The Islamic University – Gaza
Research and Postgraduate Affairs
Faculty of Information Technology
Master of Information Technology

الجامعـــــــة الإســـــلامية – غـزة
شئون البحث العلمي والدراسات العليا
كـليـــة تـكنـولـوجـيــا المعـلـومــات
ماجستير تـكنـولـوجـيـا المـعلـومــات

Steganography within LSB and Second LSB with Randomness Depending on Indicators Using Secret Key

إخفاء المعلومات في LSB و ثاني LSB عشوائيًا اعتمادًا على المؤشرات باستخدام المفتاح السري

Wesam Monir Saqer

Supervised by

Dr. Tawfiq Barhoom

Associate Professor – Applied Computer Technology

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Environmental Sciences

February/2017

إقـــــــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Steganography within LSB and Second LSB with Randomness Depending on Indicators Using Secret Key
إخفاء المعلومات في LSB و ثاني LSB عشوائيًا اعتمادًا على المؤشرات باستخدام المفتاح السري

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل الاخرين لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

Declaration

I understand the nature of plagiarism, and I am aware of the University's policy on this.

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted by others elsewhere for any other degree or qualification.

Student's name:	وسام منير صقر	اسم الطالب:
Signature:		التوقيع:
Date:	28/2/2017	التاريخ:

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

الجامعة الإسلامية-غزة
The Islamic University of Gaza

الجامعة الإسلامية غزة
The Islamic University – Gaza
1396 - 1978 - ١٣٩٨ - ١٩٧٨

مكتب نائب الرئيس للبحث العلمي والدراسات العليا
هاتف داخلي: 1150

الرقم: ج س غ/35/ Ref:
التاريخ: 2017/02/28 Date:

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **وسام منير صالح صقر** لنيل درجة الماجستير في كلية ***تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

إخفاء المعلومات في LSB وثاني LSB عشوائياً اعتماداً على المؤشرات باستخدام المفتاح السري
Steganography within LSB and Second LSB with Randomness Depending on Indicators Using Secret Key

وبعد المناقشة التي تمت اليوم الثلاثاء 01 جمادى الثاني 1438هـ، الموافق 2017/02/28م الساعة الثانية عشر والنصف ظهراً ، في قاعة مؤتمرات القدس، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. توفيق سليمان برهوم	**مشرفاً و رئيساً**	[illegible]
د. إياد محمد الأغا	**مناقشاً داخلياً**	[illegible]
د. محمد عوض عوض الله	**مناقشاً خارجياً**	[illegible]

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية ***تكنولوجيا المعلومات*** / **برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

نائب الرئيس لشئون البحث العلمي والدراسات العليا

[illegible]

أ.د. عبدالرؤوف علي المناعمة

شئون البحث العلمي والدراسات العليا
الجامعة الإسلامية غزة
The Islamic University Gaza
Research & Graduate Affairs

Abstract

Steganography is the art of science that is concerned with hiding communications by hiding secret information inside a medium as a carrier, called cover medium, to be sent over communication channels to meant parties. Information hiding could be done with simple and direct methods; however we should increase hidden information security as possible by developing and using more robust ways.

One of the most used techniques is Lest Significant Bit (LSB) Substitution. The direct approach that uses LSB Substitution is Sequential LSB that embeds data sequentially, but it is too simple and easy to attack. So, to increase hidden information security, we must use this technique of hiding in random way. Many algorithms were set to increase the security of the hidden data, each of which has its own mechanism of data hiding randomization.

This research introduces a relatively new algorithm of data hiding, called Indicators-based LSB, which embeds data randomly to increase the hidden data security. The algorithm implements randomness using indicators, which makes embedding operation moves forward and backward through cover mediums during hiding process. Also, for increasing hidden information security, a secret key is used through hiding process. A secret key is a sequence of characters defined by users.

There are several types of cover mediums as images, audios, videos, etc. However, image based steganography is the most common system used, since digital images are widely used over the Internet, so images were used through our research as the cover mediums for experiments and image quality metrics as Peak signal-to-noise ratio (PSNR) and mean square error (MSE) were used for evaluation. Also, stego images were subjected to steganalysis, which is the art of detecting steganography, to test the algorithm robustness.

According to the tests and results, the randomness of the algorithm is extremely satisfied, so it is hard to attack the resulting stego files. Also the embedding operation of the algorithm results in stego files with high quality, so it doesn’t arouse any suspicion.

Keywords- Steganography, Information Hiding, Information Security, Steganalysis, Randomness.

الملخص

إخفاء المعلومات Steganography هو العلم الذي يختص بإخفاء التواصل عن طريق إخفاء المعلومات المرسلة داخل وسيط ناقل يدعى cover medium و إرساله عبر قنوات الاتصال إلى الأطراف المقصودة، إخفاء المعلومات يمكن أن يتم بالطرق البسيطة و المباشرة، إلا أنه يجب زيادة أمن المعلومات المخفية قدر المستطاع عن طريق تطوري و استخدام طرق أكثر قوة.

أحد أكثر تقنيات الإخفاء استخدامًا هي تقنية استبدال الخانة الأقل وزنا Lest Significant Bit (LSB) Substitution، الطريقة المباشرة التي تستخدم تقنية استبدال LSB هي LSB المتسلسلة التي تخفي البيانات تسلسليًا، ولكنها بسيطة جدًا و سهلة الكسر، لذلك لرفع أمن المعلومات المخفية يجب إخفاء البيانات باستخدام هذه الطريقة عشوائيًا، الكثير من الخوارزميات تم وضعها لرفع أمن البيانات المخفية، كل منها لديها آليتها الخاصة في الإخفاء العشوائي للبيانات.

هذا البحث يقدم خوارزمية إخفاء جديدة نسبيًا تدعى LSB المعتمِدة على المؤشرات Indicators-based LSB، و التي تقوم بإخفاء البيانات عشوائيًا لرفع أمن البيانات المخفية، الخوارزمية تحقق العشوائية باستخدام المؤشرات و التي تجعل عملية إخفاء البيانات تتحرك للأمام و الخلف خلال الوسيط الحاوي أثناء عملية الإخفاء، أيضًا لزيادة أمن المعلومات المخفية، يتم استخدام مفتاحًا سريًا خلال عملية الإخفاء، و المفتاح السري عبارة عن سلسلة من الرموز المعرفة من قبل المستخدم.

هناك العديد من أنواع الوسائط التي يمكن إخفاء المعلومات فيها مثل الصور، و الملفات الصوتية و ملفات الفيديو إلخ، و لكن إخفاء المعلومات داخل الصور هو النظام الأكثر شيوعًا و استخدامًا، حيث أن الصور الرقمية تستخدم بشكل واسع عبر شبكة الإنترنت، لذلك خلال بحثنا تم استخدام الصور كوسيط حاوي لإجرات التجارب، و تم استخدام مقايس جودة الصور مثل Peak signal-to-noise ratio (PSNR) and mean square error (MSE) لتقييم الصور الناتجة، أيضًا الملفات الحاوية للبيانات تم تعرضيها لعمليات اختراق و كسر إخفاء المعلومات Steganalysis، و ذلك لاختبار مدى قوة الخوارزمية.

طبقاً للنتائج فإن معيار العشوائية تحققه الخوارزمية بشكل كبير، مما يجعل عملية كسر الملفات الحاوية للبيانات صعبة، أيضًا الملفات الحاوية للبيانات و الناتجة عن عملية الإخفاء ذات جودة عالية، مما يجعلها لا تثير الاشتباه.

{يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ}

[المجادلة: 11]

Dedication

This thesis is dedicated with everlasting love to holy prophet Muhammad, peace be upon him, and all precious people in my life.

To my parents for their love, sacrifices and patience, and who always be the most reason of every success in my life.

To my brother and sisters for their love, encouragement and true feelings within their hearts.

To all my friends for being there for me all the way.

To all of my relatives who really have true love within their hearts for me.

Acknowledgment

Foremost, all praise is due to Allah (Glorified and Exalted is He) for his immeasurable blessings, graces and mercy without which none of this could have been possible.

Numerous people have provided me with guidance and support through this research. I would like to express my deep gratitude towards Prof. Tawfiq Barhoom for his continuous guidance, and encouragement. I would like to thank him for his constructive criticism and suggestions of improvement at various stages of the work. I am also sincerely indebted to Dr. Iyad AlAgha for his invaluable guidance, insightful comments and suggestions throughout my studies. I would like to thank all of the Information Technology Faculty teachers at The Islamic University for their continuous efforts, support and guidance.

I would like to express how grateful I am to my teachers at Al-Aqsa University especially Dr. Mohammed Awadallah, Dr. Mohammed AbuSaqer and Dr. Tamir Fatayer for their valuable guidance.

I would like to send special thanks to my parents, brother and sisters for their endless love and support. I am also thankful to all my friends relatives and good people around me especially my elder brother Aiman Ashor. I must also express my thanks to all my colleagues in the Faculty of Information Technology.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Amount Indicator
ASCII	American Standard Code for Information Interchange
BMP	Bitmap image
DCT	Discrete Cosine Transform
EOF	End Of File
GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
LI	Location Indicator
LSB	Least Significant Bit
MSB	Most Significant Bit
MSE	Mean Square Error
PNG	Portable Network Graphics
PoVs	Pair of Values
PRNG	Pseudo Random Number Generator
PSNR	Peak Signal-to-Noise Ratio
RGB	Red, Green and Blue
RGBA	Red, Green, Blue and Alpha
WAM	Wavelet Moment Analysis

Chapter 1
Introduction

Due to the need for transmitting secret data, steganography plays an important role in secure communications, since steganography is concerned with hiding communications. Communication hiding is accomplished by hiding secret data inside an innocent-looking medium, and sending the medium over a communication channel to the meant party. Consequently, steganography is used for protecting secret information while it is being transferred, as in military issues, if some institution needs to transfer data and protect it from spying or even in data transferring between individuals. In this chapter we give an introduction to steganography field and a brief overview about its applications and usage. We also explain all the aspects of this research and how it is organized.

1.1 Background and Context

Steganography is the art of science that is concerned with hiding secret data inside other innocent-looking data, which is called the cover, carrier or container, in order to hide communications, so no one apart from the meant parties can suspect the existence of the secret data and thus, the covert communication taking place (Johnson & Jajodia, 1998; Krenn, 2004). The aim of steganography is hiding the very existence of the secret data, in order to hide the communication taking place. Therefore, we can transfer the carrier medium with the hidden data inside over some channel to the meant recipient while no one apart knows or can suspect that there is data transferring and communication in between. So, if we have some sensitive data that should not be exposed to unauthorized parties, and we need to transfer it over an open network as the Internet, simply we can hide the data into some medium and send the carrier medium with the hidden data to the meant recipient. The object into which the data is embedded and hidden is known as cover medium (Kipper, 2003), and the resulting output known as stego-medium (Kipper, 2003), or stegogramme (Bateman & Schaathun, 2008). The stego-medium should be as identical to the cover medium as possible, so while it is being transferred, it doesn’t raise any suspicion. So, if anyone intercepts the stegogramme, it is difficult to tell that it has hidden data inside. One of the most used hiding techniques is Least Significant Bit Substitution (LSB), which depends on substituting the cover file binary sequences LSBs with secret data bits. Cover mediums could be of several types such as images, audios and videos, etc. (Neeta, Snehal, & Jacobs, 2006; Nguyen, Arch-Int, & Arch-Int, 2016). Through our work we introduce a new algorithm of data hiding using LSB substitution technique with high security and extra capacity compared to Hide & Seek or sequential LSB algorithm. We concentrate on image files as cover mediums for experiments.

1.2 Statement of the Problem

Many algorithms were set to hide secret data into cover data. LSB Substitution is a very popular technique which is used by too many algorithms. Some of those algorithms are straightforward or simple as Hide & Seek algorithm, and some are robust. So, it is necessary to develop more new algorithms with more security and robustness, and not known by attackers.

1.3 Objective

The main objective is to develop a new approach of hiding secret data with high security and extra capacity compared to other LSB-based algorithms.

1.4 Scope

Our research is about developing a new algorithm of information hiding inside cover mediums. It concentrates on image files as cover mediums. LSB Substitution technique is used by the algorithm for embedding secret data into cover mediums. Since data is embedded into spatial domain, then lossless images as PNG and BMP could be used. Hence, we selected one of these types only for the experiments which is PNG. The algorithm was evaluated by evaluating the resulting files after embedding the data inside. Since images are used as cover mediums for experiments, so images evaluation means were used for evaluating the outcome of the algorithm. Also the resulting images by the embedding operation are subjected to a steganalysis tool to measure how much the resulting mediums can withstand the attacks. Steganalysis is out of the scope, but some of its techniques are used for experiments and testing purpose. Dataset images were gathered from USC-SIPI image database, which contains the famous images globally used for steganographic algorithms evaluation such as Pepper, and randomly from Internet.

1.5 Signification

One of the most important issues for people and organizations is communicating securely. Most governments monitor communications means between people, organizations and even between other governments. Eavesdroppers spend too much effort for spying on some parties. Hence, communications may not be safe from monitoring or attacking. Therefore, both of these issues have increased the importance of finding secret communication methods. Steganography is considered one of the most fields satisfying the purpose, since its main aim is covert communication.

1.6 Limitations

Fortunately, there were no serious limitations encountered though the research, however:

- The algorithm can’t be applied to all types of images the same way. For lossless images as PNG and BMP we can embed the data directly into pixels. but it needs extra work to apply it to lossy images as JPEG, because we do the embedding through the transform domain.
- Also for audio mediums, we need to know how deal with signals to make them contain the secret data using LSB Substitution.
- The size of the data that can be embedded is restricted by the number of cover medium size, as an instance, for lossless images the more pixels we have, the more data can be embedded.
- On another hand, access to most studies and researches is limited due to monetary constraints, since most researches and studies require to be paid for in order to get them.

1.7 Overview of Thesis

This thesis is structured around five chapters as follows:

Chapter Two provides review of all steganography aspects and its related fields. It explains steganography in depth and some of its techniques, science fields that our work depends on and common metrics used for measuring and quantifying the main aspects of steganographic systems. Additionally, it discusses some of the most related work and gives a comparison between them depending on some approved evaluation aspects. Furthermore, it explains steganalysis and its techniques and introduces one of its tool that is used for experimentations.

Chapter Three explains in details our algorithm of data hiding, which is proposed as a novel solution with more security and capacity. It explains how the algorithm works, illustrates how it satisfies the aim of steganography and gives analysis of its points of strength and weakness.

Chapter Four explains the experiments done for testing the proposed algorithm and discusses the results in details showing its efficiency.

Chapter Five summarizes research findings and conclusions. It highlights the contributions of the research and gives an overview of its evaluation. Also it gives recommendations for hiding data. Finally it talks about future work.

Chapter 2
Literature Review

The idea of steganography is not new. It has been used long time ago for transmitting data. However, with the evolving of digital communication means, it has become possible to employ steganography for transferring secret data covertly through digital communication channels. Through this chapter we give in details explanations for all aspects related to steganography.

2.1 Ancient History

The term steganography is of Greek origin, Steganos means "covered" and graphia means “writing” (Gowda & Sulakhe, 2016; Holub, 2014). Then steganography which is the combination of them means “Covered Writing”(Sharif, Mollaeefar, & Nazari, 2016; Watkins, 2001). It has been used in several forms for thousands of years (Cheddad, Condell, Curran, & Mc Kevitt, 2010). In the 5th century BC Histaiacus shaved a slave’s head and tattooed a message on his skull to get the message hidden after the slave’s hair grew back. Then he dispatched the slave with the message (Bateman & Schaathun, 2008; Easttom II, 2016). Five hundred years ago, the Italian mathematician Jerome Cardan reinvented a Chinese ancient method of secret writing, which depends on using a paper mask with holes. The method was named Cardan Grille after him. Nazis invented several steganographic methods during World War II and have reused invisible ink and null ciphers (Cheddad et al., 2010). Today after the extreme development of information technology field, steganography became widely used in digital fields and its techniques evolve more and more day by day. Steganography can be used for multiple purposes, like watermarking, ownership identification and copyright protection, data authentication etc.

2.2 Steganography Nomenclature

Steganography refers to the process of hiding data within some cover medium to allow covert communication. When we need to send some secret data to some remote recipient over an open network that can be accessed by anyone, like Internet, and the data shouldn’t be exposed to unauthorized parties, then seriously the data need to be sent covertly where no one knows that there is data transferring. This covert communication is the main purpose of steganography, where we need to keep others from thinking that a secret message even exists within stego files. So, steganography aims to hide the communication by hiding the presence of the data passing. The strength of this advantage is that no one knows about the data, so it enables us to make the data avoid even the attempt of attack.

Hiding data is accomplished by embedding it into some medium which called the cover medium, such that the resulting object would be sent carrying the data to the meant party through the communication channel. The object into which the data is hidden is called the cover medium, and the resulting object is called the stego medium or stegogramme. The cover medium could be one of several types. It could be an image, audio, video, text, html or any other object. The resulting stegogramme after hiding data must be created with some restrictions to be of high quality, such that no one can suspect that it contains secret data. It should be identical to the cover medium as possible. Stego mediums with differences from the cover mediums may arouse suspicion and attract attacker’s attention. So stego mediums should have visual and statistical properties as close as possible to the cover medium properties. As stegogrammes are sent to the meant recipient, they may get subjected to attacks to find out whether they carry hidden data or not. The art concerned with detecting stegogrammes and steganographic message is known as steganalysis (Kipper, 2003). Lots of steganographic algorithms have been developed to result in stegogrammes with high quality in order to make it as difficult as possible to detect them by steganalysis means. So the main purpose of steganalysis is detecting the stegogrammes. Since steganography is concerned with allowing secret communication, not just hiding data into files. It also takes advantage of network protocols, such as TCP and SOAP, by hiding the secret data inside them or their headers, since network headers contain many fields that are either optional or unused for normal transmission. (Al-Mohammad, 2010; Lubacz, Mazurczyk, & Szczypiorski, 2012). These protocols such as ARP, TCP, UDP or ICMP protocols, are referred to as carrier-protocols (Lubacz et al., 2012).

2.3 Steganography and Cryptography

Steganography is a branch of information security field, since one of the information security concerns is protecting sensitive and secret data, which is the purpose for which steganography is used. Another branch of information security concerned with data protection is cryptography. So steganography and cryptography are cousins and intended to protect information from unauthorized parties, but the main difference between them is how each of which does so (Goel, 2008). Steganography is concerned with hiding data and hiding the very existence of the data, where cryptography is concerned with scrambling the data and make it unreadable “cipher”, so the encrypted data does not make sense to anyone but the meant parties after they decrypt it (Bateman & Schaathun, 2008; Dunbar, 2002; Johnson & Jajodia, 1998; Krenn, 2004; Morkel, Eloff, & Olivier, 2005). Encrypted data could be vulnerable because eavesdroppers are aware of its existence (Jain & Boaddh, 2016). And since attackers have the chance to apply cryptanalysis techniques over the data, then it is possible to break down the security system (Al-Mohammad, 2010). So sometimes hiding the communication is more important than protecting it. This was clearly illustrated by Simmons (1983) in the “Prisoners’ Problem” (Simmons, 1984). In Prisoners Problem, Alice and Bob are arrested and thrown in two different cells. They want to make an escape plan, but their communication is monitored and checked by a warden (Wendy). Alice and Bob must communicate invisibly in order not to arouse Wendy’s suspicion since she will transfer them to a high-security prison if she notices any suspicious communication. Alice and Bob can succeed only if they can transfer messages covertly without making Wendy suspicious. Thus, this vulnerability can be solved by hiding the message transferring from Wendy (Chandramouli, Kharrazi, & Memon, 2003). So, the communication could be hidden by hiding the data messages within an innocent-looking cover medium that does not arouse suspicion of eavesdroppers (Al-Mohammad, 2010). So, Steganography which is a kind of covert communication is concerned with not detecting the existence of secret the data because it aims at making it unknown that there is secret message passing (Stanley, 2005), where cryptography is concerned with not understanding the secret data by altering its structure (Thangadurai & Sudha Devi, 2014; Wang & Wang, 2004). Even though both cryptographic and steganographic systems provide secret communications, they are different in terms of system breaking. A cryptographic system is considered broken if an attacker can read the secret message. However, a steganographic system is considered broken if merely an attacker detects the existence or read the hidden message (Al-Mohammad, 2010). We can combine both of them to make our data more secure. While we can encrypt data by one of cryptography techniques, next we can hide it within a carrier using a steganography technique to provide extra layer of protection which is advisable by most researchers (Al-Mohammad, 2010; Bateman & Schaathun, 2008; Goel, 2008; Johnson & Jajodia, 1998; Mahmood, Azeez, & Rasool, 2014; Satar et al.). Therefore steganography role is to complement cryptography (Al-Ani, Zaidan, Zaidan, & Alanazi, 2010; Al-Mohammad, 2010). Cryptography can be divided into two types, symmetric and asymmetric (Mahmood et al., 2014). In symmetric cryptography, encryption and decryption is done using the same key. Asymmetric encryption involves pair of keys, public and private. When encryption is done using one key, decryption is done by the other (Mahmood et al., 2014).

2.4 Steganography Architecture

A steganographic system is comprised of two algorithms, the first is for hiding and the second is for retrieving. The hiding process is concerned with embedding data within the cover medium and resulting in the stegogramme. Therefore, this process should be constructed carefully to be sure the stegogramme is identical to the cover medium as possible; thus the message is sent unnoticed. Therefore, basically the components of the embedding process system consists of a secret message and a cover medium as inputs, a steganography algorithm as the method of hiding and a resulting stegogramme as the output. Also a secret key can be used for hiding the data as a third input to increase the robustness and security of the hidden data, such that there is no way the data is retrieved in the absence of the secret key even though the algorithm of hiding is known (Al-Ani et al., 2010; Al-Mohammad, 2010; Goel, 2008). On the other hand, the retrieving process is concerned with extracting data from the stegogramme. Simply this process is the inverse of the hiding process. Retrieving process takes the stegogramme and the secret key as inputs, and returns back the secret data as the output (Al-Ani et al., 2010; Al-Mohammad, 2010; Goel, 2008). Figure 2.1 shows the architecture of a steganographic system.

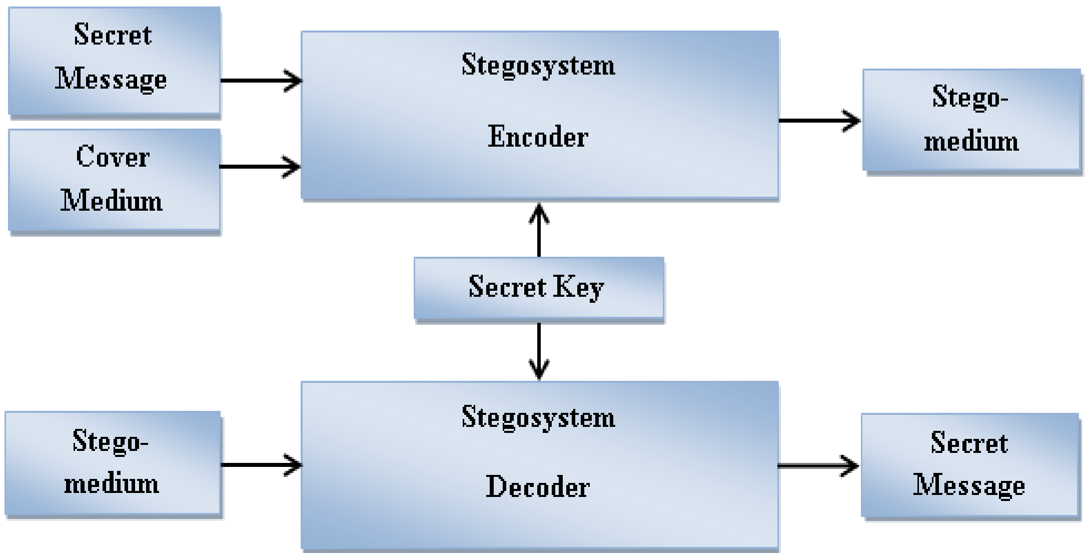


Figure (2.1): Steganography System Architecture

When we desire to hide secret data into some cover medium, the stegosystem should be designed carefully to embed the data and create a stegogramme which is an exact copy of the cover medium, or at least as close as possible, so that the adversary regards the stegogramme and the communication taking place as innocuous. After obtaining the stegogramme, it is in most cases sent to a remote recipient along with the secret key to extract the hidden message (Al-Ani et al., 2010; Al-Mohammad, 2010; Goel, 2008).

2.5 Steganography in Depth

Over time since steganography has been started being used, it has evolved and many new algorithms and techniques were developed to improve the hiding operation and increase the hidden data security. The embedding process is done by altering the contents and tweaking the values of cover mediums to make them contain the data and result in the stegogrammes. However, we can’t modify the values of all areas of the cover file. Changing values of some parts of the cover file may destroy the cover file or result in some noticeable and detectable distortion. Thus, if the distortion was perceptible, the chances of detecting the stegogramme would be so high. So, the lower the distortion, the better the chances of undetectability. Therefore essentially, steganographic systems must identify the redundant data of the cover medium. Redundant data is insignificant data that when gets modified, it has no direct impact on the overall perceptibility of the cover file, therefore the alteration of the data is not detected easily (Al-Mohammad, 2010; Morkel et al., 2005). So, any modification to these redundant bits should not destroy the integrity of the cover medium, and thus preserving the quality which in turn would enhance the imperceptibility and the undetectability of the steganographic system and resulting stegogrammes. Moreover, even if the hiding algorithm used is publicly known, if the stegogramme has no suspicious changes or indications, no one can figure out the presence of hidden data. So, steganographic systems should produce stegogrammes as identical to the cover medium as possible, such that it doesn't arouse suspicion and it makes it hard for steganalysts to detect steganography in stego mediums that is identical to innocent mediums (Bateman & Schaathun, 2008).

Hidden data security is enhanced by enhancing the imperceptibility or the robustness. Also, some algorithms increase the capacity of cover mediums for secret data. So the key properties of steganographic systems that must be considered are imperceptibility, robustness and capacity (Bateman & Schaathun, 2008; Saidi, Hermassi, Rhouma, & Belghith, 2016; Sumathi, Santanam, & Umamaheswari, 2014). So, we can consider them as criteria of efficiency of steganographic algorithms and systems:

1. **Imperceptibility or Undetectability:** Imperceptibility is how much the stego file has no perceptually detectable change or distortion. Thus, it depends on the quality of the resulting stego file to be as identical to the cover object as possible. This is done by avoiding making noticeable change in the resulting stego medium. (Bateman & Schaathun, 2008; Krenn, 2004; Morkel et al., 2005; Sumathi et al., 2014). Additionally, the stego-medium must not be statistically perceptual, thus it should has statistics identical to the cover medium (Al-Mohammad, 2010).
2. **Robustness:** Robustness is the degree of how much the steganographic system can withstand against steganalysis and attacks, and how difficult to determine

whether the stegogramme contains hidden data or not (Bateman & Schaathun, 2008; Sumathi et al., 2014; Wang & Wang, 2004). Robustness involves withstanding hidden data detection, extraction and destruction by steganalysis means.

3. **Capacity or Payload:** Capacity is determined by the maximum amount of secret data that can undetectably be embedded inside the cover medium. Hiding data within cover files could be done sometimes with huge amount, but it would be so obvious that the resulting stego files have hidden data inside. So, increasing the capacity of an algorithm must be done with maintaining the quality of the cover files, and with least possible affecting to its properties (Al-Mohammad, 2010; Kipper, 2003).

However, there is tradeoff between imperceptibility and capacity, where embedding more data introduces more artifacts into cover mediums and then increases the perceptibility of hidden data (Al-Mohammad, 2010). Subsequently, data embedding should be as small as possible, since typically the more the embedded data, the more the cover medium is altered, the easier for steganalysts to detect the stegogramme (Bateman & Schaathun, 2008; Kipper, 2003; Sumathi et al., 2014). So it is difficult to increase the capacity and maintain the imperceptibility at the same time (Al-Mohammad, 2010).

2.6 Steganography Classifications

Several approaches were set for classifying steganographic systems, but there are two general approaches. The first is based on the type of cover file while the second is based on the hiding method used (Al-Mohammad, 2010).

2.6.1 Cover Type-Based Classification

Since we can hide the data inside multiple types of cover mediums, Thus steganography could be classified according to the cover medium type that is used for hiding the data within as:

1. Image steganography.
2. Audio steganography.

3. Video steganography.
4. Text steganography.
5. HTML steganography.
6. Network steganography.

The classification is shown below in Figure 2.2:

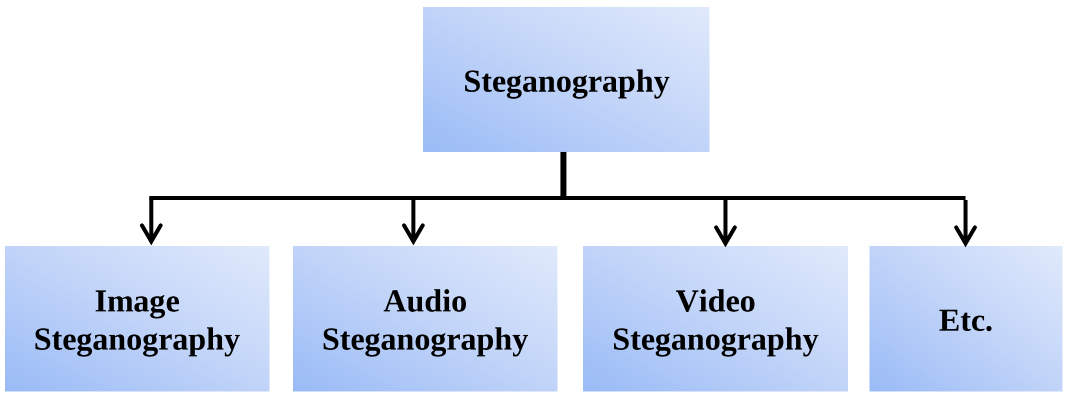


Figure (2.2): Cover Type-Based Classification

However, the properties of cover files types vary, thus the hiding processes themself vary in accordance with the type of the cover medium.

2.6.2 Hiding Method-Based Classification

Steganography could be also classified according to the method of data hiding. Subsequently, steganography can be split into three approaches of hiding data (Al-Mohammad, 2010):

1. Insertion-based method.
2. Substitution-based method.
3. Generation-based method.

The classification is shown below in Figure 2.3:

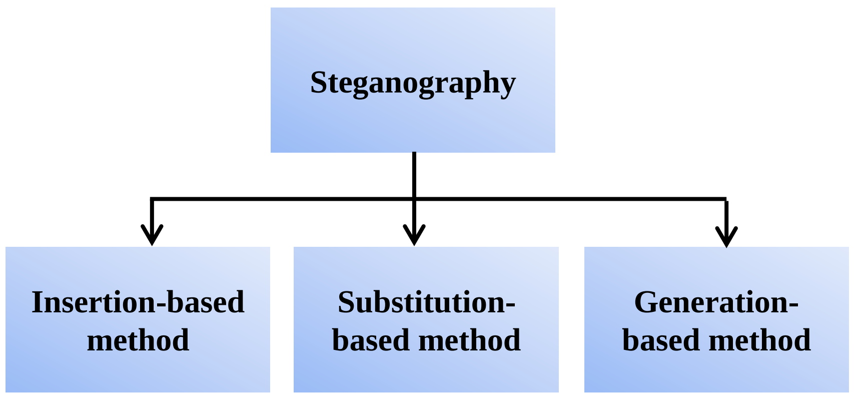


Figure (2.3): Hiding Method-Based Classification

2.6.2.1 Insertion-based method

Insertion-based method works by finding areas in cover files which are ignored by applications that read these cover files, and hiding the secret data within these areas. This method involves a defect and an advantage. The defect is, since this method inserts the data inside the cover file, then the file of the resulting stegogramme would be larger than the cover medium size. However, since in most cases the original cover file would not be available for comparing, this method may be good as long the stego medium size is reasonable. The advantage is that it doesn’t change the content of the cover file, so it preserves the quality of the cover file and there wouldn’t be any detectable or perceptible change in the stegogramme. Also by using this method, we can hide any amount of data inside the cover medium, however with adding too much data, the resulting stegogramme will be very suspicious.

As an example, some files have a flag called EOF or end-of-file marker. This flag is used by the applications to find the end of a file in order to stop processing it. One of the ways to hide data is just to insert it after the EOF marker of the cover file and the application will ignore the hidden data when reading the resulting stego-file (Cheddad et al., 2010; Eric, 2003). As an instance, if an image is used as a cover file, simply the message is inserted after the EOF tag of the image file and when opening the stego-image by any photo application, it will just display the image ignoring anything coming after the EOF marker. On the other hand, as mentioned the weakness is that the size of the resulting stegogramme is the sum of the sizes of both the secret data and the cover file, which may arouse the suspicion if the size is too large to be of the cover file (Cheddad et al., 2010; Eric, 2003). Another example is writing the secret data between end-text and begin-text markers in Microsoft Word files (after the end-text and before the next begin-text). And according to the configuration of Microsoft Word, it ignores anything written in such areas, so the hidden message would not appear when the document is read by Microsoft Word application (Eric, 2003).

2.6.2.2 Substitution-based method

Substitution-based method depends on finding insignificant areas or information in cover files and replacing these areas values with the secret data. So the sizes of both the cover file and the stegogramme are identical, since the cover data are just modified without any data adding, and this is the main advantage of this method over insertion-based method. On the other hand, the quality of the cover file could be degraded because of the modification. And the amount of secret data that can be embedded is restricted by the size of the insignificant information that can be replaced or overwritten (Eric, 2003).

2.6.2.3 Generation-based method

Generation-based method works by generating the cover file into which the data would be hidden into. So it doesn’t require an existing cover file. The main advantage of this method is that the stegogramme is not a cover file with distortion or extra size than the original cover file. However, the generated files might be unrealistic to end users ,since they consist of random content. So, probably random-looking images is suitable for this kind of information hiding (Eric, 2003).

2.7 Least Significant Bit Substitution

One of the earliest and most popular steganography techniques is Least Significant Bit Substitution technique (LSB) (Easttom II, 2016; Juneja & Sandhu, 2013). In Computer science, the term Least Significant Bit refers to the smallest (right-most) bit of a binary sequence (Bateman & Schaathun, 2008). LSB substitution technique is defined as hiding the secret data bit into the LSB of the cover binary sequence, by replacing the LSB value of the cover binary sequence with secret bit value, regardless of the order of the cover binary sequences used to contain the secret data, and if the hiding order is sequential or random. So if we have a cover binary sequence of 01101100 and a secret bit with value of 1, and we need to hide it using LSB substitution technique, then simply we replace the LSB of the cover binary sequence with the secret bit and the cover binary sequence becomes 01101101. The simplest algorithm that hides data using LSB substitution is the Hide & Seek algorithm, which embeds secret bits into the LSBs of the cover binary sequences in sequential fashion from the beginning to the end of the cover medium. When we want to embed a byte of secret data, we just take the eight bits of the secret byte, and replace the least bit of a series of eight binary sequences of the cover data with these secret bits and so on. Simply we replace the least bit of each binary sequence of the cover binary sequences with our secret bits. Thus, every secret byte consumes eight cover binary sequences to get embedded into (Morkel et al., 2005).

Suppose a binary sequence unit of the cover data is one byte and we have the secret byte 01101010 which we want to embed into a series of cover bytes using LSB substitution technique. The process would be done as shown below in Table 2.1:

Table (2.1): Series of Cover Bytes Before and After Embedding by Sequential LSB

Cover byte Index	The series of cover bytes before the embedding process	The resulting stego bytes after the embedding process
0	10111000	1011100(0)
1	10000001	1000000(1)
2	10100001	1010000[0]
3	01111100	0111110[1]
4	01110110	0111011(0)
5	10010000	1001000[1]
6	11010011	1101001(1)
7	01010010	0101001(0)
(b) The bit new value is identical to its original **[b]** The bit new value is different from its original		

There are several advantages of LSB substitution technique. First it doesn’t affect the size of the cover data because it does not increase or decrease the number of the cover data bytes. It just replaces some of the cover data bits with our secret bits without affecting the size. Next, it does not make noticeable changes to the cover data and this is according to two factors. Firstly, the change occurs in the least bit (right-most) which has the least weight among all the bits in a byte. Simply if the value of this bit is changed from 0 to 1 or from 1 to 0, in either cases the value of the byte is changed only by 1 increasingly or decreasingly as shown in Table 2.1 at byte 3 and 2 respectively. This amount of change is neither noticeable by human visual system nor human auditory system. Secondly some of the bits are replaced with its same value as shown in Table 2.1 above, for example, at byte 7 there is no change done to the byte. Thus, on average half of the cover bits are modified using the maximum cover size (Johnson & Jajodia, 1998; Morkel et al., 2005; Wang & Wang, 2004). So, if we hide some data into, for example an image, the change of the image is not detectable at all by human eye, even if we also used the second least significant bit for hiding (Morkel et al., 2005), where it has been claimed by (Ker, 2007) that it is better to use two bit planes than one. LSB from the viewpoint of capacity consumes moderate amount of cover bytes for embedding certain amount of secret data. Where for each secret byte to get embedded, it needs 8 bytes of the cover. So, to hide the data we just need to use a suitable-sized cover, and secret data could be compressed before embedding.

2.8 LSB Substitution and Image-Based Steganography

To a computer, an image is collection of numbers that constitute different light intensities of image areas, and the numeric values forms a grid of points referred to as pixels (Morkel et al., 2005). Image steganography is concerned with developing and enhancing techniques and algorithms of hiding data inside images. Image steganography is the most common system used among steganography categories, because images are widespread over Internet and web (Al-Mohammad, 2010). Hiding data into images could be done either into spatial domain or into transform domain images (Morkel et al., 2005; Silman, 2001; Sumathi et al., 2014; Wang & Wang, 2004). Spatial domain techniques use lossless images as PNG and BMP, and transform domain techniques use lossy images as JPEG. Spatial domain techniques have more capacity than transform domain techniques, however transform domain techniques are more robust (Al-Mohammad, 2010).

2.8.1 LSB Substitution into Spatial Domain

Spatial domain is the image plane itself; the collection of pixels that composes an image (Kipper, 2003). In spatial domain techniques, data embedding is done by encoding the secret data bits directly into image pixels values (Goel, 2008; Jain & Boaddh, 2016). The image pixels are tweaked to contain the secret data bits. The most technique used in spatial domain is LSB substitution, since LSB substitution technique embeds the secret bits directly into the cover file. So, in image steganography, the secret data is being embedded directly into the LSBs of cover image pixels values. LSB substitution technique could be used for embedding secret data either in sequential or random fashion. The simplest form of spatial domain LSB substitution is the method known as Hide & Seek (Bateman & Schaathun, 2008). In Hide & Seek, the LSBs substitution is done sequentially, starting from the first binary sequence unit until the end of the cover file. However, it would be easy for a steganalyst to retrieve the secret data (Morkel et al., 2005). Therefore, many researchers developed plenty of algorithms that do the embedding in randomized manners, in which the locations that contain the data are scattered and not sequential (Bateman & Schaathun, 2008). For instance, the Hide & Seek method itself was applied in randomized mode by shuffling the image pixels using a Pseudo Random Number Generator (PRNG) according to a seed before embedding the secret data. Then the secret data is embedded within the shuffled image data using Hide & Seek method. Finally, the image pixels is inversely shuffled back using the same seed to obtain the image in the original order, but with scattered hidden data inside (Bateman & Schaathun, 2008). Spatial domain techniques are applicable to lossless image compression as PNG, BMP and GIF (Goel, 2008).

2.8.2 LSB Substitution into Transform Domain

Transform domain techniques are methods with Lossy compression used for reducing image size, without reducing its quality to noticeable degree by naked eyes.

It involves several methods as Discrete Cosine Transform (DCT), Fast Fourier Transform, etc. Images of transform domain like JPEG could be used for steganography, where data embedding is done through the transform process by encoding the data bits into transform domain coefficients (Bateman & Schaathun, 2008; Jain & Boaddh, 2016). Hiding in this fashion is more difficult to detect than spatial domain as steganalysts have to do more effort to find the embedding artifacts (Al-Ani et al., 2010; Bateman & Schaathun, 2008). LSB substitution could be used with JPEG, but with some difference from spatial domain. For instance, JSteg algorithm embeds the secret data inside the LSBs of the DCT coefficients, rather than pixel values as in spatial domain (Bateman & Schaathun, 2008). Also sequential hiding was not considered very secure. So OutGuess algorithm was developed to improve JSteg algorithm by randomizing the embedding process (Bateman & Schaathun, 2008). The randomizing is done in the same way of randomized Hide & Seek approach, where the coefficients are shuffled randomly using a PRNG according to a seed. The embedding within the shuffled coefficients is performed using the technique of JSteg. Finally the shuffle operation is inversed in order to get the coefficients back in the correct order.

2.9 Related Work

Many steganographic methods were set with various advantages and weaknesses. The study in (Sumathi et al., 2014) covers in details various steganographic techniques and classifications. Too many LSB-based approaches were set for hiding data. The main purpose of these approaches is increasing security of hidden data. Security of LSB-based approaches is implemented mostly by hiding secret data with randomness and scattering it within the entire cover medium.

2.9.1 Image Steganography

Image steganography techniques are concerned with hiding data inside image files (Gowda & Sulakhe, 2016). Many researchers proposed LSB-based approaches for hiding data within images.

(Karim, Rahman, & Hossain, 2011) proposed using a secret key to decide the positions within the cover image to hide the secret data into. Also for deciding the positions of hiding, beside the secret key they use the red channel of the cover image. Simply for each cover pixel, the LSB value of the red color is XORed with the current bit of the secret key. If the result of the XOR operation is 0, then the current secret bit is embedded into the LSB of the blue color of the current pixel. Otherwise it is embedded into the LSB of the green color and so on. So, it is difficult to retrieve the hidden data in absence of the secret key, which in turn increases the robustness of the stego-image.

The algorithm is robust due to hiding with randomness and secret stego-key, but on the other hand, it doesn't add much randomness, since it moves among the pixels sequentially from the beginning to the final pixel and just hide either inside green or blue channel. For imperceptibility, it is so high due to embedding into one color per pixel, which means one byte out of three for RGB images, or one byte out of four for RGBA images whose LSB is altered (RGBA pixel is a pixel with alpha value. Alpha is used for pixel transparency to determine how opaque or how transparent a pixel is). However on the other hand, the capacity is so low, since we need 3 cover bytes to embed one bit, which means 24 cover bytes for each secret byte.

In research of (Akhtar, Johri, & Khan, 2013) Rivest Cipher 4 algorithm (RC4) is used with a stego-key to randomize the embedding of secret data over the entire cover image. They use RC4 algorithm with a stego-key to generate random order of the cover pixels locations. Then they hide the secret data into the pixels according to the random order. They also have introduced a new technique called bit-inversion to improve the stego image quality. The technique works by splitting the pixels into four sections according to the third and second bits values. The first section is of all the pixels that have the third and second bits with 00 values. The second section is of all pixels with third and second bits of 01 values, and so on, the third is of 10 values and the fourth is of 11 values. Finally, for each section, the count of changed and unchanged pixels is found, and if the number of changed pixels is greater than the unchanged pixels, then the LSBs of the section is inverted.

The robustness is increased due to adding the randomness, by using RC4 algorithm and a secret stego-key to generate random order for pixels locations. So the robustness is increased because to retrieve the hidden data, the sego-key must be known in order to find out the pixels locations order of the hiding process.. Imperceptibility was really improved by bit-inversion technique which reduces the number of changed pixels and ensures that in any case the changed pixels are less than 50% of the entire pixels. The capacity is as much as sequential LSB, so it is relatively good.

(Islam, Siddiqa, Uddin, Mandal, & Hossain, 2014) proposed an algorithm called filtering-based that uses LSB in different way than usual. The algorithm doesn't embed within the LSBs the secret data bits, instead it embeds indications about whether a pixel contains hidden data or not. First they check what pixels are more, the lighter or the darker. Since the pixel consists of three colors, and each color is represented by one byte, then pixels are considered lighter when their colors MSBs have two or three bit values of 1. Thus, darker pixels have two or three MSBs of 0 value. The algorithm hides the secret data into the pixels with greater count among the darker and lighter. Also not all of the selected pixels are used to contain secret data, only those which match a condition that is part of the hiding process. The algorithm finds the decimal value Pn of the MSBs of the three colors of each pixel. The Pn value would be between 0 and 7. Inside the third byte of the pixel, if the value of the bit with index equal to Pn is equal to the secret bit, then LSB of the third byte is set equal to 1 as indication that this byte contains a secret bit, otherwise the LSB is set equal to 0.

The robustness of the algorithm is accomplished due to the high randomness. As we see, it doesn't hide into all pixels, only darker or lighter. The secret bits aren't embedded into the LSB of the selected pixel, instead the LSB would contain an indication to tell if the pixel contains a secret bit, and if so, the secret bit doesn't always exist in a fixed bit, but in some random bit, depending on a condition. The strength point is that a secret bit is not embedded inside the LSBs, but in another bit and randomly. Because there is no much change, the imperceptibility is very high too. However, the capacity of this algorithm is too low, since it searches for pixels that have the same value as the secret bits by chance, which in most cases is rare to find. So it is really not applicable with any amount of secret data, unless the secret data is too small and the cover medium is relatively too big.

(Laha & Roy, 2015) proposed that when using LSB substitution, instead of simply substituting the secret data bits for cover LSBs, XOR operation is performed between least two significant bits of each cover byte and the secret data bits, and then the result is substituted for the least two significant bits of the cover byte. They also used Genetic algorithm (GA) to optimize stego image quality.

The most advantage is that the secret data is not embedded into cover image, instead embeds the result of XOR operation between the cover image and the secret data. Therefore, for retrieving the hidden data, the cover image must be available, which make it hard for those who don’t have the original cover object to extract the data, which in turn increases the robustness.

In the research of (Singh & Kaur, 2015), hiding into image process is done first inside odd pixels then even pixels. When hiding into a pixel, the algorithm embeds two bits into red byte LSBs, two bits into green byte LSBs and four bits into blue byte LSBs.

The algorithm is not very robust, because the hiding is not completely random. The hiding is done into odd pixels sequentially, then into even pixels in the same way. Also it exploits two LSBs of red and green colors and four bits of blue color, which in turn reduce the imperceptibility. The capacity is very high due to using eight bits per pixel.

2.9.2 Audio Steganography

Audio and video is considered good carrier because of the redundancy (Asad, Gilani, & Khalid, 2011). Audio steganography is one of the popular data hiding techniques, which embeds secret data into audio signals. It is based on the masking of human auditory system (Tayel, Gamal, & Shawky, 2016). Many researchers used the LSB substitution technique with improvement to hide data inside audio mediums.

In research of (Asad et al., 2011) two techniques were introduced, Sample Selection which is used to determine the cover samples inside which secret data is embedded and Bit Selection which is used to select the cover bits within a sample. The audio file consists of consecutive samples, but according to Sample Selection technique, not all of them are used to contain data. During the embedding process, after embedding the secret data within the current sample, the Sample Selection technique is used to determine the next sample, where the next sample is equal to the decimal value of the first 3 MSBs of the current sample plus one. So, if the current sample is of index 6 and the value of its first 3 MSBs is 010, which is equal to 2 in decimal, then the next sample would be 3 samples ahead, which is the sample of index 9. After determining the next sample, another technique called Bit Selection is used for determining the bit of the sample to embed the secret data inside. If the first two MSBs of a sample are equal to 00, the third LSB will be replaced with secret message bit. If the first two MSBs are equal to 01, the second LSB will be replaced and if the first two MSBs are either 10 or 11, the first LSB will be replaced with the secret message bit.

The techniques proposed by the researchers make the steganographic approach robust due to adding randomness to the hiding process, where not all samples would contain hidden data. Additionally samples selecting process is random and not regular. Moreover, within the sample, data is embedded into one of the least three significant bits randomly, which increase the robustness. The imperceptibility is high, since the algorithm embed into very little amount of cover samples. According to the Sample Selection technique, the number of samples skipped between the consecutive samples that selected to contain data is at least one, and at most 8. So, on average 4 samples are skipped at a time, which means 25% of the cover samples would contain secret data. The capacity is very low because many samples are skipped, however, since the researchers hide into audio and audio files have much redundancy, then there might be enough capacity for the secret data.

2.9.3 Video Steganography

Video is combination of images and audio, so both image steganography and audio steganography can be used. Videos are considered good carriers in terms of having much capacity because of the abundance of redundancy, and it is hard to detect small changes in the whole video stream (Yadav, Mishra, & Sharma, 2013). LSB substitution is an efficient method for video steganography and many researchers proposed methods to employ LSB substitution in video steganography.

(Balaji & Naveen, 2011) introduced a new method that according to set of functions assigns a frame of the cover video to be the index frame, which is used to locate the frames that would contain the secret data. Functions are also used to identify the frames within which the data would be embedded. The functions depend on the characteristics of the video. They used LSB substitution technique for embedding the secret data. The remaining frames which do not contain secret data are also filled with some random data to increase randomness.

Robustness is high, since secret data is scattered through random frames depending on a set of calculations. Additionally, random data is inserted into frames that don't contain secret data. The imperceptibility is high because of using video mediums, which consist of plenty of frames (images), and it is hard to detect any difference through the frames stream. The capacity is high too, due to using videos as carriers, which have too much redundancy.

The proposed algorithm, which is called Indicator-based LSB, is explained in details including the terms of robustness, imperceptibility and capacity and all of its aspects in section 3.1.

Table (2.2): Comparison of LSB-based algorithms explained in related work based on Imperceptibility, Robustness and Capacity

Cover Type	**Algorithm**	**Imperceptibility (stegogramme quality)**	**Robustness**	**Capacity**
Image	Hide & Seek (sequential LSB) (Bateman & Schaathun, 2008)	High On average, LSBs of 50% of cover bytes are changed	Very Low Sequential embedding	Moderate One bit per byte, relatively good

Cover Type	Algorithm	Imperceptibility (stegogramme quality)	Robustness	Capacity
Image	A New Approach for LSB Based Image Steganography using Secret Key (Karim et al., 2011)	Very High Due to embedding one bit per pixel (one bit within one byte out of three)	Moderate Much more than sequential LSB Due to • using secret key • adding little randomization	Low Due to embedding one bit per pixel (one bit within one byte out of three bytes)
Image	Enhancing the Security and Quality of LSB based Image Steganography (Akhtar et al., 2013)	Very High Bit-inversion technique	High Due to randomness by using RC4 algorithm with secret key	Moderate The same as LSB
Image	An Efficient Filtering Based Approach Improving LSB Image Steganography using Status Bit along with AES Cryptography (Islam et al., 2014)	Very High More than sequential LSB by nearly 83.33% Due to • Hiding into pixels not bytes, one byte out of three • Only lighter or darker pixels not all of them • Among the selected pixels, only pixels matched with a condition	Very High Much Randomization Due to • After determining the lighter or darker, only the matched pixels would contain secret bits • secret bits are not embedded within LSBs, but in random bits, LSBs just contain indications	Very Low Less than Sequential LSB by nearly 83.33% Due to • Hiding into pixels not bytes, one byte out of three • Only lighter or darker pixels not all of them • Among the selected pixels, only pixels matched with a condition

Cover Type	Algorithm	Imperceptibility (stegogramme quality)	Robustness	Capacity
Image	An Improved Image Steganography Scheme with High Visual Image Quality (Laha & Roy, 2015)	High Due to • Embedding into only the least two significant bits • Using Genetic Algorithm to optimize stego image quality	Moderate Due to • No randomness • No secret key • Only it is hard to retrieve the hidden data without the availability or having the original cover object	Moderate More the LSB by 100% Where every secret byte needs 4 cover bytes
Image	Odd-Even Message Bit Sequence Based Image Steganography (Singh & Kaur, 2015)	Low Due to changing too much at cover mediums, where 2 bits at each red byte, 2 bits at each green byte and 4 bits at each blue byte are used to contain secret data	Moderate Randomization due to hiding into odd then even pixels. Also red and green channels have different payload of secret data from blue channel	Very High Due to embedding 8 bits per pixel
Audio	An Enhanced Least Significant Bit Modification Technique for Audio Steganography (Asad et al., 2011)	High Due to • Hiding only into only one of the least three significant bits at a time • Hiding into little amount of the cover bytes, on average 25% of the cover bytes would contain data	High Much Randomization	Low Due to • Embedding only one bit inside a byte • Using on average only 25% of the cover bytes to contain secret data

Cover Type	Algorithm	Imperceptibility (stegogramme quality)	Robustness	Capacity
Video	Secure Data Transmission Using Video Steganography (Balaji & Naveen, 2011)	High Due to • Using video mediums, and it is hard to detect any difference through the video stream • Using LSB technique	High Due to • Distributing secret data through random frames depending on set of calculations • Inserting random data into the frames that don't contain secret data	High Due to • Using video medium, which has a lot of redundancy and capacity
Image	Indicators-based LSB	High Almost as LSB	Very High • Much more Randomization due to moving forward and backward	High Much more than LSB Due to • Embedding into three bytes of image pixels • Embedding sometimes two bits

2.10 Image Quality Metrics PSNR and MSE

Peak signal-to-noise ratio (PSNR) and mean square error (MSE) are the most common and widely-used metrics for image quality evaluation (Al-Mohammad, 2010). PSNR measures the similarity between two images (how two images are close to each other), while MSE measures the difference between two images (how different two images are from each other) (Al-Mohammad, 2010). Therefore, image quality is better with higher value of PSNR and smaller value of MSE. The best image quality is when MSE value is very small or going to be zero, since the difference between the original image and the reconstructed image is negligible (Al-Mohammad, 2010). For PSNR, the higher the PSNR value, the better the degree of imperceptibility, since the similarity between the original image and the reconstructed image is high (Al-Mohammad, 2010). However, PSNR values between 20 and 40 can be considered as typical values (Eric, 2003). For example, it is difficult to recognize any difference between a grey-scale cover image and its stego image if the PSNR value exceeds 36 dB (Al-Mohammad, 2010). PSNR and MSE are defined as follows (Al-Mohammad, 2010):

$$\mathrm{MSE} = \left(\frac{1}{\mathrm{MN}}\right) \sum_{\mathrm{i}=1}^{\mathrm{M}} \sum_{\mathrm{j}=1}^{\mathrm{N}} \left(\mathrm{X}_{\mathrm{ij}} - \overline{\mathrm{X}}_{\mathrm{ij}}\right)^2 \qquad (2.1)$$

$$\mathrm{PSNR} = 10.\log_{10} \frac{\mathrm{I}^2}{\mathrm{MSE}} \mathrm{db} \qquad (2.2)$$

Where:

X_{ij} is the i^{th} row and the j^{th} column pixel in the original image,

$\overline{\mathrm{X}}_{\mathrm{ij}}$ is the i^{th} row and the j^{th} column pixel in the reconstructed image,

M and N are the height and the width of the image,

I is the dynamic range of pixel values, or the maximum value that a pixel can take, for 8-bit images: I=255.

However, the MSE for color images is defined as follows (Al-Mohammad, 2010):

$$\mathrm{MSE}_{\mathrm{AVG}} = \frac{\mathrm{MSE}_{\mathrm{R}} + \mathrm{MSE}_{\mathrm{G}} + \mathrm{MSE}_{\mathrm{B}}}{3} \qquad (2.3)$$

Where: R MSE, G MSE and B MSE are the MSE of red, green, and blue respectively.

2.11 Steganalysis Principles

Steganalysis is the art of identifying and detecting stegogrammes that contain hidden data (Bateman & Schaathun, 2008). So the main aim of steganalysis is merely detecting stego files, however beside stego-mediums detection, it involves data extraction and data destruction (Al-Mohammad, 2010). Whilst the aspiration of recovering the message should be considered extremely unlikely because to recover the hidden data, the steganalyst needs to discover the hiding algorithm, and the hidden data itself in most cases would have been encrypted before getting embedded. So, steganalysis succeeds and the steganography system is broken if merely an attacker detects the existence of the hidden data. Persons apply steganalysis with the intent of intercepting and detecting stego-mediums and the hidden data in the communication channels are referred to as steganalysts (Kipper, 2003). Generally, modifying some parts of a cover file to embed secret data inside, changes the properties of this file in some way, and this can be a sign of the presence of hidden data (Al-Mohammad, 2010). Therefore, applying a comparison between a stego file and its corresponding cover file may reveal the existence of the hidden message. Thus, to avoid such a comparison, cover files used for hiding data should not be publicly available, and after the embedding, they may have to be destroyed (Al-Mohammad, 2010). After embedding the secret data within a cover medium, the resulting stegogramme is in most cases sent to a remote recipient over some communication channel. During the way it may get subjected to steganalysis. After detecting a stego-medium by steganalysis techniques, attacker may attack the communication in different forms. Steganography attacks could be categorized into three kinds in accordance with the role of the steganalyst as passive attack, active attack and malicious attack (Al-Mohammad, 2010; Bateman & Schaathun, 2008; Kipper, 2003):

1. **Passive Attack** is when the warden just observes the communication and permits or prevents the message delivery without performing any modification to the stego mediums. Therefore, the communication between two parties will be blocked in case the warden suspects that a secret communication is taking place.
2. **Active attack** is when the warden alters the detected stegogrammes and causes distortion to them during the communication, so that the communication is prevented. In such attack, the attacker may aims to alter the passing files, even though there is no suspicion, in order to destroy any hidden data might be existed.

3. **Malicious Attack** is when the warden replaces the hidden message with a fake message, and tries to impersonate one of the communicating parties to trick them. On the other hand, this type of attack is too hard to apply, because the attacker must know everything about the hiding process, like the algorithm used and the secret key if exists. Moreover, this attack is kind of easy to be detect by the receivers, since they would notice that the hidden messages don't make sense.

2.11.1 Steganalysis Techniques

As defined before, steganalysis is the science concerned with detecting the covert communication taking place by detecting hidden data within stego mediums passing in between. Steganographic systems leave behind in stego files some traces as a result of embedding the secret message inside. These traces make the stego files detectable in some way. So, steganalysis focuses on taking advantage of these traces to detect the stego files.

2.11.1.1 Targeted Steganalysis

Targeted steganalysis techniques are designed in direct accordance with a specific methods of embedding, where they attempt to discover stego files by checking the known side-effects of specific steganographic algorithms, so it requires to have deep knowledge about the steganographic algorithm that the attack is targeted to (Bateman & Schaathun, 2008).

2.11.1.1.1 Visual Attacks

Visual Attacks are the process of examining the subject file or certain components of it by naked eye to identify any obvious inconsistencies with assistance of software (Bateman & Schaathun, 2008). Of course stego files with quality degradation as a result of steganographic manipulation look suspicious than the cover files, and could be detected by naked eye. So, the first rule to avoid visual attacks is that a steganographic system should keep quality of cover files as hiding data inside. When steganalysts perform visual attacks, they concentrate in isolation on the likely areas of embedding inside stego files to detect signs of manipulation.

The most common visual attack combats LSB-based steganography, and is made based on the fact that the structure of LSBs of a cover image does not match the structure of message bits (Bateman & Schaathun, 2008). The attack depends on viewing the LSB plane of the suspect image to identify any inconsistency that indicates existence of hidden data. For Images, there are almost even values as there are odd, which means there are as many 0's as there 1's in its LSB plane (Bateman & Schaathun, 2008). When the text meant to be hidden, it is converted to binary or ASCII, the resulting bit stream would contain unequal numbers of 0's and 1's, where the number of 0,s is larger than the number of 1's (Bateman & Schaathun, 2008). Thus, replacing the LSB values with the ASCIIs of the text would increase 0's and in turn result in inconsistency in the LSB plane. So, the part of LSB plane that has hidden data would be visually different from the clean part. Steganalysts who perform visual attacks search for signs of embedding in the LSB plane by searching for such difference. Figure 2.4 shows an example of visual attack on a true color PNG image, where image (a) is clean and image (b) is the same image after being manipulated to contain secret data.

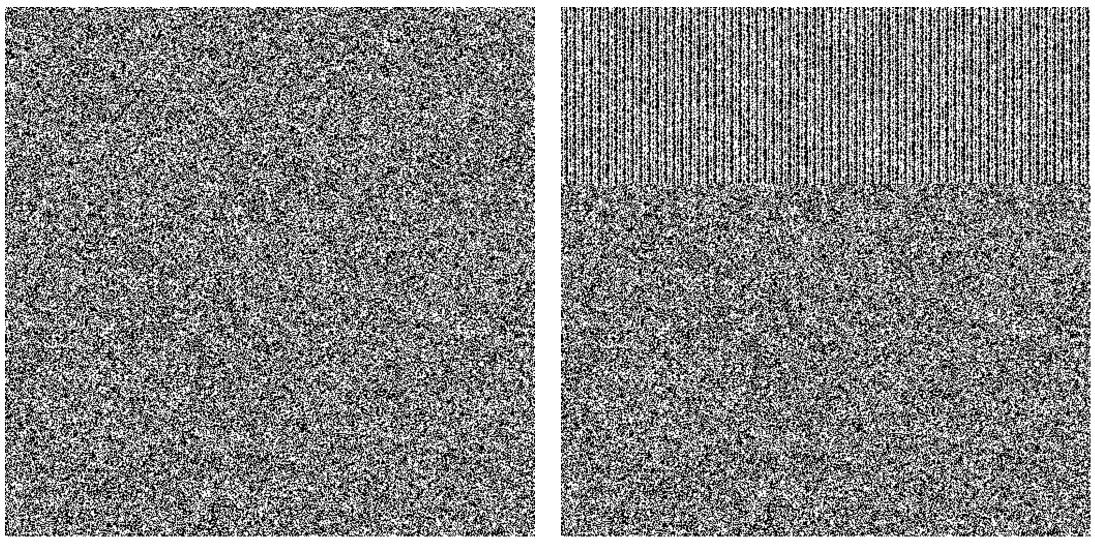

(a) Clean Image (b) Stego Image

Figure (2.4): The LSB Bit Plane Before and After Embedding Unencrypted Data

It is clear that image (b) has hard indication of embedding data within the first 33% of the LSB plane of the image. This kind of attack enables steganalyst to figure out the length of the hidden message. Also it is obvious that this attack doesn't need the original image to work, since the discrepancy is detected by checking the LSB plane of the suspect image.

Visual attack fails when the secret data is encrypted before getting embedded inside the cover image. This is because the binary ASCII of the text has often 0's more than 1's, and by encrypting the secret data there will usually be a more even distribution of 1's and 0's (Bateman & Schaathun, 2008). Figure 2.5 shows the same image of Figure 2.4, but image (b) here contains encrypted data.

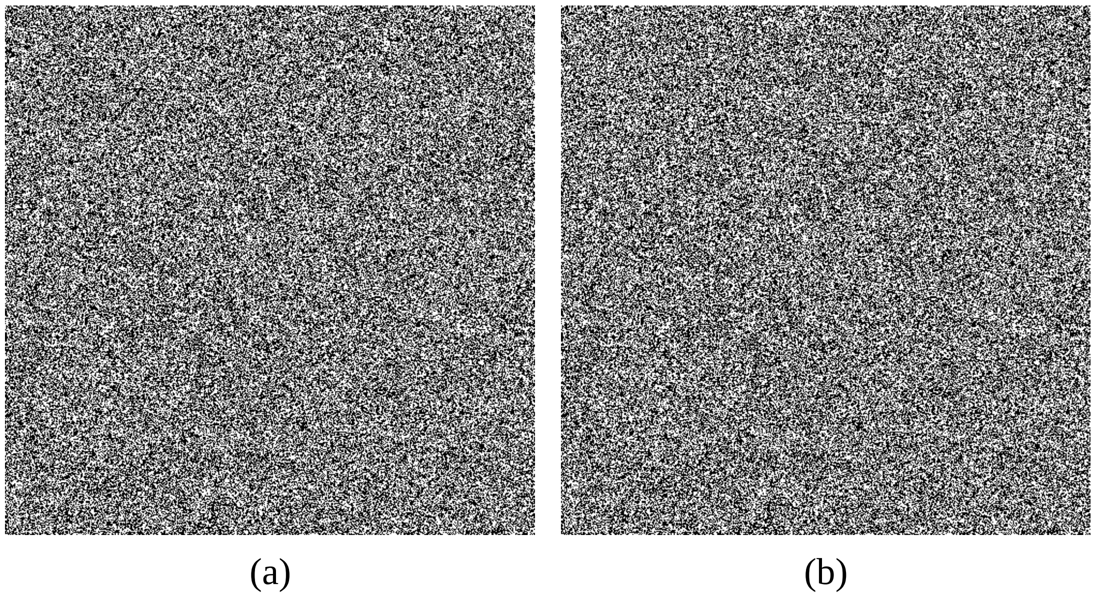


Figure (2.5): The LSB Bit Plane Before and After Embedding Encrypted Data

It is obvious that the image after embedding the secret data has no discrepancy, and this is due to hiding encrypted data. Also hiding the data randomly not sequentially results in no discrepancy within the image LSB plane. This will be shown later in the experiments and results chapter when our algorithm is used for hiding the data randomly. So, to defeat the visual attack, first the secret data must be encrypted before the embedding, next the embedding must be performed randomly and not sequentially (Bateman & Schaathun, 2008).

Another way of visual attacks is done if the steganalyst has access to the clean image. This is referred to as known cover attack. When the clean image is available, the steganalyst has the ability to get the LSB plane of both, the clean and the suspect image, and calculate the difference between them by subtracting one from the other to identify the identical and the different regions of the images.

Finally, the most trait that makes visual attacks too hard to perform is that visual attacks are not automated, and needs human to check each image for identifying the suspect images.

2.11.1.1.2 Statistical Attacks

These attacks depend on detecting the modifications which have occurred in the statistical properties of cover files (Al-Mohammad, 2010). Statistical Attacks may reveal that a file had been modified, but it can’t identify which technique was used for modification. Statistical Attacks are often preferred because they can be automated (Bateman & Schaathun, 2008). For images, there are several statistical properties which can be analyzed such as standard deviation, differential values, median, skew and kurtosis (Al-Mohammad, 2010). There are several statistical attacks, and here are some of them:

Chi-square Test (Westfeld & Pfitzmann, 1999) enables steganalysts to compare the statistical properties of a suspect image with the theoretically expected statistical properties of its counterpart, where the degree of similarity of them is a measure of the probability of embedding. The test is based on statistical analysis of Pairs of Values (PoVs) that are exchanged during message embedding (Bateman & Schaathun, 2008; Westfeld & Pfitzmann, 1999).

Histogram Attacks depend on histogram analysis to identify whether there is steganography or not (Bateman & Schaathun, 2008). For instance, the Difference Histogram Analysis (Zhang & Ping, 2003) is a statistical attack on an image's histogram, measuring the correlation between the LSB and all other bit planes.

RS Analysis (Fridrich, Goljan, & Du, 2001) can detect 24-bit color images and 8-bit grayscale with randomly scattered LSB embedding by inspecting the differences in the number of regular and singular groups for the LSB and shifted LSB plane.

Sample Pair Analysis (Dumitrescu, Wu, & Wang, 2003) is a steganalysis technique for LSB substitution depending on a finite state machine that can detect randomly embedded messages in LSB Plane.

Primary Sets (Dumitrescu, Wu, & Memon, 2002) is a steganalysis technique that can detect the randomly embedded messages in LSBs of natural continuous-tone images.

2.11.1.2 Blind Attacks

These attacks attempt to evaluate the probability of embedding based solely on the data of the suspect image, even when it is not known how the data might has been embedded. It assumes that nothing is known about either the algorithm or the cover image (Bateman & Schaathun, 2008). Some of the most popular blind attacks are Wavelet Moment Analysis (WAM), Calibration Based Attacks and Farid’s Wavelet Based Attack (Goel, 2008).

2.11.2 StegExpose – Steganalysis Tool for Detecting Steganography in Images

StegExpose is a steganalysis tool specialized in detecting steganography of LSB substitution in lossless images such as PNG and BMP (Boehm, 2014). StegExpose can be run in the background analyzing multiple images without human supervision, returning a detailed steganalytical report once the tool has finished its job. StegExpose is derived from an intelligent and thoroughly tested combination of pre-existing LSB steganalysis methods which are Chi-square Attack (Westfeld & Pfitzmann, 1999), RS Analysis (Fridrich et al., 2001), Primary Sets (Dumitrescu et al., 2002), Sample Pairs Attack (Dumitrescu et al., 2003) and Difference Histogram analysis (Zhang & Ping, 2003).

2.12 Summary

In this chapter we have introduced the reader to the main issues of steganography. We have given an introduction and identified the core concepts and principles of this field. We have illustrated steganography and its aspects in depth. Also we have covered some other subjects concerning steganography. We have explained image steganography and its categories. Furthermore, we have discussed some of the most related work. Finally we have covered Steganalysis and its techniques, and talked about one of its tools which is going to be used for testing.

Chapter 3
Proposed Solution

Since steganography has been discovered, too many algorithms were developed for enhancing information hiding and covert communication in turn. Information hiding algorithms have their own points of strength and weakness. In this chapter, a new algorithm is presented for hiding data inside cover mediums depending on indicators.

3.1 Embedding Process

In our approach of concealing data, we hide the secret data bits into the least one or two significant bits (right-most bits), but this process is done depending on indicators. Images are chosen as cover mediums. Before the embedding process starts, the cover image is split into bytes. The bytes represent the color channels of the cover image pixels, where every three bytes represent the Red Green and Blue of a pixel. Therefore, data is hidden into the three color channels of each pixel. So, the cover image is treated as a stream of bytes through the hiding process.

Through the hiding process, an indicator is used to identify the byte into which we embed the secret bit(s), and another indicator to determine how many bits to embed at a time. An indicator is a fixed bit in each byte of the cover bytes other than the least two bits, because the least two bits are used to contain the secret data bits. Through the hiding process, each cover byte is used to tell which is the cover byte to embed the current secret bit(s) into and how many secret bit(s) to embed. At each iteration, the byte currently being used to find out where and how many secret bits to embed is called the Indicator Byte. Inside the Indicator Byte, the bit that tells us where to embed the secret bit(s) is called the Location Indicator (LI), and the bit that tells us how many secret bits to embed at once is called the Amount Indicator (AI). Let's assume the Location Indicator is the fourth bit (the bit with index 3 from the right), and the Amount Indicator is the third bit (the bit with the index of 2 from the right). According to the Location Indicator value inside the Indicator Byte (the byte that currently the embedding operation is performed according to its indicator bits values), we determine the byte into which we hide our secret data. If the value of the

Location Indicator is zero, then our current secret bit(s) will be embedded into some cover byte of the previous bytes before the Indicator Byte itself, and if the value of the Location Indicator is 1, our current secret bit(s) will be embedded into exactly the next cover byte after the Indicator Byte. Through the hiding process we scan the Location Indicator (fourth bit) of each cover byte of the cover data to identify where to hide the current secret bit(s) of the secret data. This procedure is done starting from the second cover byte as the Indicator Byte to ensure that there is a previous byte, and the final Indicator Byte is the one before the last byte of the cover data to ensure there is a next byte.

As we see when the hiding process is directed to embed into the next byte according to the value of the Location Indicator, the embedding is done into exactly the next byte after the Indicator Byte. But when the embedding is aimed to be done into the previous byte it is not the same. The previous byte is not always the byte exactly before the indicator byte. Most times, it is before the indicator byte by random number of bytes. So we can’t tell which byte before the indicator byte is the previous byte directly. Consequently, a mechanism should be followed for controlling the process of hiding.

In the beginning of the hiding process, the first byte would be the previous byte and the second byte is the Indicator Byte. As long as the indicator bytes hide into the next byte, the first byte remains the previous byte that would be used by any Indicator Byte that decides to embed secret data inside the previous byte. When some Indicator Byte uses the previous byte, the next unused byte will be the previous byte. If the first Indicator Byte, which has the index 1, embedded the secret bit(s) into the next byte of index 2, that means the first two bytes of indices 0 and 1 would be empty of secret data, and the third byte with index 2 would be full of secret data. So, the first two bytes of indices 0 and 1 would be the previous bytes for the next two iterations that decide to embed into the previous byte. This is because as we see, the previous byte is before the Indicator Byte, and the next byte is after the Indicator Byte, so the Indicator Byte itself is skipped. Subsequently, this Indicator Byte would be the next previous after the current previous is used. So, to control the process of hiding, we need to constantly determine and be aware of the previous byte that is going to be used whenever the process decides to embed into the previous byte, which is called current-previous, and to be aware of the previous byte for next time we need to embed into the previous, which is called next-previous. Thus, simply in the beginning of the hiding process, the first Indicator Byte would be the second byte, which is the byte of index 1. The current-previous would be the byte of index 0, and the next-previous would be, just like the indicator byte, the byte with index 1, because the first two bytes would not be used as next bytes. Now as long as each Indicator Byte is hiding into the next byte, the current-previous and the next-previous never change. When some indicator byte decides to embed into the previous byte, the secret data is embedded into the current-previous, then the current-previous pointer moves to the next-previous and the next-previous pointer moves to the next Indicator Byte after the current indicator byte. That means every time after the embedding is done into the current-previous, the next-previous becomes the current-previous and the next Indicator Byte after the current Indicator Byte becomes the next-previous. Finally the indicator just steps ahead to the next Indicator Byte and so on.

Also through the hiding process, we don't always embed only one bit at a time. We may embed one bit or two bits into the cover byte. This is done depending on another indicator bit of the Indicator Byte, which called the Amount Indicator (AI). Again let's assume the Amount Indicator which tells us how many secret bits to embed at a time is the third bit (the bit with the index of 2 from the right). If the value of the Amount Indicator is 0, we embed only one bit, and if it is one, we embed two bits at once into the cover byte. This operation adds more randomness to the hiding process because the amount of the embedded bits is not fixed. So, it is hard to identify the number of embedded bits into each cover byte without checking the Amount Indicator value. On the other hand, this operation increases the capacity of the hiding process on average by 50% over normal LSB substitution technique, which embed only one bit at a time, and this is another advantage besides increasing the randomness. Hiding two bits at once causes some change to the byte value, but this change is at the range from 0 to 3 at maximum, which is a very small change and not noticeable by human eye or ear.

So simply we scan all the cover bytes of the cover data, and check the values of the indicator bits (the Location Indicator and the Amount Indicator) in each cover byte. The Location Indicator is the indicator that tells us where to hide. If its value is zero, then we hide into some previous byte, and if it is one, we hide into exactly the next Indicator Byte after the current Indicator Byte. The Amount Indicator is the indicator that tells us how many bits to hide at once. If its value is zero, then we hide only one bit, and if it is one, we hide two bits at once.

To make the algorithm more robust and secure, a secret steganographic key is used through the hiding process. Steganographic key is used for controlling the embedding and extracting process (Westfeld & Pfitzmann, 1999). The secret key is a series of bits which can be represented with one dimensional circular array. At each cycle of the hiding process, the Location Indicator bit, which is used for deciding the byte into which we embed the current secret bit, is XORed with the current bit of the secret key. If the resulting value of the XOR operation is zero, then we hide into some previous byte, and if it is one, we hide into exactly the next indicator byte. Also the Amount Indicator bit, which is used for deciding how many bits to embed, is XORed with the next bit of the secret key, and if the resulting value of the XOR operation is zero, then we hide one bit, and if it is one, we hide two bits at once. The secret key array is circular, so every time we reach to its end, we return to the beginning and so on. So we can use secret keys of any size. A secret key is chosen and entered by the user. Figure 3.1 shows the flow chart of the hiding process.

As an example for clarifying how the algorithm works, suppose we want to use Indicators-based LSB Algorithm to hide the following secret bits 1110110001100110 starting from the right bit, into the below series of cover bytes using letter M with ASCII of 01001101 as the secret key, then the hiding process is going to be as next:

Secret data bits: 1110110001100110

Secret key bits: 01001101

Cover bytes to hide the secret data into:

0	1	2	3
10111000	10000001	10100001	01111000
4	5	6	7
01110010	10011100	11010001	01010111
8	9	10	11
01010101	10110101	11010011	11110000

The steps of the hiding process are explained next in Table 3.1 step by step:

Table (3.1): The Iterations of the Hiding Process of the Example

Before Embed		**In Byte Index**	**In Bits Val**	**Key Bits Val**	**XOR**		**Dest Byte Index**	**Dest Bits Val**	**Sec Bits Val**	**After Embed**	
CP	**NP**				**Val**	**Mn**				**CP**	**NP**
0	1	**1**	00	01	01	P2	**0**	00	10	1	2
1	2	**2**	00	11	11	N2	**3**	00	01	-	-
1	2	**3**	10	00	10	N1	**4**	0	0	-	-
1	2	**4**	00	01	01	P2	**1**	01	11	2	5
2	5	**5**	11	01	10	N1	**6**	1	0	-	-
2	5	**6**	00	11	11	N2	**7**	11	00	-	-
2	5	**7**	01	00	01	P2	**2**	01	11	5	8
5	8	**8**	01	01	00	P1	**5**	0	0	8	9
8	9	**9**	01	01	00	P1	**8**	1	1	9	10
9	10	**10**	00	11	11	N2	**11**	00	11	-	-

Abbreviations: CP= Current-Previous, NP= Next-Previous, In= Indicator,
Val= Value, Mn= Mean, Dest= Destination, Sec= Secret.

As shown in Table 3.1, the hiding process is not done sequentially like Hide & Seek algorithm (sequential LSB Substitution). The secret bits are hidden into cover bytes randomly depending on the values of the indicators bits of the cover bytes and the secret key. For example, in the first iteration, the current-previous points to byte 0 and the next-previous points to byte 1. The Indicator Byte of index 1 is going to a decide about the current embedding, so the indicators bits of the Indicator Byte of index 1 (00) is XORed with the current secret key bits (01). The result of the XOR operation was (01), which means to hide in the current-previous two bits (0 means to hide into the previous and 1 means to hide 2 bits). After the embedding has been done, the current-previous was moved to the next-previous and the next-previous and the indictor were moved to the next Indicator Byte.

As we see in Table 3.1, the order of the cover bytes that were embedded into is 0, 3, 4, 1, 6, 7, 2, 5, 8, 11 and the amount of embedded bits into each of which respectively is 2, 2, 1, 2, 1, 2, 2, 1, 1, 2. Here we can realize that unlike sequential LSB approach, the hiding process is not sequential and on the other hand, some bytes contain only one secret bit and others contain two bits. Table 3.2 below shows the cover bytes before and after the hiding, where it shows the status of the cover bytes and what LSBs were changed and what were not.

Table (3.2): Series of Cover Bytes Before and After Embedding by Indicators-based LSB Algorithm

Cover byte Index	**The series of cover bytes before the embedding process**	**The resulting bytes after the embedding process**
0	**10111000**	**101110[10)**
1	**10000001**	**100000[11)**
2	**10100001**	**101000[11)**
3	**01111000**	**011110(01]**
4	**01110010**	**0111001(0)**
5	**10011100**	**1001110(0)**
6	**11010001**	**1101000[0]**
7	**01010111**	**010101[00]**
8	**01010101**	**0101010(1)**
9	**10110101**	**10110101**
10	**11010011**	**11010011**
11	**11110000**	**111100[11]**

(b) **The bit new value is identical to its original**
[b] **The bit new value is different from its original**
(bb] **Only the right bit new value is different from the original**
[bb) **Only the left bit new value is different from the original**
(bb) **Both new values are identical to the original**
[bb] **Both new values are different from the original**

For robustness, since we use indicators to identify the byte to hide into, this procedure increases the randomness of the hiding process, because we sometimes embed into the previous byte and other times into the next byte. Hence, we can see that the hiding doesn’t keep moving forward after each hiding iteration. Since it embeds into next and previous, then it keeps moving forward and backward. Moreover, since the previous byte is not fixed for each Indicator Byte, where the previous byte may be any byte of the bytes before the Indicator Byte, then it moves backward random number of locations, which adds extra randomness to the process of data hiding. According to the Amount Indicator we sometimes hide one bit and other times we hide two bits. Finally using the secret key increases the security, where it becomes much harder to retrieve the hidden information in the absence of the secret key. All of these factors increase the randomness of the hiding process and thus make the process of retrieving the hidden data by unauthorized parties much more complex, which increases the robustness and subsequently the security, and this is the main aim of the proposed approach.

For imperceptibility, the algorithm results in stego images with high imperceptibility according to the experiments discussed in section 4.2.1.

For capacity, due to embedding into three colors of each pixel, not only one color as done when image pixels are used as units of embedding, the capacity of cover images is increased over embedding into only one color of each pixel by 66%. Also, because of embedding sometimes two bits, depending on the Amount Indicator, the capacity is increased on average by 50% over embedding one bit constantly.

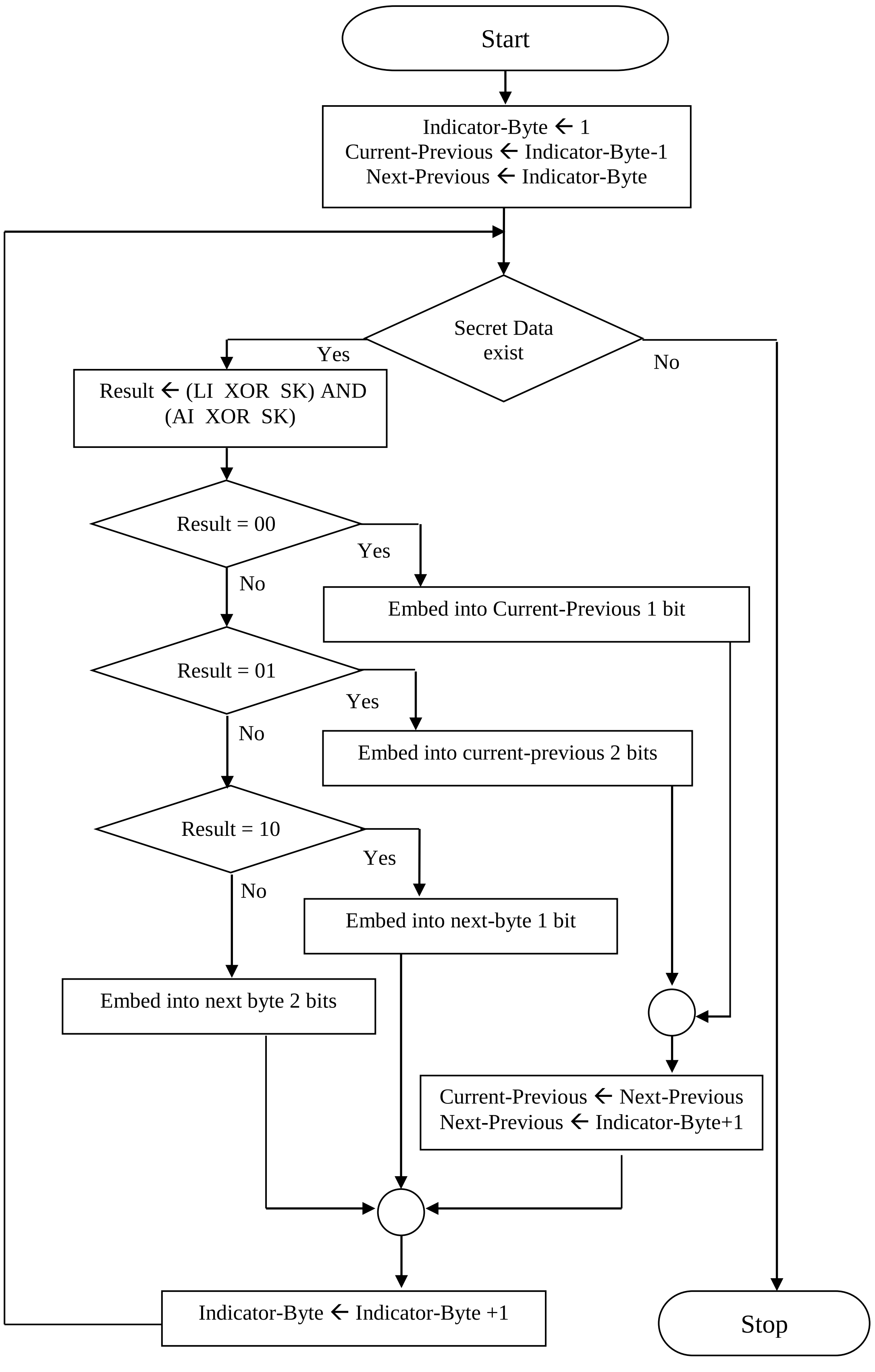

Start
Indicator-Byte ← 1
Current-Previous ← Indicator-Byte-1
Next-Previous ← Indicator-Byte
Secret Data exist
Yes
No
Result ← (LI XOR SK) AND (AI XOR SK)
Result = 00
Yes
No
Embed into Current-Previous 1 bit
Result = 01
Yes
No
Embed into current-previous 2 bits
Result = 10
Yes
No
Embed into next-byte 1 bit
Embed into next byte 2 bits
Current-Previous ← Next-Previous
Next-Previous ← Indicator-Byte+1
Indicator-Byte ← Indicator-Byte +1
Stop

Figure (3.1): The Embedding Flow Chart, LI= Location Indicator, AI= Amount Indicator, SK= Secret Key

3.2 Retrieving Process

The retrieving process is simply the inverse of the embedding process. So as we depend on indicators through embedding process, we use the same indicators for retrieving. When we want to extract hidden data from the stego object, we check both, the Location Indicator and the Amount Indicator using the same secret key used through the embedding process to identify the location and the amount of the embedded bits. Simply we scan each cover byte, as an Indicator Byte, sequentially and check its indicator bits (the Location Indicator and the Amount Indicator). The Location Indicator value is XORed with the secret key, if the result is 0, then the secret bit(s) is retrieved from some previous byte, and if it is 1, then it is retrieved from the next byte. Also the Amount Indicator is XORed with the secret key, and if the result is 0, then one bit is retrieved, otherwise 2 bits are retrieved.

The steps of the retrieving the hidden data from the stego bytes resulted through the embedding example in section 3.1 is explained in Table 3.3 step by step. The retrieving process is done using the same secret key. Through the retrieving process, the first bit embedded is the first bit retrieved and so on.

Secret key bits: 01001101

Stego bytes which contain the hidden data, where the underlined least bits have embedded data.

0	1	2	3
101110**10**	100000**11**	101000**11**	011110**01**
4	5	6	7
0111001**0**	1001110**0**	1101000**0**	010101**00**
8	9	10	11
0101010**1**	10110101	11010011	111100**11**

Table (3.3): The Iterations of the Retrieving Process of the Example

Before Embed		In Byte Index	In Bits Val	Key Bits Val	XOR		Dest Byte Index	Dest Bits Val	After Embed	
CP	**NP**				**Val**	**Mn**			**CP**	**NP**
0	1	**1**	00	01	01	P2	**0**	10	1	2
1	2	**2**	00	11	11	N2	**3**	01	-	-
1	2	**3**	10	00	10	N1	**4**	0	-	-
1	2	**4**	00	01	01	P2	**1**	11	2	5
2	5	**5**	11	01	10	N1	**6**	0	-	-
2	5	**6**	00	11	11	N2	**7**	00	-	-
2	5	**7**	01	00	01	P2	**2**	11	5	8
5	8	**8**	01	01	00	P1	**5**	0	8	9
8	9	**9**	01	01	00	P1	**8**	1	9	10
9	10	**10**	00	11	11	N2	**11**	11	-	-

Abbreviations: CP= Current-Previous, NP= Next-Previous, In= Indicator, Val= Value, Mn= Mean, Dest= Destination, Sec= Secret.

As we see the retrieved data after applying the retrieving process is 1110110001100110, which is identical to the original data.

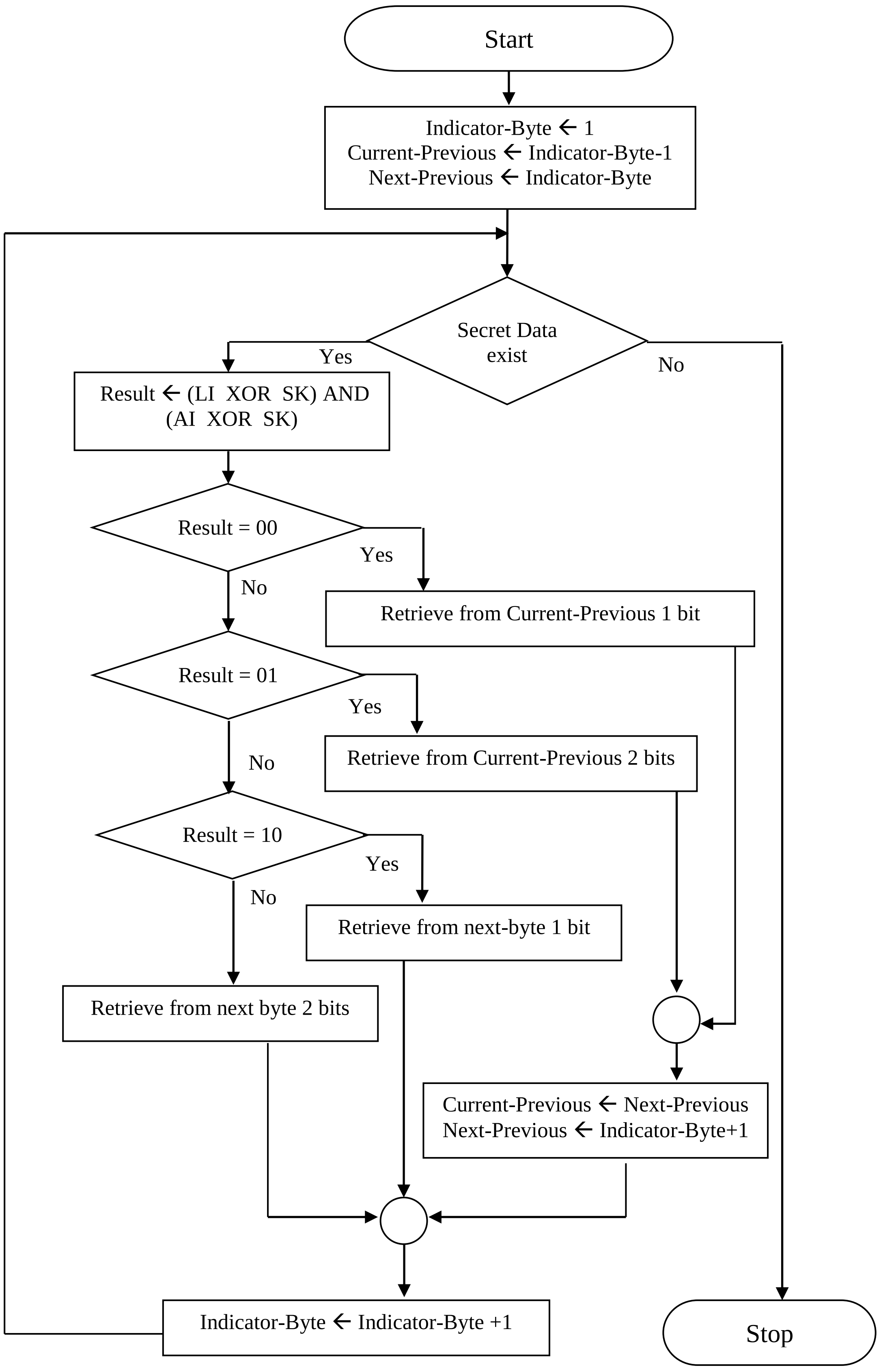

Start
Indicator-Byte ← 1
Current-Previous ← Indicator-Byte-1
Next-Previous ← Indicator-Byte
Secret Data exist
Yes
No
Result ← (LI XOR SK) AND (AI XOR SK)
Result = 00
Yes
No
Retrieve from Current-Previous 1 bit
Result = 01
Yes
No
Retrieve from Current-Previous 2 bits
Result = 10
Yes
No
Retrieve from next-byte 1 bit
Retrieve from next byte 2 bits
Current-Previous ← Next-Previous
Next-Previous ← Indicator-Byte+1
Indicator-Byte ← Indicator-Byte +1
Stop

Figure (3.2): The Retrieving Flow Chart, LI= Location Indicator, AI= Amount Indicator, SK= Secret Key

3.3 Summary

In this chapter we have introduced a new algorithm of hiding data inside cover mediums. We have illustrated how it works and how it satisfies the randomness over sequential LSB. We also have shown its points of strength and weakness. Finally we have explained the retrieving process which is simply the invers of the embedding process. In the next chapter, the proposed algorithm is tested over an image dataset, and the results are discussed in details.

Chapter 4
Experiments and Results

As every scientific field, steganography has evaluation scheme for steganographic systems, to identify which algorithm is better. Currently, no test or measure is considered as standard. However, there are guidelines and general procedures can be used for evaluation (Al-Mohammad, 2010). In this chapter, we present the experiments we carried out to evaluate the proposed Indicators-based LSB algorithm. Also, we introduce and illustrate the measures we considered for evaluating our steganographic system effectiveness and efficiency. The evaluation is done to find out how good is the algorithm in general, after evaluating all of the aspects considered by steganography.

4.1 Steganography Aspects for Evaluation

To evaluate steganography algorithms, we need to take into account the purpose of steganography field to measure the degree of how much an algorithm meets that purpose. As clarified before, the main purpose of steganography is hiding the communication to preserve the security of the information. Steganography implements covert communication by hiding the presence of the secret data inside stego mediums. For hiding the presence of the secret data, stego files must not arouse any suspension to avoid getting detected. Thus, the first aspect of steganography algorithms to evaluate is the imperceptibility which is concerned with making the stego files perceptually undetectable, and this is done by making stego files as identical to the cover files as possible. For images, the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR) are the metrics of the difference and similarity between images before and after processing. So, through the tests, they are going to be the metrics of the difference and similarity between cover images and stego images. Another aspect that could be considered is the capacity of the algorithm. Since we need to hide the data for transferring it over the Internet, so the larger the amount of data that can be loaded and sent at once the better the algorithm. Here we encounter the fact that the more data we hide inside a cover file, the more distortion we cause, which in turn increases the probability of detectability. Hence, as mentioned, there is a tradeoff between capacity and imperceptibility, and it is obvious that imperceptibility is more important to maintain, since it is a main component of the hidden data security. Therefore, increasing the imperceptibility is considered a significant contribution. Additionally, increasing the capacity is a good contribution, but with maintaining the imperceptibility. The last aspect is the robustness which is the degree of how much an algorithm can resists steganalysis. Robustness depends on the algorithm mechanism of data embedding. So, there are no metrics for robustness, and its evaluation is the evaluation of the strength of the algorithm itself in terms of complexity and randomness. However, for measuring the robustness, some steganalysis methods would be applied to see how much the algorithm can resist attacking by passing the attacks without getting detected. Hence, we want to measure the percentage of the data that can get embedded to the size of the cover image without getting detected when the stego image is subjected to steganalysis. Also, since we use LSB technique for data hiding, we would show the least and the second least bit planes of one image as a sample to check if there are any visual signs of embedding.

4.2 Experiments and Results Discussion

In this section we show the experiments that have been done and discuss the results. To measure the efficiency of our algorithm, the algorithm has been tested over a dataset of cover images. Our dataset consists of ten true-color images, which are gathered from two sources. First source is USC-SIPI image database (The USC-SIPI Image Database, 1977), which contains the famous images globally used for steganographic algorithms evaluation such as Pepper. Secondly, some random images collected from the Internet. Since we embed the secret data inside the bytes that consists the pixels, so the number of bytes that can be used depends on the number of the cove image pixels. Since the algorithm embeds in the spatial domain, lossless images as PNG and BMP should be used as cover images. All the images were chosen of one type which is PNG, to make the type of image a fixed factor through the experiments. The images consists of seven images with dimensions of 512×512, two images of dimensions of 256×256 and one image of dimensions of 1030×1060. All the cover images are shown below in Table 4.1.

Table (4.1): Cover Images for Experiments

No	Name	Type	Dimensions (pixel)	Number of Bytes	Image
1	Splash	PNG	512 × 512	786,432	
2	Sailboat on lake	PNG	512 × 512	786,432	
3	Airplane F-16	PNG	512 × 512	786,432	
4	Nature	PNG	512 × 512	786,432	
5	Parking	PNG	512 × 512	786,432	
6	Peppers	PNG	512 × 512	786,432	
7	House 2	PNG	512 × 512	786,432	
8	House 1	PNG	256 × 256	196,608	
9	Tree	PNG	256 × 256	196,608	
10	MATLAB	PNG	1030 × 1060	3,275,400	

4.2.1 Imperceptibility and Detectability Tests and Results

For performing the experiments, we have embedded gradual amounts of data into the cover images. The amount of hidden data is not fixed, rather it is relative to the number of the image pixels. Data that would be hidden into an image is generated to fill certain percentage of the image pixels. For each image, Data is embedded within 10% of the image, where we increase the percentage each time by 5 until 45%, so we obtain eight stego images containing gradual data amounts. Then MSE and PNSR values are found for each stego image to measure the imperceptibility and the impact of the hiding process. All stego images are subjected to a the StegExpose steganalysis tool to see the undetectability. The results are shown below in Table 4.2.

Table (4.2): Results of Stego Images Obtained by Embedding Experiments

NO	Name	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
1	Splash	10%	14,160	0.14	56.78	✓
		15%	21,225	0.20	55.03	✓
		20%	28,291	0.27	53.79	✓
		25%	35,380	0.34	52.82	✓
		30%	42,460	0.41	52.03	✓
		35%	49,550	0.48	51.36	✓
		40%	56,590	0.54	50.78	✕
		45%	63,641	0.61	50.28	✕
2	Sailboat on lake	10%	14,183	0.14	56.83	✓
		15%	21,296	0.20	55.06	✓
		20%	28,397	0.27	53.81	✓
		25%	35,461	0.34	52.84	✓
		30%	42,545	0.41	52.05	✓
		35%	49,634	0.47	51.38	✕
		40%	56,708	0.54	50.80	✕
		45%	63,801	0.61	50.29	✕
3	Airplane F-16	10%	14,133	0.13	56.89	✓
		15%	21,221	0.20	55.12	✓
		20%	28,305	0.27	53.87	✓
		25%	35,394	0.34	52.88	✓
		30%	42,469	0.40	52.08	✓
		35%	49,563	0.47	51.41	✕
		40%	56,617	0.54	50.83	✕
		45%	63,724	0.61	50.31	✕

NO	Name	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
4	Nature	10%	14,175	0.13	56.87	✓
		15%	21,258	0.20	55.12	✓
		20%	28,329	0.27	53.87	✓
		25%	35,385	0.33	52.90	✓
		30%	42,455	0.40	52.11	✓
		35%	49,543	0.47	51.43	✕
		40%	56,628	0.53	50.85	✕
		45%	63,717	0.60	50.34	✕
5	Parking	10%	16,744	0.15	56.47	✓
		15%	25,102	0.22	54.75	✓
		20%	33,422	0.29	53.57	✓
		25%	41,713	0.35	52.66	✓
		30%	50,013	0.42	51.92	✓
		35%	58,305	0.49	51.27	✕
		40%	66,617	0.55	50.70	✕
		45%	74,899	0.62	50.21	✕
6	Peppers	10%	14,160	0.14	56.82	✓
		15%	21,232	0.20	55.06	✓
		20%	28,287	0.27	53.82	✓
		25%	35,335	0.34	52.85	✓
		30%	42,394	0.41	52.05	✕
		35%	49,455	0.47	51.39	✕
		40%	56,494	0.54	50.80	✕
		45%	63,531	0.61	50.29	✕
7	House 2	10%	14,095	0.13	57.01	✓
		15%	21,248	0.20	55.23	✓
		20%	28,336	0.26	53.94	✓
		25%	35,392	0.33	52.95	✕
		30%	42,462	0.40	52.15	✕
		35%	49,542	0.46	51.47	✕
		40%	56,606	0.53	50.88	✕
		45%	63,701	0.60	50.36	✕
8	House 1	10%	3,675	0.14	56.68	✓
		15%	5,501	0.21	54.95	✓
		20%	7,331	0.28	53.72	✕
		25%	9,145	0.34	52.75	✕
		30%	10,933	0.41	51.98	✕
		35%	12,729	0.48	51.29	✕
		40%	14,518	0.55	50.72	✕
		45%	16,300	0.62	50.22	✕

NO	Name	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
9	Tree	10%	3,614	0.14	56.68	✓
		15%	5,400	0.21	54.99	✕
		20%	7187	0.28	53.73	✕
		25%	8,960	0.34	52.79	✕
		30%	10,736	0.41	52.02	✕
		35%	12,502	0.48	51.35	✕
		40%	14,289	0.54	50.78	✕
		45%	16,067	0.61	50.27	✕
10	MATLAB Logo	10%	58,104	0.15	56.44	✓
		15%	86,940	0.22	54.70	✓
		20%	116,099	0.29	53.44	✓
		25%	145,546	0.37	52.46	✓
		30%	174,352	0.44	51.70	✓
		35%	203,861	0.51	51.06	✓
		40%	232,956	0.58	50.50	✓
		45%	262,409	0.66	49.95	✓
		50%	290,986	0.73	49.49	✓
		55%	320,105	0.81	49.05	✕

For MSE, which represents the statistical difference between cover and stego images, as we see from Table 4.2, the MSE values of the stego images ranges from 0.13 as the smallest value to 0.66 as the largest value among all the resulting stego images. Since the MSE value which is too small or close to zero indicates that the difference is negligible, and the MSE values of our results satisfies that criterion, then this is an indication that the statistical difference between the cover and the stego images is too small, and thus the stego images are not perceptually detectable, which means high imperceptibility.

For PSNR, which represents the similarity between cover and stego images, the values ranges from 49.95 as the smallest value to 57.01 as the largest value among all the resulting stego images. Since PSNR values exceed 40 db, then the algorithm is considered very imperceptible.

So as we see from the results, on average MSE ranges from 0.14 to 0.62 and PSNR ranges from 56.75 to 50.25 when the hidden data fills from 10% to 45% of the cover image. Which means, both metrics MSE and PSNR indicate that the algorithm works with high imperceptibility. Finally, Figure 4.1 and Figure 4.2 show the charts of how on average MSE increase and PSNR decrease for the resulting stego images as the hidden data increase.

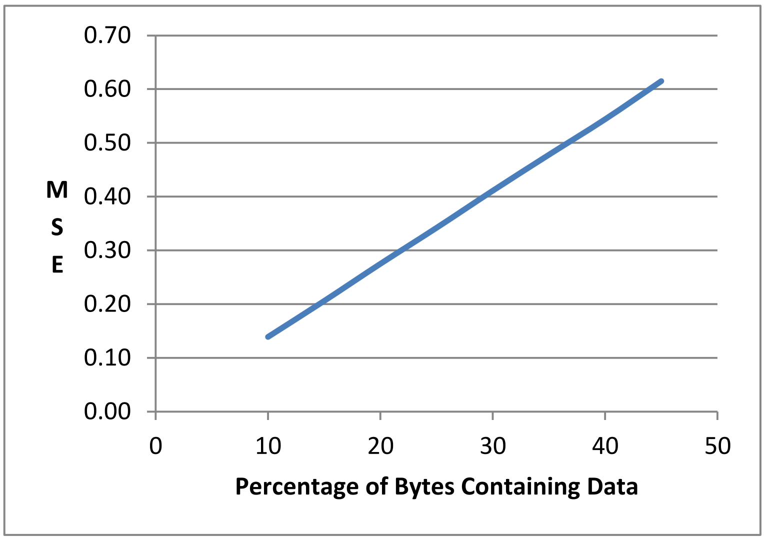


Figure (4.1): MSE Values Chart

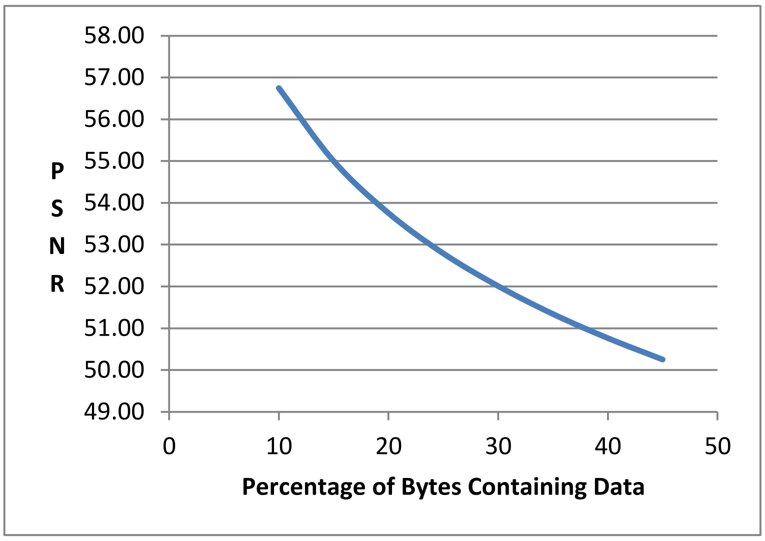


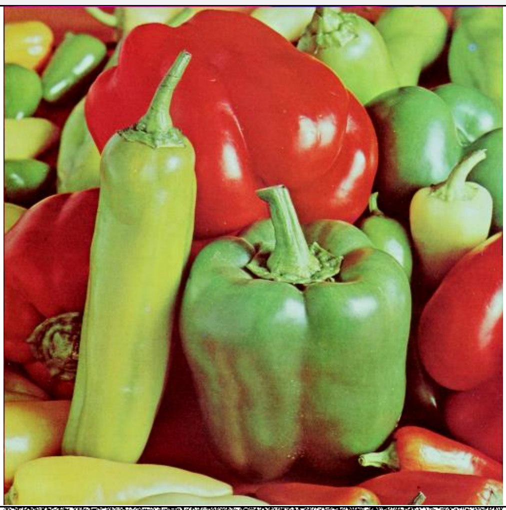
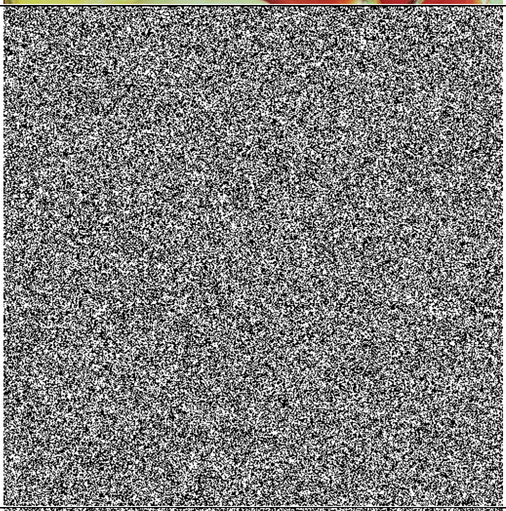
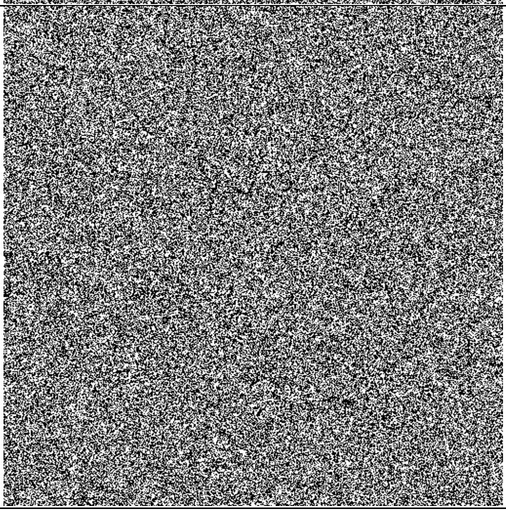
Figure (4.2): PSNR Values Chart

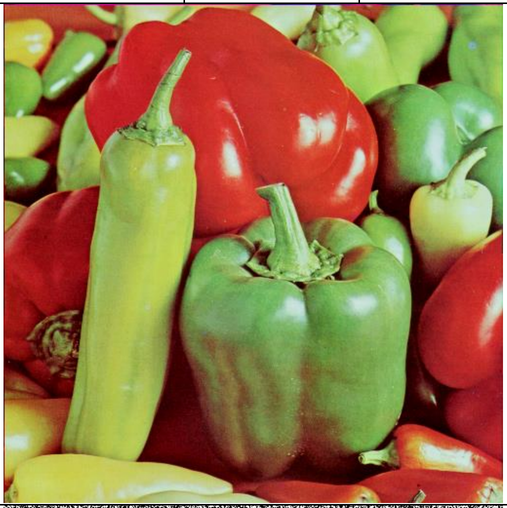
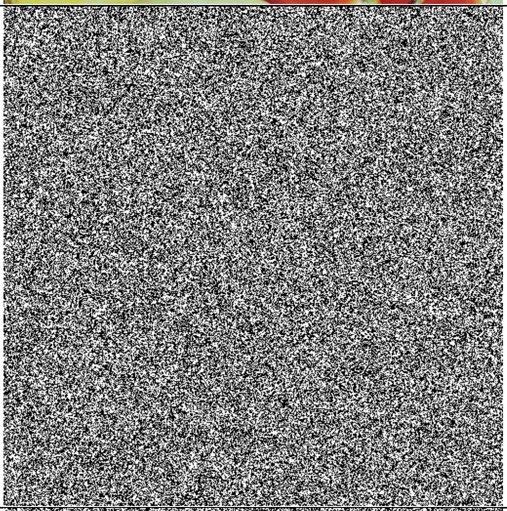
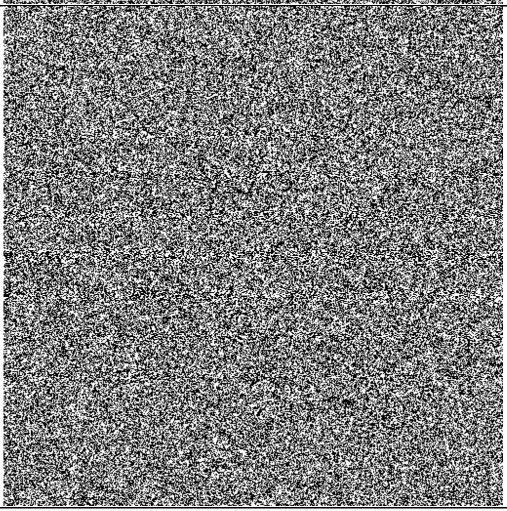
Furthermore, all the stego images were subjected to a steganalysis tool. As shown in Table 4.2, for images of dimensions of 512×512, most of the images were detected when the embedded data has filled more than 30% of the cover bytes. However, it cannot be considered as a rule that filling less than 30% of an image makes it undetectable, since some images are detected with much less percentage of hidden data, as happened to image 8, which was detected when the embedded data has exceeded 20% of the cover bytes. We have made tests over many images with different characteristics, like dimensions and color diversity, but no general rule of detectability threshold was found. Through experiments we noticed that the factor controlling how detectable a stego image is, is the structure of the LSBs values of the cover image itself. Since images have general pattern for their statistical characteristics, as PoVs of LSBs and histograms (Bateman & Schaathun, 2008), then, most of statistical attacks algorithms depend on these characteristics patterns to decide whether an image is suspicious or not. For example, PoVs characteristic that occurs as a result of bit-flipping through secret data embedding is used by Chi-square Test for detecting stego images. So, when attacks find that the characteristics of the image in question are out of the general pattern, then the image is marked as suspicious. So, through data embedding, we must take care when selecting the suitable image. Indeed, changing many LSBs changes the statistical characteristics of the image and makes them remarkably out of the general pattern that these characteristics belong to. As a result, the resulting images would be considered having signs of data hiding, which increases the likelihood of detectability. Thus, hidden data should not be of relatively big amount compared to the cover image size to avoid leaving signs of manipulation as possible. Our algorithm embeds data sometimes into the LSB and sometimes into Least two Significant Bits at once. By experiments, embedding data constantly into the Least Two Significant Bits at once is much less detectable than embedding inside only the LSB. Also embedding into only the 2nd LSB is less detectable than into Least Two Significant Bits at once.

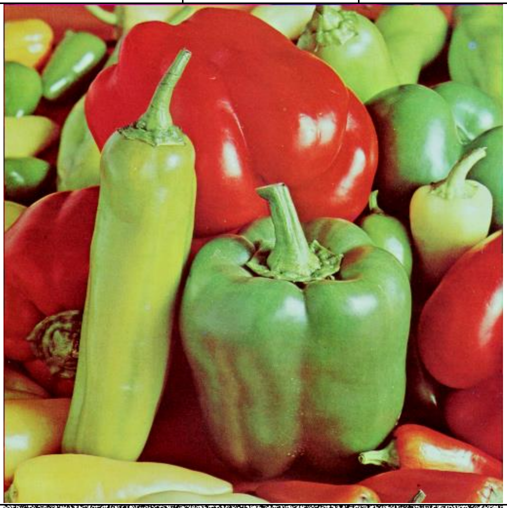
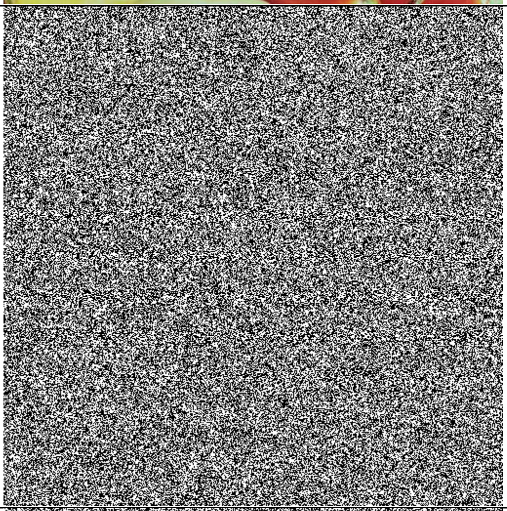
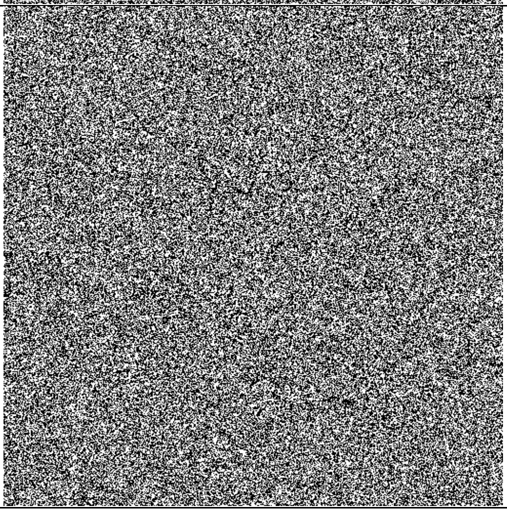
4.2.2 Visual Attack Test of LSB and 2nd LSB Planes

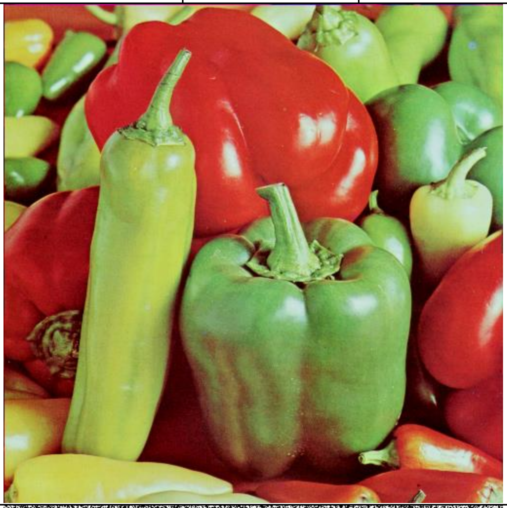
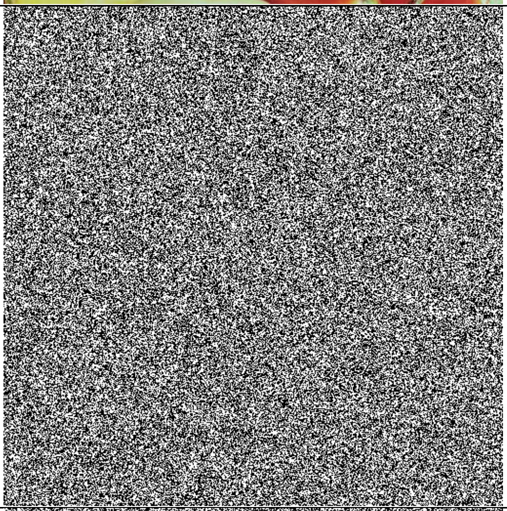
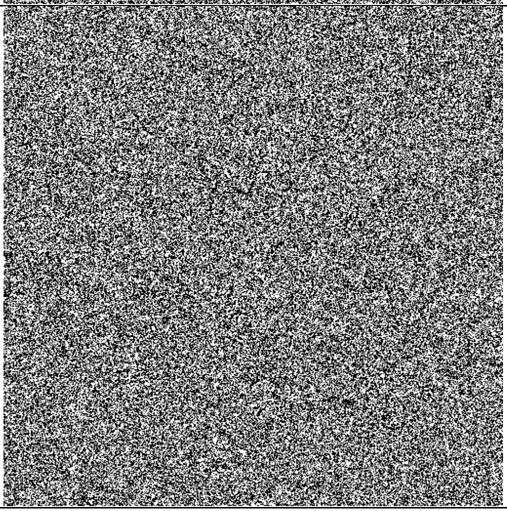
As shown before in section 2.11.1.1.1, altering LSB values sequentially causes inconsistency in the LSB plane. So, the part of LSB plane that has hidden data is visually different from the clean part. Steganalysts performing visual attacks search for signs of data hiding in the LSB plane by searching for such a difference. By hiding secret data randomly, as Indicators-based algorithm does, we can avoid causing any inconsistency in LSB plane. As a sample, Table 4.3 shows the LSB plane and the 2nd LSB plane of the resulting stego images of cover image 6, which is called Peppers.

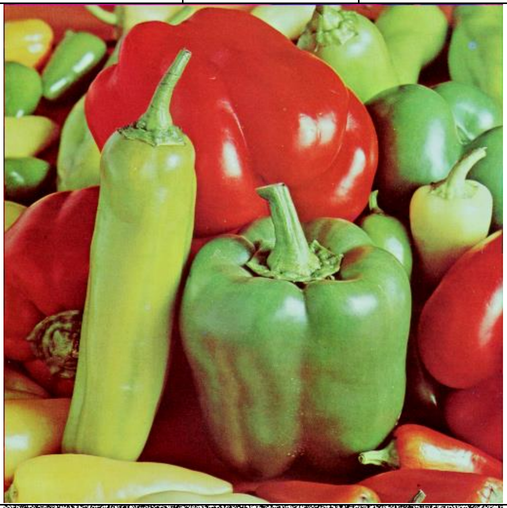
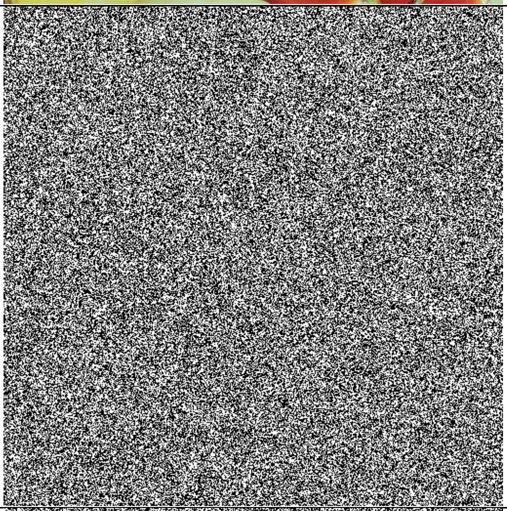
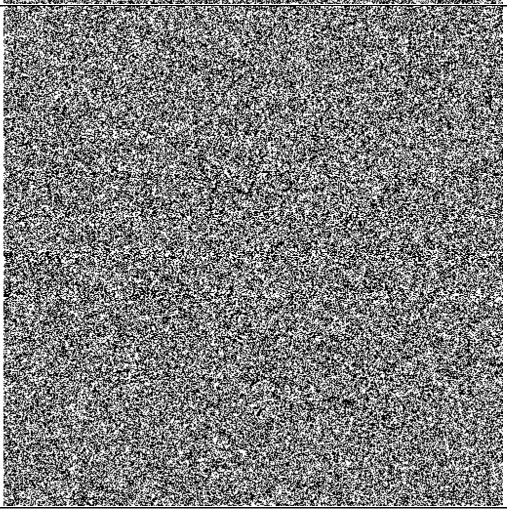
Table (4.3): Visual Attack Results for LSB and 2nd LSB Planes

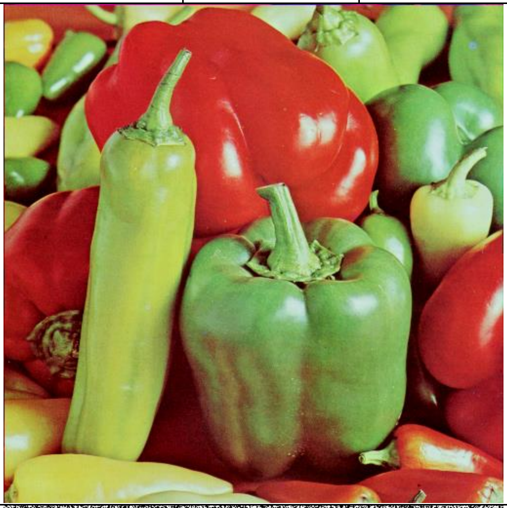
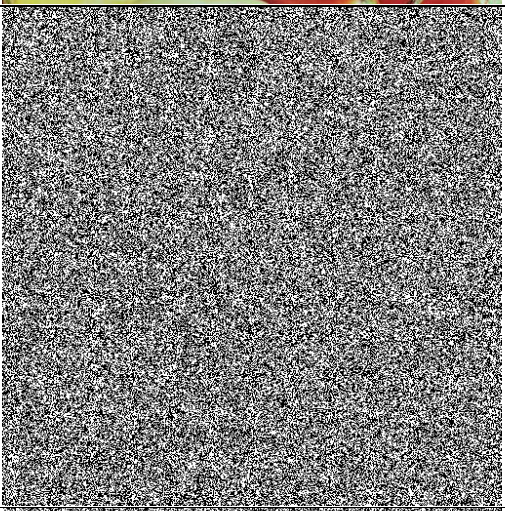
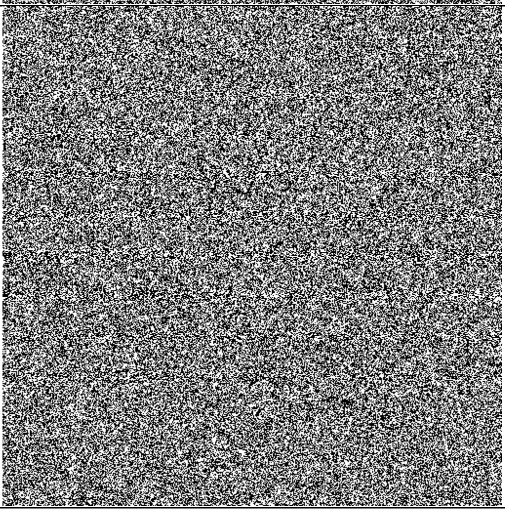
Original Cover Image without Embedded Data	
Cover Image	
LSB Plane	
2nd LSB Plane	

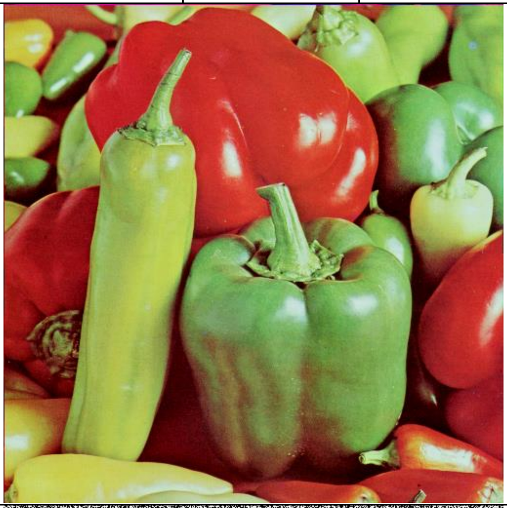
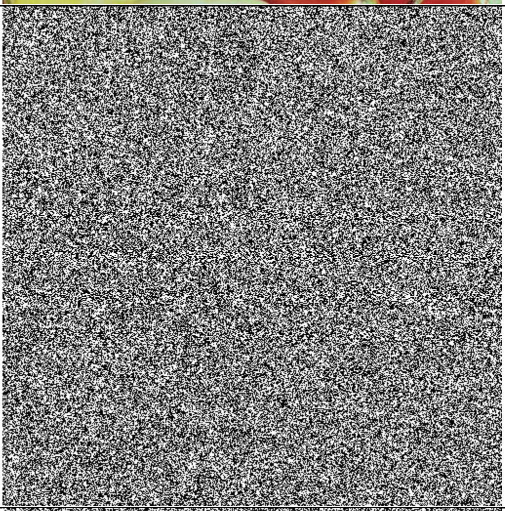
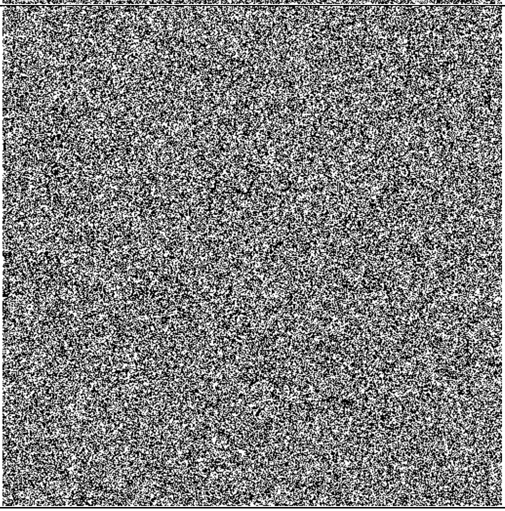
Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	10%	14,160	0.14	56.82	✓
Stego Image					
LSB Plane					
2nd LSB Plane					

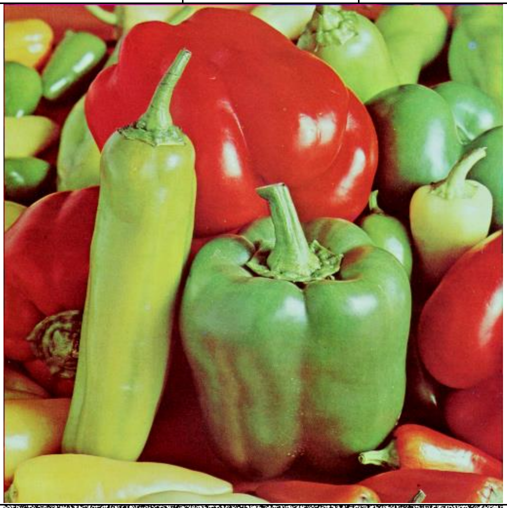
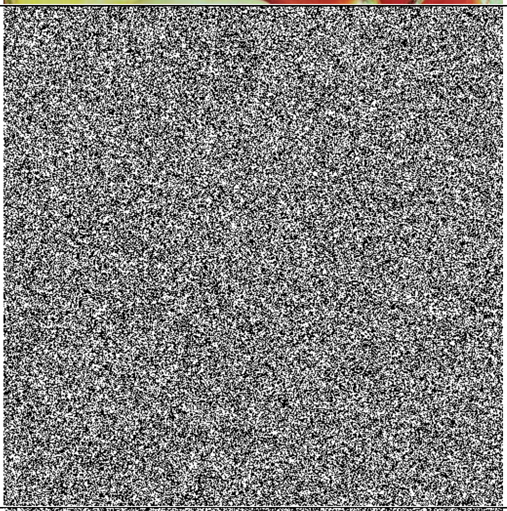
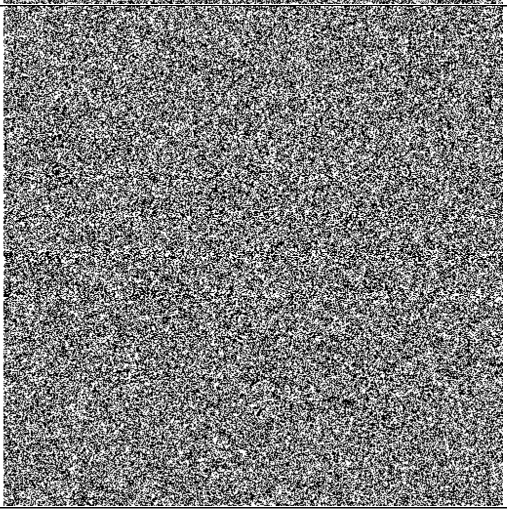
Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	15%	21,232	0.20	55.06	✓
Stego Image					
LSB Plane					
2nd LSB Plane					

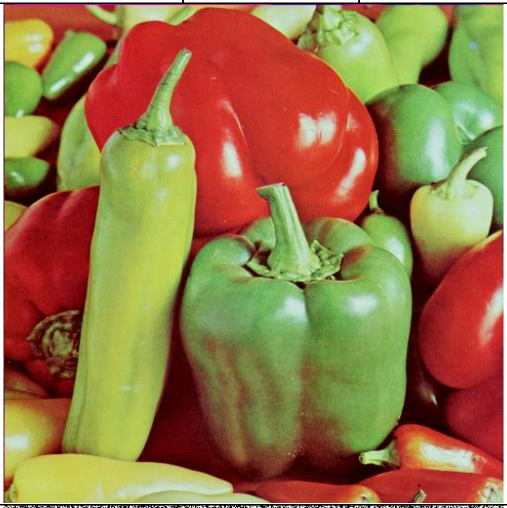
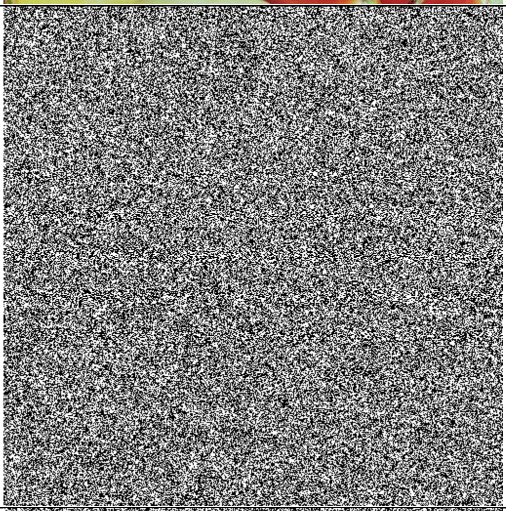
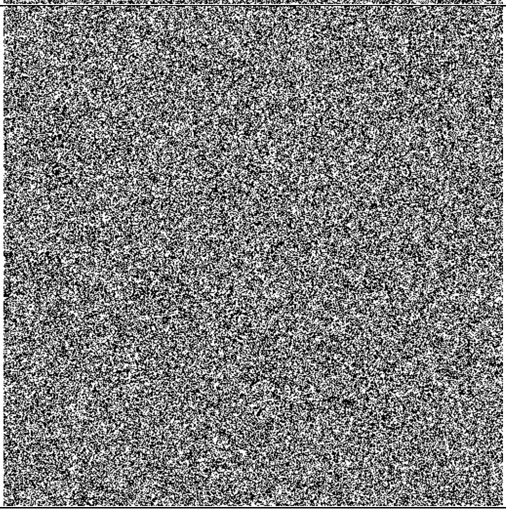
Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	20%	28,287	0.27	53.82	✓
Stego Image					
LSB Plane					
2nd LSB Plane					

Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	25%	35,335	0.34	52.85	✓
Stego Image					
LSB Plane					
2nd LSB Plane					

Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	30%	42,394	0.41	52.05	✕
Stego Image					
LSB Plane					
2^{nd} LSB Plane					

Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	35%	49,455	0.47	51.39	✕
Stego Image					
LSB Plane					
2nd LSB Plane					

Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	40%	56,494	0.54	50.80	✕
Stego Image					
LSB Plane					
2nd LSB Plane					

Stego Image Properties	Percentage of Bytes Containing Data	Hidden Data Size (Byte)	MSE	PSNR	Undetected
	45%	63,531	0.61	50.29	✕
Stego Image					
LSB Plane					
2nd LSB Plane					

As shown it Table 4.3, it is clear that due to hiding data randomly, there is no inconsistency in the LSBs planes no matter how much the hidden data is. Therefore, performing visual attacks would not work.

4.3 Summary

In this chapter we have presented several types of experiments conducted to test and evaluate the efficiency of the proposed algorithm. Also we have shown by results that the algorithm works with good efficiency at data hiding.

Chapter 5
Conclusion, Recommendations and Future Work

Here in this chapter, the conclusion is given. We also give some recommendations for using steganography the best way and reducing the risk of failure as possible. Finally, we talk about some future work and research.

5.1 Conclusion

Steganography techniques and algorithms are developed to improve data hiding process aspects which are imperceptibility, capacity and robustness. This thesis introduces a novel algorithm based on LSB substitution to address and improve the robustness and capacity while keeping high imperceptibility. The main contribution of our work is the new way of data hiding that apply randomization based on indicators.

Imperceptibility of the algorithm is measured by measuring the MSE and PSNR of the resulting stego images, and all of the values were very high.

Robustness is enhanced by embedding secret data randomly depending on indicators. Most LSB-based algorithms add randomness to the hiding process to increase the robustness, where randomization makes it harder to detect and extract hidden data.

Capacity is increased by embedding sometimes secret data into 2nd LSB beside the LSB. Decision of embedding data into 2nd LSB is made depending on an indicator. On average the algorithm increases the capacity over embedding into only the LSB by 50%.

5.2 Recommendations

Steganographic systems cannot be absolutely secure, so it is important to take care when hiding secret data. First, the cover image should be of suitable size to contain the secret data, such that the data dose not fill on average more than 20% of a cover image. Second, it is preferred that the cover image is of medium dimensions or larger such as 512 × 512. Third, when intend to embed data, the cover image should be new and not be available. Fourth, the data must be minimized as possible, for example, if the secret data is of text type, then spaces must be removed, where the ASCII of the space is 32 which is 0010 0000 in binary. As we see the ASCII of the space consists of seven zeros and a one. Since each word of a text is followed by a space, then there would be a lot of spaces to hide, and that means for each space, seven zeros are hidden to a one, which would increase zeros in LSBs for the ones. We can remove the spaces and each word is capitalized to separate words of the text. Fifth, secret data could be encrypted before embedding, to increase the protection and to change ASCII of the text characters.

5.3 Future Work

In some researches it has been claimed that embedding data into Least Two Significant Bits is less detectable than into only the Least Significant Bit. Also we noticed that embedding into only the 2^{nd} LSB is less detectable than into only the LSB or into the Least Two Significant Bits at once. So, we intend to do experiments on huge dataset of images to figure out characteristics of using this approach. Also we intend to do more researches to find out how statistical attacks work through detecting stego images to improve the mechanism of data hiding.

References

Akhtar, N., Johri, P., & Khan, S. (2013). *Enhancing the security and quality of LSB based image steganography.* Paper presented at the Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on.

Al-Ani, Z. K., Zaidan, A., Zaidan, B., & Alanazi, H. (2010). Overview: Main fundamentals for steganography. *arXiv preprint arXiv:1003.4086*.

Al-Mohammad, A. (2010). *Steganography-based secret and reliable communications: Improving steganographic capacity and imperceptibility.* Brunel University, School of Information Systems, Computing and Mathematics Theses.

Asad, M., Gilani, J., & Khalid, A. (2011). *An enhanced least significant bit modification technique for audio steganography.* Paper presented at the Computer Networks and Information Technology (ICCNIT), 2011 International Conference on.

Balaji, R., & Naveen, G. (2011). *Secure data transmission using video Steganography.* Paper presented at the Electro/Information Technology (EIT), 2011 IEEE International Conference on.

Bateman, P., & Schaathun, H. G. (2008). Image steganography and steganalysis. *Department Of Computing, Faculty of Engineering and Physical Sciences, University of Surrey, Guildford, Surrey, United Kingdom, 4th August*.

Chandramouli, R., Kharrazi, M., & Memon, N. (2003). *Image steganography and steganalysis: Concepts and practice.* Paper presented at the International Workshop on Digital Watermarking.

Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal processing, 90*(3), 727-752.

Dumitrescu, S., Wu, X., & Memon, N. (2002). *On steganalysis of random LSB embedding in continuous-tone images.* Paper presented at the Image Processing. 2002. Proceedings. 2002 International Conference on.

Dumitrescu, S., Wu, X., & Wang, Z. (2003). Detection of LSB steganography via sample pair analysis. *IEEE transactions on Signal Processing, 51*(7), 1995-2007.

Dunbar, B. (2002). A detailed look at Steganographic Techniques and their use in an Open-Systems Environment. *Sans Institute, 2002*, 1-9.

Easttom II, W. C. (2016). *Computer security fundamentals*: Pearson IT Certification.

Eric, C. (2003). Hiding in plain sight, Stegnography and the art of Covert Communication. *Wiley, Indianapolis, Indiana, ISBN, 10*, 0471444499.

Fridrich, J., Goljan, M., & Du, R. (2001). *Reliable detection of LSB steganography in color and grayscale images.* Paper presented at the Proceedings of the 2001 workshop on Multimedia and security: new challenges.

Goel, P. (2008). *Data Hiding in Digital Images: A Steganographic Paradigm.* Indian Institute of Technology–Kharagpur.

Gowda, S. N., & Sulakhe, S. (2016). *Block Based Least Significant Bit Algorithm For Image Steganography*.

Holub, V. (2014). *Content Adaptive Steganography–Design and Detection.* Citeseer.

Islam, M. R., Siddiqa, A., Uddin, M. P., Mandal, A. K., & Hossain, M. D. (2014). *An efficient filtering based approach improving LSB image steganography using status bit along with AES cryptography.* Paper presented at the Informatics, Electronics & Vision (ICIEV), 2014 International Conference on.

Jain, R., & Boaddh, J. (2016). *Advances in digital image steganography.* Paper presented at the Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016 International Conference on.

Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer, 31*(2), 26-34.

Juneja, M., & Sandhu, P. S. (2013). Data Hiding with Enhanced LSB Steganography and Cryptography for RGB Color Images. *International Journal of Applied Research, 3*(5), 118-120.

Karim, S. M., Rahman, M. S., & Hossain, M. I. (2011). *A new approach for LSB based image steganography using secret key.* Paper presented at the Computer and Information Technology (ICCIT), 2011 14th International Conference on.

Ker, A. D. (2007). Steganalysis of embedding in two least-significant bits. *IEEE Transactions on Information Forensics and Security, 2*(1), 46-54.

Kipper, G. (2003). *Investigator's guide to steganography*: crc press.

Krenn, R. (2004). Steganography and steganalysis. *Retrieved September, 8*, 2007.

Laha, S., & Roy, R. (2015). *An improved image steganography scheme with high visual image quality.* Paper presented at the Computing, Communication and Security (ICCCS), 2015 International Conference on.

Lubacz, J., Mazurczyk, W., & Szczypiorski, K. (2012). Principles and overview of network steganography. *arXiv preprint arXiv:1207.0917*.

Mahmood, N. R., Azeez, A. A., & Rasool, Z. N. (2014). Public Key Steganography. *International Journal of Computer Applications, 100*(8).

Morkel, T., Eloff, J. H., & Olivier, M. S. (2005). *An overview of image steganography.* Paper presented at the ISSA.

Neeta, D., Snehal, K., & Jacobs, D. (2006). *Implementation of LSB steganography and its evaluation for various bits.* Paper presented at the Digital Information Management, 2006 1st International Conference on.

Nguyen, T. D., Arch-Int, S., & Arch-Int, N. (2016). An adaptive multi bit-plane image steganography using block data-hiding. *Multimedia Tools and Applications, 75*(14), 8319-8345.

Saidi, M., Hermassi, H., Rhouma, R., & Belghith, S. (2016). A new adaptive image steganography scheme based on DCT and chaotic map. *Multimedia Tools and Applications*, 1-18.

Satar, S. D. M., Hamid, N. A., Ghazali, F., Muda, R., Mamat, M., & An, P. K. Secure Image Steganography Using Encryption Algorithm.

Sharif, A., Mollaeefar, M., & Nazari, M. (2016). A novel method for digital image steganography based on a new three-dimensional chaotic map. *Multimedia Tools and Applications*, 1-19.

Silman, J. (2001). Steganography and steganalysis: an overview. *Sans Institute, 3*, 61-76.

Simmons, G. J. (1984). *The prisoners' problem and the subliminal channel.* Paper presented at the Advances in Cryptology.

Singh, S., & Kaur, J. (2015). Odd-Even Message Bit Sequence Based Image Steganography. *International Journal of Computer Science and Information Technologies, 6*(4).

Stanley, C. A. (2005). Pairs of Values and the Chi-squared Attack. *Department of Mathematics, Iowa State University*.

Sumathi, C., Santanam, T., & Umamaheswari, G. (2014). A Study of Various Steganographic Techniques Used for Information Hiding. *arXiv preprint arXiv:1401.5561, 4*(6).

Tayel, M., Gamal, A., & Shawky, H. (2016). *A proposed implementation method of an audio steganography technique.* Paper presented at the Advanced Communication Technology (ICACT), 2016 18th International Conference on.

Thangadurai, K., & Sudha Devi, G. (2014). *An analysis of LSB based image steganography techniques.* Paper presented at the Computer Communication and Informatics (ICCCI), 2014 International Conference on.

Wang, H., & Wang, S. (2004). Cyber warfare: steganography vs. steganalysis. *Communications of the ACM, 47*(10), 76-82.

Watkins, J. (2001). Steganography-Messages Hidden in Bits. *Multimedia Systems Coursework, Dept of Electronics and CS, University of Southampton, SO17 1BJ, UK*.

Westfeld, A., & Pfitzmann, A. (1999). *Attacks on steganographic systems.* Paper presented at the International workshop on information hiding.

Yadav, P., Mishra, N., & Sharma, S. (2013). *A secure video steganography with encryption based on LSB technique.* Paper presented at the Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on.

Zhang, T., & Ping, X. (2003). *Reliable detection of LSB steganography based on the difference image histogram.* Paper presented at the Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on.

Boehm, B. (2014). StegExpose - A Tool for Detecting LSB Steganography, University of Kent, School of Computing, https://github.com/b3dk7/StegExpose, 6/2/2017

The USC-SIPI image database, http://sipi.usc.edu/database/database.php, 18/4/2017