete message or not.

Wendy should not be able to distinguish in any sense between cover-objects (objects not containing any secret message) and stego-objects (objects containing a secret message). In this context, steganalysis refers to the body of techniques that aid Wendy in distinguishing between cover-objects and stego-objects. It should be noted that Wendy has to make this distinction without any knowledge of the secret key which Alice and Bob may be sharing and sometimes even without any knowledge of the specific algorithm that they might be using for embedding the secret message. Hence steganalysis is inherently a difficult problem. However, it should also be noted that Wendy does not have to glean anything about the contents of the secret message *M*. Just determining the existence of a hidden message is enough. This fact makes her job a bit easier (Kharrazi, Husrev and Memon 2004).

2.2 Literature Review Analysis and Comparisons

Information hiding can be broken down into different areas. Steganography can be used to hide a secret message. In this case the aim is to prevent the message being detected by an attacker.

The other major area of steganography is copyright marking, where the inserted message is used to emphasize copyright over a document.

Watermarking is a notably different form of steganography because of its purpose. It is used primarily to denote ownership or authorship of a particular file while steganography, by definition, is used to pass secret messages. With watermarking, the mark can either be hidden or visible to accomplish its purpose.

The most common usage of watermarking is for copy-write protection. In this case, the watermark must be robust, meaning it cannot be removed or damaged through compression, cropping, rotating, resizing, or trimming the data.

There are two strategies to handle the image steganography techniques:

2.2.1 Spatial Domain Strategy

It is to deal with the image domain also known as spatial domain. Image domain techniques encompass bit-wise methods that apply bit insertion and noise manipulation and are sometimes characterized as "simple systems" (Neil F. and Jajodia S. 1998-a). The image formats that are most suitable for image domain steganography are lossless and the techniques are typically dependent on the image format.

In image domain, the message is imbedded in the intensity of image or in the spatial domain of the image directly. There are many techniques used in this domain. The most important and the easiest one is called Least Significant Bit (LSB) Technique that is considered in our research.

Least significant bit (LSB) is a common, simple approach to embed information in a cover image (Neil F. and Jajodia S. 1998-a). The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. When using a 24-bit image, a bit of each of the red, green and blue color components can be used, since they are each represented by a byte. In other words, one can store 3 bits in each pixel. To understand how the LSB technique is worked we should give the following example:

Suppose a pixel contains the color values 255, 255, 255 which are 11111111, 11111111, 11111111 in binary (a very bright white). The data we want to hide is the first three bits of some ASCII code **011**

The values of the pixel representation will be changed in the following order:

11111111 becomes 1111111**0**

11111111 stays as 1111111**1**

11111111 stays as 1111111**1**

So the resulting color is 254, 255, 255 which is a shade of white that a human eye can not detect the indiscernible difference. So the stego-image is nearly similar to the cover-image.

Popular steganographic tools that are based on LSB-embedding vary in their approach for hiding information. Methods like Steganos and Stools use LSB embedding in the spatial domain, while others like Jsteg and OutGuess embed the message in the frequency domain (to be discussed below) (Avcibas, Memon and Sankur 2003).

As the resolution and depth of color increase in an image, the impact of manipulating the LSB becomes less noticeable. Thus, high resolution images are preferred for use as cover images (Silman 2001).

Changing the LSBs causes an imperceptible change to the digital image. Without a direct comparison between the original image and the altered image, there isn't any strong way to tell that something is changed (Al-Jaber and Aloqily 2003).

With a well-chosen image, one can even hide the message in the least as well as second to least significant bit and still not see the difference (Neil F. and Jajodia S. 1998-a).

Some researchers can hide the data in the 4-LSB (Alkhraisat 2005). Habes used an algorithm to hide a maximum capacity of data in the cover image. His algorithm used the 4-LSB of the pixel representation for hiding the message. In our research, Flipping Embedded Text (FET) technique can be applied not only in the 1-LSB but also it can be extended to use 4-LSB with minimal distortion in each LSB level. In addition, FET technique does not need the cover image in the receiver side to extract the secret message. But according to (Alkhraisat 2005), the extracting process needs both the cover and stego images.

(Fridrich 2006) introduced an embedding scheme that used more numbers of selected pixels to embed the message in their LSB although all pixels of the cover image are used to achieve minimum distortion in the cover image. But, in our research, we are using only minimum number of pixels that contain the appropriate message.

There are many techniques which used the LSB approach to hide a secrete message. Jsteg uses the traditional LSB method i.e. it replaces the LSB of the frequency domain of the cover image sequentially bit-by-bit. Which causes the secrete message becomes easy to extract and discover. Also, especially in the case of small message size, the bearing pixels are grouped in the first part of the image and the other parts are unused, which indicates that there is something hidden. While in our research we adopt the random-walk algorithm to spread the message over the entire image, which added difficulties for the attacker to detect the existence of such a secret message.

Out-Guess technique selects the used pixels randomly by using the stego-key as a seed of a pseudo random number generator. The bearing pixels were selected randomly, but in this process, a chance of collision may happen i.e. the same pixel may be selected twice in the random process (this can be done in the case of a large size of data to be hidden). To overcome this problem, a set is used to keep track of the selected pixel. The set was checked every time a random process is done, and if the selected pixel is repeated it will be ignored, and another one should be taken. But the collision case in our research will not happen because our random-walk algorithm is simple and does not depend on the pseudo random number generator. Our random walk algorithm can be described as a flip-flop algorithm; it is to say that it passes the bearing pixels randomly but in a rhythmic way. While Out-Guess hides the message in the Discrete Cosine Transformation (DCT) of the image, our technique hides the message in the spatial domain.

Three different basic aspects in information-hiding systems are closely related with each other: capacity, security, and robustness. Capacity refers to the amount of information that can be hidden in the cover medium. Security refers to guard's inability to detect hidden information, and robustness refers to the amount of modification the stego-medium can withstand before an adversary can destroy hidden information. So these requirements depend on the purpose of steganography.

Capacity is an important factor in captioning applications where a lot of information is embedded into a cover image which is usually related to the same, current picture. For example, when medical images were transmitted, the personal data, and the diagnosis could be embedded into the same picture.

Security (imperceptibility) is important when a secret communication occurs between two parties and the fact of a secret communication is kept to be secret. The method will be called secure if the stego-images should have the same statistical properties as the set of cover-images. If there exists an algorithm that can guess whether or not a given image contains a secret message with a success rate better than random guessing, the steganographic system is considered broken (Fridrich, Goljan and Soukal 2004).

Robustness mostly used in watermarking and all copyright protecting applications which they are always demand robust steganographic method, i.e., where the embedded information cannot be removed without serious degradation of the image.

The information hiding problem space was characterized by a trade-off between robustness and bandwidth. When the capacity is high, we will have less security and less robustness. So, there is an opposite relation between capacity and either robustness or security. As one increases, the other must decrease.

Obviously, the less information we embed into the cover image, the smaller the probability of introducing detectable artifacts by the embedding process. Each steganographic method seems to have an upper bound on the maximal safe message length (or the bit rate expressed in bits per pixel or sample) that tells us how many pseudorandom bits can be safely embedded in a given image without introducing any statistically detectable artifacts. This limit is called the steganographic capacity (Fridrich, Goljan and Soukal 2004).

A notion of “steganographic capacity” which means “how many bits can be hidden in an image using LSB based technique without causing statistically significant modifications” is introduced by (Chandramouli and Memon 2001). Their results are able to provide an upper bound on this capacity. They derive a closed form expression in terms of the number of bits that are hidden. We think that applying our proposed technique before using the form expression proposed by Chandramouli and Memon will increase the upper bound of steganographic capacity since our proposed technique uses the LSB based algorithm and succeeds in decreasing the number of distortions. This minimal distortion makes warden Wendy undistinguished whether the cover object contains a secret message or not. So we can pass more capacity of data before the warden Wendy can notice it.

So far, we discussed the LSB of the image to hide the information, our proposed algorithm is considered as simple system since it depends heavily on LSB technique; it does not use complex methods or composite methods such as those of both frequency domain and spatial domain to obtain the results. It does not change the size of the cover image file as some techniques do such as S-Tool.

The proposed algorithm introduced some improvements into the traditional LSB technique. It raises the performance of the LSB technique by hiding a high capacity of text data in the image with low Bit Error Rate. So the security and the invisibility of the system are increased. Another good feature of our proposed algorithm is that the cover image file is not needed to extract the text from the stego-image.

However, another way to hide information is to use the palette of the image. Such a technique is to alter the order of the colors in the palette or use least significant bit encoding on the palette colors rather than on the image data, for example GIF images, are another popular image file format commonly used on the Internet. By definition, a GIF image cannot have a bit depth greater than 8, thus the maximum number of colors that a GIF can store is 256. GIF images are indexed images where the colors used in the image are stored in a palette, sometimes referred to as a color lookup table (Morkel, Eloff and Olivier 2005). Each pixel is represented as a single byte and the pixel data is an index to the color palette. The colors of the palette are typically ordered from the most used color to the least used colors to reduce lookup time.

GIF images can also be used for LSB steganography although extra care should be taken. The problem with the palette approach used with GIF images is that should one change the least significant bit of a pixel, it can result in a completely different color since the index to the color palette is changed. If adjacent palette entries are similar, there might be little or no noticeable change, but should the adjacent palette entries be very dissimilar, the change would be evident (Neil F. and Jajodia S. 1998-a).

One possible solution is to sort the palette so that the color differences between consecutive colors are minimized (Kharrazi, Husrev and Memon 2004). Another solution is to add new colors which are visually similar to the existing colors in the palette. Using this approach, one should thus carefully choose the right cover image. Unfortunately any tampering with the palette of an indexed image leaves a very clear signature, making it easier to detect.

A final solution to the problem is to use greyscale images. In an 8-bit greyscale GIF image, there are 256 different shades of grey (Neil F. and Jajodia S. 1998-a). The changes between the colors are very gradual, making it harder to detect.

Finally, since the color palette based images are not of our concern, we advice the interested reader to consult (Neil F. and Jajodia S. 1998-a) for more details.

2.2.2 Transform Domain

Frequency Domain Strategy, is to deal with the transform domain which known as frequency domain. Steganography in the transform domain involves the manipulation of algorithms and image transforms. These methods hide messages in more significant areas of the cover image making it more robust. Data embedding performed in the transform domain is widely used for robust watermarking. Many transform domain methods are independent of the image format and the embedded message may survive conversion between lossy and lossless compression (Morkel, Eloff and Olivier 2005).

Image domain techniques embed messages in the intensity of the pixels directly, while in transform domain techniques, images are first subjected to transformation and then the message is embedded in the transformed image (Morkel, Eloff and Olivier 2005).

To understand the steganography algorithms that can be used when embedding data in the transform domain, one must first explain the type of file format connected with this domain - although our research field is not in transform domain -. The JPEG file format is the most popular image file format on the Internet, because of the small size of the images. For JPEG, the Discrete Cosine Transform (DCT) is used, but there are similar transforms, for example the Discrete Fourier Transform (DFT). These mathematical transforms convert the pixels in such a way as to give the effect of “spreading” the location of the pixel values over part of the image (Morkel, Eloff and Olivier 2005). The DCT transforms a signal from an image representation into a frequency representation, by grouping the pixels into 8×8 pixel blocks and transforming the pixel blocks into 64 DCT coefficients each. A modification of a single DCT coefficient will affect all 64-image pixels in that block.

The next step is the quantization phase of the compression. Here another biological property of the human eye is exploited: The human eye is fairly good at spotting small differences in brightness over a relatively large area, but not so good as to distinguish between different strengths in high frequency brightness (Moerland T. 2003). This means that the strength of higher frequencies can be diminished, without changing the appearance of the image. JPEG does this by dividing all the values in a block by a quantization coefficient. The results are rounded to integer values and the coefficients are encoded using Huffman coding to further reduce the size (Morkel, Eloff and Olivier 2005).

Originally it was thought that steganography would not be possible to use with JPEG images, since they use lossy compression which results in parts of the image data being altered. One of the major characteristics of steganography is the fact that information is hidden in the redundant bits of an object and since redundant bits are left out when using JPEG it was feared that the hidden message would be destroyed. Even if one could somehow keep the message intact it would be difficult to embed the message without the changes being noticeable because of the harsh compression applied. However, properties of the compression algorithm have been exploited in order to develop a steganographic algorithm for JPEGs (Morkel, Eloff and Olivier 2005).

It is neither feasible nor possible to embed information in an image that uses lossy compression, since the compression would destroy all information in the process. Thus it is important to recognize that the JPEG compression algorithm is actually divided into lossy and lossless stages. The DCT and the quantization phase form part of the lossy stage, while the Huffman encoding used to further compress the data is lossless.

Some steganographic algorithms can either be categorized as being in the image domain or in the transform domain depending on the implementation, such as, Patchwork technique. A disadvantage of the patchwork approach is that only one bit is embedded. One can embed more bits by first dividing the image into sub-images and applying the embedding to each of them (Petitcolas, Anderson and Kuhn 1999). The advantage of using this technique is that the secret message is distributed over the entire image, so should one patch be destroyed, the others may still survive. This however, depends on the message size, since the message can only be repeated throughout the image if it is small enough. If the message is too big, it can only be embedded once (Neil F. and Jajodia S. 1998-a).

Another technique is the Spread Spectrum technique, in which hidden data is spread throughout the cover-image making it harder to detect (Johnson and Katzenbeisser 1999).

Spread spectrum communication can be defined as the process of spreading the bandwidth of a narrowband signal across a wide band of frequencies. This can be accomplished by adjusting the narrowband waveform with a wideband waveform, such as white noise. After spreading, the energy of the narrowband signal in any one frequency band is low and therefore difficult to detect. In spread spectrum image steganography the message is embedded in noise and then combined with the cover image to produce the stego image. Since the power of the embedded signal is much lower than the power of the cover image, the embedded image is not perceptible to the human eye or by computer analysis without access to the original image (Johnson and Katzenbeisser 1999).

To understand more in the frequency domain and it’s techniques we recommend the reader to see (Neil F. and Jajodia S. 1998-a) and (Morkel, Eloff and Olivier 2005).

There exists a large selection of approaches to hide information into images. All the major image file formats have different methods of hiding messages, with different strong and weak points respectively. Where one technique lacks in payload capacity, the other lacks in robustness or imperceptibility.

Our work lies in the area of LSB used in grey-scale image. It aims to increase the invisibility aspect while attempting to embed a high capacity that the grey image permitted.

Chapter 3

System Model and Analysis

3.1 System Model

In our system, we will consider a source of messages that generates *M* message that is to be hidden in an object, *C*. Without loss of generality, we assume a text message *M* is to be hidden in a digital image, *C*. The resultant Stego-Image, *S*, contains the hidden text. Our model is a bit different of the general model of steganography[1]. In our model the text message *M* should be flipped if the resultant BER > $\text{BER}_{\text{threshold}}$. In our model the message should be subjected to encryption to increase the security of the stego-system. After that the message should be embedded into the image using one of the three algorithms (sequential, odd-even and random walk algorithms), our stego-system choose the best algorithm that causes a less BER taking into account the flipping process of the message *M*. The following figure describes our proposed model.

[1] See Fig. 1

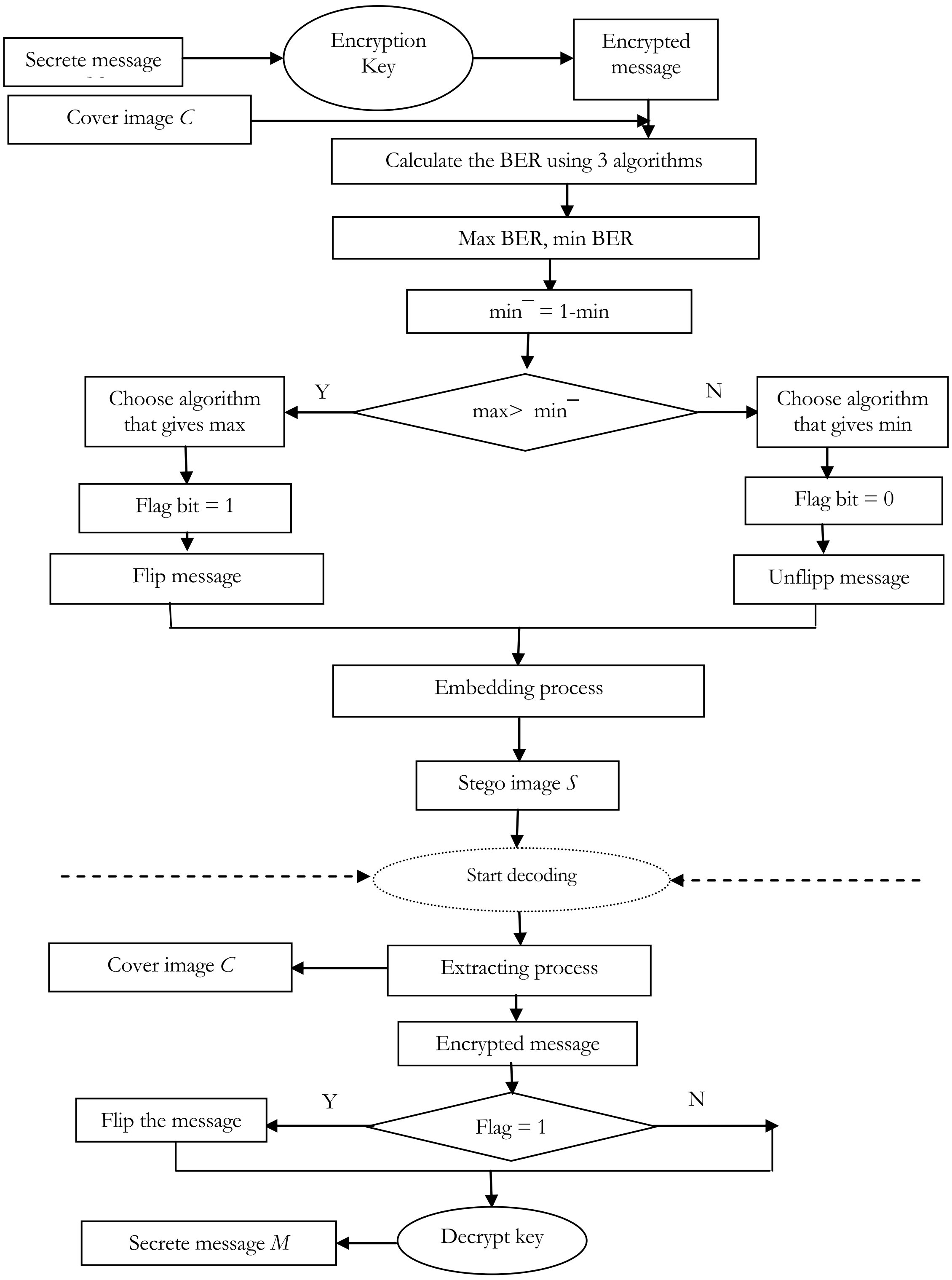

Figure 3: Our proposed model

Our system model converts the text message *M* to a stream of binary bits, as shown in Fig. 3, the first step is to encrypt the message using 4-bits key. The encryption is added extra security to the embedding and extracting procedures. In this process two codes should be used, a private code which known only to the sender and receiver and public code which may known to any one. Each code consists of four bits, the 4-bits codes are chosen optimally[1].

The encryption process using 4-bits encryption code is described as follows:

In the encryption process two codes are used; a private code and public code. The private code is used as a window for the public code in encoding and decoding text; in encoding process the public code is AND-ed with the private code, the result of AND-ing processed is 4 encoding bits, where these 4-bits is XOR-ed with each others, the result of XOR-ed process is a single bit, this bit is XOR-ed with the first bit of the stream bits of the text data, after each encoding process of data bits, the public code is rotated left once, so another code will be formulated. This new code is AND-ed again with private code and the result of AND-ing process is XOR-ed with each other and the result is XOR-ed again with the next data bit, and so on. This process continues until encoding all data bits. This process is repeated at the receiver side to decode data bits from the received stego-image.

[1] The selection of optimal encryption is out of scope of our research. We use encryption to add some difficulty to our system in order to prevent the attacker to reveal the message.

The encryption process can be clarified in Fig. 4, below.

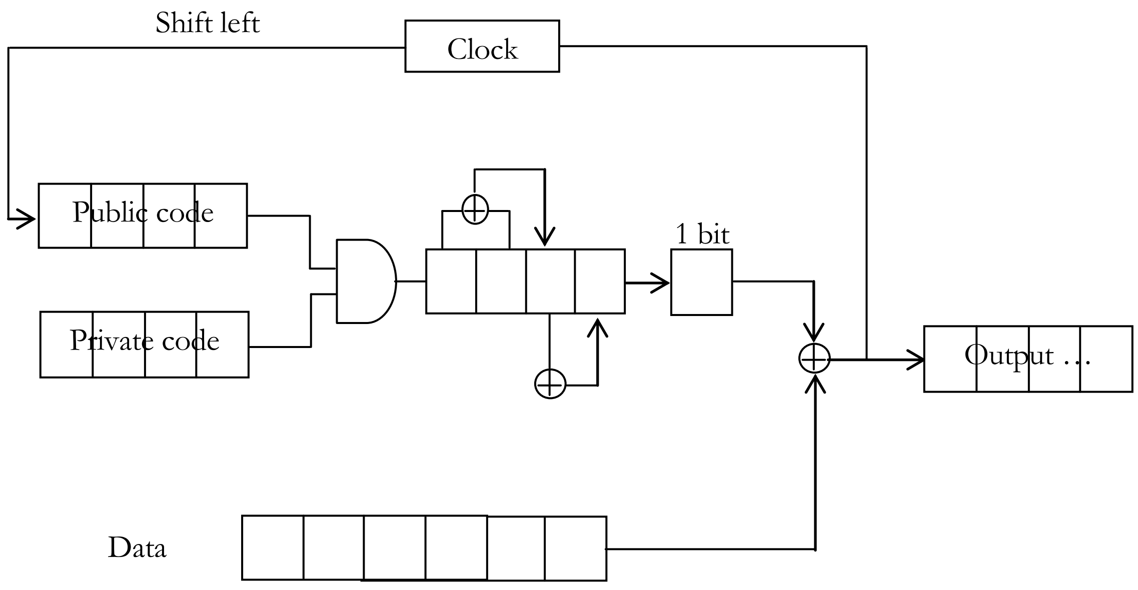

Figure 4: Encryption method diagram.

One should note that the rotated procedure of the public code in which it rotates the bits of the code one bit to the left is used to add more complexity on the encryption algorithm. In addition the rotated process adds more randomness to the encrypted message in which the same letter of the message has different encrypted result which prevents the warden to expect the value of any letter of the original message.

To explain the encrypted process we introduce the following example:

Assume the stream of bits to be encrypted is [10101] and the private code = [1001] and the public code = [1011]. The first step is AND-ing the two codes. So,

[1001]. [1011] = [1001]

The second step is to XOR-ed the result with each other. So,

$1 \oplus 0 = 1 \oplus 0 = 1 \oplus 1 = 0$ so we have 0 as a result of XOR-ed process. This bit is XOR-ed with the first bit of the data stream. i.e. $0 \oplus 1 = 1$ which is the first encrypted bit of the message. Then, the public code is rotated to the left one step, hence, the code [1011] becomes [0111], then we repeat the process again.

[0111].[1001] = [0001], then XOR-ed with each other the result is 1. This result bit is XOR-ed with the second bit of the data stream. So, $1 \oplus 0 = 1$ is the encrypted bit.

The public code is rotated again so we have a new code of [1110]

[1110].[1001] = [1000]

The result of XOR-ed with each other is 1. This bit is XOR-ed with the third bit of the data stream 1 and the result is 0, and so on. The process is repeated until all bits of data stream bits are encrypted.

So, the encryption process add extra one level security in steganographic system. The conventional cryptography (symmetric key) is used. The same key is used for both encryption and decryption procedures. Our encryption method is classified as a bit complicated; because of circulated procedure which tends the same letter in different location in the text has different encrypted code.

In our proposed model we use three different algorithms, these embedding algorithms are used to calculate the BER based on hamming distance measurements, we should choose the maximum BER and the minimum one of the used algorithms, then we compare the maximum BER with the complement of the minimum one, if the maximum BER is greater than minimum complement BER then we choose the algorithm that gives maximum BER and the encrypted binary bits of the message should be flipped to achieve BER always less than $BER_{threshold}$. The flag bit is set to 1 to indicate that the encrypted message was flipped. On the other hand, if the minimum complement of BER is greater than the maximum BER then the algorithm that gives the lowest BER should be chosen and the message should not be flipped, i.e., the flag bit is 0. After the stego-image *S* is formulated, it is sent to receiver.

The receiver extracted the encrypted message from the stego-image *S* and decrypts it using the same encrypted keys as mentioned before. The receiver investigates the flag bit to determine whether the received message needs to be flipped or not in order to recover the original message *M*.

To clarify our proposed model, we divide the system model into two major parts, the text model, and the image model.

3.1.1 Text Model

In the text model, we assume that a transmitter generates a text message contains all possible alphabets (characters) in any language. The source generates the message that contains M distinct characters. The i-th character, $1 \le i \le K$, denoted as m_i is generated by the source with probability $P(m_i)$, where K is the number of characters in the source. Depending on the language itself, the characters are generated with different probabilities from the source, e.g., if we consider an English text message, due to its nature, the repetition of the character "e" differs from the repetition of, say, character "q".

Accordingly, it is impractical to assume that the characters are generated with equally probable distribution, but, it is language dependent, that is to say, for $m_i \neq m_j$, we have, $P(m_i) \neq P(m_j) \; \forall^{\, i \neq j}$.

However, as our objective is to hide the generated text into a digital image, we have to deal with the text in its binary form rather than in its character form. In English, as any other language, the characters are to be mapped to ASCII codes, thus, the source can be seen as a sequence of bits generator, where we, moreover, assumed that the source generates a total number of bits, say, T. Accordingly, the generated alphabet is ASCII encoded into a vector **t** of binary bit stream of length T. One should note that, the probability density function[1] (p.d.f) of the generated characters is a discrete non-uniform function, however, after mapping characters to their corresponding ASCII codes, the generated bits (0 and 1) have a discrete uniform p.d.f. of equal probability, namely, half. Such a source that generates sequence of bits independently is referred to as a Discrete Memmoryless Source (DMS) (Armand, 2003). From the first axiom of probability definition, for large text, it is clear that $P(0) = P(1) = 1/2$.

In general, given a certain specific text mapped into ASCII binary bits, the resultant number of 0's differ from the number of 1's, we can say, $P(0) = p$, and $P(1) = 1\text{-}p$. The average amount of information, denoted by the binary Entropy function (Shannon, 1948), is given by

$$E = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{(1-p)}. \tag{3.1}$$

[1] In literature, sometimes it is referred to as probability mass function (pmf)

The entropy and related information measures provide useful descriptions of the long term behaviour of random processes. It is the number of bits on the average required to describe the random variable, i.e., the entropy of a random variable is a lower bound on the average number of bits required to represent the random variable say *X*. Note that entropy is a functional of the distribution of *X*. It does not depend on the actual values taken by the random variable *X*, but only on the probabilities (Cover and Thomas 1991).

The binary entropy function is shown in Fig. 5 as a function of the probability *p*. We can see that the convex binary entropy function has its maximum value 1 when $p = 1\text{-}p = 1/2$ and symmetrically, decreases around *p*. The binary entropy function, moreover, has zero values if and only if $p = 0$ or 1.

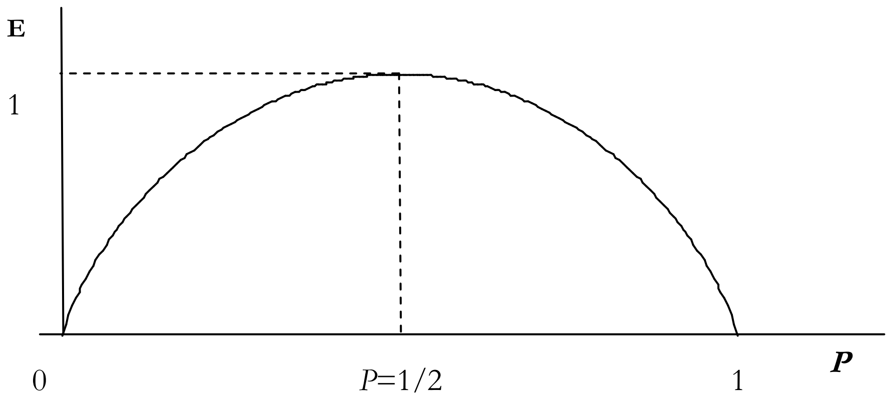

Figure 5: Binary entropy function

Under the assumption that $p\ (0) = p$, for a text contains only 0's, the entropy function $E = 0$, as the number of 0's decrease, the entropy function increases to reach its maximum value when $p = 1/2$. That is to say, when the number of 0's is equal to number of 1's. This makes sense, because when $p = 0$ or 1, the variable is not random and there is no uncertainty. Similarly, the uncertainty is maximum when $p = 1\text{-}p$, which also corresponds to the maximum value of the entropy. Thus, the entropy function, *E*, can be considered as a good measure of the statistical distribution of 0's and 1's within a specific text. Such a concept will be considered when discussing the embedding of a binary mapped-text into a digital image in Chapter 5.

3.1.2 Image Model

An image is defined as a collection of real numbers that determines different light intensities in different areas of the image (Honeyman and Provos, 2003). This numeric representation forms a grid and each point is named a pixel. The image can be represented as a 2-

dimensional function of two real variables $f(x,y)$[1], where x,y represent the spatial coordinates, and the function amplitude f given at any pair of coordinates is called the intensity or grey level of the image at that point. When x, y and the amplitude f are all finite and discrete quantities, we have a digital image; otherwise, it is an analogue image (Gonzalez, Woods and Eddins 2004).

A digital image consists of a rectangular map of the image's pixels that can be represented as bits. The number of bits used for each pixel is called the bit depth. The bit depth of monochrome (binary image) is 1 bit (two-states case, either 1 or 0 to represent black or white), while the bit depth of the grey image is 8 bits to display 256 different shades of grey. Finally, the true color image used the RGB model[2] of 24-bit depth.

No doubt that the image file size increases with the number of pixels. Aiming at simplifying the matters, and due to the fact that the greyscale images are considered a good type to hide data in spatial domain[3] (Neil F. and Jajodia S. 1998-a), in our research, we will consider the greyscale image to hide the text file, although the RGB image is able to hide larger text.

Referring to previous discussion, the 2-dimensional image, say the cover image C, can be seen as a 2-dimensional matrix of order $a \times b$, each pixel position is determined by the spatial coordinates (x,y), where $0 \leq x \leq a$, and $0 \leq y \leq b$, the value of such a pixel is determined by the amplitude f of a decimal value encoded into 8-bit binary codeword. The 8-bit binary value determines the image grey level.

Let us assume the cover image C is $a \times b$ pixels of total number of bits, N, under the assumption that each pixel is encoded using 8-bits codeword, then $N = 8 \times a \times b$. As the text size is directly related to the image size, it is clear that the maximum text size is bounded by $N/8$, i.e., $T \leq N/8$, of bits that can be hidden in the image pixels if we assume that the information is to be hidden only in the Least Significant Bit (LSB)[4] of each pixel codeword (Chandramouli and Memon 2001). Our objectives are to hide a large amount of T which known as a system capacity.

3.2 System Capacity Calculations

The length of the text should be bounded by the image pixels. The maximum length size of a text can be embedded into the LSB of the image is T bytes, but some of the Least Significant Bits of the image should be reserved to help the receiver to well decode and extract the text from the image. One bit was reserved as a flag bit which is using as a control of flipping process. Flag bit is useful to indicate that the text was flipped or not. Also we need some bits

[1] So far, the image we consider is the grey image.

[2] All color variations of 24-bit image are derived from three primary colors: red, green and blue, and each primary color is represented by 8 bits

[3] Data is hidden in the least significant bits of the image

[4] Hereinafter, we will not apply any data compression techniques through our analysis.

of the image to represent the length of the text to send it to the receiver with the image, in order to help the receiver to exactly find the bits of the message inside the image.

In general, the formula is manipulated as follows:

Assume the image size $S = a \times b$ pixels, and the high capacity of data can be embedded is T. one should note that the embedding is done in the LSB of each pixel.

$$S = a \times b \text{ bytes}$$
$$T = S/8 \text{ bits}$$

Convert the decimal value of T to binary. Moreover, BT is the binary of T, $BT = \text{Bin}(T)$

In other words, we want to find the minimum numbers of bits that represents the text size T.

Assume the number of bits that can represent the text length T is l, which is defined as the amount of information (Shannon, 1948) included in l, given by

$$l = \text{I}(l) = \log_2 (BT) \tag{3.2}$$

Here we deal with the BT as a decimal value.

Lemma (1):

Here we intend to prove expression in (3.2).

We note that the number of the LSB's of the image is T and a text file should be embedded in these LSB's. The question is how many bits that the text size T needs to be implemented in binary representation, we assume this number of bits representation is l. So, $T \leq 2^l$. By taking $\log_2$ for both sides, we have $\log_2 T \leq \log_2 2^l$, $\log_2 T \leq l$.

Thus, $l \geq$ Ceiling ($\log_2 T$), i.e., $l \geq \lceil \log_2 (T) \rceil$, where $\lceil x \rceil$ is the least maximum integer greater than x, and l is the minimum number of bits needed to represent the maximum text size used by specific image.

■

In our case, we use Lenna 256 $\times$ 256 pixels.

$$S = 256 \times 256 = 65536$$

$$T = \frac{65536}{8} = 8192$$

$$BT = Binary(T) = Binary(8192) = 10000000000000$$

$$l \geq \log_2(10000000000000)$$

$$l \geq 13$$

So, the minimum number of bits that represent the maximum text length in this case is 13.

To help receiver to extract the text, some of bits must be reserved; one bit is reserved as a flag bit to indicate whether the text is flipped or not. To help the receiver to know the length of the text, so *l* is reserved. In addition, four bits are reserved as encryption key to help the receiver to decrypt the message text.

So the number of bits that must be reserved are *l*+1+4 bits and the maximum text size can be embedded in the image exactly is: Max size = $T - l - 5$.

3.3 System Analysis

The image will be distorted according to the process of hiding information, specifically, through hiding information inside image pixels, we sometimes need to change the LSB in each pixel codeword, and hence distortion of the image will occur. Such a distortion generates random errors in the original image, which we refer to as a Bit Error Rate (BER)[1]. Simply, we consider the system performance in terms of BER, which is defined as the average number of bits in the LSB of each pixel codeword that are changed through hiding the text information. Thus, our objective is to embed a high capacity of information into a cover greyscale image, *C*, with minimum BER, i.e., the average number of distortions of the image LSB's should be minimized. The following expression describes the optimization problem in which our objective is to hide as large as possible of information bits *T* (hereinafter, it is referred to as system capacity) in the LSB of the pixel codeword in the cover image *C* under the BER constraint

[1] One should notice that this BER differs from the conventional BER in the literature,

$$\begin{cases} \text{Objective} & \max_{Q}\{T\} \\ \text{Constraint} & \text{BER} \le \text{BER}_{\text{Threshold}}, \quad T \le Q \end{cases} \tag{3.3}$$

where Q is the number of the LSB's of the image pixels.

However, we should hide the sequence of text bits into a greyscale image, specifically, into the LSB of image pixels. In this context, we assume the LSB's of cover image C is **q** vector of binary bits of size Q where $Q = N/8$, the i-th element in vector **q**, say $q_i \in \{0, 1\}$ where $1 \le i \le Q$, it is clear that $q_i = 0$ if the corresponding decimal representation of the pixel is even and $q_i = 1$ if the corresponding decimal representation is odd. Similarly, the secrete message M can be seen as a vector of binary bits, **t** where $t_i \in \{0,1\}$ and ($1 \le i \le T$). Fig. 6 illustrates the text vector **t** to be hidden in the LSB vector **q**.

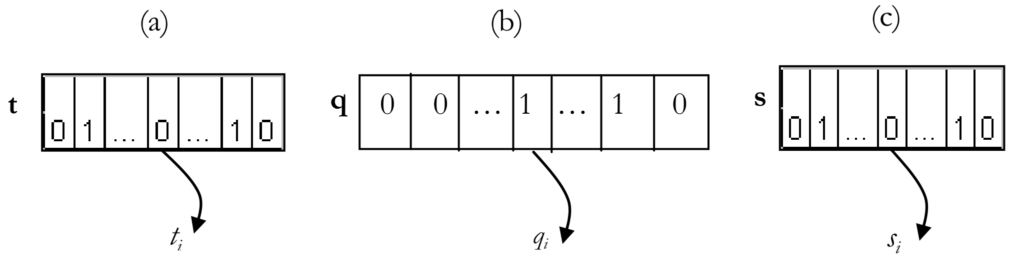

Figure 6: (a) Vector **t**, (b) Vector **q,** and (c) Vector **s**

As shown in the figure, the problem is completely simplified to hide a vector of binary bits **t** into a vector of binary bits **q** to formulate a vector of binary bits **s**

In general, the p.d.f. of a random binary sequence can be illustrated in Fig. 7, where the probabilities of occurrence of 0 and 1, in the average, are equal. Accordingly, in their binary form, similar p.d.f. and similar Entropy functions are generated for both models, namely, text and image models.

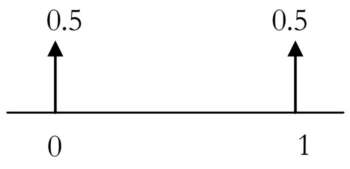

Figure 7: The p.d.f of random binary bits

Recall the entropy function and assume E_t and E_q represent the binary Entropy functions $p(0) = p$ in the vectors **t** and **q**, respectively. Without loss of generality, if the embedding process is applied, the probabilities of p and $1-p$ will affect the distribution of the image. To show the effect of entropy measurements, we first need to define the difference of entropy expressed in the distance (L) between vector E_t and E_q and the difference of probabilities (β) between P_t and P_q (see Fig. 8 below) given as

$$L = | E_q - E_t | \tag{3.4-a}$$

$$\beta = | P_q - P_t | \tag{3.4-b}$$

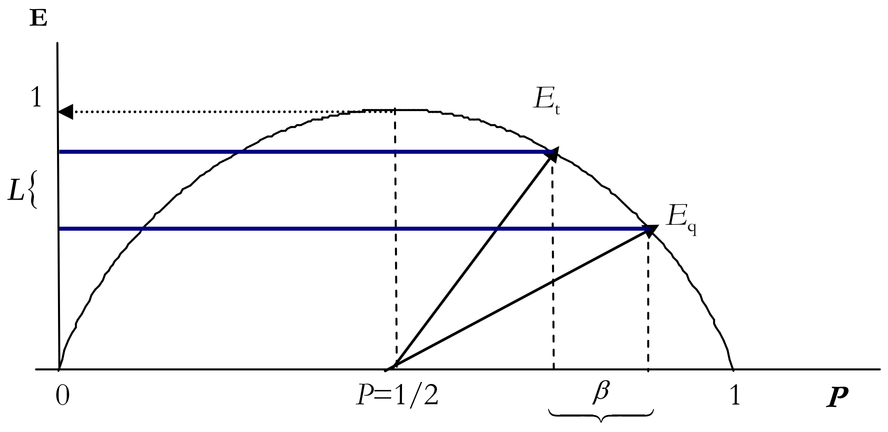

Figure 8: Entropy diff. L and probability diff. β

Cases Study:

Case 1:

When E_t and E_q are in the same half, and they are coincide. In this case the entropy difference $L = 0$ and the probability difference $\beta = 0$ too. This situation represents the best case since we can guess that the probability distribution of the bit 0 in both vectors is almost close to each other in the average. Such a result can be used to indicate that the number of 0's in text and image are approximately equal in average. Case 1 is shown in Fig. 9 below.

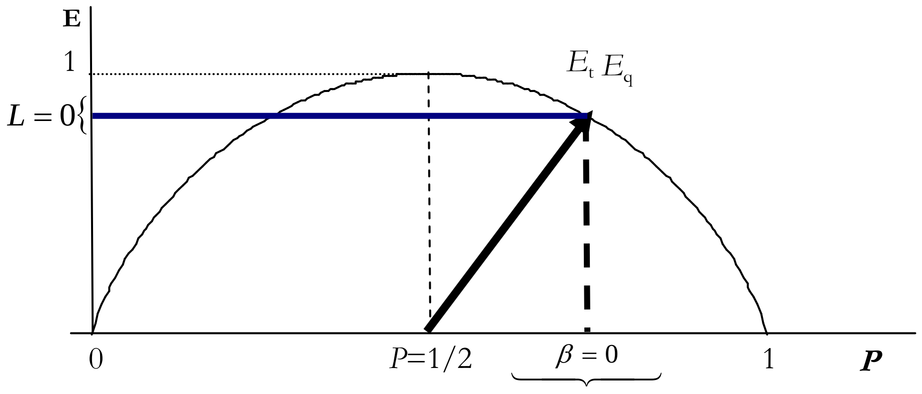

Figure 9: Case 1 in which E_t and E_q are coincide in the same half

Case 2:

If the two vectors are in the same half and they have a maximum L=1 and β = ½, this case represents the worst case since the distribution of bits in both vectors are uncorrelated. One should note that flipping process and unflipping are the same and does not make sense, because the flipping procedure will remain L=1 and β = ½, so, this case may cause large BER. Case 2 is shown in Fig. 10 below.

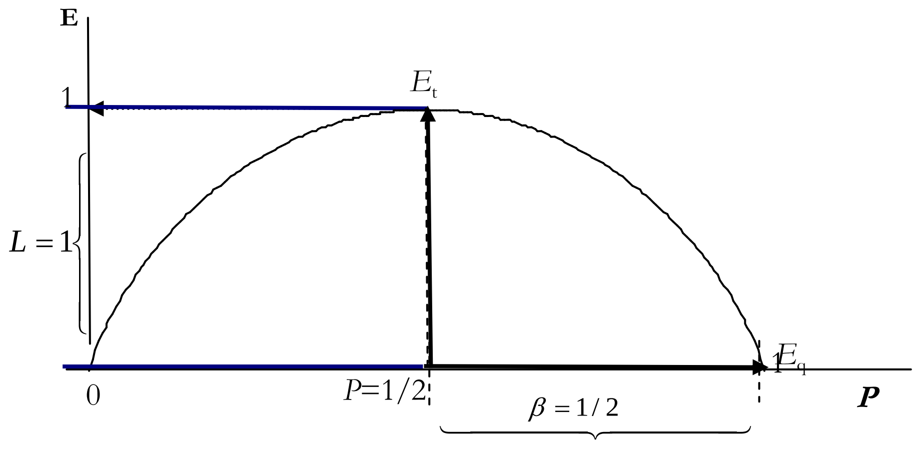

Figure 10: Case 2 in which E_t and E_q are uncorrelated

Case 3:

If the two vectors are in the same half but not in the extreme case (case 2), we always have the absolute difference of probabilities between P_t and P_q β < ½, and the entropy difference L <1. In this case the information of bit 0 in the text and image are close to each other when the entropy difference is as minimum as possible, since the probability difference β becomes minimum. So, we can achieve a good result as the two vectors close to each other until they are coincide, we reach the best case. One should note that there is no need to flip the text vector because the flipping procedure in this case increases β which leads to uncorrelated distribution of bit 0 in both vectors. Case 3 is shown in Fig. 8 above.

Case 4:

If the two vectors are not in the same half, and the entropy difference $L = 0$, this means that the distribution of 0's in the text and image are opposite of each other. One should note that $\beta \neq 0$, so it may be more than ½ or less. If $\beta \leq ½$ then the text vector should not be flipped even the flipping operation leads the best case to be occurred, because the flipping process is determined according to β value, the flipping occurs only if $\beta > ½$. Case 4 is shown in Fig. 11 below.

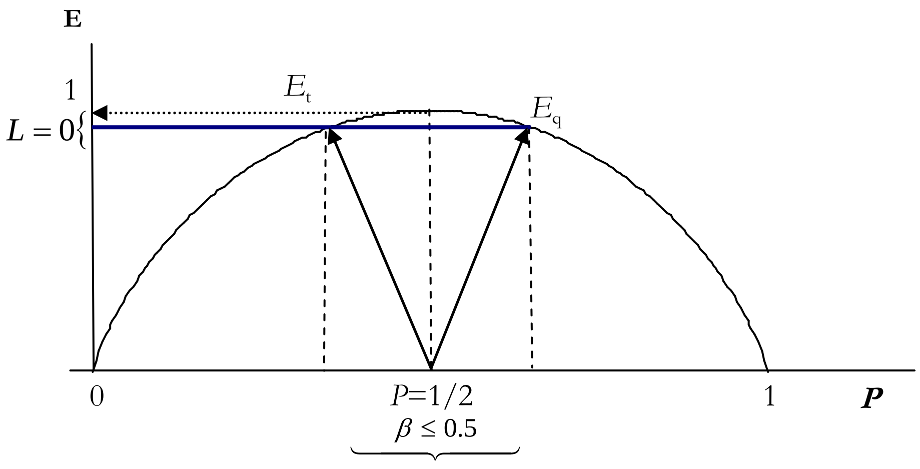

Figure 11: Case 4 in which E_t and E_q in different halves and no flip needed

Case 5:

This case is the same as case 4 but the probability difference $\beta > ½$, in this case E_t vector must be flipped Since the statistical average of bit 0 in both cases are completely opposite to each others. The flipping operation moves the vector E_t to $\bar{E}_t$ in the other half and it coincides over the image vector E_q which in turn represents case 1, the best case, since the probability difference β is flipped and minimized to $\beta' = 0$. Case 5 is shown in Fig. 12 below.

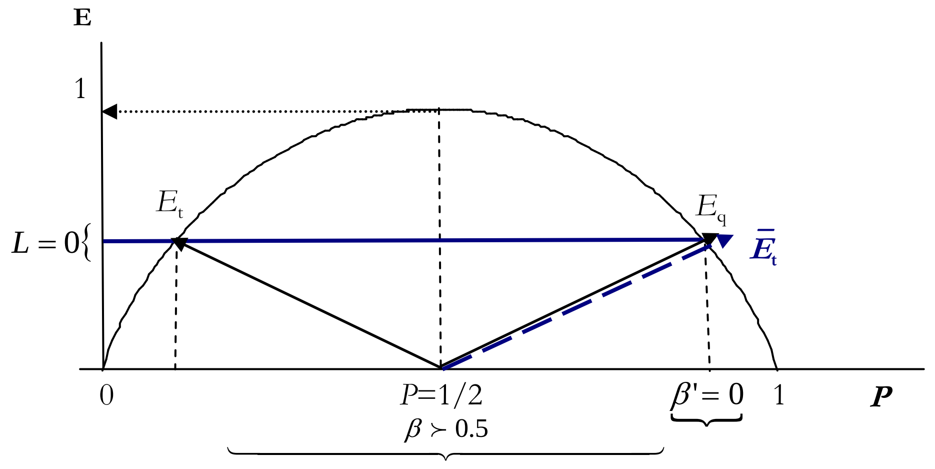

Figure 12: Case 5 in which β > 0.5 and the flip is needed

Case 6:

In this case the two vectors are in two different halves in which $0 < L < 1$, on the other hand, the probability difference $\beta \leq ½$, in this case there is no need to flip E_t even it may leads to minimum β because the flipping process determined by $\beta >$ ½, also the statistical distribution of bit 0 in both vectors is still closed to each other, such a case is shown in Fig. 13 below.

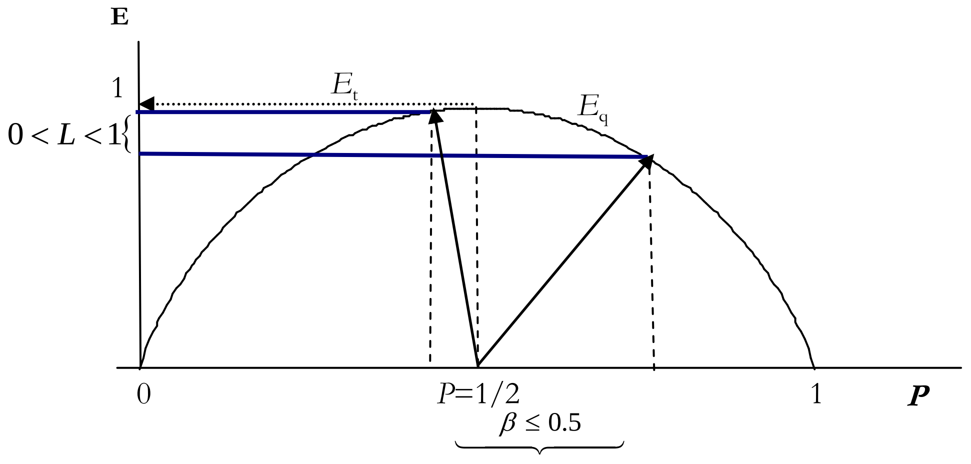

Figure 13: Case 6 in which E_t and E_q in different halves and no need to flip E_t

Case 7:

This case is similar to previous case 6 but there is a big difference, $\beta > ½$, in this case, since β is increased and exceeded 0.5 the text vector must be flipped to a new location $\bar{E}_t$ in the second half because the statistical average of bit 0 in both cases are completely uncorrelated, on the other hand β is minimized as a result of flipping procedure, i.e. the statistical distribution of bit 0 in $\bar{E}_t$ is correlated to E_q. One should note that the flipping process does not affect L but it affects β, in other words L after flipping remains as it is and β is minimized and it is changed to β'. Such a case is illustrated in Fig. 14.

Accordingly, in order to correlate the statistical distribution of bit 0 in both text and image, we need to minimize β, such a minimization is carried out by flipping process, which in turn, rotates the text vector E_t from the first half to a new position $\bar{E}_t$ in the second half.

From the figure, it is clear that β' after flipping becomes smaller than β before flipping. In this case we can easily expect that the interchange (flipping) of bit value during hiding the text information generates small BER.

From Fig. 14 one should note that when E_t is moved closely to E_q , L is increased and β is decreased, in other words, when E_t is closing to E_q this means that the distribution of bit 0 in both vectors are becomes correlated. This case was chosen to discuss in the simulation process in chapter 6.

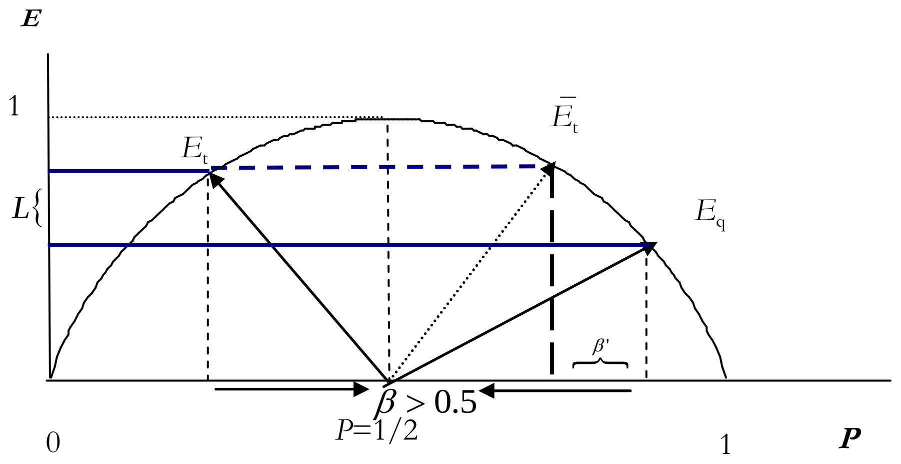

Figure 14: Minimizing β by rotating E_t to $\bar{E}_t$

Based on the aforementioned discussion, we introduce and discuss the system performance considering the hamming distance measurements, and the entropy measurements in Chapter 4 and Chapter 5, respectively.

Chapter 4

System Performance Based on Hamming Distance Measurement

In this chapter we define the system performance in terms of BER that should be calculated according to one main measurement, namely, Hamming Distance Measurement. In this measurement method, several algorithms should be tested and compared.

4.1 Hamming Distance Measurement

Due to hiding process, the embedding of text code vector **t** into image code vector **q** may cause bits distortion in the image vector. As we sometimes need to change the LSB, say q_i, of the pixel codeword when the text bit, t_i, to be hidden differs from q_i. We define a new vector **d** as the difference between the text vector **t** and image vector **q**. Such a difference can be calculated by XOR-ing **t** and **q** bit-by-bit. When $t_i=q_i$, the difference is 0, whereas we have 1 if q_i not equal t_i. That is to say, $d_i = t_i \oplus q_i$ where $1\leq i \leq D$, where D is the total number of bits in vector **d**, $\oplus$ is an XOR operation and $d_i \in \{0, 1\}$.

The weight of vector **d**, *w* (**d**) is the number of 1's in **d**, gives the so-called Hamming distance between **t** and **q** (Fearghail 2007).

The hamming distance is given by

$$w(d)=\sum_{i=1}^{T} d_i, \qquad (4.1)$$

where $d_i = t_i \oplus q_i$ and $T \leq Q$.

We use the hamming distance between the text vector **t** and the image vector **q** to determine if the text needs to be flipped or not. The flipping operation is carried out according to the value of the hamming distance. As *w* (**d**)/*D* exceeds 1/2 which means that the number of distortion is large and the text must be flipped to achieve the minimum distortion due to embedding process.

In this case the vector **t** becomes **t'** and $\mathbf{đ}_i = \mathbf{t'}_i \oplus \mathbf{q}_i$ where $1 \leq i \leq Ď$ and

$w(\mathbf{đ})/Ď < ½$ and must be less than $w(\mathbf{d})/D$

One should note that $w(\mathbf{đ})/Ď = 1 - w(\mathbf{d})/D$

But the question is, how the receiver should know that the text is flipped or not? The answer is to use the flag bit, which is a bit reserved in one of the image pixels, namely, in the last pixel of the image. The flag bit determines whether the text message *M* is being flipped or not. When the value of the weight vector *w*(**d**) (hamming distance) is greater than half of the number of image vector **q** the flag bit is set to be 1 otherwise it is 0. The following formula is used to assign the flag bit

$$\begin{cases} 1 : \dfrac{w(d)}{D} > \dfrac{1}{2}, \\ 0 : \text{otherwise} \end{cases} \qquad (4.2)$$

The flipping process reduces the value of the hamming distance which is in turn minimizes the BER. In another words, the upper bound of acceptable BER or (BER $_{\text{threshold}}$) is 50% of image vector, so we are always looking for BER < BER $_{\text{threshold}}$. However, the flag bit helps the receiver to decide on whether the text message is flipped or not.

So far, we didn't discuss the position of pixels to be used through embedding process. Moreover, the way (algorithm) of picking up the pixels that are used to hide the text message are not yet determined. This yields to ask the question, how to construct the image vector **q?**

Toward this end, we propose three different algorithms that are employed to decide which pixel is used. In the next three sub-sections, we introduce the sequential algorithm which the traditional LSB is used, odd-even algorithm which is our own method, and random walk algorithm which is also our own method but differ in the way of constructing the bearing pixels and not in the idea of randomization process.

4.1.1 Sequential Algorithm

As we mentioned before, the vector **q** represents the LSB of codeword pixels of the cover image *C*. The picking of the bearing pixels plays a big role in the stego-system and affects the embedding process. This section was dedicated to discuss the sequential way of choosing the bearing pixels.

The cover image *C* is represented as 2-dimensional matrix of real numbers used to represent the value of image pixels. In the sequential algorithm, we are tracing the image matrix horizontally row-by-row, starting from picking the first element (pixel) in the first raw of the matrix[1]. In the image, each pixel is then mapped into its ASCII binary codeword. In each codeword we allocate, sequentially, the LSB into vector **q**. The resultant **q** vector includes all LSB of the codeword pixels arranged in order one-by-one. Note that the arrangement of binary bits in vector **q** is related to the sequential algorithm of selected bearing pixels. After the vector **q** is formulated, the embedding process of the text vector **t** is done sequentially, i.e., the first bit of text vector **t,** t_1 is then embedded into q_1 in **q**. the error will occur if $t_1 \neq q_1$.

It's clear that the arrangement of the image vector **q** affects the hole embedding process and determines the BER. As we mentioned before, the BER which is based on hamming distance measurements determines whether to flip the bits in **t** or not. If $w(\mathbf{d})/D > 1/2$, then we have to flip all bits in **t**, otherwise the direct embedding (without flipping) will generate BER ≤ $\text{BER}_{\text{threshold}}$.

Some numerical examples of the BER are given in Chapter 6. Aiming at reducing BER, other algorithms are employed, first, we relax our assumption of sequentially picking up the pixels into an odd-even algorithm in which the odd pixels are selected first, then the even pixels.

4.1.2 Odd-Even Algorithm

In this section, we discuss another algorithm of picking up the bearing pixels called the odd-even algorithm. In this algorithm we select the pixel of the first position (of course, the first after the overhead pixels) and left the next one. If all odd positions are traced and there LSB's are allocated into vector **q**, then the even positions are similarly traced and allocated in the empty spaces after the allocation of odd pixels into **q.** Note that the tracing of image pixels is carried out row-by-row of the cover matrix *C* in the way such that take-and-leave. When the vector **q** is formulated using odd-even algorithm, the embedding process of vector **t** is executed sequentially (bit-by-bit), and the hamming distance-based BER measurements are used to determine whether to flip t or not.

[1] One should grasp that the first pixels are left as overhead to help extracting the text.

Finally, the last adopted algorithm is to pick up the pixels in a random fashion known to both transmitter and receiver. Such an algorithm is to be discussed in the following sub-section.

4.1.3 Random Walk Algorithm

In this section, we should discuss the last algorithm of picking up the bearing pixels which is referred to as random walk algorithm. The pixels in the image matrix are traced randomly according to a specific function known to both sides, transmitter and receiver. In random walk algorithm, we assume *S* to represent the total number of pixels in the image (cardinality of the matrix) seen as a matrix of order $a \times b$, given by

$$S = a \times b\,. \tag{4.3}$$

We need now to encode the position of each pixel using a binary codeword of maximum codeword length given by

$$K = \log_2 S\,. \tag{4.4}$$

Each position of the pixel, *PN* is encoded from its decimal value to the corresponding binary code. Let the pixel position in a matrix of order $a \times b$ is given by a pair (x, y), where $1 \leq x \leq a$, and $1 \leq y \leq b$. The nth position of the pixel *PN*, located at (x, y) has a decimal value given by

$$PN = b(x-1) + y\,. \tag{4.5}$$

Using this *PN* decimal number to encode the position to the corresponding binary codeword of length *K*, as discussed before.

Accordingly, each pixel position in the image *C* is represented by *K*-bits codeword.

Fore example, the Lenna image shown in Fig. 15 includes $S = a \times b = 256\times256$ pixels.

Figure 15: Lenna image

The required number of bits to represent all pixels positions in the Lenna image shown in Fig. 15, S, is $K = 16$ bits, i.e. each pixel position in the image can be represented using 16 code length, e.g., Pixel number 1, PN =1 is represented as 0000000000000001, PN =2 is represented as 0000000000000010, and similarly PN = 65536 (256×256) is represented as 1111111111111111.

To choose the pixel in the random walk algorithm, we read the first representation from write to left, i.e. pixel number 1 becomes pixel number 32768 which represented as 1000000000000000 and pixel number 2 becomes pixel number 16384 that represents 0100000000000000, and so on. One should note that this algorithm does not depend on a pseudo number generator, but it is closer to a rhythmic flip-flop method. So, the vector **q** can be formulated by allocating the LSB of each visited pixel. Remember that the embedding of vector **t** into vector **q** is done sequentially, and the BER based on the hamming distance measurements is used to flip **t** or not.

After all algorithms are applied in the hamming distance measurements, we should adopt the best algorithm. Assuming that the sequential algorithm reach some value of $BER < BER_{threshold}$, we didn't stop, we should try the other algorithms in order to reach the minimum BER. One notes that, if an algorithm is adopted for a specific cover image and a specific text, it is not necessarily adopted for another image and another text. So the hall algorithms should be applied, then a comparison between the three algorithms should be carried out and then, an optimum algorithm should be adopted to achieve the best system performance.

Finally, the best BER i.e. the minimum BER of the three algorithms should be applied in the stego-system.

Chapter 5

System Performance Based on Entropy Function Measurement

5.1 The Entropy Function

In this chapter the entropy function measurements are used to calculate the system performance. The entropy function is a measure of the amount of information required on the average to describe the random variable. Shannon in his seminal work (Shannon, 1948) identifies the entropy in terms of information amount of uncertainty.

$$I = p \log_r \frac{1}{p} \quad , \tag{5.1}$$

where p is the probability of the information occurred and r is the logarithmic base which is in our case, $r = 2$.

As we mentioned before, we have a text vector **t** of binary bits 0's and 1's, furthermore, we have an image vector **q** of 0's and 1's too. Aiming to enhance the system performance, we are looking for a relationship between the statistical behavior of bit, say, 0 in the text vector **t** and image vector **q**. without loss of generality, one should note that the analysis are applicable for either bit 0 or 1.

The embedding process of the text vector **t** into image vector **q** to conduct the stego-image vector **s** is illustrated in Fig. 16,

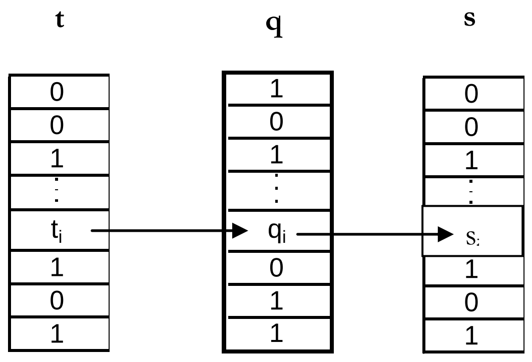

Figure 16: The transition process of **t** into **q** to form **s**

where we assume that each bit of **t** vector is embedded into its corresponding bit of **q** vector and results in **s** vector, i.e., t_i embedded into q_i. to generate s_i.

Fig. 16 shows the embedding process of **t** into **q**. Through such a process, the value of q_i might be changed to s_i depending on the value of t_i. Recall that if $t_i=q_i$ no change in q_i, on the contrary, the value of t_i replaces the value of q_i if they are different; in this case an error is generated. In telecommunication literature such an embedding process can be seen as a transmitter (text), receiver (image), and noise (flipping). The noisy channel is known as a cascaded Binary Symmetric Channel (BSC) (Haykin, 1993) shown in Fig. 17, in which the bits 0 and 1 in **t** occurs with $P_t(0)$ and $P_t(1)$, respectively. In image vector **q**, the bit 0 has the probability $P_q(0)$ and the bit 1 has the probability $P_q(1)$. Similarly, in stego-image vector **s**, $P_s(0)$ and $P_s(1)$.

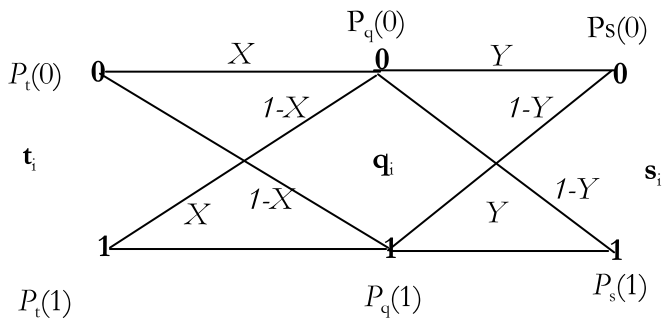

Figure 17: Cascaded Binary Symmetric Channel (BSC)

Note that,

$$P_t(0) = \frac{N(0)}{T}, \qquad \text{and} \qquad P_t(1) = \frac{N(1)}{T}, \tag{5.2}$$

where *N(i)* denotes the number of bit *i* and *T* is the total number of bits in **t**. The probabilities given in (5.2) are known and can be easily measured from **t**, hence, the entropy of bits 0 and 1.

Similarly,

$$\begin{aligned} P_q(0) &= \frac{N(0)}{Q} \text{and}, P_q(1) = \frac{N(1)}{Q} \\ P_s(0) &= \frac{N(0)}{S} \text{and}, P_s(1) = \frac{N(1)}{S} \end{aligned} \tag{5.3}$$

Assume that the embedding process generates error when 0 is converted into 1 and 1 is converted into 0. Let *X* denotes the probability of having 0 in **q** given that **t** generates 0 and let *Y* denotes the probability of having 0 in **s** given that we have 0 in **q**, which is the case of no errors (correct decision). Moreover, let *1-X* is the probability of having 0 in **q** given that **t** generates 1 and 1-*Y* is the probability of having 0 in **s** given that we have 1 in **q**, which is the case of an error (error decision). Similarly for bit 1.

We can develop a transition matrix **Λ** that represents the aforementioned probabilities given by

$$\mathbf{\Lambda} = \begin{bmatrix} X & 1-X \\ 1-X & X \end{bmatrix} = \begin{bmatrix} P(0/0) & P(0/1) \\ P(1/0) & P(1/1) \end{bmatrix} \tag{5.4}$$

Note that the summation of each row is 1. From the total probability definition, (Haykin, 1993) the probabilities of bits 0 and 1 in **q** are given by

$$\begin{aligned} P_q(0) &= P_t(0)X + P_t(1)(1-X) \\ P_q(1) &= P_t(0)(1-X) + P_t(1)X \end{aligned} \tag{5.5}$$

In matrix notation,

$$\begin{bmatrix} P_q(0) \\ P_q(1) \end{bmatrix} = \begin{bmatrix} X & 1-X \\ 1-X & X \end{bmatrix} \begin{bmatrix} P_t(0) \\ P_t(1) \end{bmatrix}$$

$$\mathbf{P_q} = \Lambda \mathbf{P_t} \tag{5.6}$$

Solving the system of two dimensional linear equations in (5.6) for the conditional probabilities *X* and 1-*X*, we have

$$X = \frac{P_q(0) - P_t(1)}{P_t(0) - P_t(1)},$$

$$X = \frac{P_q(1) - P_t(0)}{P_t(1) - P_t(0)} \tag{5.7}$$

$$X = \frac{P_q(0) + P_t(0) - 1}{2P_t(0) - 1}$$

Similarly, for the stego-image *S*,

$$\begin{aligned} P_s(0) &= P_q(0)Y + P_q(1)(1-Y) \\ P_s(1) &= P_q(0)(1-Y) + P_q(1)Y \end{aligned} \tag{5.8}$$

By solving this linear equation we obtain,

$$Y = \frac{P_s(0) - P_q(1)}{P_q(0) - P_q(1)}$$

$$Y = \frac{P_s(1) - P_q(0)}{P_q(1) - P_q(0)} \tag{5.9}$$

After solving (5.9)[1] we obtain

$$Y = \frac{P_s(0) + P_q(0) - 1}{2P_q(0) - 1} \tag{5.10}$$

Recall the entropy function[2], we assume $E_t(0)$, $E_q(0)$, $E_s(0)$ are the entropies of a bit 0 in vectors **t** , **q** and **s** respectively.

$$E_t(0) = \sum_{i=0}^{1} P_t(i) \log_2 \frac{1}{P_t(i)} \tag{5.11}$$

$$E_q(0) = \sum_{j=0}^{1} P_q(j) \log_2 \frac{1}{P_q(j)} \tag{5.12}$$

$$E_s(0) = \sum_{k=0}^{1} P_s(k) \log_2 \frac{1}{P_s(k)} \tag{5.13}$$

Note that $E_t(0)$ is

$$\begin{aligned} E_t(0) &= P_t(0) \log_2 \frac{1}{P_t(0)} + P_t(1) \log_2 \frac{1}{P_t(1)} \\ &= P_t(0) \log_2 \frac{1}{P_t(0)} + (1 - P_t(0)) \log_2 \left(\frac{1}{(1 - P_t(0))} \right) \end{aligned} \tag{5.14}$$

Similarly for $E_q(0)$ and $E_s(0)$.

The results of calculation based on the Entropy function measurements are given in the next chapter. One should note that the Entropy measurements given in this chapter is independent on the used algorithm, that is why we will apply these measurements only on the random walk algorithm to construct the vector **q**.

[1] See Appendix A

[2] See Eq. (3.1) in chapter 3

Chapter 6

Simulation and Experimental Results

In this chapter, we first consider the hamming distance-based measurements, and then we introduce the entropy-based measurements.

6.1 Hamming Distance-Based Measurements

In the hamming distance measurements, we used three different algorithms to achieve the least BER. When they are applied in the experiment, our duty is to judge which algorithm gives lower BER than others for different texts and one specific image.

We adopt the method based on hamming distance measurements to decide on the best used algorithm to pick up or to construct the vector **q** from a given specific image. The steps of our method are defined below:

1) Assign a fixed size specific image that is used to formulate the **q** vector used to hide the text. In our example, we used a Lenna image of size 256×256 pixels. The generated **q** vector from Lenna image is of fixed size which is 8 KB.

2) Construct the vector **t,** serially bit-by-bit, from the ASCII encoded text of different sizes. We assume a text of 100 byte, then incremented by 100 byte every time we execute the measurements.

3) Construct the vector **q** from Lenna image using the three different algorithms. The resultant 3 **q** vectors are different of the locations of zeros and 1s.

4) For the text of size 100 bytes, we calculate the hamming distance between **t** and **q**, once without flipping of vector **t** and the other with flipping of vector **t**.

5) Calculate the BER of each algorithm without and with flipping the vector **t**. The BER is calculated using the weight of the hamming distance vector divided by the length of **t** vector. One should note that for such a fixed text size the BER is given by

$$\text{BER} = w(\mathbf{d})/T \tag{6.1}$$

6) The values of BER of each algorithm without flipping and with flipping, for different text sizes, are tabulated in Table 1 given below. The process is repeated for each text size incremented by 100 bytes each time[1].

[1] In the table, we use increment of 500 to skip the problem of table size.

Table1: Text size versus BER of the three algorithms

Text Size (Bytes)	Image vector	Sequential		Odd-Even		Random Walk	
		BER		BER		BER	
		Flip	No Flip	Flip	No Flip	Flip	No Flip
500	8 K	0.4862	0.5138	0.2585	0.7415	0.0325	0.9675
1000	8 K	0.4902	0.5098	0.2565	0.7435	0.0628	0.9372
1500	8 K	0.4898	0.5102	0.2558	0.7442	0.0972	0.9028
2000	8 K	0.4898	0.5102	0.2558	0.7442	0.1284	0.8716
2500	8 K	0.4905	0.5095	0.2538	0.7462	0.1590	0.841
3000	8 K	0.4911	0.5089	0.2533	0.7467	0.1834	0.8166
3500	8 K	0.4924	0.5076	0.2527	0.7473	0.2161	0.7839
4000	8 K	0.4938	0.5062	0.2526	0.7474	0.2463	0.7537
4500	8 K	0.4942	0.5058	0.2969	0.7031	0.2781	0.7219
5000	8 K	0.4975	0.5025	0.3408	0.6592	0.3102	0.6898
5500	8 K	0.4946	0.5054	0.3806	0.6194	0.3394	0.6606
6000	8 K	0.4963	0.5037	0.4084	0.5916	0.3691	0.6309
6500	8 K	0.4958	0.5042	0.4387	0.5613	0.3980	0.602
7000	8 K	0.4982	0.5018	0.4567	0.5433	0.4278	0.5722
7500	8 K	0.4975	0.5025	0.4781	0.5219	0.4611	0.5389
8000	8 K	0.4987	0.5013	0.4945	0.5055	0.4904	0.5096

In Fig. 18 we show the relation between the flipping and non-flipping techniques in sequential algorithm.

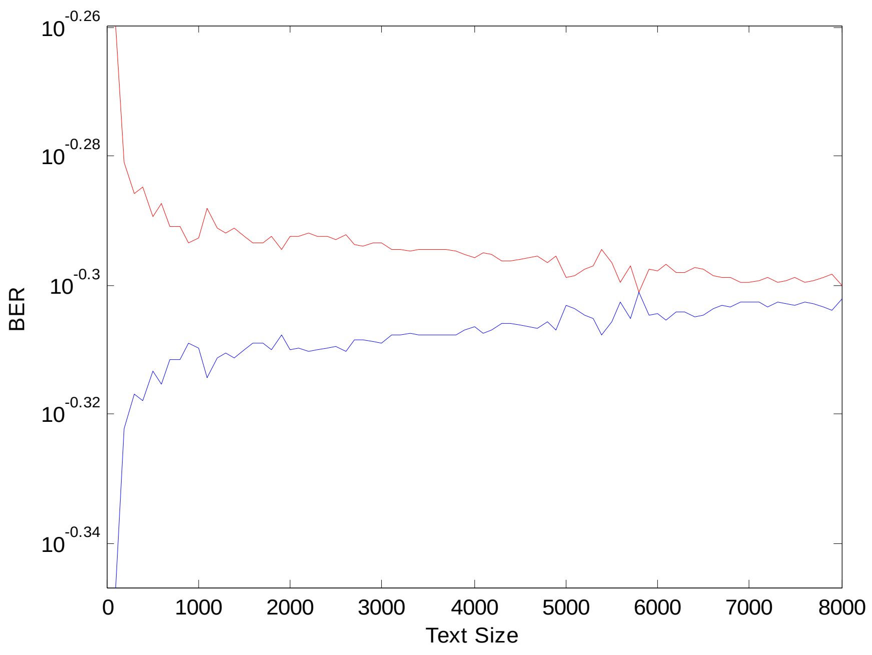

Figure 18: Text size Vs. BER for sequential algorithm with flip and no flip

The larger the text size, the more BER we have. With flip, it is noticed that flipping behaves better, especially, in the case of small size text, however, when the text size goes larger, we see that the flipped text and the unflipped one are almost the same. That is to say, the flipping/no flipping process of large text approaches almost the same BER.

The algorithm's behaviour manifests itself in small texts more effectively than larger texts. Also, when the text size is so large, BER increases but its change becomes so slow.

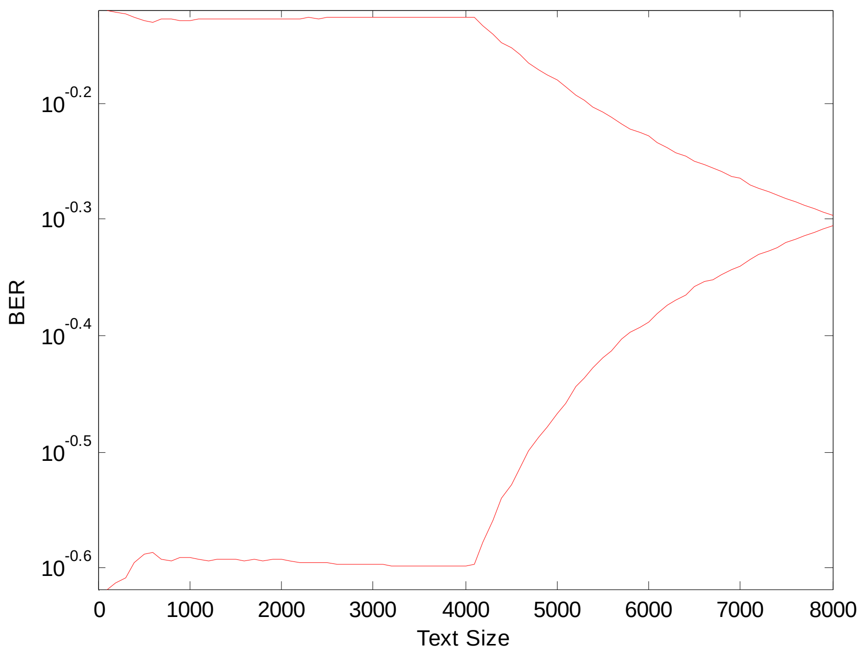

Figure 19: Text size with odd-even flip and no flip algorithms

Similarly, in odd-even algorithm, the BER increases with the text size in case of no flipping as shown in Fig. 19. When the text size goes larger and larger, the flipped and unflipped texts appear to be almost the same. This algorithm seems to be more practical than sequential algorithm especially in the first half of text size because the gap between flipped and unflipped texts is remarkable; hence, the flipping process is so effective in this algorithm. However, it should be mentioned that it is not difficult to predict the algorithm's behaviour when the text size goes larger and larger. That is to say, both flipping and unflipping of the text yields almost same BER performance.

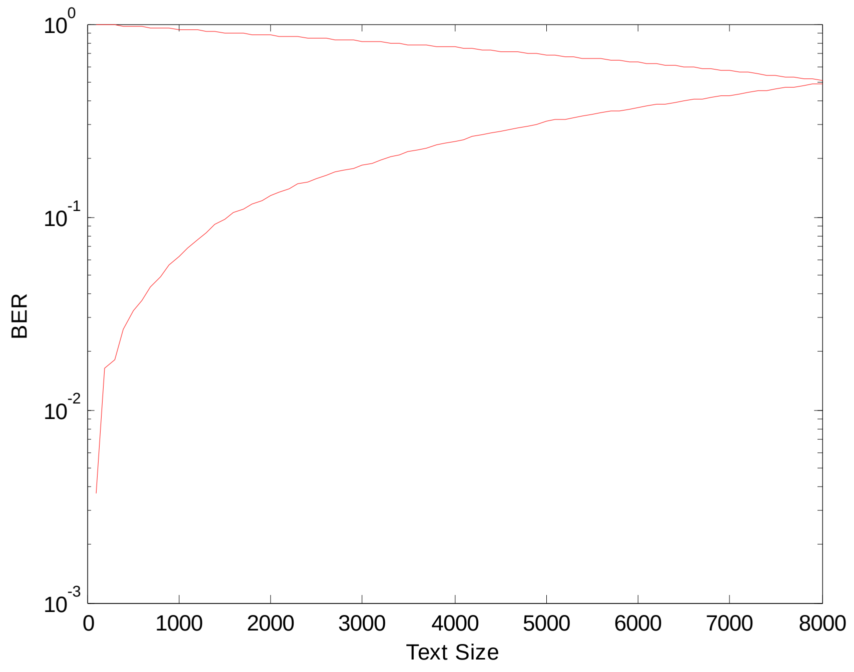

Figure 20: Text size with random walk flip and no flip algorithms

By the same token, as shown in Fig. 20, the BER increases with text size in random walk algorithm. This shows that the larger the text, the less effective the flipping process is.

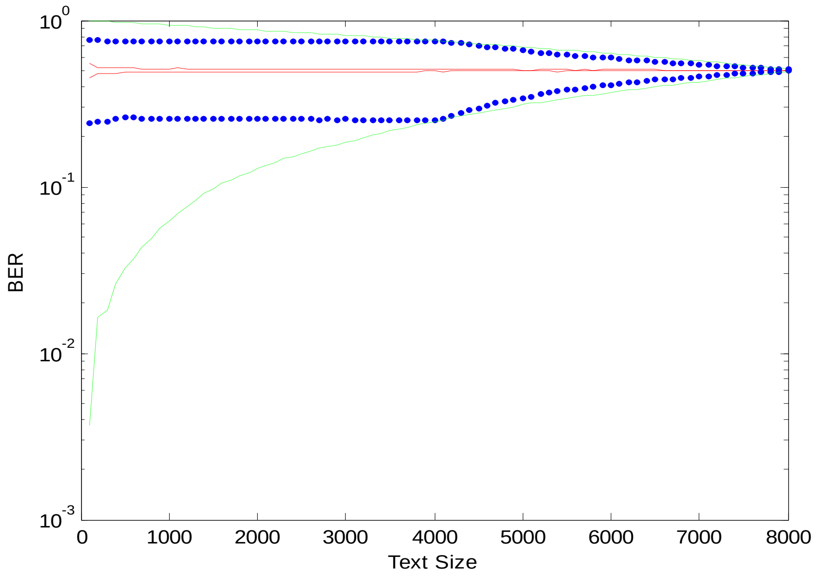

Figure 21: Text size with all flip and no flip algorithms

For comparison purposes, Fig. 21 shows all algorithms, flipped and unflipped. It can be seen that the sequential algorithm (red color) is the least effective as it shows a slight gap between flipped and unflipped texts, which means, the flipping process adds no noticeable effective value on the BER performance comparable with others algorithms. As for the odd-even algorithm (blue color), it is more effective as it shows a wider and higher gap between the flipped and unflipped texts. The random walk algorithm (green color) is the most effective one because it exhibits so huge gap between the flipped and unflipped texts. Also, this algorithm distinguishes itself from other algorithms in that it distributes the errors or distortions in all parts of the image and not in one part or specific area of the image as it occurs in other algorithms, because, the bearing pixels are chosen randomly, thus, in case of error, it will be scattered over the image. Our results suggest that the Random-Walk algorithm is the most proper algorithm to hide a text message into a grey-scale image.

6.2 Entropy-Based Measurements

In this section, the relation between the entropy that is defined as the average amount of information, (Wikipedia®, 2007) and the BER should be introduced. One should note that the entropy is a measure of the numbers of 0's and 1's in vectors **t** and **q**, and does not show how 0's and 1's are arranged. Accordingly, it is worthy to stress the fact that the measurements here are independent on how the vectors are arranged, i.e., in the entropy measurements, the three algorithms don't affect the entropy-based measurements. For simplicity and without loss of generality, we will adopt the random walk algorithm[1] to generate the **q** vector.

In this section, the entropy of bit 0 was calculated for each text size and for the LSB of a specific image; vector **q** was formulated using random walk algorithm. After that, the entropy difference (*L*) between the text vector **t** and the image vector **q** was computed for each text size. Table 2 shows these calculations are related to text size and the BER.

Table 2: Text size with the entropy, entropy difference and BER

Text Size	Text Entropy $E_t(0)$	Entropy Diff. (*L*)	BER
500	0.9915	0.0085	0.0325
1000	0.9927	0.0072	0.0628
1500	0.9926	0.0073	0.0972
2000	0.9934	0.0065	0.1284
2500	0.9932	0.0067	0.1590
3000	0.9935	0.0065	0.1834
3500	0.9936	0.0064	0.2161
4000	0.9937	0.0063	0.2463
4500	0.9940	0.0060	0.2781
5000	0.9940	0.0061	0.3102
5500	0.9942	0.0058	0.3394
6000	0.9943	0.0056	0.3691
6500	0.9946	0.0054	0.3980
7000	0.9945	0.0054	0.4278
7500	0.9945	0.0055	0.4611
8000	0.9946	0.0054	0.4904

Based on the above calculations, we introduce several graphs to show how the entropy function behaves related to text size, BER, etc.

[1] Note that our stego-system checks all algorithms and uses the best one.

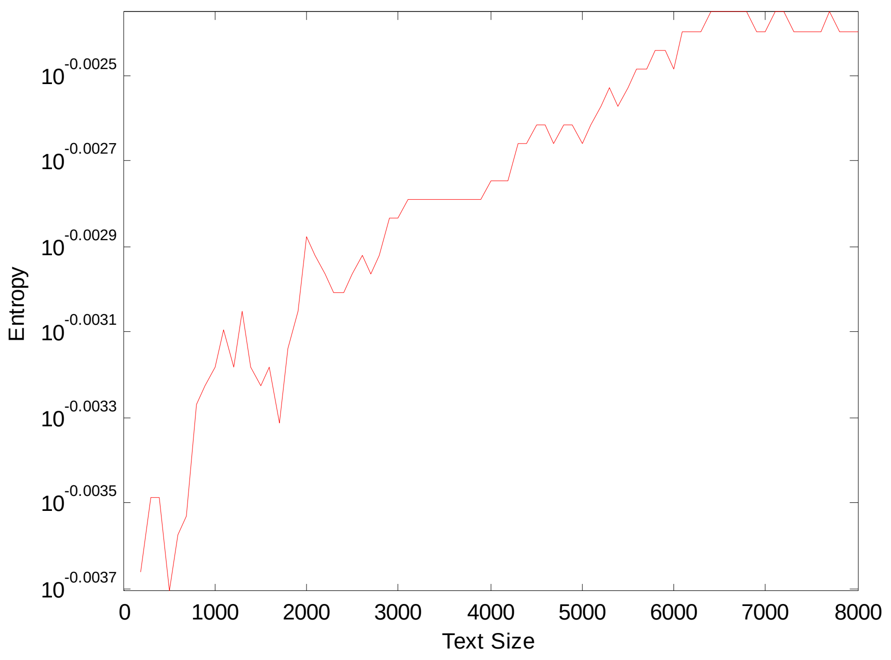

Figure 22: Text size vs. entropy

Fig. 22 shows the expected result of the entropy of the text versus the text size. As the entropy is defined as the average amount of information of certain source, we expect the increasing of the entropy as the text size increases.

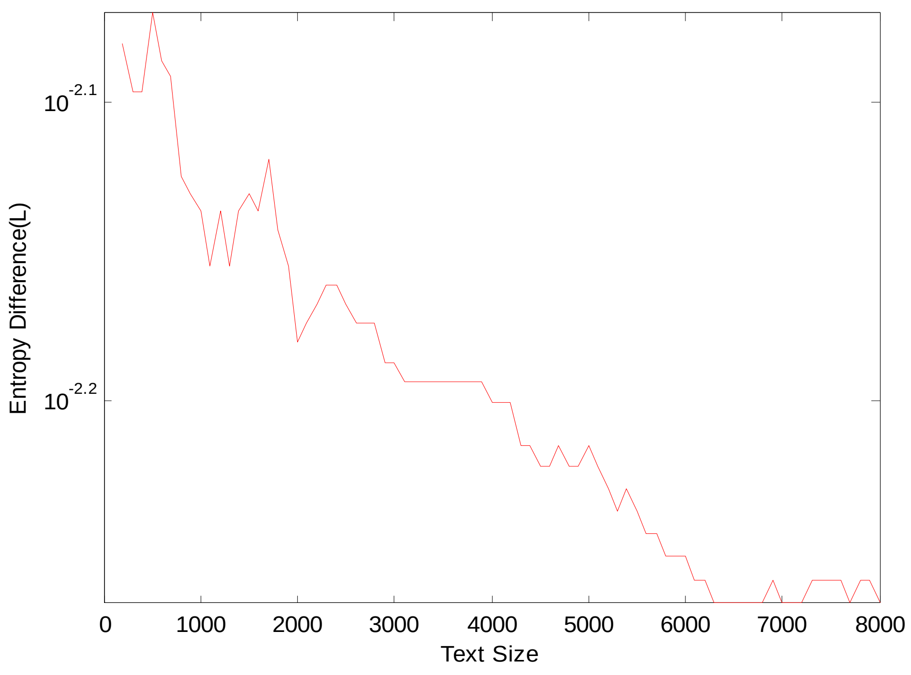

Figure 23: Text size with entropy difference between vector **t** and vector **q**

In Fig. 23, we show the relationship between L (the absolute difference between the entropy of text vector **t** and the entropy of image vector **q**), and the text size. As shown in the figure, the difference L decreases with text size almost exponentially. It is worthy to note that for small text size, the difference is noticeable and it becomes smaller with larger text. Aiming at explaining such a result, when L is small, this means that $E_t(0)$ and $E_q(0)$ are almost equal and L becomes 0 when $E_t(0) = E_q(0)$. Such case happened in large text size, since the distribution of 0's and 1's, on the average, becomes the same.

It is important to note that our entropy-based measurements show that, for large text size, the difference becomes zero, which means that the two vectors **t** and **q** almost have the same distribution of zeros and ones. Such a result coincides with that obtained in hamming distance based measurements, where we showed that for large text size, the probability of error (BER) approaches half, which means flipping/unflipping becomes not significant to be considered. This result also explains the behaviour of the BER in large text size, where we showed that BER in large text size approaches ½ as T approaches infinity, however, in entropy-based measurements, we showed that as the text size approaches infinity, L approaches zero, which means that $E_t(0)$ equals $E_q(0)$, i.e., the 0 and 1 distribution becomes similar that is equally likely in large scale text size. Accordingly, we conclude that for $L = 0$, we have equal probable distribution of 0's and 1's that results in BER being independent on neither algorithm nor flipping process. We can say, these results coincide with the hamming distance measurements given in Chapter 4, which is one of the main objectives of our research.

Recall the case 7 from the cases study in Chapter 3, in that case we showed that when the vector **t** is moving closely to vector **q** the entropy difference L is increased and the probability difference β is decreased, the aforementioned case is applicable and matched with the result in Fig. 24. In this figure the value of BER is decreased while the entropy difference L is increased, since when the entropy is increased the vector E_t is closing to E_q which means that the statistical distribution of bit 0 in both vectors is becoming correlated to each other. So these results explain exactly case number 7.

The entropy-based measurements give a first-shot on deciding whether the assigned image is suitable or not for a certain text to be embedded in, by measuring L, if L is relatively large, the first indication we have is to employ the algorithm and flip/unflip selection to have the lower BER, on the contrary, for relatively small L, the BER approaches ½, and no significance to employ selection criterion.

For example, if the required BER is smaller than 0.1, the entropy difference L must be larger than, approximately, 7×10^{-3}. The example is taken from Fig. 24 shown below. In the figure, we show the relationship between the entropy difference and the BER

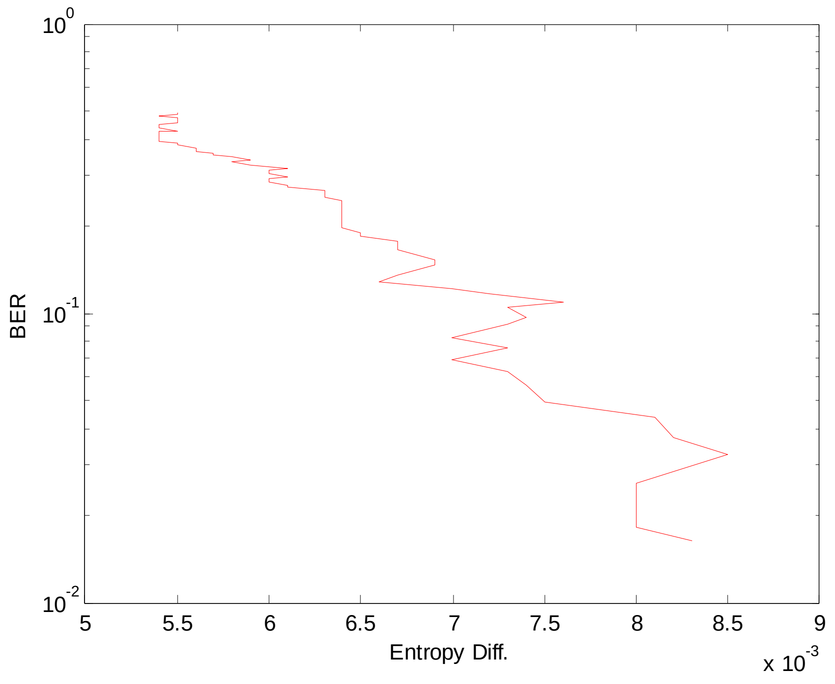

Figure 24: Entropy diff. with BER

Fig. 25 shows the relation between the BER and the entropy difference *L*. In this case we use a fixed size of text data with different image sizes; we note that the BER is symmetrically around 0.5. If the BER goes large and exceeds the $BER_{threshold}$ the flipping process is necessary to keep BER as minimum as possible. The following graph shows the relation between the entropy difference *L* and the BER. From the figure we can say that the flipping process always keeps the BER under $BER_{threshold}$.

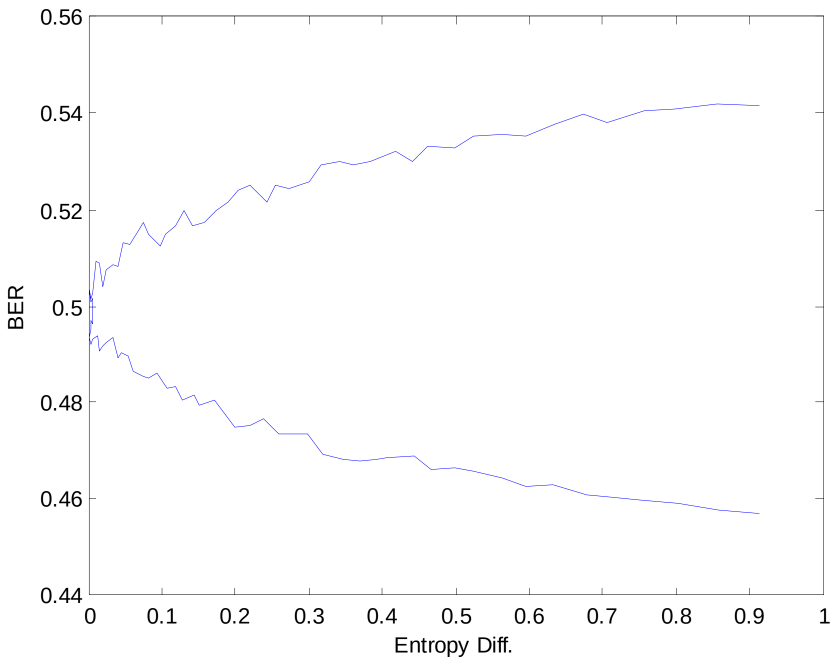

Figure 25: Entropy diff. of different images with fixed text size and BER

Chapter 7

Conclusions

7.1 Concluding Main Results

In this study, it was found out that several texts can be embedded into the image using Flipping Embedded Text technique (FET) with minimum BER based on a newly proposed technique, namely, entropy function measurement. Our main results can be concluded as:

1. We found out that among the three embedding algorithms, the random walk algorithm on the average behaves better than other two algorithms especially in small text sizes as shown in Fig.21. The FET technique is used to minimize the number of distortions and the artefacts occurred in the cover image through embedding process.

2. It was concluded that there is a relation between the entropy based measurements and the BER.

3. It is shown through our results that if the BER increases and exceeds a pre-specified threshold value (0.5), the FET technique might be used to ensure that the BER is always less than $BER_{threshold}$.

4. The study concluded that the BER increased if the entropy difference between the image entropy and the text one was decreased. We can say that the entropy measurement is useful in studying the distribution of bits 0's and 1's in both image and text as a first indication to determine whether the image distribution of bits is suitable for a specific text or not.

7.2 Open Problems and Future Work

Although there are a lot of techniques that can be employed to enhance our proposed system and several questions were raised during our study, we mention some of them:

1. The study suggested using a compression text to enable the sender to embed a large text data into the image.

2. As a future work, the researcher is encouraged to use methods of error detection codes and error correction codes to overcome the changes happened through communication.

3. We are encouraged to use the frequency domain embedding rather than spatial domain to make the stego system more robust and to keep the message without any distortions because of the image processing such as image compression, cropping, sampling etc…

4. In this research, the entropy-based measurements, as we discussed, can be helpful not only to determine if the image is suitable for a specific text, but also to determine and to choose the best image from number of images that are suitable for the appropriate text. More deep analysis is required to take the full benefits of the entropy concepts. For example, researchers can use the conditional entropy; relative entropy to determine the most suitable image for the specific text, i.e. the text can be embedded in this image with the lowest BER.

Bibliography

Al-Jaber, A, Aloqily, I. (2003): “High Quality Steganography Model with Attacks Detection” Asian Network for Scientific Information. Pakistan Journal of Information and Technology 2 (2):116-127, ISSN 1682-6027.

Alkhraisat H. (2005): “4 least Significant Bits Information Hiding Implementation and Analysis” GVIP 05 Conference, 19-21 December 2005, CICC, Cairo, Egypt.

Armand, M. (2003): “On the use of analogies in a first course on information theory” World Transactions on Engineering and Technology Education. Vol. 2, No. 3, 2003 UICEE.

Avcibas, I, Memon, N, Sankur, B. (2003): “Steganalysis Using Image Quality Metrics” IEEE Transactions On Image Processing, Vol. 12, No. 2, February 2003.

Bret D. (2002), “A detailed look at Steganographic Techniques and their use in an Open-Systems Environment”. SANS Institute.

Cachin Ch. (2005): “Digital Steganography”. IBM Research, Zurich Research Laboratory, CH-8803 Ruschlikon, Switzerland, February 17, 2005.

Chandramouli, R, Memon N. (2001): “Analysis of LSB based image steganography techniques”. Proc. of ICIP, Thessaloniki, Greece.

Cover, Th, Thomas J. (1991): Elements of Information Theory, John Wiley & Sons, Inc. ISBN 0-471-20061-1.

Cummins, J, Diskin, P, Lau, S, Parlett, R. (2004): “Steganography and Digital Watermarking”, School of Computer Science, the University of Birmingham.

Cvejic, N. (2004): “Algorithms for audio watermarking and steganography” Department of Electrical and Information Engineering, Information Processing Laboratory, University of Oulu, Finland.

Davidson I, Goutam P. (2004): “Locating Secret Messages in Images” Computer Science, ACM 1-58113-888-1/04/0008.

Fearghail, S. (2007): Information Theory and Coding, Dublin Institute of Technology (www.electronics.dit.ie/staff/sofearghail/Courses/DT080/AD_Telecoms/Information%20and%20coding.doc 08.08.2007).

Fridrich J. (2006): “Minimizing the Embedding Impact in Steganography” Conference ACM Multimedia Security’06, Sep 26–27, 2006, Geneva, Switzerland -58113- 000-0/00/0000.

Fridrich J, Goljan M and Soukal D. (2004): “Searching for the Stego-Key” DBLP: conf/sswmc/2004. Security, Steganography, and Watermarking of Multimedia Contents, p 70-82.

Gonzalez, R, Woods, R, Eddins, S. (2004): Digital image processing using MATLAB, 1’st Edition, Prentice Hall, ISBN 0130085197 USA.

Hany F, Siwei L. (2006): "Steganalysis Using Higher-Order Image Statistics". IEEE. Transactions on Information Forensics and Security, Vol. 1, No. 1. 1556-6013.

Haykin, S. (1993): Communication Systems, 3rd Edition. John Wiley & Sons, Inc. New York.

Honeyman P, Provos N. (2003): “Hide and Seek: An Introduction to Steganography” IEEE Computer Society. 1540-7993/03.

Judge, J. (2001): “Steganography: Past, Present, Future”. SANS Institute. As a part of the Information Security Reading Room.

Kessler, G. (2004): “An Overview of Steganography for the Computer Forensics Examiner” Computer and Digital Forensics Program Champlain College Burlington, Vermont. Forensic Science Communications July 2004 – Volume 6 – Number 3.

Kharrazi M, Husrev T, Memon N. (2004): “Image Steganography: Concepts and Practice” WSPC, Polytechnic University, Brooklyn, NY 11201, USA. April 22, 2004 1:49.

Moerland, T. (2003): “Steganography and Steganalysis” May 15, 2003.

Morkel T, Eloff J, Olivier M. (2005): “An Overview Of Image Steganography”. Fifth Annual Information Security South Africa Conference (ISSA2005), Sandton, South Africa, June/July 2005.

Moskowitz, I, Longdon, G, Chang, L. (2000): “A New Paradigm Hidden in Steganography” US Government work. Research supported by the Office of Naval Research. NSPW2000, Ballycotton, Co. Cork, Republic of Ireland.

Neil F, Jajodia S. (1998-a): “Exploring Steganography: Seeing the Unseen”. George Mason University. IEEE Computer.

Neil F, Jajodia S. (1998-b): “Steganalysis: The Investigation of Hidden Information”. IEEE Information Technology Conference, Syracuse, New York, USA, September 1st - 3rd, 1998.

Neil, F, Katzenbeisser S. (1999): “A survey of Steganographic Techniques – Information Hiding techniques for steganography and watermarking Chapter 3”.

Network Associates, Inc (1990-1998): An Introduction to Cryptography. PGP*, Version 6.0.2 printed in the USA. (www.eie.fceia.unr.edu.ar/ftp/Comunicaciones/AN%20INTRODUCTION%20TO%20CRYPTOGRAPHY.PDF).

Petitcolas, F, Anderson, R, Kuhn, M. (1999): “Information Hiding - A Survey” Proceedings of the IEEE, special issue on protection of multimedia content, 87(7):1062-1078.

Shannon C. (1948): “Mathematical Theory of Communications” Bell System Technical Journal. vol. 27, pp. 379-423 and 623-656, July and October, 1948.

Silman J. (2001): “Steganography and Steganalysis: An Overview” SANS Institute. As a part of the Information Security Reading Room.

Simmons G. (1983): “The prisoners’ problem and the subliminal channel” Sandia National Laboratories. Albuquerque NM 87185, Springer-Verlag.

Vidyasagar M, Song H and Elizabeth Ch. (2005): "Fingerprinted Secret Sharing Steganography for Robustness against Image Cropping Attacks" IEEE, 3rd International Conference on Industrial Informatics.

Wikipedia®, (2007): Information Theory, (http://en.wikipedia.org/wiki/Information_theory, September 2007).

Appendix A

$$P_s(0) = P_q(0)Y + P_q(1)(1-Y)$$

$$= \ (P_t(0)X + P_t(1)(1-X))Y + (P_t(0)(1-X) + P_t(1)X)(1-Y)$$

$$= P_t(0)XY + P_t(1)Y - P_t(1)XY + [P_t(0) - XP_t(0) + P_t(1)X](1-Y)$$

$$= P_t(0)XY + YP_t(1) - XYP_t(1) + P_t(0) - XP_t(0) + P_t(1)X - YP_t(0) + XYP_t(0) - XYP_t(1)$$

$$= 2XYP_t(0) - XP_t(0) - YP_t(0) + P_t(0) + YP_t(1) - 2XYP_t(1) + XP_t(1)$$

$$P_s(0) + XP_t(0) - P_t(0) - XP_t(1) = 2XYP_t(0) - YP_t(0) + YP_t(1) - 2XYP_t(1)$$

$$P_s(0) - XP_t(0) - P_t(0) - XP_t(1) = Y[2XP_t(0) - P_t(0) + P_t(1) - 2XP_t(1)]$$

$$Y = \frac{P_s(0) + XP_t(0) - P_t(0) - XP_t(1)}{2XP_t(0) - P_t(0) + P_t(1) - 2XP_t(1)}$$

From equation (4.5)

$$P_q(0) = P_t(0)X + P_t(1)(1-X)$$

$$P_q(0) = P_t(0)X + P_t(1) - P_t(1)X$$

$$P_q(0) - P_t(1) = X(P_t(0) - P_t(1))$$

$$X = \frac{P_q(0) - P_t(1)}{P_t(0) - P_t(1)}$$

Substituting the X value in Y

$$Y = \frac{P_s(0) + [\frac{P_q(0) - P_t(1)}{P_t(0) - P_t(1)}]P_t(0) - P_t(0) - [\frac{P_q(0) - P_t(1)}{P_t(0) - P_t(1)}]P_t(1)}{2P_t(0)[\frac{P_q(0) - P_t(1)}{P_t(0) - P_t(1)}] - P_t(0) - 2P_t(1)[\frac{P_q(0) - P_t(1)}{P_t(0) - P_t(1)}] + P_t(1)}$$

$$= \frac{P_s(0) + \frac{P_t(0)P_q(0) - P_t(0) + P_t(0)^2}{P_t(0) + P_t(0) - 1} - P_t(0) - \frac{P_q(0)P_t(1) + P_t(1)^2}{P_t(0) + P_t(0) - 1}}{\frac{2P_t(0)P_q(0) - 2P_t(0)(1 - P_t(0))}{P_t(0) + P_t(0) - 1} - P_t(0) - \frac{2P_t(1)P_q(0) + 2P_t(1)^2}{P_t(0) + P_t(0) - 1} + P_t(1)}$$

$$= \frac{P_s(0) - P_t(0) + \frac{P_t(0)P_q(0) - P_t(0) + P_t(0)^2}{2P_t(0) - 1} - \frac{P_q(0) + P_q(0)P_t(0) + (1 - P_t(0))^2}{2P_t(0) - 1}}{\frac{2P_t(0)P_q(0) - 2P_t(0) + 2P_t(0)^2}{2P_t(0) - 1} - \frac{2P_q(0) + 2P_t(0)P_q(0) + 2(1 - P_t(0))^2}{2P_t(0) - 1} + 1 - 2P_t(0)}$$

$$= \frac{\frac{P_s(0)(2P_t(0) - 1) - 2P_t(0)^2 + P_t(0) + P_t(0)P_q(0) - P_t(0) + P_t(0)^2 - P_q(0) + P_q(0)P_t(0) + 1 - 2P_t(0) + P_t(0)^2}{2P_t(0) - 1}}{\frac{2P_t(0)P_q(0) - 2P_t(0) + 2P_t(0)^2 - 2P_q(0) + 2P_t(0)P_q(0) + 2 - 4P_t(0) + 2P_t(0)^2 + 2P_t(0) - 1 - 4P_t(0)^2 + 2P_t(0)}{2P_t(0) - 1}}$$

$$= \frac{2P_s(0)P_t(0) - P_s(0) + 2P_t(0)P_q(0) - P_q(0) - 2P_t(0) + 1}{4P_t(0)P_q(0) - 2P_q(0) - 2P_t(0) + 1}$$

$$= \frac{P_s(0)[2P_t(0) - 1] + P_q(0)[2P_t(0) - 1] - [2P_t(0) - 1]}{[2P_t(0) - 1][2P_q(0) - 1]}$$

$$Y = \frac{P_s(0) + P_q(0) - 1}{2P_q(0) - 1}$$