IMPROVING DATA HIDING CAPACITY IN CODE BASED STEGANOGRAPHY USING MULTIPLE EMBEDDING

A THESIS SUBMITTED TO
THE SCHOOL OF COMPUTING

BY

KATANDAWA KINGSLEY A

2301171012

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF INFORMATICS
IN
THE SCHOOL OF COMPUTING

TELKOM UNIVERSITY
2019

APPROVAL PAGE

Approval of the School of Computing of Telkom University
I certify that this thesis satisfies all the requirements as a thesis for the degree of Master Informatics.
Date Jul 10 , 2019 (*the date can be set manully)

(Dana Sulistyo Kusumo, Ph.D.)
Head of Master Informatics
This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Informatics.

Date Jul 10 , 2019

(Supervisor's name)
Ari Moesriami Barmawi

(Co-Supervisor's name)
Co-Supervisor

Examining Committee Members.
Date Jul 10 , 2019

(Jury's name) (Chairperson of the jury) :

(Jury's name) (jurys member) :

(Jury's name) (jurys member) :

SELF DECLARATION AGAINST PLAGIARISM

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

Month/Date/Year Jul 10 , 2019
Katandawa Kingsley A

Signature: _____

Month/Date/Year Jul 10 , 2019
A. M Barmawi

Signature: _____

Month/Date/Year Jul 10 , 2019
Name, last name of the Co-Supervisor: Co-Supervisor Name

Signature: _____

ABSTRACT

In this modern era there is a rapid increase in use of internet to exchange sensitive information. However, communication via the internet is unsecure and unrealible. Due to these factors, data hiding techniques has been proposed to increase the confidentiality and security of sensitive information. Moreover, Crandall [4] introduced Code Based Steganography, which merges steganography with coding theory. It implemented matrix encoding using linear codes to increase the visual quality of stego image by preserving high embedding capacity. A. M. Molaei et al [12] proposed a steganography scheme which implemented Reed Muller codes and modulus function in attempt to increase the embedding capacity. These are fault tolerant schemes have ability to recover secret messages from attacks using error detection and correction. However, the existing schemes have low embedding capacity (150%) and low PSNR value (48dB). To overcome this problem, this paper proposed the multiple embedding method that aims to re-embed secrets bits on the same LSBs of the selected pixels based on a stego key. The experiments results shows that the proposed method achieved higher embedding capacity (450%) three times more than A. M. Molaei's method. The proposed method also obtained a higher PSNR value of 51dB and a higher error correction capability.

Keywords: Data hiding, Ruller Muller codes, Secret Sharing, Re-embedding, Embedding Capacity

ABSTRAK

Abstract in bahasa Indonesia.

Kata kunci:

DEDICATION

This thesis is compiled with the support of my family. I praise GOD and thanks to all who have helped, directed, and supported this work. I dedicate the work to my beloved parents ALICE and BOB.

Finally this thesis is dedicated to all informatics students in the world. I hope that this research may provide valuable contribution in Computer Science.

ACKNOWLEDGMENTS

This thesis is compiled with the effort, help, and support from both students and lecturers. I would like to express my deepest gratitude and thanks to:

1. .....

2. .....

PREFACE

In thesis writing, the most difficult part to write is Chapter 1 (Introduction/The Problem). As they say, the most difficult part of any endeavor is the starting point. This is because the first chapter is where you conceptualize your entire research. The whole research/thesis can be reflected in Chapter 1 including expected results or outcomes. For your guidelines, please read the following sample format of Chapter 1.

Bandung,

YOUR NAME

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF TERMS

Terms	Definition
Classes	Number of individual in biometrics data
Sample	Number of images can be used to represent population in a class.
...	...

LIST OF NOTATIONS

Symbols	Definition
$\mapsto$	Mapping operator
(x_i, y_i)	Data point
$\|A\|$	Cardinality of A
$\|p - q\|$	Absolute of $p - q$
$\mathbb{R}$	Sets of real number

CHAPTER 1

INTRODUCTION

1.1 Rationale

In this modern era there is a rapid increase in use of internet to exchange data. However, the internet over which the exchange of sensitive data is done is unsecure and unrealible. Due to these factors, the confidentiality and security of the sensitive information has emerged as a major concern and top priority. Data hiding is therefore a method of facilitating covert communication in the form of concealing information in a host media called a cover and then being able to extract the message from the cover [15]. The secret data that are applicable include text, images, videos and audio and so as is the cover media. There are two main types of data hiding techniques used for protecting sensitive data from unauthorised access and or tampering these are Watermarking and Steganography [5].

In Watermarking, the hidden or embedded data has a relation with the host or cover media. In Steganography the concealed secret data (e.g text, image etc) and the cover media (e.g text, image, audio, video etc) have no relation, the secret data hidden in stego image is required to be undetectable. Coding Theory is responsible for recovering messages sent via a noisy channel. It extends the message by adding redudant bits which enables error detection and correction [17]. Currently in Steganography, Coding theory serves a major part. Its main objective is to recover hidden messages from the attacked stego images.

A good steganography scheme has three characteristics: high embedding capacity, high embedding efficiency and security [19]. The embedding capacity describes the size of hidden secret message that can be transmitted. Embedding efficiency refers to the visual quality of the scheme and less image distortions are more secure because it does not raise any suspicion to adversaries. Highly secure and robust steganographic schemes have ability to resist against attacks. Current code based steganography scheme proposed by A. M. Molaei [12] have low embedding capacity (150%) and low embedding efficiency of PSNR value of 48dB. This is due to the adopted LSB embedding method of higher order significant bits of the cove pixels. The proposed encoder with code rate $\frac{1}{2}$ extended the secret data by 2. The number of cover pixels to be embedded is increased by 2 also and because of that, the image quality reduces significantly. Therefore an introduction of data hiding techniques that improves the embedding capacity at the same time assuring invisibility of embedded data are preferable.

This research proposes a high capacity data hiding mechanism for gray scale images that maintains the invisibility of embedded data. It is based on multiple embedding of secret data on few randomly selected pixels to solve the low embedding capacity problems. The proposed method randomly selects a maximum of $\frac{3}{4}$ of the cover image pixels for multiple embedding to guarantee a highly imperceptible stego image. The secret data is encoded using Reed Muller error correction codes before embedding process to increase the robustness of the scheme.

For random selection of cover pixels to be embedded, Blum Blum shub, a cryptographically secure pseudo-random number generator (CSPRNG) [7] was adopted. The seed used in CSPRNG and the number of embedding cycles was communicated between sender and receiver using a (2,2) secret sharing scheme [3]. This increases the security of the hidden data. The senders' secret share is encoded with Reed Muller codes before embedding on the stego image to increase its resistance against attacks. Furthermore, the Modulus function proposed in [16] and LSB embedding were adopted during embedding process to ensure good visual quality stego images. Experiment results shows that it also benefits the scheme by preventing burst errors which are common in existing methods during secret message recovery in destroyed stego images. The proposed method provides high PSNR, high embedding capacity (450%) and superior error correction capabilities. The error correction capability of the proposed scheme increases as the size of the secret image increases.

1.2 Theoretical Framework

The proposed method conceals the secret message efficiently and securely by re embedding it on few randomly selected pixel positions of the cover image to avoid raising suspicion. The implementation of the proposed method is described as follows:

Suppose there are two parties Bob and Alice who agree to communicate a secret image using the rules of the proposed method. Bob is represented as the secret message Sender and Alice is represented as the Recipient of the secret message. Both Bob and Alice would determine two images from the database on the cloud to be used as cover and the other as an innocent image.
Bob would encode the secret message using a linear error correction code to enable secret image recovery from attacked stego. He would generate a sequence of random integers using random number generator. The random integers in the sequence represented the pixels positions to be embedded on the cover. Bob generates a secret key which is a binary image that consist of the seed which initialised the random number generator, the number of cycles on each re embedded pixels and the dimensions of the embedded secret image.

To increase the security of the secret message, Bob perform visual cryptography on the generated secret key. Visual cryptography generates two scrambled shares that do not reveal the contents of the secret key. He randomly embeds the encoded secret message and one secret share on the randomly generated pixels of the cover using multiple embedding method and the agreed upon innocent image. Multiple embedding method allowed more than 1 secret message bits to be embedded on the LSB of the cover pixels. A stego image and a stego agreed image are produced after the embedding process.

Bob establishes a secure channel with Alice to communicate securely the other generated share. However, Bob uses a public channel ie via internet to transmit the stego image and a stego agreed image. Meanwhile, Alice would download the stego image and a stego agreed image. To retrieve the hidden secret image, Alice would need to extract the hidden share and overlap it with the other share that she already possesses. If Alice is not an adversary the secret key is reconstructed from the overlapping of the two shares. Alice retrieves visually the contents of the secret key and uses seed to regenerate the embedded pixel positions and the number of cycles utilized in multiple embedding. With that information, Alice was able to retrieve the encoded secret data from the stego image using stego agreed image. The retrieved data is decoded using the same error correction code to recover the secret data that had been attacked or manipulated. Finally, Alice would possess the hidden secret image

1.3 Conceptual Framework/Paradigm

The basic concept of the proposed method is to improve the embedding capacity in code based steganography using multiple embedding method combined with cryptography techniques. The idea of multiple embedding is to embed more than one secret data on a single pixel's LSB. This was made possible performing XOR on the secret data to be ebedded on each pixel. Multiple embedding also increased the robustness of code based steganography.

In the embedding process of this research, a high performance error correction code called Reed Muller(1, 4) is used to encode the secret message. The error correction code was introduced to increase the robustness of the hidden secret message. The linear code consist of 18,25% error correction capability. After the encoding process, a sequence of random integers were generated using a pseudo random number generator(PRNG) called Blum Blum shub. The introduced PRNG was introduced because of its increased security and period. The sequence of integers represented the pixel positions on the cover image that would be embedded with the secret data. Furthermore, an efficient method of increasing the embedding capacity called Multiple embedding was implemented to form a stego image. The seed of the PRNG and the number of cycles of the multiple embedding was used

to generate a binary image called a secret key. The binary image was communicated between the sender and the recipient using Visual Cryptography techniques. Upon receiving the Stego image, the recipient reconstructs the secret key to retrieve the seed and number of cycles. The recipient retrieves the embedded secret data by regenerating the sequence of integers and extract the LSBs. The original secret data is obtained by decoding the extracted LSBs.

1.4 Statement of the Problem

A. M. Molaei's Method had two major weaknesses: Low embedding capacity(%) and Low visual quality (PSNR) of the stego. The low embedding capacity was caused by restricting embedding of secret data to first and second LSBs of the cover pixels only. The visual quality of the stego is low because of all the pixels of the cover were embedded. Furthermore, the embedding of second LSBs causes an increase in the difference between the original image and the resulting stego thereby decreasing the overal quality of the stego. A. M. Molaei's Method also suggested error correction codes with low error correction capability which resulted in low robustness of the scheme.

1.5 Objective

The objective of the research is to improve the data hiding capacity of a code based steganography scheme while maintaining a good visual quality of the stego image.

1.6 Hypotheses

In this research we introduce a code based steganography scheme that uses multiple embedding and secret sharing to achieve a high embedding capacity. Linear Block Codes were used to increase the robustness of the stego image against noise. LSB embedding using modulus function was implemented to embed the secret data efficiently and to ensure good visual quality of stego is maintained. The implementation of the proposed method is by re embedding the secret data on the same pixel of the cover image. To achieve this a key is required to hold the Number of cycles per pixel used in re embedding and the seed used to randomly select pixels in embedding process. The stego key was distributed between both parties using visual cryptography to increase the security of the hidden secret data. So, if the extra secret image bits are re embedded 6 times on the $\frac{3}{4}$ pixels of the cover image then the hiding capacity is increased by 3.

1.7 Assumption

There is one assumption in this thesis.The sender and receiver agreed on one single image before exchanging the secret message and the resulting agreed image after embedding process is exchanged via a secure channel.

1.8 Scope and Delimitation

The parameters of the error correction codes ie r and m should be sent to the related party(receiver). The parameters r and m represent the order of the Reed Muller code and m is a positive integer such that $0 \leq r \leq m$ [10]. A good choice of m and r ensures a high performance error correction code.

1.9 Significance of the Study

This research can be used in various areas and fields such as in intellectual property applications [6], diplomatic and personal. It can also be used in military and intelligency organizations to communicate classified files. Moreover the research can be used for medical purposes to secure the information of a patient such as X-Rays and computed tomography(CT-Scan) [2]

CHAPTER 2

REVIEW OF LITERATURE AND STUDIES

This chapter discusses the detailed description of concepts and theories useful in code based steganography. The concepts such as Reed Muller codes and Modulus function were implemented in both A. M. Molaei and proposed method. While the concepts such as Visual cryptography and Blum Blum Shub were implemented in proposed method only. After description of concepts and theories, A. M. Molaei's Method is discussed in terms of the embedding and extraction process.

2.1 Reed Muller Codes

In this section the reed Muller error correction codes are discussed. In Coding theory, error correction codes is a function that expresses numbers in a sequence such that error detection and correction is performed based on the remaining numbers after introduction of errors [18].There are two categories of Error correction codes which are Block and Convolution codes. Block codes act on a message block of k bits to generate N fixed output data bits, where $N \geq k$.

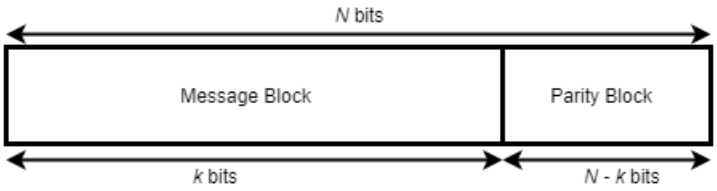


Figure 2.1: Embedding process

In this section a linear block code called Reed Muller codes are explained. It is employed because of their simplicity in encoding messages and decoding received data [10]. Given m and r be the two positive integers and $0 \leq$ r $\leq$ m. An r^th-order Reed-Muller code, RM(r, m) has a message size, k, a code length, N and minimum distance, d_{min} , where

$$k = \sum_{i=0}^{r} \binom{m}{i} \tag{2.1}$$

$$N = 2^m \tag{2.2}$$

$$d_{min} = 2^{m-r} \tag{2.3}$$

and the notation is written as:

$$\left[2^m, \quad k, \quad 2^{m-r}\right] - code$$

The Reed muller code can correct a maximum number of errors, t given by

$$t = \left\lfloor \frac{2^{m-r} - 1}{2} \right\rfloor \tag{2.4}$$

The choice of m and r values determines the message size k of the Encoder, the block size N, minimum distance of the code and these have impact on the error correction capability of the code.

For example, given m = 4 and r = 1, then k = 5, N = 16, $d_{min} = 8$ and t = 3.

2.1.1 Construction of Reed Muller Codes

There are many ways that can be used to construct Reed Muller codes. In study we define Reed Muller codes recursively. To define Reed Muller codes inductively, let the 0^{th} order, RM(0, m) defined by two vectors 0,1 over Galois field, F_2^N that are repetition codes [13]. For $0 \leq$ r $\leq$ m, the r^{th} order Reed-Muller code R(r,m) is defined recursively by

$$RM(r, m) = (u, u + v) : u \in RM(r, m - 1), v \in RM(r - 1, m - 1) \tag{2.5}$$

where u and v represent the code words of the previously constructed codes .For example, if r = 1 and m = 2 ie RM(1,2), the (u, u+v) construction results in:
u $\in$ RM(1,1) and v $\in$ RM(0,1). Note that RM(1,1) = 00 01 10 11 and $u_0 = \mathbf{00}$, $u_1 = \mathbf{01}$, $u_2 = \mathbf{10}$, $u_3 = \mathbf{11}$. Also note that RM(0,1) = 00 11 and $v_0 = 00$, $v_1 = 11$. Therefore the resulting code of RM(1,2) is:

$$RM(1, 2) = \left\{ \begin{array}{cccc} \mathbf{00}00 & \mathbf{01}00 & \mathbf{10}00 & \mathbf{11}00 \\ \mathbf{00}11 & \mathbf{01}11 & \mathbf{10}11 & \mathbf{11}11 \end{array} \right\}$$

2.1.2 Encoding Reed Muller Codes

This section describes how a binary message is encoded using Reed Muller Encoder. Firstly, determine the dimensions of the Generator matrix,$G_{r,m}$ of an r^{th}-order Reed Muller code, RM(r, m) with block length of 2^m using equation(5). Then Generator matrix of RM(r, m) is generated and expressed as follows [13]:

$$G_{r,m} = \begin{bmatrix} G_{r,m-1} & G_{r,m-1} \\ \mathbf{0} & G_{r-1,m-1} \end{bmatrix} \tag{2.6}$$

where $G_{0,m}$ is a vector of length 2^m consisting of all ones and $G_{m,m}$ is an identity matrix, $\mathbf{I}_{2^m}$.
To encode a message, $u \in F_2^k$, a binary multiplication of the message and the Generator matrix is performed to form a code word, $c \in F_2^N$ as follows:

$$c = u * G_{r,m} \tag{2.7}$$

For example, assuming a message $u = (u_0, u_1, u_2, u_3, u_4) = (10101)$ is encoded using RM(1,4), the Generator Matrix, $G_{1,4}$ is calculated using equation (6)

$$G_{1,4} = \begin{bmatrix} G_{1,3} & G_{1,3} \\ \mathbf{0} & G_{0,3} \end{bmatrix} = \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \\ 1111111111111111 \end{bmatrix}$$

The codeword, $c = (c_1, c_1, ..., c_{15}, c_{16}) \in RM(1,4)$ is then calculated as:

$$(c_1, c_1, ..., c_{15}, c_{16}) = [u_4, u_3, u_2, u_1, u_0] * \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \\ 1111111111111111 \end{bmatrix}$$

$$c = u * G_{r,m} = [10100] * \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \\ 1111111111111111 \end{bmatrix} = [0000111001101011]$$

2.1.3 Decoding Reed Muller Codes

There are different techniques used for decoding Reed-Muller codes, the most common and easily implementable is majority logic decoding. The decoding process converts a code words, c to message, u. The decoding process consist of 2 steps which are majority logic decoding and converting c to u, these steps are described as follows [10]:

1. Majority decoding: This process is responsible for error correcting of the received vector. To perform majority decoding first consider the code word, c formed from encoding an input message, $u = (u_0, u_1, ..., u_{k-1})$ is represented as:

$$c = (c_0, c_1, ..., c_{N-1}) = u_0 v_0 + \sum_{1 \le i_1 \le m} u_{i_1} v_{i_1} + \sum_{1 \le i_1 \le i_2 \le m} u_{i_1} u_{i_1} v_{i_1} v_{i_1} + ... + \sum_{1 \le i_1 \le i_2 \le ... \le i_r \le m} u_{i_1, i_2, ..., i_r} v_{i_1} v_{i_1}, ..., v_{i_r} \tag{2.8}$$

   if $x = (x_0, x_1, ..., x_{N-1})$ is the received vector, then there are r + 1 stages of the decoding process. For $1 \le i_1 \le i_2 \le ... \le i_{r-l} \le m$ where $0 \le l \le r$, the following index set is formed :

$$S = \{a_{i_1 - 1} 2^{i_1 - 1} + a_{i_2 - 1} 2^{i_2 - 1} + ... a_{i_r - l - 1} 2^{i_{r-l} - 1} : a_{i_j - 1} \in 0, 1 for 1 \le j \le r - l\} \tag{2.9}$$

   S is a set of 2^{r-l} nonnegative integers that are less than 2^m. Assuming E is a set of integers $0, 1, ..., m-1$ not in $i_1 - 1, i_2 - 1, ..., i_{r-l} - 1$, such that

$$E = \{0, 1, ..., m-1\} \backslash \{i_1 - 1, i_2 - 1, i_{r-l} - 1\} = \{j_1, j_2, ..., j_{m-r+l}\} \tag{2.10}$$

   where $0 \le j_1 \le j_2 \le ... \le j_{m-r+l} \le m - 1$. Then a set of integers S^c is formed as

$$S^c = \{d_{j_1} 2^{j_1} + d_{j_2} 2^{j_2} + ... d_{j_{m-r+l} - 1} 2^{j_{m-r+l}} : d_{j_t} \in 0, 1 for 1 \le t \le m - r + l\} \tag{2.11}$$

   Then the set of indices, B is formed for each q $\in S^c$ such that:

$$B = q + S = \{q + s : s \in S\} \tag{2.12}$$

   Then the following equation, (14) will determine the decision equations in l-th decoding stage:

$$A^{(l)} = \sum_{t \in B} x_t^{(l)} \tag{2.13}$$

   Relation (15) consist of 2^{m-r+l} equations at each stage and follows a binary addition

rule. The input message $u_{i_1 i_2 ... i_{r-l}}$ is decoded as $u^*_{i_1 i_2 ... i_{r-l}} = 0$ if the decision equations results in zero. And it is decoded as $u^*_{i_1 i_2 ... i_{r-l}} = 1$ if the decision equations results in one.

When decoding process has finished l stages, a modified received vector is formed as follows:

$$x^{(l)} = x^{(l-1)} - \sum_{1 \le i_1 \le ... \le i_{r-l+1} \le m} u^*_{i_1, i_2, ..., i_{r-l+1}} v_{i_1} v_{i_1}, ..., v_{i_{r-l+1}} \tag{2.14}$$

$x^{(l-1)}$ is the modified received vector in the l-th stage of decoding process and $x^{(0)} = x$. Then proceed to the next stage and repeat the above process until the end of the r + 1 stage to decode all received bits

2. Converting r' to u: In this process, the modified received code word is converted to message vector. The following equation is used to determine the message vector:

$$u = r' * G^T_{r,m} \tag{2.15}$$

The following example illustrate the example of the decoding process. From equation (8) a code word can be expressed in terms as (9):

$(c_1, c_1, ..., c_{15}, c_{16}) = (u_0, u_1 + u_0, u_2 + u_0, u_2 + u_1 + u_0, u_3 + u_0, u_3 + u_1 + u_0, u_3 + u_2 + u_0, u_3 + u_2 + u_1 + u_0, u_4 + u_0, u_4 + u_1 + u_0, u_4 + u_2 + u_0, u_4 + u_2 + u_1 + u_0, u_4 + u_3 + u_0, u_4 + u_3 + u_1 + u_0, u_4 + u_3 + u_2 + u_0, u_4 + u_3 + u_2 + u_1 + u_0)$.

Using equation (11) to decide a set of indices of u and equation (12) to decide a set of indices of c for each corresponding u. Then the check-sums for u_1, u_2, u_3 and u_4 are created using (14):

$u_1 = c_1 + c_2 = c_3 + c_4 = c_5 + c_6 = c_7 + c_8 = c_9 + c_{10} = c_{11} + c_{12} = c_{13} + c_{14} = c_{15} + c_{16}$
$u_2 = c_1 + c_3 = c_2 + c_4 = c_5 + c_7 = c_6 + c_8 = c_9 + c_{11} = c_{10} + c_{12} = c_{13} + c_{15} = c_{14} + c_{16}$
$u_3 = c_1 + c_5 = c_2 + c_6 = c_3 + c_7 = c_4 + c_8 = c_9 + c_{13} = c_{10} + c_{14} = c_{11} + c_{15} = c_{12} + c_{16}$
$u_4 = c_1 + c_9 = c_2 + c_{10} = c_3 + c_{11} = c_4 + c_12 = c_5 + c_{13} = c_6 + c_{14} = c_7 + c_{15} = c_8 + c_{16}$

So if the received code word is: [1 1 1 0 1 1 0 0 1 0 1 1 1 0 1 1]
We calculate the check-sums for u_1, u_2, u_3 and u_4 using equation ():

$u_1 : \{0, 1, 0, 0, 1, 0, 1, 0\}, u_2 : \{0, 1, 1, 1, 0, 1, 0, 1\}, u_3 : \{0, 0, 1, 0, 0, 0, 0, 0\}, u_4 : \{0, 1, 0, 1, 0, 1, 1, 1\}$

Considering the majority of each u_1, u_2, u_3 and u_4, we conclude that $u_1 = 0, u_2 = 1, u_3 = 0$

and $u_4 = 1$. To determine u_0, use equation (15) to calculate the modified received vector:

$$\begin{aligned} x &= [1110110010111011] - [10100] * \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \\ 1111111111111111 \end{bmatrix} \\ &= [1110110010111011] - [0011001111001100] \\ &= [1101111101110111] \end{aligned}$$

Using value of x,we decide that $u_0 = 1$. The corrected vector, cc is calculated as:

$$\begin{aligned} cc &= 0011001111001100 - 1111111111111111 \\ &= 1100110000110011 \end{aligned}$$

The equation (15) is used to decode the code word cc, to form the message vector uu:

$$\begin{aligned} uu &= r * G_{1,4}^T \\ &= 0000111001101011 * \begin{bmatrix} 10000 \\ ... \\ 11001 \\ ... \\ 11111 \end{bmatrix} \\ &= 10100 \end{aligned}$$

2.2 Modulus Function

This section explains the LSB embedding using the modulus function. It is a steganographic technique used to increase the visual quality of the stego image using modulus functions. It was proposed by Thien and Lin [16].
The expression of modulus function is defined as c = a mod b, where a is the dividend, b is the divisor, and c is the remainder. For example, 3 mod 2 equals 1, that is, the division of 3 by 2 leaves a remainder of 1. Let x be a pixel used to hide data in the cover, z be the decimal value of the block/unit bits in the range 0 to $2^n - 1$. First, the embedding process of modulus function is employed to hide z_i unit in the i^{th} pixel of cover, x_i to compute the following equation:

$$dd_i = z_i - (x_i \bmod 2^n). \tag{2.16}$$

Where n is the index of the low order bit of each pixel to be embedded.

The result dd_i is applied to determine the minimum variance using the following equations:

$$dd_i' = \left\{ \begin{array}{lll} dd_i & \text{if} & -\left\lfloor \frac{2^n-1}{2} \right\rfloor \leq dd_i \leq \left\lceil \frac{2^n-1}{2} \right\rceil \\ dd_i + 2^n & \text{if} & -2^n + 1 \leq dd_i \leq -\left\lfloor \frac{2^n-1}{2} \right\rfloor \\ dd_i - 2^n & \text{if} & \left\lceil \frac{2^n-1}{2} \right\rceil \leq dd_i \leq 2^n \end{array} \right\} \tag{2.17}$$

The value of dd_i can exceed the range 0 to 255, so equation (19) is used determine the final pixel value, xx_i after embedding

$$x_i' = \left\{ \begin{array}{lll} x_i + dd_i' & \text{if} & 0 \leq x_i + dd_i' \leq 255 \\ x_i + dd_i' + 2^n & \text{if} & x_i + dd_i' \leq 0 \\ x_i + dd_i' - 2^n & \text{if} & x_i + dd_i' \geq 255 \end{array} \right\} \tag{2.18}$$

The extraction of the embedded data embedded in the n low order bits is then done using the following equation:

$$z_i = x_i' \bmod 2^n \tag{2.19}$$

2.3 Visual Cryptography scheme

This section describes the concept of Visual cryptography. It is a cryptographic method used to protect secret message by encrypting a secret message(image, text) into a number of shares and decrypting the secret message by superimposing the encrypted shares [3]. The decryption process does not require any computation. In k out of k visual cryptography scheme, the dealer encrypts a binary secret image into k shares and distributes them to k participants. To decode the secret image, all k participants' shares are required, while in k out of n threshold scheme only k participants' shares are required. Decoding is a process of overlapping all k number of shares.

In this research a 2 out of 2 visual cryptography scheme, (2,2) VCS has been implemented. The encoding and decoding processes have been described as follows.

2.3.1 Encoding process

The encoding process is discussed in this section. It generates 2 shares from a secret message S. Fig 2.2 shows how balck and white pixels are encoded to generate two shares $S1$ and $S2$.

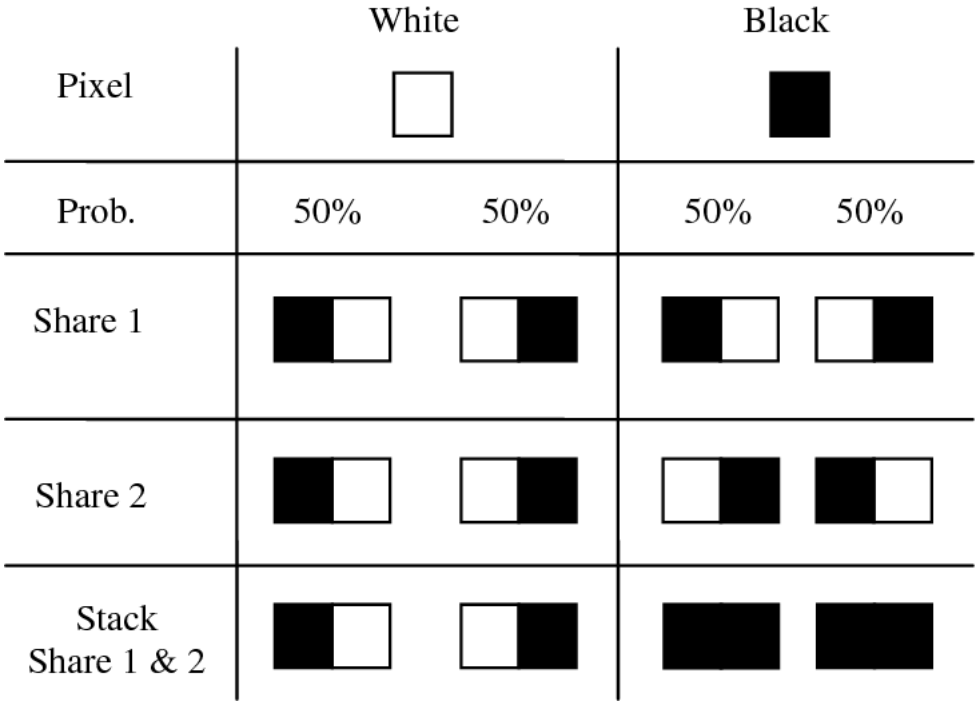


Figure 2.2: Construction of (2,2) Visual Cryptography Scheme [1]

The encoding process of each pixel black and white is done using the following process:

1. Represent the pixels of the secret image as an n x m Boolean matrix S consisting of 0s and 1s ie $S = (s_{ij})_{nxm}$ where $s_{ij} = 1$ or 0, n and m are height and width of secret image for $1 \leq i \leq n$ and $1 \leq j \leq n$. Black and white pixels are represented by 0 and 1 respectively.

2. Construct the basis matrices S_0 and S_1 for white and black pixels as follows:

$$S_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}, S_1 = \begin{bmatrix} 10 \\ 01 \end{bmatrix}$$

3. Generate collections of matrices C_0 for white pixels by permuting the columns of matrices S_0. And also generate collections of matrices C_1 for black pixels by permuting

the columns of matrices S_1. The collections C_0 and C_1 are represented as follows [3]:

$$C_0 = \{ \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \begin{bmatrix} 01 \\ 01 \end{bmatrix} \} \text{ and } C_1 = \{ \begin{bmatrix} 10 \\ 01 \end{bmatrix}, \begin{bmatrix} 01 \\ 10 \end{bmatrix} \}$$

4. The dealer encrypts a white pixel by randomly selecting one matrix $C_0^{(i)} \in C_0$ and a black pixel by randomly selecting one matrix $C_1^{(i)} \in C_1$ where $i = 0 or 1$. The first row of the selected matrix $C_0^{(i)}$ or $C_0^{(i)}$ is used for share 1 $S1$ and the second row for share 2 $S2$. Therefore each pixel white or black is represented as two sub pixels on each of the shares $S1$ and $S2$.

2.3.2 Decryption process

The decoding process is described in this section. It regenerates the secret message from the two encrypted shares $S1$ and $S2$ using the following process:

1. Superimpose $S1$ and $S2$. When the sub pixels of $S1$ and $S2$ are correctly aligned, the black pixels in the stacked shares are represented by the Boolean OR of the rows in the matrix and the secret image is visible.

The reconstructed secret image of (2, 2) VCS consist of parameters, pixel expansion m = 2 ie two sub pixels in S_1 and S_2 for each pixel of the secret image, relative difference α between the reconstructed white and black pixels is calculated by:

$$\alpha = \frac{h(S_0).h(S_1)}{m} = \frac{1}{2}$$

where $h(S_0)$ and $h(S_1)$ is the hamming weight corresponding to the basis matrices S_0 and S_1. And contrast β is calculated as follows:

$$\beta = \alpha.m = \frac{1}{2}.2 = 1$$

Therefore the scheme has a 50% loss of contrast

2.4 Blum-Blum-Shub Method:

This section explains the Blum Blum Shub method. It is simple pseudorandom number generator (PRNG) proposed by Lenore Blum, Manuel Blum and Michael Shub [7]. The algorithm generates a sequence of random numbers as follows:

1. Generate p and q. These are two prime numbers.

2. Determine the divisor n of the modulus function. It is the product of the prime numbers p and q:

$$n := p.q \tag{2.20}$$

3. Determine the seed s: Select a random integer within the range 1 and $n-1$. ie $s \in_R [1, n-1]$.

4. Calculate the initial sequence value x_0 by:

$$x_0 = s^2 (\text{ mod } n) \tag{2.21}$$

5. The sequence is determined as:

$$x_t = x_{t-1}^2 (\text{ mod } n) \tag{2.22}$$

Example
Let $n = p.q = 7 \ . \ 19 = 133$. And choose $s = 100$. We have $x_0 = 100^2 (\text{ mod } 133) = 25$. Then the sequence is $x_1 = 25^2 (\text{ mod } 133) = 93$, $x_2 = 93^2 (\text{ mod } 133) = 4$, $x_3 = 4^2 (\text{ mod } 133) = 16$, $x_4 = 16^2 (\text{ mod } 133) = 123$.

2.5 A. M. Molaei's Method

In this section, Molaei's method is described. This method combines coding theory concepts and steganography to achieve higher embedding capacity and robustness against different kinds of noise attacks [12]. However, this method has low embedding efficiency (PSNR) and embedding capacity because it embeds the first and second LSBs of all pixels of the cover. The method is also less secure since it embeds the secret data serially on the cover image. It consist of two phases: Embedding and Extraction Phase.

2.5.1 Overview of Embedding phase

The embedding phase is responsible for hiding secret data into cover media. It hides a secret binary data SD in a grayscale cover image C of height H and width W and generates a stego image C'. The block diagram of the embedding process is shown in Fig 2.3.

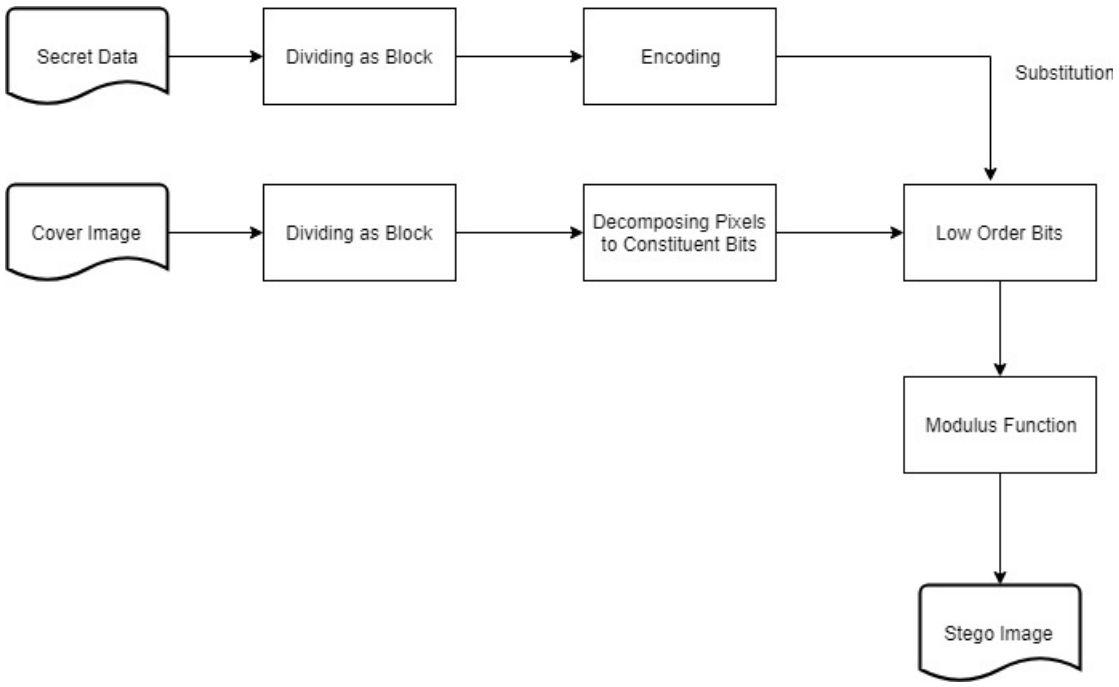


Figure 2.3: Embedding process

The embedding process is done using the following process:

1. **Divide secret data into blocks:** The secret data is divided into blocks that will enable easy encoding of the secret data. The first B bits are divided into nb_0 blocks $b = (b_0, b_1, ..., b_{k_0-1})$ of equal sizes equal to k_0, where $1 \leq B \leq \frac{H \text{ x } W}{2}$, k_0 is the message block size of the RM(1,3) code. The first number of blocks nb_0 is calculated as follows:

$$nb_0 = \left\lceil \frac{B}{k_0} \right\rceil \tag{2.23}$$

   Considering $SD > B$, the remaining $(SD - B)$ bits are also divided into nb_1 blocks $bb = (bb_0, bb_1, ..., bb_{k_1-1})$ of sizes k_1, where $1 \leq (SD - B) \leq \frac{H \text{ x } W}{4}$, and k_1 is the message block size of RM(2,5).The second number of blocks nb_1 is calculated as follows:

$$nb_1 = \left\lceil \frac{(SD - B)}{k_1} \right\rceil \tag{2.24}$$

2. **Encode the secret blocks:** The blocks of the secret data are encoded to enable error detection and correction to be performed from attacked stego images. Consider all nb_0 blocks, using equation (2.7), encode each block b using RM(1,3) code to form code word $c = (c_0, c_1, ..., c_{N_0-1})$ of size N_0, where N_0 is block size of RM(1,3) code.

Consider all nb_1 blocks,using equation (2.7), encode each block bb using RM(2,5) codes to form code word $cc = (cc_0, cc_1, ..., cc_{N_1-1})$ of size N_1, where N_1 is block size of RM(2,5) code.

3. **Divide the pixels of I:** To be able to embed the sequence of c into the cover I, firstly divide the all pixel of I into blocks p of size N_0.

4. **LSB embedding using Modulus Function:** In this process the encoded secret blocks are hidden or embedded in the pixels of the cover image. The two sequence of code words c and cc are embedded on the first and second LSB respectively.
   For each block of nb_0 blocks, consider the decimal value of each c_i to be embedded in each p_i and calculate the difference value dd_i using equation (2.16). Determine the difference dd'_i using equation (2.17). The modified pixel value pp_i is calculated by equation (2.18).

5. Perform the same procedure to embed the nb_1 blocks of cc into sequence of modified pixel blocks pp. The output of the embedding process are modified pixel values pp' which forms the stego image C'

The equations (2.16), (2.17) and (2.18) were implemented to reduce the difference between the original pixel values and modified pixel values. This reduces the overal Mean Square error of between the original pixels and the embedded pixels. Thereby increasing the visual quality of the stego. The embedding process only hides secret data in both first and second LSBs of the cover.

2.5.1.1 Determining the Embedding Capacity of A. M. Molaei's Method

The maximum size of secret data D_{max} that can be embedded on a cover I is determined as follows:

$$D_{max} = \sum_{i=0}^{n} \left\lfloor \frac{H * W}{2^m} \right\rfloor * k \tag{2.25}$$

where k is the message block size of RM(r, m) encoder and n is the number of LSBs used for embedding. The embedding capacity, EC is then determined by the following equation:

$$EC = \frac{|D_{max}|}{H * W} \tag{2.26}$$

2.5.2 Overview of Extraction phase

In this section the extraction process is discussed. The extraction phase aims to retrieve the hidden data successfully form the received stego image C'. The block diagram of the embedding process is shown in Fig 2.4.

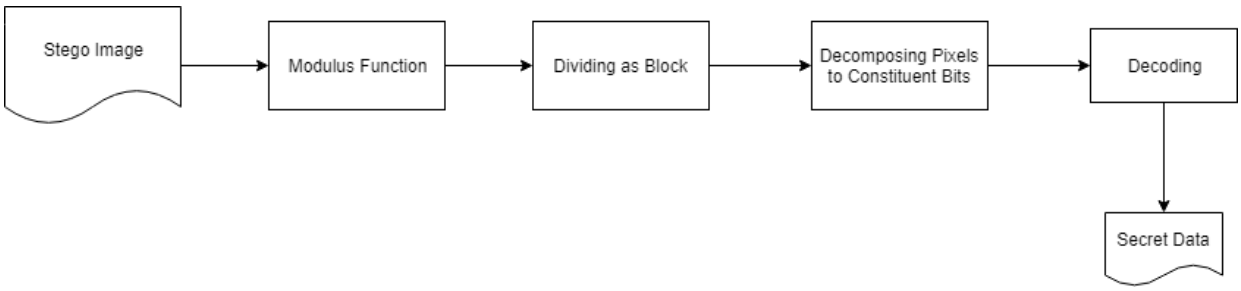


Figure 2.4: Extraction process

The secret data SD is retrieved from the Stego image C' using the following process:

1. **Modulus Function**: The process aims to extract the secret encoded bits first. The equation (2.19) is used to extract the decimal values z_{ii} of the secret data, for $1 \leq ii \leq H$ x W and $n = 1$.

2. **Divide extracted data into blocks**: The extracted data is divided into blocks that will enable easy decoding of the extracted data. The first C bits of z_{ii} are divided into nc_0 blocks $c = (c_0, c_1, ..., c_{N_0-1})$ of equal sizes equal to N_0, where $1 \leq C \leq H$ x W, N_0 is the block size of the RM(1,3) code. The first number of blocks nc_0 is calculated as follows:

$$nc_0 = \left\lceil \frac{B}{N_0} \right\rceil \tag{2.27}$$

3. **Decode the extracted blocks:** The blocks are decoded to recover the secret data using majority decoding as described in section(2.1.3). For all the nc_0 blocks, each code word block c is decoded using RM(1,3) decoder to form a secret block $b = (b_0, b_1, ..., b_{k_0-1})$

4. Increase n to extract data in second Lsb and repeat step (1) to (3) using $1 \leq ii \leq \frac{H \text{ x } W}{2}$, number of blocks $nc_1 = \left\lceil \frac{H \text{ x } W}{2 * N_1} \right\rceil$, where N_1 is the block size of the RM(2,5) code. Decode using RM(2,5) decoder

5. **Output:** The output of the process is a sequence of bits from the Decoder which represent the secret data.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter discusses about the Research Design, System design and Implementation of the proposed method. The last section discusses about the scenarios of the experiments conducted to test the Code based steganography.

3.1 Research Design

To increase the embedding capacity of code based steganography, this research used multiple embedding method combined with various cryptography techniques. Multiple embedding aims to embed more than one secret bits on each randomly selected pixel of the cover image. The proposed method only embeds on the first LSB of the selected pixels. Multiple embedding was made possible by performing XOR on the secret bits that has to be embedded on each pixel. The idea of Multiple embedding also contributes to the increased of robustness of code based steganography.
To increase the robustness of the scheme, the secret message was encoded by high performance error correction code called Reed Muller(1, 4) before embedding it in to the cover image. After the encoding process, a sequence of random integers of size equal to $\frac{3}{4}$ of the cover pixels was generated using a pseudo random number generator(PRNG) called Blum Blum shub. This ensured that only are embed to increase the embedding efficiency. Random embedding of the secret data was done to increase the security of the hidden secret data. The sequence of integers represented the pixel positions on the cover image that would be embedded with the secret data.
Multiple embedding of the encoded secret data was conducted on the $\frac{3}{4}$ of the cover pixels. The seed used by PRNG and the number of cycles of the multiple embedding were used to generate a binary image called a secret key. The binary image was communicated between the sender and the recipient using Visual Cryptography techniques. Upon receiving the Stego image, the recipient reconstructs the secret key to retrieve the seed and number of cycles. The recipient retrieves the embedded secret data by regenerating the sequence of integers and extract the LSBs. The original secret data is obtained by decoding the extracted LSBs.

3.1.1 System Design and Implementation

This section discusses about a steganographic technique that offers high embedding capacity, high error correction capabilities and superior visual quality. The proposed method is discussed in detail. It consist of two main processes, embedding process and extraction process.

3.1.2 Overview of Embedding Process

In this section, the embedding process is described. The process is responsible for embedding secret data securely. There are three input images to this process, the cover image CI, agreed image A and secret image SI. Fig 3.1 shows how a secret message is embedded in a cover using an agreed image to generate Agreed and Stego image.

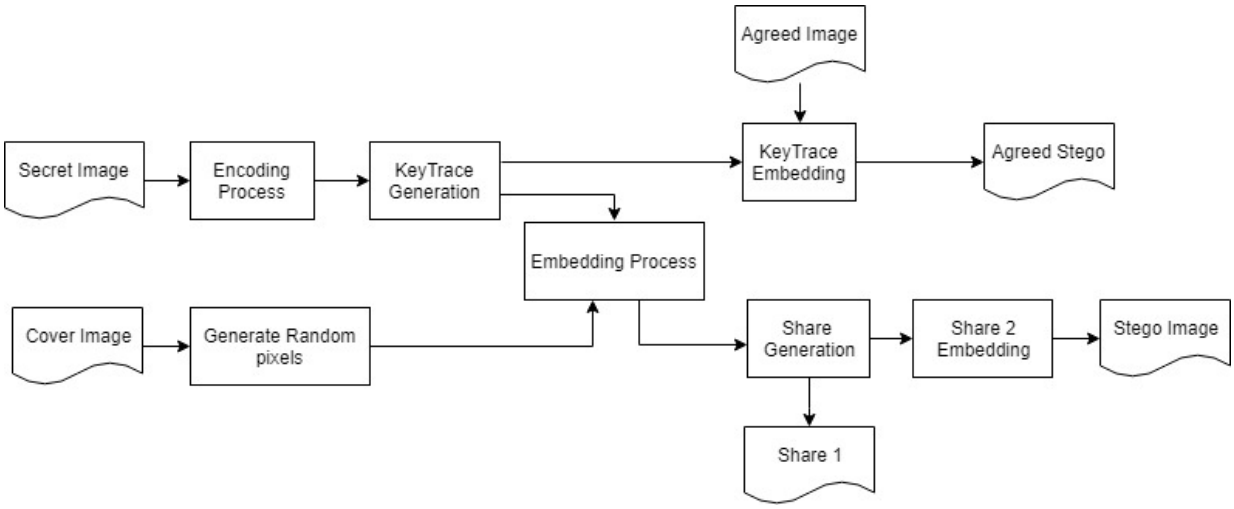


Figure 3.1: Data Embedding procedure

The secret image pixels are first decomposed into bits and are encoded with Reed Muller error correction codes. First order Reed Muller RM (1, 4) codes have been introduced because of their high Error Correction Capacity, efficiency and relatively easy to decode. To increase the security of the secret data, a pseudorandom number generator (PRNG) was introduced to randomly generate pixels to be embedded from the cover. If the seed of the PRNG is not known then the hidden secret data cannot be extracted correctly.

The scheme implements LSB embedding using modulus function to embed the encoded secret data on $\frac{3}{4}$ of the cover image pixels. Only $\frac{3}{4}$ of the cover image pixels were used to increase the visual quality of the final Stego image. If the size of the encoded secret data exceeds $\frac{3}{4}$ of the cover image pixels, the data is re embedded on the same randomly generated pixels using a key. The key contains Seed and number of cycles. The Key is securely distributed to the receiver using (2,2) VCS mentioned in section (4). This further increases the security of the embedded secret data.

The following definition discusses how the embedding capacity is determined using the proposed method.

Definition 3.1: *Given p pixels of a Cover image CI of height H and width W where* $p \leq \frac{3}{4}\ x\ H\ x\ W$. *If n secret bits are re embedded on each LSB of the p pixels, then the*

total secret bits S_T that can be embedded on the cover image CI is:

$$S_T = p \ . \ (n+1) \tag{3.1}$$

and the Embedding Capacity EC expressed as a percentage is determined by

$$EC = \frac{S_T}{H * W} = \frac{p \ . \ (n+1)}{H * W} \tag{3.2}$$

Example:
Consider a Cover image of size $H = 512$ and $W = 512$. Assuming $n = 5$ secret bits are re embedded on the LSBs of $p = \frac{3}{4}$ x H x $W = \frac{3}{4}$ x 512 x 512 = 196608 pixels. The total secret bits S_T that can be embedded on the cover image is calculated as:

$$S_T = 196608 \ . \ (5+1) = 1179648 \ bits$$

and the Embedding Capacity EC is calculated by

$$EC = \frac{1179648}{512 * 512} = 450\%$$

The following sections describes each sub process of the embedding process in detail as follows:

3.1.2.1 Encoding Process

The Reed Muller encoder is responsible for encoding secret data before embedding such that the secret data is recoverable using error correction if the stego image encounters attacks. The encoding process consist of the following sub processes as shown of Fig 3.2.

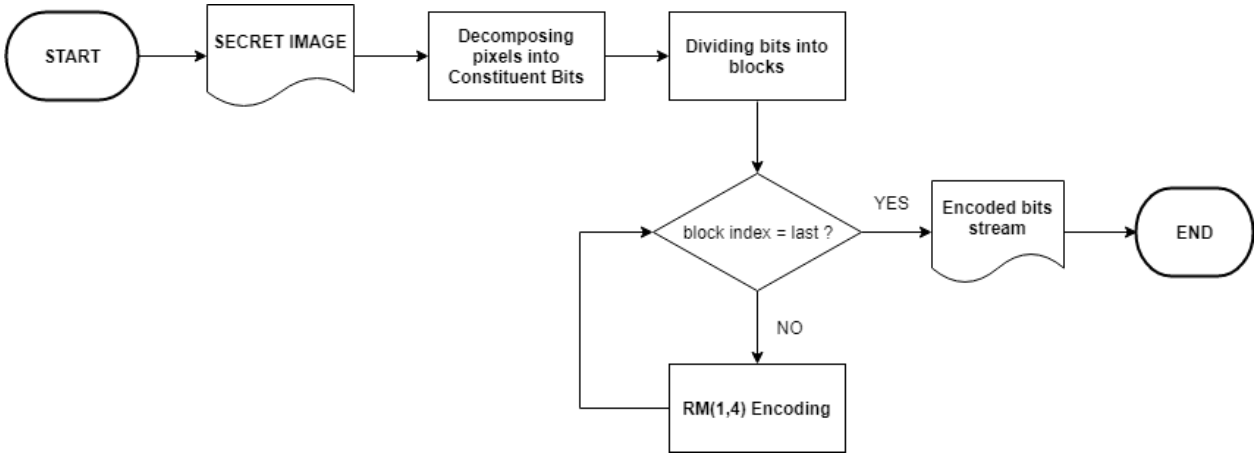


Figure 3.2: Encoding Process

The secret image SI with height H_0 and width W_0 is decomposed into bits and encoded

by RM(1,4) Encoder to generates sequence of code words C using the following three process:

1. *Decomposing pixels into Constituent Bits*: The pixel values of the secret image are converted into bits because of the binary Reed Muller codes that has been introduced. Each pixel of SI (s_i, $1 \leq i \leq H_0$ x W_0) is converted to a byte. Each byte is broken down into bits and the output of the process is a sequence of all bits $sb = (sb_0, sb_1, ..., sb_Y)$ where $Y = 8$ x H_0 x W_0.

2. *Dividing bits into blocks*: The process divides secret bits into equal sized blocks as required by the Encoder such that the encoding process is more efficient. The input to the process is a sequence of secret bits sb and message length, k_2 of the RM(r_1,m_1) encoder. The following definition describes how the blocks are created as follows:

   Definition 3.2: *Assume $Y \nmid k_2$ the secret bits sb are divided into n blocks $mb^{(q)} = (mb_0^{(q)}, mb_1^{(q)}, ..., mb_{k_2-1}^{(q)})$ of length k_2 where*

$$n = \left\lceil \frac{Y}{k_2} \right\rceil \tag{3.3}$$

   and ($1 \leq q \leq n$).
   The number of zero bits Z_n that needs to be added to the n^{th} block are determined by:

$$Z_n = (n * k_2) - Y \tag{3.4}$$

   Example:
   Assuming the secret bits $sb = [1010111100110]$ are to be encoded by RM(1,4) encoder. Using equation (1), the message length $k_2 = 5$. The secret bits size $Y = 13$. Therefore $13 \nmid 5$. The number of blocks n is calculated as:

$$n = \left\lceil \frac{Y}{k_2} \right\rceil = \left\lceil \frac{13}{5} \right\rceil = 3$$

   The number of zero bits Z_n added to 3^{th} block are determined by:

$$Z_n = (3 * 5) - 13 = 15 - 13 = 2. \tag{3.5}$$

   Therefore the blocks to be encoded are : $mb^{(1)} = [10101]$, $mb^{(2)} = [11100]$ and $mb^{(3)} = [110\mathbf{00}]$

3. *RM (r_1,m_1) Encoding*: The Encoder is responsible for converting the secret message bits into code words which will enable error detection and error correction. The

input to the process at each instance is a block $mb^{(q)}$. For all n blocks, encode each $mb^{(q)}$ using RM(r_1,m_1) error correction code using equation (7) to form codewords $cw^{(q)} = (cw_1^{(q)}, cw_2^{(q)}, ..., cw_{N_2}^{(q)})$ where $N_2 = 2^{m_2}$. Decompose the code words $cw^{(q)}$ into constituent bits, for $(1 \leq q \leq n)$. Therefore the output of the Encoder is a sequence of bits, $eb = eb_0, eb_1, ..., eb_{E_T}$ formed from decomposing all code words, where the total encoded bits size E_T after encoding process is calculated as:

$$E_T = \frac{8 * H_0 * W_0 * N_2}{k_2} \tag{3.6}$$

3.1.2.2 Key Trace Generation

This section discusses about how the Key Trace is generated. Key Trace are the sequence of bits generated from XORing the encoded bits and the chosen Reference Key bit value. The main purpose of this process is to increase the overall hiding capacity of the scheme by re embedding secret bits on the pixels of the cover image. Fig 3.3 shows how the Key Trace bits are generated.

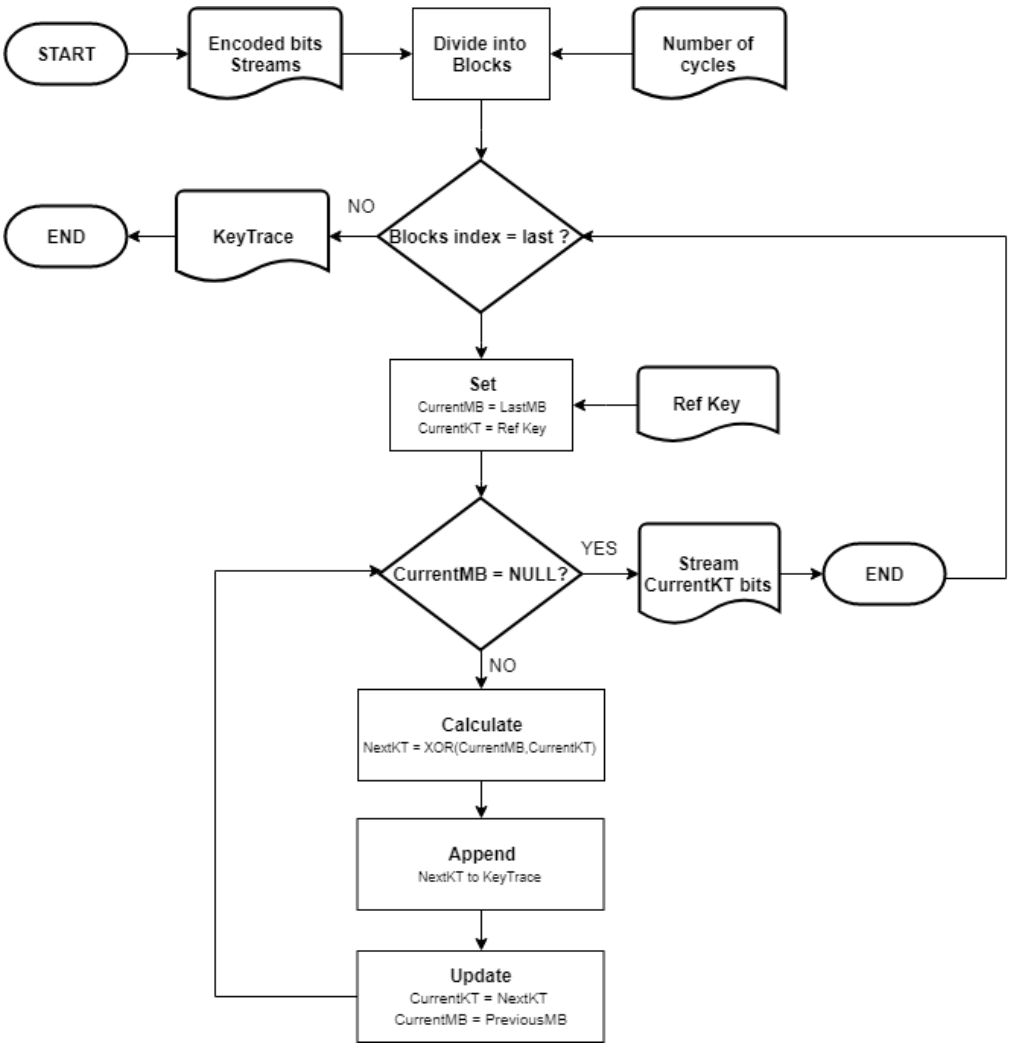


Figure 3.3: Key Trace Generation

This process is executed if and only if the size of the encoded bits, E_T is greater than $\frac{3}{4}$ x H_1 x W_1, where H_1 and W_1 are the height and width of the cover image I. The key trace K_T is generated from the encoded bits eb using the following 4 steps:

Step 1 Input sequence of encoded bits eb and Reference Key. A reference key, R is any binary value of choice ie $R \in 1, 0$.

Step 2 Determine how many bits are to be embedded on each pixel of the cover, called number of cycles, N_c by:

$$N_c = \frac{E_T}{\frac{3}{4} * H_1 * W_1} \tag{3.7}$$

Step 3 Divide the encoded bits, eb to h blocks $db = (db_1, db_2, ..., db_{\lceil N_c \rceil})$ and/or h' blocks $db' = (db'_1, db'_2, ..., db'_{N_c})$.
where $(h = \frac{3*H_1*W_1*(N_c - \lfloor N_c \rfloor)}{4}$ and $(h' = \frac{3*H_1*W_1}{4} - \frac{3*H_1*W_1*(N_c - \lfloor N_c \rfloor}{4})$

Step 4 Consider each of the blocks db and db' , the Key Trace block CK and Cover embeddable bit E are determined using algorithm 1:

Algorithm 1 Generation of Key Trace block and Cover Embeddable Bit

Input: Blocks db or db', Number of cylces N_c, Reference Key R

Output: KeyTrace block CK and Cover Embeddable bit E

1 $t \leftarrow N_C$

$CK_{t-1} \leftarrow db_t \oplus R$

while $t > 0$ **do**
  if $t == 1$ **then**
    $E \leftarrow CK_t \oplus db_t$
    Append E to E_m
  else
    $CK_{t-1} \leftarrow db_t \oplus CK_t$
    $t \leftarrow t - 1$

For each block of size n the process creates a key Trace block of size n-1. The output are two sequences of bits,
a). Key Trace, K_T which is a decomposition of all blocks CK_i into constituent bits.

$$|K_T| = \lceil N_c \rceil \, (\frac{3 * H_1 * W_1 * (N_c - \lfloor N_c \rfloor)}{4} + N_c(\frac{3 * H_1 * W_1}{4} - \frac{3 * H_1 * W_1 * (N_c - \lfloor N_c \rfloor)}{4}) \tag{3.8}$$

b). For embedding to cover, E_m.

3.1.2.3 Generate Random pixels

This section discusses how the random pixel positions are generated. The process uses a Blum Blum Shub a cryptography pseudo random number generator discusses in section

(2.4) to generate random sequence of integers between 1 and H_1 x W_1. The generated integers represent pixel positions on the cover image that are used for embedding secret bits, E_m. The secret bits are embedded according to the order of the sequence generated. If the seed of the PRNG is not known then the embedded pixel positions are not known and the secret message cannot be extracted correctly. This process was introduced to increase the secrecy of the embedded secret message.
To generate a sequence of random pixel positions, $pp = pp_1, pp_2, ..., pp_L$, the function inputs
a). Seed
b). Length of pp, L where

$$L = \left\{ \begin{array}{lll} E_T & \text{if} & 1 \leq E_T \leq (\frac{3}{4} * H_1 * W_1) \\ \\ \mid E_m \mid & \text{if} & E_T \geq (\frac{3}{4} * H_1 * W_1) \\ \\ \end{array} \right\} \tag{3.9}$$

The output is a random sequence of integers values $pp = (pp_1, pp_2, ..., pp_L)$ where $1 \leq pp_i \leq H_1 * W_1$. For example: 120132 145 9987 97 103110 125 14285 92 117 197 203210

3.1.2.4 Embedding Process

This section discusses how the secret bits are embedded into the cover image. It is responsible for efficiently hiding the encoded secret bits on randomly generated Cover Image pixels.
The process require two inputs :
a). The sequence of embedding bits, EB, where:

$$EB = \left\{ \begin{array}{lll} eb & \text{if} & 1 \leq E_T \leq (\frac{3}{4} * H_1 * W_1) \\ \\ E_m & \text{if} & E_T \geq (\frac{3}{4} * H_1 * W_1) \\ \\ \end{array} \right\} \tag{3.10}$$

b). Sequence of randomly generated integers, pp

LSB embeddding using modulus function as described in section (2.2) was used to embed all bits of EB on Cover pixels of index pp. It is implemented because of its simplicity in embedding and extraction. The embedding process is done as follows:

Step 1: Consider the decimal value of each bit in EB to be embedded on each pixel px_i on index pp_i.

Step 2: For each EB_i and px_i with $n = 1$, calculate the difference value dd_i using equation (2.16). Determine dd'_i using equation (2.17). And lastly, the modified pixel value pp'_i is calculated by equation (2.18).

Step 3: Perform the same procedure for $1 \leq i \leq L$. The output of the embedding process are modified pixel values pp' which forms the stego image I'

3.1.2.5 KeyTrace embedding

In this section, a sequence of Key Trace bits, K_T are embedded in the image A of height H_2 and width W_2. The image A is a grayscale image that has been agreed upon between Sender and Receiver before communication of the secret message. To embed K_T the following process was performed:

1. Determine the number of LSBs nn to be embedded: The value of nn is calculated as follows:
$$nn = \frac{\mid K_T \mid}{H_2 \text{ x } W_2} \tag{3.11}$$
If nn is a not an integer, then there are two sets of block sizes,

   (a) The first $(\lceil nn \rceil * H_2 * W_2)$ bits are divided into blocks w of size $\lceil nn \rceil$, such that $w = (w_1, w_2, ..., w_{\lceil nn \rceil})$ and

   (b) The remaining $|K_T| - (\lceil n \rceil * H_2 * W_2)$ bits are divided into blocks, w of size n, such that $w = (w_1, w_2, ..., w_{nn})$ where $n \geq 1$.

   But if nn is an integer, then all K_T bits are divided into blocks w of size nn, such that $w_i = (w_1, w_2, ..., w_n n)$.

2. Perform LSB embedding using Modulus function: To embed the Key trace bits on the agreed image, the modulus function was applied to increase the visual quality of the resulting stego. The method reduces the difference between the original and final pixel values using equations (16) and (17). To embed the all blocks w, the following process was performed:

3. Consider the decimal value z of each block in w to be embedded on each pixel pa of image A. For the first $(nn - \lfloor nn \rfloor) * H_2 * W_2)$ decimal values we have $n = \lceil nn \rceil$, calculate the difference value dd_i using equation (16). Determine dd'_i using equation (17). The modified pixel value pa'_i is calculated by equation (18).

4. Perform the same procedure for the last $(H_2 * W_2) - (nn - \lfloor nn \rfloor) * H_2 * W_2)$ decimal values where $n = \lfloor nn \rfloor$. The output of the embedding process are modified pixel values pa' which forms the stego image A'.

3.1.2.5 Share Generation

In this section, the process is designed to securely communicate the seed, number of cycles, N_c and height H_0 and width W_0 of the secret image. A Key in form of binary image is generated based on these 4 parameters. (2,2) visual cryptography was later implemented on the Key to generate shares because of its low complexity nature in encoding and decoding. The following generation of key and shares is described as follows:

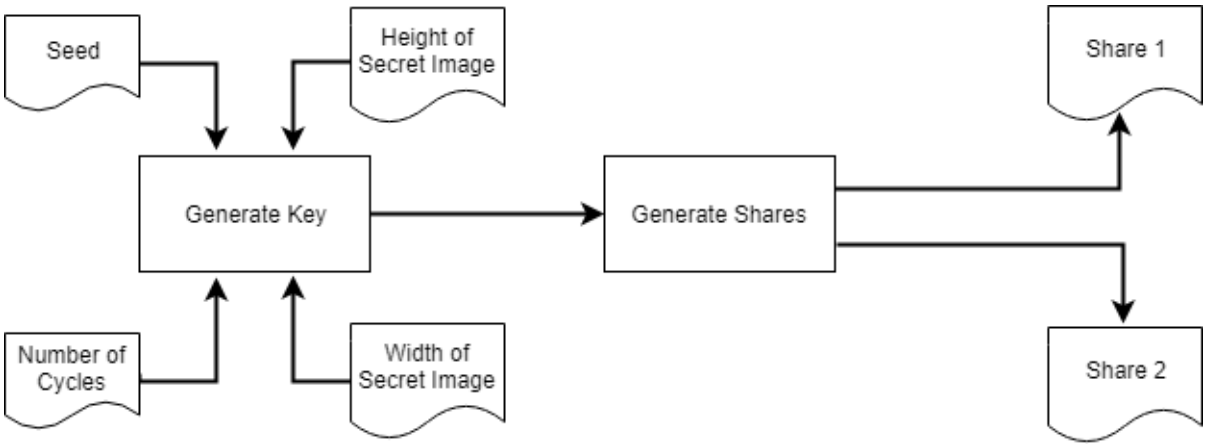


Figure 3.4: Share Generation

1. Generate Key: The function of this component is to create a secret key during the embedding process to enable easy and successful extraction of the hidden secret image, SI. The process require 3 input texts:

   (a) Seed, SE: An integer that initialize the PRNG.

   (b) Number of Cycles, N_c: it is the result after dividing the Total encoded secret bits by the of the cover image.

   (c) Height H_0 and width W_0 of the secret image : The output of the process is a binary image consisting of SE, N_c, H_0 and W_0.

   The three input text are converted from text to binary image using the following process:

   (a) Concatenate SE, N_c, H_0 and W_0 separating each with a space character to form, Txt. Such that, $Txt = SE + ' '+ N_c + ' ' + H_0 + ' ' + W_0$.

(b) Represent each character in Txt by (0, 1)-matrix, T_g of dimensions m' x n' with 0s and 1s arranged to show the character Txt_g visually, where $1 \leq g \leq |Txt|$, $m' = 16$ and $n' = 14$.

(c) Combine all T_i to form one binary matrix, T' of size m'' x n'', where $m'' = m' * |Txt|$ and $n'' = n' * |Txt|$

Example

Assuming the seed $SE = 262139$, Number of cycles $N_c = 0.01$, Height of secret image $H_0 = 10$ and Width of secret image $W_0 = 6$. The string $Txt =$ 262139 0.01 10 6, and the binary image T' is shown in fig 3.5.

Figure 3.5: Example of Key Generation

2. Generate Share: (2, 2) visual cryptography is implemented on T' as discussed in section (2.3) to increase the security of the secret key while requiring less computation time. The input to the process is a binary matrix, $T' = (t'_{ij})_{mm*nn}$ where $t'_{ij} =$ 1 or 0. The process perform encryption of T' as described in (2.3.1) to generate 2 random binary matrixes, $T'_1 = (t'_{ij})_{mm*2nn}$ and $T'_2 = (t'_{ij})_{mm*2nn}$ where $t_{ij} = 1$ or 0, referred to as Share 1 $S1$ and Share 2$S2$ respectively. $S1$ is distributed securely to the receiver while $S2$ is to be embedded on the stego image.

Example

Considering the binary image formed in fig 3.5, the (2,2) Visual cryptography will create two shares $S1$ and $S2$ which are scrambled and cannot reveal any information about the secret image. Fig 3.6 and 3.7 shows the generated shares $S1$ and $S2$

Figure 3.6: Share 1

Figure 3.7: Share 2

3.1.2.6 Share 2 embedding

In this section, the embedding of Share 2 $S2$ is described. Firstly, $S2$ is encoded with RM(r_1,m_1) code as discussed in section (2.1) to enable error correction of error bits. The

encoded $S2$ is embedded on Stego I' such that the correct receiver, possessing $S1$, can extract $S2$, decode and be able to stack both shares to retrieve SE, N_c, H_0 and W_0. The emdding of $S2$ to generate a new stego image is done by the following process:

1. Input the Stego image, I', T_2' and message length k_2 of RM(r_1,m_1) encoder.
2. Divide $S2$ into linear blocks of size k_2: The binary matrix, T_2' is decomposed to single linear block of bits, $T_2'' = t_1'', t_2'', ..., t_{n''*2m''}''$. Divide, T_2'' to nt blocks, $tb = (tb_0, tb_1, ..., tb_{k_1-1})$ of size k_1, where $nt = \left\lceil \frac{n''*2*m''}{k_1} \right\rceil$
3. Encode each tb with RM(r_1,m_1) error correction code discussed in section (2.1.2) to form codewords $ct = (ct_1, ct_2, ..., ct_{N_1})$ where $N_1 = 2^{m_1}$.
4. Output is a sequence of bits CT of size ET_T formed after decomposing all code words, such that the total encoded bits size, ET_T of Share 2 bits after encoding process is calculated as:

$$ET_T = \frac{n'' * 2 * m'' * N_1}{k_1} \tag{3.12}$$

5. LSB embedding using modulus function : The method described in section (2.2) was used to embed all bits of CT on Stego I' pixels. The embedding process is performed as follows:

   (a) Consider the decimal value of each bit in CT to be embedded on each pixel pp'. For each CT_f and pp'_f with $n = 1$, calculate the difference value dd_f using equation (2.16). Determine dd'_f using equation (2.17). The modified pixel value pp''_f is calculated using equation (2.18).

   (b) Perform the same procedure for $1 \leq f \leq ET_T$. The output of the embedding process are modified pixel values pp'' which forms the final stego image I''.

3.1.2.6.1 Determining the Maximum size of Secret Key and Share 1 and 2

In this section the maximum size of Share 2 is determined. The size of the secret key, share 1 and 2 depends on the number of digits of each of Seed SE, Number of cycles N_c, Height H_0 and Width W_0. To determine the maximum size of the secret key, share 1 and 2, we consider the maximum digits of each SE, N_c, H_0 and W_0. In addition to that, we consider the maximum space and dot(for decimal numbers) characters

Firstly, determine the maximum number of digits for each of the parameters SE, N_c, H_0 and W_0 as follows:

1. To determine the maximum number of digits of SE, consider a gray-scale cover image CI of size 512 x 512. The maximum number of pixels to be embedded with the secret message is $\frac{3}{4} * H_1 * W_1 = \frac{3}{4} * 512 * 512 = 196608$ pixels. According to PRNG in section (2.4), generate prime numbers p and q such that their product $n \approx 196608$. For example, $p = 421$ and $q = 467$ generates $n = 196607$. The seed SE is a random integer in the range 1 and n - 1. Therefore the maximum number of digits of SE is the number of digits of n - 1, SE is a 6 digit number.

2. The Number of cycles, N_c is to 3 significant numbers. Therefore the maximum number of digits of N_c is 4

3. The H_0 and Width W_0 have a maximum of 3 digits each.

Therefore the total maximum number of digits of secret key = 6 + 4 + 3 = 13. Each digit is represented as a 16 x 14 binary matrix as stated in section (3.1.2.5) such that each digit consist of 16 * 14 = 224 pixels. The total number of pixels for all digits is : 13 * 224 = **2912** pixels.
Secondly, determine the maximum number of digits for each of the parameters SE, N_c, H_0 and W_0 as follows:

1. The parameters SE, N_c, H_0 and W_0 are separated by a space character and is represented as a 16 x 8 binary matrix. It consist of 16 * 8 = 128 pixels. There are 3 space characters separating the parameters, therefore the total number of pixels of all spaces: 3 * 128 = **384** pixels.

2. The dot (for decimal) in N_c is represented as 16 x 6 matrix. The total number of pixels of dot character : 16 * 6 = **96** pixels.

The maximum number of pixel of secret key SK_max : 2912 + 384 + 96 = **3392** pixels.
Using (2,2) VCS as discussed in section (2.3) the shares 1 and 2 are generated from the secret key. The scheme has a pixel expansion $m = 2$ ie the maximum number of pixels of each share 1 and share 2 S_max : m * SK_max = 2 * 3392 = **6784** pixels.

3.1.3 Overview of Extraction Phase

The process aim to retrieve the embedded secret message. It requires a stego image, Share 1, share 2 and the Key Trace to successfully recover the secret message. Fig 3.5 shows how the extraction process is conducted.

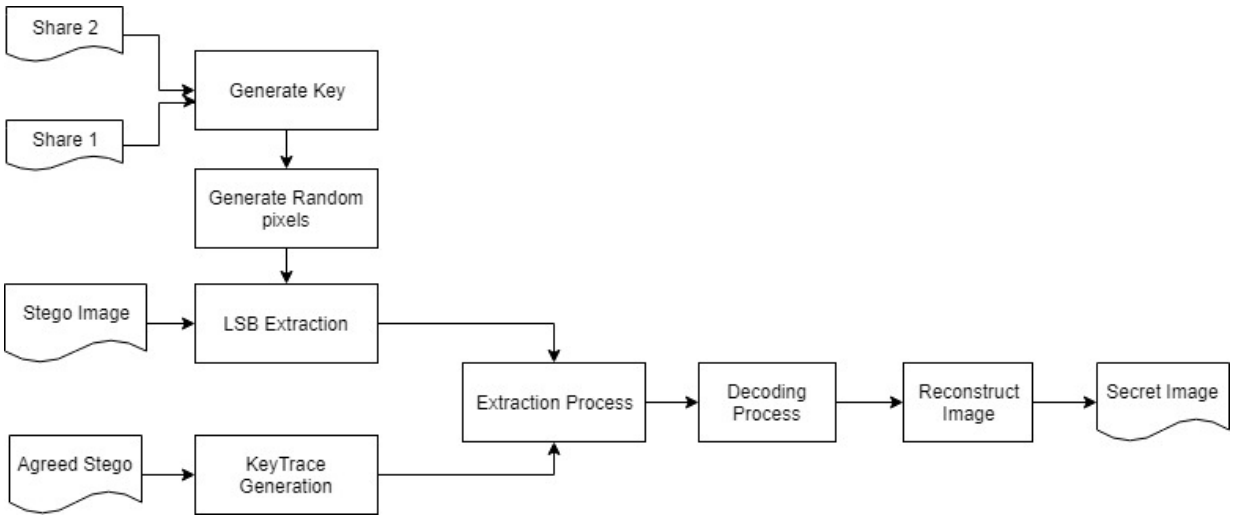


Figure 3.8: Extraction procedure

To begin the extraction process, Share 1 is overlapped/stacked with Share 2 to construct the Key. Once the key is successfully constructed the Seed and the number of cycles becomes visible ie the stacked image will show the seed and Number of Cycles. The pseudo random number generator with the correct seed produces integers that represent positions on the stego to extract the secret message using the key trace. Each of the sub processes of the extraction process is discussed in detail as follows:

3.1.3.1 Generate Key

This section discusses how the key image is regenerated using (2,2) VCS. To extract a secret message successfully, a key should be reconstructed first because it possesses the seed and number of cycles used in embedding process. To reconstruct the key we overlap the extracted Share 2 and Share 1, S1 that the receiver already possess. To extract $S2$ from input Stego image, I'' the following process is performed:

1. Extract the encoded Share 2 bits: To extract $S2$ consider all stego pixels, $sp = sp_q, sp_{q+1}, ..., sp_{H_1 * W_1}$ where q is the marker for starting point of embedded $S2$ bits. Apply equation (13) to every sp to form a sequence of integers $sl = sl_1, sl_2, ..., sl_E$ where n = 1 and $E = (H_1 * W_1) - q$ and $sl_i = 1$ or 0
2. Convert each sl element to binary to form a sequence of bits sl'.
3. Divide sequence st' to nt' blocks $ct = (ct_1, ct_2, ..., ct_{N_1})$ of length N_1 where $nt' = \frac{(H_1 * W_1) - q}{N_1}$
4. Decode each block ct with RM(r_1, m_1) as shown in section (2.2) to form message blocks $tb' = tb'_1, tb'_2, ..., tb'_{k_1}$.

5. Decompose all message blocks tb' to form a single linear block B.

6. Using dimensions of Share 1, n" x m", create a binary matrix/image, S2 from sequence, B. If S2 and S1 are stacked and aligned correctly as described in (4), Seed, SE, number of cycles, N'_c, and dimensions of secret image, H'_0 x W'_0 becomes visible.

3.1.3.2 Generate Random pixels

The purpose of this function is to generate same integer values as in embedding process. The generated integers represent the indices/positions of the pixels embedded with secret bits. The seed, S' initializes Blum Blum Shub to generate the random sequence of integers, $pp = pp_1, pp_2, ..., pp_L$ as discussed in section (3.1.2.3)

3.1.3.3 LSB Extraction

The purpose of this component is to extract the encoded secret bits that were embedded in less or equal to $\frac{3}{4}$ of the cover image. If the encoded bits E_T are greater than $\frac{3}{4} * H_1 * W_1$, then the extracted bits act as a reference to the other bits embedded to on that same pixel. But if E_T are less or equal to $\frac{3}{4} * H_1 * W_1$ then the extracted bits represent the encoded bits E'. The extraction of lsbs is done as follows:

1. Input the randomly generated pixel positions, pp from stego image I''.

2. Retrieve the pixels px' on indices pp in same order. Consider all stego pixels $ps' = ps'_1, ps'_2, ..., ps'_L$. Extract the hidden bits by applying equation (19) to every px' to form a sequence of integers $eb = eb_1, eb_2, ..., eb_L$ where n = 1 and $ez_i = 1$ or 0.

3. Convert each eb' to binary to form a sequence of bits $eb' = eb'_1, eb'_2, ..., eb'_L$

3.1.3.4 Key Trace Extraction

The purpose of this component is to extract bits that were embedded in Agreed image A'. These bits are called the Key trace, K_T. This process is executed if and only if $E_T \geq (\frac{3}{4} * H_1 * W_1)$. In this section, a sequence of Key Trace bits, K_T is extracted from image A' of size H_2 x W_2. To extract the K_T bits the following process was performed:

1. Determine the block size value of ns by: $ns = \frac{|K_T|}{H_2 x W_2}$. If ns is a not an integer, then there are two sets of block sizes,

   (a) Apply equation (19) to the first $\lceil ns \rceil$ pixels of A', to retrieve a sequence of integers,$w' = w'_1, w'_2, ..., w'_{\lceil ns \rceil}$ where $ns = \lceil ns \rceil$.

   (b) if $((H_2 * W_2) - \lceil ns \rceil) \geq 1$,

(c) b). Apply equation (13) to the last $((H_2 * W_2) - \lceil ns \rceil$ pixels of A', to retrieve a sequence of integers,$w'' = w''_1, w''_2, ..., w''_{((H_2*W_2)-\lceil n \rceil)}$ where $ns = ns$.

2. Append sequence w" to w' such that ww = w" + w'.

3. Convert all ww to binary to produce a combined sequence of bits $wb = wb_1, wb_2, ..., wb_{E_T}$ called Key Trace K_T.

3.1.3.5 Extraction process

In this component, the bits that were re-embedded on each pixel are retrieved. This process is only executed if the secret message embedded was greater than the of the cover image. The process inputs Key Trace K_T bits wb and the LSBs extracted from the stego ez' and reference key R and extract the re-embedded bits as follows:

1. Divide the Key Trace bits wb to blocks $kb = (kb_1, kb_2, ..., kb_{\lfloor N_c \rfloor})$ and/or h' blocks $kb' = (kb'_1, kb'_2, ..., kb'_{N_c})$.

   where $(h = \frac{3*H_1*W_1*(N_c - \lfloor N_c \rfloor)}{4}$ and $(h' = \frac{3*H_1*W_1}{4} - \frac{3*H_1*W_1*(N_c - \lfloor N_c \rfloor}{4})$

2. To each created block kb insert wb_l on first index and R on last index of the block such that $kb = (wb_l, kb_1, kb_2, ..., kb_{\lfloor N_c \rfloor}, R)$ and/or $kb' = (wb_l, kb'_1, kb'_2, ..., kb'_{N_c}, R)$, for $1 \leq l \leq E_T$

3. Consider each of the blocks kb and kb' , the sequence of encoded bit block Eb is determined using algorithm 2:

Algorithm 2 Generation of Encoded bit block

Input: Blocks kb or kb', $\lceil N_c \rceil$, Number of blocks h, h'

Output: sequence of Encoded bit blocks EB

$EB \leftarrow [\,]$

$j \leftarrow 1$

$H \leftarrow h + h'$

for $l \leftarrow 1$ *to* H **do**

  while $j < \lceil N_c \rceil$ **do**

    $ebb^l_j \leftarrow kb^l_j \oplus kb^l_{j+1}$.

  Append ebb^l_j to EB

4. Output is a sequence of of bits EB formed after decomposing all message blocks ebb to constituent bits. The sequence, EB is similar to the encoded sequence bits.

3.1.3.6 Decoding process

In this process, the encoded bits are decoded. The decoder is responsible for performing error detection and correction to each of the input bits as stated in section 2.2 .The inputs to this process are the encoded sequence bits, EB, and block length, N_1. The decoding process is performed as follows:

1. Determine if $1 \leq E_T \leq (\frac{3}{4} * H_1 * W_1$ then divide sequence EB to n blocks $cw = (cw_1, cw_2, ..., cw_{N_1})$ of length N_1 where $n'' = \frac{(H_1 * W_1) - q}{N_1}$
2. Decode each block of cw with RM(r_1, m_1) Decoder as shown is section (2.2) and use equation (15) to form message blocks $mb' = mb'_1, mb'_2, ..., mb'_{k_1}$.
3. Output a sequence of decoded bits, $sb = (sb_0, sb_1, ..., sb_Y)$ formed after decomposing all message blocks mb' to constituent bits, such that $Y = 8 * H_0 * W_0$

3.1.3.7 Reconstruct Image

In this section the the decoded bits are reconstructed to form secret image SI. The process inputs the decoded bits, sb and the secret image dimensions, H_0 x W_0 from the generated Key to generate the secret image using the following process:

1. Divide the sequences of bits sb to bytes $mb = mb_1, mb_2, ..., mb_8$.
2. Convert each byte mb to decimal to form a sequence of integer $s' = s'_1, s'_2, ..., s'_{H_0 \text{ x } W_0}$ where $(0 \leq s'_i \leq 255)$.
3. From the sequence of integers, s' ,create a H_0 x W_0 matrix that is the representation of the secret image.

3.1.4 Security Model

In this section, the security protocol of the system is discussed. It demonstrates how the information is exchanged between the sender and recipient. The security model is divided into 3 phases, Registration, Authentication and Stego exchange. The assumption in this model is that the Agreed image is communicated earlier between the sender and the receiver via a private channel. Table 3.1 provides a description of the functions used in the

security model.

Notation	Function	Notes
Generate(.)	Inputs key and generates shares using VC	Outputs 2 shares
Extract(.)	Obtains the LSBs of randomly generated pixels of image	
Construct(.)	Overlaps 2 shares (OR operation)	Regenerates secret key
Hash(.)	Scrambles the input values to data of same size(One way)	
isEqual	Compares 2 Hash values if there are equal	Returns TRUE if Equal otherwise FALSE
RandEmbed(.)	Implements PRNG to generate pixel values to be embedded	

Table 3.1: Functions in Security model

The security model of the proposed method is shown on Fig 3.6

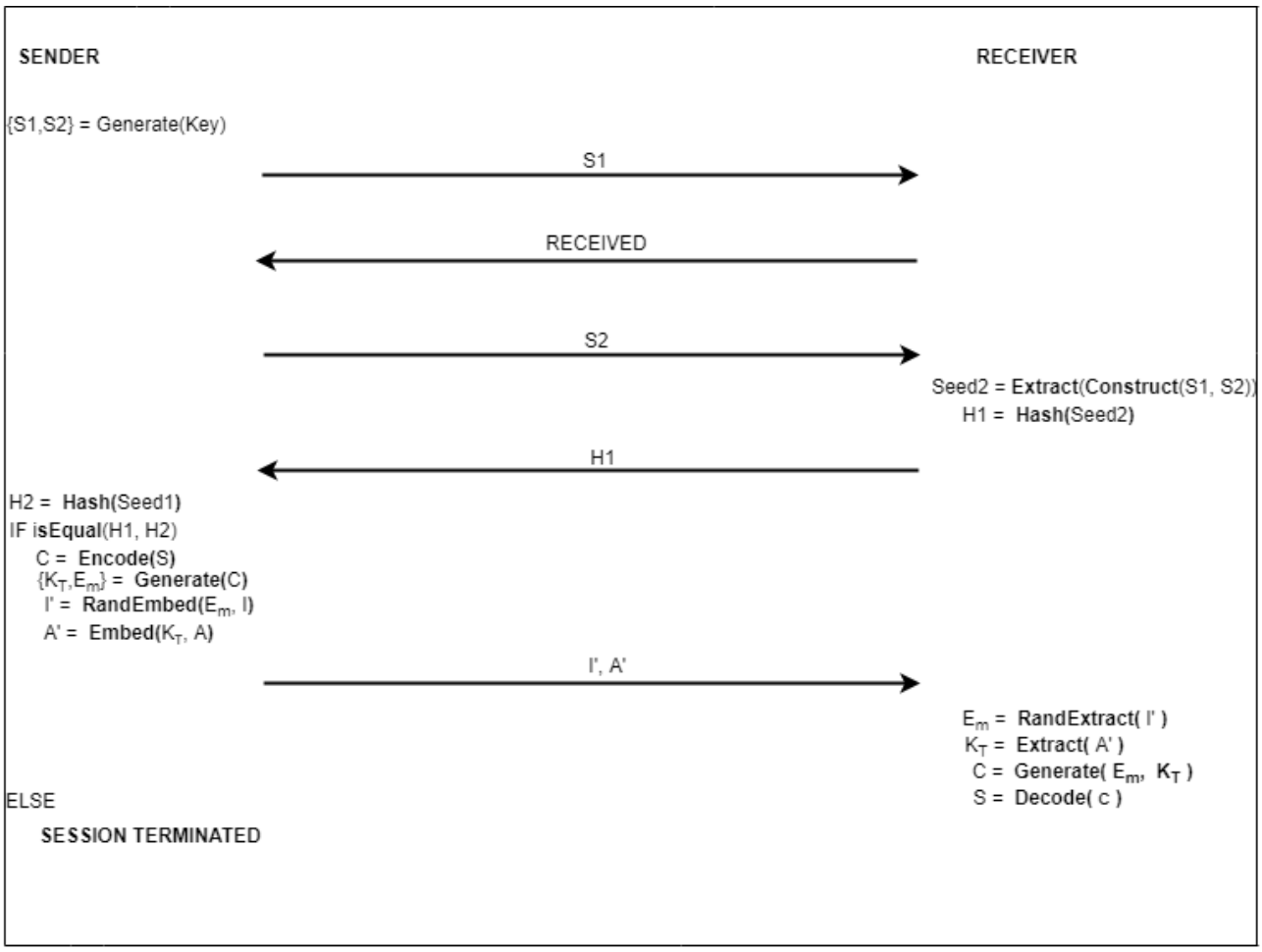


Figure 3.9: Security Model

1. **Registration Phase:** It is the first phase of the security model and is illustrated on Fig 3.9. The purpose of this phase is to enable the sender to generate two shares S1 and S2 from the secret Key based on visual cryptography. The sender will possess S2 and transmit S1 to the receiver via a secure channel.

2. **Authentication Phase:** It is the second phase of the security protocol and is illustrated in Fig 9. Its function is to verify the validity of the receiver. The sender sends S1, and the receiver upon receiving S1, stacks the two shares and try to extract the seed visually. The extracted seed is hashed, H1 and send back to sender. The sender hashes the seed it possesses to form H2, and upon receiving H1, the two hashes are compared. If the two hash values are equal then the receiver is valid and if not then he/she is not.

3. **Stego exchange**: It is the last phase of the security protocol and is illustrated on Fig 10. The purpose of this phase is to communicate a secret image between two users. It involves all the steps as described in sections 3.1.2 and 3.1.3

3.2 Experiment Scenario

To evaluate the performance of the proposed method compared to other methods an experiment is conducted based on four main quantitative measures. Fig 3.7 illustrates the experiment design suggested for the proposed method.

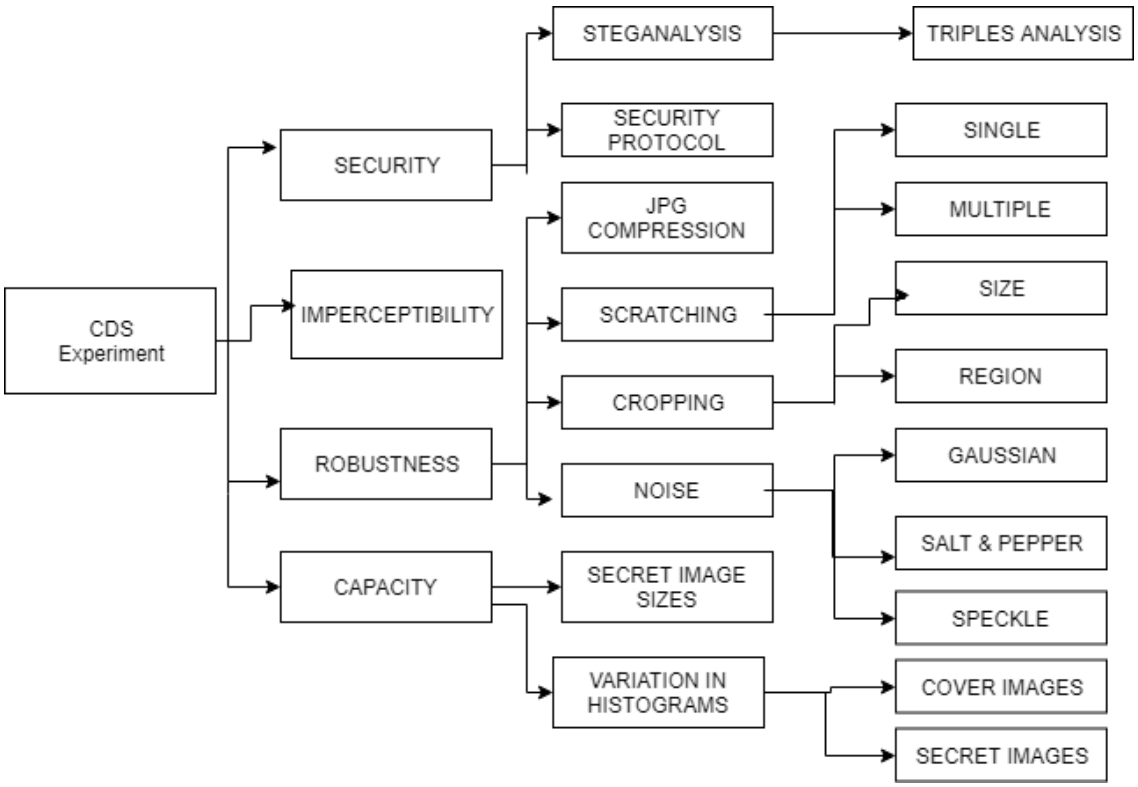


Figure 3.10: Experiment Scenario

3.2.1 Capacity

In this section, the embedding capacity of both the proposed and method is examined. The ratio of the number of secret bits embedded in a Cover image to the number of pixels of the Cover image is evaluated.The embedding capacity, EC is then determined by the following equation:

$$EC = \frac{|S|}{H * W} \tag{3.13}$$

Where—S— is the number of secrets bits embedded, H and W is the height and length of the image respectively. The resulting payload is expressed as a percentage. To successfully evaluate the performance of the proposed method in terms of the capacity, the following two main scenarios are considered:

1. Variation in pixel values: Cover images and secret images of various image histograms are considered. We consider forty (40) grayscale Cover images of dimensions 512 x

512 and forty grayscale secret images of various sizes. For each cover image with a unique tonal, all secret images of various tonal distributions are embedded.

2. Secret image sizes: For Each secret image, consider 33 different sizes of that secret image. Each was embedded in different covers and the resulting capacity was recorded.

The results for each group are tabulated and the PSNR value is determined and compared with A. M. Molaei's Method.

3.2.2 Imperceptibility

A steganography scheme is imperceptible if a human eye cannot distinguish between the cover and the stego image. The concern in this section is the visual quality of the resulting Stego image. Groups of secret images with same image size and various tonal distributions are formed. Each secret image in a given group is embedded in the cover using both proposed and A. M. Molaei's Method.
The Imperceptibility is then measured by the Mean Square Error (MSE) and Peak to Signal Noise Ratio (PSNR).
The MSE is calculated by:

$$MSE = \frac{1}{H * W} \sum_{i=1}^{H} \sum_{j=1}^{W} (I_{ij} - I'_{ij})^2 \tag{3.14}$$

Where I is the Cover image and I is the stego image. H and W are the height and the width f the cover image or stego image.

$$PSNR = 10log_{10} \frac{255^2}{MSE} \tag{3.15}$$

The results for each group are tabulated and the PSNR value is determined and compared with A. M. Molaei's Method.

3.2.3 Robustness

In this section we examine the proposed methods ability to overcome various attacks. This enables the receiver to retrieve the secret message which has been destroyed in case of destruction of the stego image intentionally or unintentionally. The quantitative measure to be determined is the PSNR of the recovered secret image. It is determined using equation (3.15) and (3.15):
Reed Muller codes RM (1, 4) has been implemented in this method because of its higher

error correction capability. This is shown as follows:
Since r = 1 and m = 4. Therefore message size k is calculated as follows:

$$k = \sum_{i=0}^{r} \binom{m}{i} = \binom{4}{0} + \binom{4}{1} = 5bits$$

And a code word of size N calculated as follows:

$$N = 2^m = 2^4 = 16bits$$

$$ECC = \frac{\text{number of correctable bits}}{\text{total bits}} = \frac{3}{16} = 18.75\%$$

After each embedding scenario has been conducted as described in the section 6.1.1, the result stego image is exposed under the following attacks

1. Noise Attack: Each resulting stego image is exposed to the following noise attacks:

   (a) **Gaussian Noise:** It was named after Carl Friedrich Gauss [14].Gaussian Noise is a statistical noise having a probability density function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution. In other words, the values that the noise can take on are Gaussian-distributed. For Gaussian Noise, two variations in variance were considered in experiments ie Gaussian noise: mean 0 variance 0.01 and Gaussian noise: mean 0 variance 0.1. Fig 3.8 shows an example of a stego image under Gaussian noise attack.

(a) Original Image: Lena (b) Gaussian: Variance 0.01 (c) Gaussian: Variance 0.1

Figure 3.11

   (b) **Salt and Pepper:** It is a form of noise also seen on images. It is also known as impulse noise. This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels [8]. Salt and Pepper noise comes in various forms of different densities. Two variations in density were considered in experiments ie Salt and Pepper

noise: density 0.05 and Salt and Pepper noise: density 0.5. Fig 3.9 shows an example of a stego image under Salt and Pepper noise attack.

(a) Original Image: Lena (b) Salt and Pepper: density 0.05 (c) Salt and Pepper: density 0.5

Figure 3.12

(c) **Speckle noise:** Speckle noise is the noise that arises due to the effect of environmental conditions on the imaging sensor during image acquisition. Speckle noise is mostly detected in case of medical images, active Radar images and Synthetic Aperture Radar (SAR) images [11]. Two variations in variance were considered in experiments ie Speckle noise: variance 0.04 and Speckle noise: variance 0.4. Fig 3.9 shows an example of a stego image under Speckle noise attack.

(a) Original Image: Lena (b) Speckle: Variance 0.04 (c) Speckle: Variance 0.4

Figure 3.13

After the stego is exposed to these Noise attacks, the extraction process retrieves the secret image and performs error correction. The PSNR of the retrieved secret image is determined and tabulated. The PSNR of all the retrieved secret images is determined and compared with A. M. Molaei's Method.

2. Cropping Attack: Stego images of same ranges in PSNR values are grouped and

exposed to cropping attacks. Cropping attacks was done based on different cropping ratios and regions. Each stego image within a specific group is exposed to similar cropping ratios and cropping regions as with others within the same group. The stego images will undergo the extraction process to retrieve the secret image by conducting error correction. For each set or group, the PSNR value of the retrieved secret image is determined and tabulated and compared with the A. M. Molaei's Method.

(a) Original Image: Lena (b) Cropped region:80-0-340-512 (c) Cropped region:80-45-345-345

Figure 3.14

3. Scratching Attack: All stego images within the same groups as mentioned in 6.1.2.2 are exposed to scratching attacks. Scratching attacks comes in two different forms ie difference in length, same width and difference in width, same length. After each form of scratching attack on the stego image is conducted error correction is used to retrieve the secret image. The PSNR of the retrieved secret image is determined, tabulated and compared with previous methods.

(a) Original Image: Lena (b) Single Scratch (c) Multiple Scratches

Figure 3.15

4. JPEG Compression: It stands for Joint Photographic Experts Group. JPEG is a commonly used method of lossy compression for digital images, particularly for those

images produced by digital photography. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.[2] JPEG compression is used in a number of image file formats. JPEG is the most common image format used by digital cameras and most common format for storing and transmitting photographic images on the Web.

To conduct experiment, all stego images embedded with different sizes of the same secret message were exposed to JPG attacks. After exposing the stegos to JPEG compression attack, extraction of the secret image is done by implementing error correction to recover the secret image. The PSNR of the recovered secret image is determined, tabulated and compared with A. M. Molaei's Methods.

(a) Original Image: Lena

(b) JPEG compressed Lena

3.2.4 Security

This section discusses about the security of the code based steganography scheme. A Steganalysis technique called Triples proposed by Andrew D. Ker [9] was used to determine the security of the scheme and is described as follows:

3.2.4.1 Triples Analysis

In this section a brief description of the Triples analysis is provided. It was implemented to determine how the proposed and A.M.Molaeis method perform under steganalysis attack. It is a detector used for simple LSB steganography in digital images. It uses the structural and combinatorial properties of the LSB embedding method to detect and estimate the length of the hidden messages. Triples has proved to be a more powerful and sensitive detector than other structural detectors such as RSA, SPA etc. It utilizes the effects of LSB changes on arbitrary groups of samples ie 3-tuples of pixel groups. Triples assumes that the Cover image is fixed and a random hidden message is embedded. The Triples analysis process is described as follows:

Suppose the hidden message has length $2pN$, where $0 \leq p \leq 0.5$ is unknown to the

detector, is embedded using LSB embedding of a random selection of samples independent of the content of the cover or hidden message [9] . A trace set $C_{m,n}$ is created in which all pixel samples $s_1, , s_N$ are drawn, where n, m ¿ 0 and N is number samples. The probability of transition from one trace subset to another is $p^i(1p)^(3i)$, where i is the length of the shortest path between them. The trace set and subsets are constructed as follows

$$C_{m_1,\ldots,m_{g-1},n_{g-1}} = (s_1, , s_g) \in T|[\frac{s_{i+1}}{2}] = [\frac{s_i}{2}] + m_i \text{ and } [\frac{s_{i+2}}{2}] = [\frac{s_{i+1}}{2}] + n_i \text{for each} 1 \leq i \leq g \tag{3.16}$$

$$E_{m_1,\ldots,m_{g-1},n_{g-1}} = (s_1, , s_g) \in T|s_{i+1} = s_i + m_i \text{ and } s_{i+2} = s_{i+1} + n_i \text{ with } m_i \text{ even} \tag{3.17}$$

$$O_{m_1,\ldots,m_{g-1},n_{g-1}} = (s_1, , s_g) \in T|s_{i+1} = s_i + m_i \text{ and } s_{i+2} = s_{i+1} + n_i \text{ with } m_i \text{ odd} \tag{3.18}$$

When a single sample has the LSB altered a tuple transitions to either of the following trace subsets $E_{2m,2n}$, $O_{2m,2n}$, $E_{2m+1,2n-1}$, $O_{2m,2n-1}$, $E_{2m,2n+1}$, $O_{2m-1,2n+1}$, $E_{2m+1,2n}$, and $O_{2m-1,2n}$. Given a stego image, consider each trace set $C_{m,n}$ in turn and count the trace subsets to make a vector x'.

$$x'' = T_3^{-1}\ x' \tag{3.19}$$

Then we can hypothesize a value of p and form estimates for the sizes of the trace subsets of the cover image using the transition matrix, T_3. The parity symmetry is checked if its true for both covers and also stego images when both m and n are even or equal. Considering one case of parity symmetry, $e_{2m+1,2n+1} = o_{2m+1,2n+1}$. To use the generalized framework to make an estimate of p, we compute error terms for each m and n, $\epsilon_{m,n} = e'_{2m+1,2n+1} - o'_{2m+1,2n+1}$. Then we find the value of p which minimizes the sum-square of the errors. Firstly, using the transition matrix, the number of errors is determined by

$$\epsilon_{m,n} = \frac{1}{8}(d_0(1+q)^3+d_1(1-q)(1+q)^2+d_2(1-q)^2(1+q)+d_3(1-q)^3) = \frac{1}{8}((d_0+d_1+d_2+d_3)+q(3d_0+d_1-d_2-3d_3$$

(3.20)

Where d_i are the differences of various combinations of $e_{2m+i,2n+i}$ and $o_{2m+1,2n+1}$ subsets Find the value of q to minimize $S(q) = \sum_{m,n} \epsilon^2_{m,n}$. We have

$$S(q) = \frac{1}{64}\sum_{m,n} c_0^2+q(2c_0c_1)+q^2(2c_0c_2+c_1^2)+q^3(2c_0c_3+2c_1c_2)+q^4(c_2^2+2c_1c_3)+q^5(2c_2c_3)+q^6(c_3^2) \tag{3.21}$$

Differentiating the above equation (10) and solving for q will give up to 5 roots for q. All roots inside the range (10, 10/11) are discarded because they will give wrong estimates of p outside (0.05, 0.55)). Substitute the remaining roots back into (10) to determine the

location of the minimum q'. Finally, the estimate of p, $p' = \frac{1}{2}(1 - \frac{1}{q})$.

3.2.4.2 Determine Proportion from the Triples Results

Proportion was used to compare the performance of A.M.Molaeis method and proposed method under steganalysis. Using the results of True p value (T_p) and Estimated p value (E_p) for each secret message size recorded and proportion P was calculated as follows:

$$P = \frac{E_p}{T_p} \tag{3.22}$$

A larger value of P means that the estimated p value is closer to the true p value. It means that the probability of attacker to predict the true p value is high. A smaller P value means that there is a bigger difference between the estimated p value from the true value so the attacker cannot easily predict the true p value. A high value of proportion means that the there is a smaller difference between the Estimated p value and the True p value. That means the Triples estimated the p value almost accurately.

3.3 Population/Sampling

The data set for this experiment were taken from the steganography standard test collection on the internet. It consist of three sets, each with 30 images described as follows :

1. Cover images: 512 x 512 gray-scale images of various image histogram structure .
2. Secret images: gray-scale images of various image histogram structure and 33 different sizes.
3. Agreed images: 600 x 600 gray-scale images of various image histogram structure and 33 different sizes. The size of the Agreed images was suggested after an investigation was done to find out the most common image size of used on social media such as Facebook, Twitter and Instagram.

.

APPENDIX shows a sample drawn from the three sets of Cover images and Secret Images respectively.

3.4 Instrumentation and Data Collection

Various gray-scale images of different forms and sizes were implemented to investigate the impact of improving embedding capacity in Code Based steganography using Multiple embedding. The forms and variations of image histograms had an impact on the embedding capacity of the secret messages.

3.5 Tools for Data Analysis

The output for Capacity and Imperceptibility experiments are stego images. The analysis was conducted as a comparison of embedding capacity (%) and the PSNR between the proposed and A.M.Molaei's Method. For robustness, the retrieved secret image was considered as the output. The analysis was conducted as a comparison of the PSNR of the retrieved secret image between the proposed and A.M.Molaei's Method.

CHAPTER 4

PRESENTATION, ANALYSIS AND INTERPRETATION OF DATA

This chapter describes the results of the experiments conducted to observe the performance of the Code Based Steganography using multiple embedding. The chapter consist of three sections mainly, the data representation, data analysis, and summary of findings.

4.1 Presentation of Data

The data presentation section comprises of three sub sections, Capacity evaluation, Imperceptibility evaluation and Robustness evaluation. The embedding capacity evaluation discusses the results of the experiment by observing the size of the secret image (bits) embedded into the cover image using both the A. M. Molaeis and proposed methods. The Imperceptibility evaluation presents the results of the experiment by observing the ability to distinguish between stego and cover image using human eye. This parameter is also measured by the Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) and results are presented. The Robustness evaluation discusses the result of the experiment by observing the ability to recover the secret image from an attacked stego image using both the A. M. Molaeis and proposed method.

4.1.1 The Experiment result for Capacity Evaluation

The objective of this part of the experiment was to determine the embedding capacity of the cover images using both the A.M.Molaei's and the proposed methods.
The experiment was conducted to 30 different cover images of size (512x512) but of various histograms as shown in APPENDIX. The proposed and A.M.Molaei's methods were both used to embed 20 different types of secret images, of different sizes in each cover image during conducting the experiment. Table 4.1 summaries the results of this experiment.

Table 4.1: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
26.png	9.png	480	0.37	0.59	75.577	62.636
		960	0.73	1.17	72.246	62.242
		1920	1.46	2.34	69.408	61.548

Table 4.1: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		3840	2.93	4.69	66.435	60.573
		7680	5.86	9.38	63.464	59.342
		15360	11.72	18.75	60.467	57.379
		26880	20.51	32.81	58.012	55.736
		30720	23.44	37.5	57.419	54.406
		46080	35.16	56.25	55.688	53.579
		61440	46.88	75	54.406	52.395
		65536	50	80	54.102	51.997
		76800	58.59	93.75	53.344	51.985
		92160	70.31	112.5	52.669	52.203
		107520	82.03	131.25	52.015	51.986
		122880	93.75	150	51.386	52.070
		131072	100	160	51.126	51.988
		138240	105.47	168.75	50.667	51.980
		153600	117.19	187.5	49.746	52.047
		168960	128.91	206.25	49.149	52.071
		184320	140.63	225	48.537	52.067
		196608	150	240	48.133	52.088
		215040		262.5	47.729	52.079
		230400		281.25	47.045	51.982
		245760		300	46.751	52.054
		261120		318.75		51.989
		276480		337.5		52.022
		291840		356.25		51.978
		307200		375		52.382
		322560		393.75		51.993
		337920		412.5		52.019
		353280		431.25		51.983
		368640		450		52.077
	15.png	480	0.37	0.59	75.614	62.285
		960	0.73	1.17	72.289	62.173
		1920	1.46	2.34	69.419	61.508
		3840	2.93	4.69	66.482	60.454
		7680	5.86	9.38	63.478	58.983

Table 4.1: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		15360	11.72	18.75	60.446	57.454
		26880	20.51	32.81	58.014	55.563
		30720	23.44	37.5	57.392	54.407
		46080	35.16	56.25	55.789	53.437
		61440	46.88	75	54.435	52.377
		65536	50	80	54.120	51.982
		76800	58.59	93.75	53.386	51.962
		92160	70.31	112.5	52.698	52.105
		107520	82.03	131.25	52.018	51.982
		122880	93.75	150	51.405	52.035
		131072	100	160	51.147	51.982
		138240	105.47	168.75	50.702	52.027
		153600	117.19	187.5	49.789	52.066
		168960	128.91	206.25	49.170	52.028
		184320	140.63	225	48.549	52.040
		196608	150	240	48.113	52.016
		215040		262.5	47.842	51.899
		230400		281.25	47.049	52.045
		245760		300	46.746	51.989
		261120		318.75		52.012
		276480		337.5		51.983
		291840		356.25		52.039
		307200		375		51.982
		322560		393.75		52.016
		337920		412.5		51.973
		353280		431.25		52.034
		368640		450		52.077
17.png	18.png	480	0.37	0.59	75.432	62.463
		960	0.73	1.17	72.246	62.176
		1920	1.46	2.34	69.408	61.497
		3840	2.93	4.69	66.502	60.423
		7680	5.86	9.38	63.468	58.283
		15360	11.72	18.75	60.462	57.387
		26880	20.51	32.81	58.012	55.538

Table 4.1: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		30720	23.44	37.5	57.441	54.418
		46080	35.16	56.25	55.687	53.406
		61440	46.88	75	54.461	52.391
		65536	50	80	54.118	51.976
		76800	58.59	93.75	53.382	51.979
		92160	70.31	112.5	52.670	52.195
		107520	82.03	131.25	52.014	51.972
		122880	93.75	150	51.419	52.037
		131072	100	160	51.118	51.969
		138240	105.47	168.75	50.693	51.972
		153600	117.19	187.5	49.757	52.036
		168960	128.91	206.25	49.150	52.040
		184320	140.63	225	48.538	52.035
		196608	150	240	48.123	52.040
		215040		262.5	47.838	52.066
		230400		281.25	47.041	51.975
		245760		300	46.752	52.041
		261120		318.75		51.968
		276480		337.5		52.312
		291840		356.25		51.976
		307200		375		52.039
		322560		393.75		51.975
		337920		412.5		52.014
		353280		431.25		51.039
		368640		450		52.039

Table 4.1 shows that the proposed method achieves a greater embedding capacity accompanied with a good visual quality for all cover images as compared to the A. M. Molaeis method. The proposed method achieves an embedding capacity of 450% with PSNR values ranging from [51.7 52.4] dB while the maximum embedding capacity for the A. M. Molaeis Method is 150% with PSNR values varying from [47.5 48.2] dB.

Table 4.2 summaries the results of this experiment in accordance to execution time of

embedding process and the extraction process for each experiment.

SECRET SIZE (bits)	Secret Share Size (Bits)	Execution Time with Secret Sharing				Execution Time without Secret Sharing			
		EMBEDDING TIME (ms)		EXTRACTION TIME (ms)		EMBEDDING TIME (ms)		EXTRACTION TIME (ms)	
		PREVIOUS METHOD	PROPOSED METHOD	PREVIOUS METHOD	PROPOSED METHOD	PREVIOUS METHOD	PROPOSE D METHOD	PREVIOUS METHOD	PROPOSED METHOD
480	5632	8	644	63	363	8	401	63	154
960	5632	46	716	117	456	46	333	117	237
1920	5632	67	741	239	603	67	388	239	416
3840	5632	132	812	511	906	132	499	511	696
7680	5632	261	977	953	1606	261	682	953	1416
15360	5632	514	1407	1923	2929	514	1084	1923	2514
26880	5632	861	2034	3759	4963	861	1711	3759	3963
30720	5184	1008	2242	4252	5618	1008	1904	4252	4798
46080	5632	1488	3046	6029	8366	1488	2751	6029	7786
61440	4544	2029	4076	7873	10890	2029	3609	7873	9320
65536	6080	2164	4457	8135	11759	2164	3933	8135	10320
76800	6080	2517	5255	9090	13698	2517	4567	9090	11863
92160	5632	3011	6123	11755	16706	3011	5531	11755	13213
107520	6080	3507	6981	13471	19366	3507	6472	13471	15642
122880	4992	3949	7580	14938	22045	3949	7021	14938	17232
131072	6080	4300	8189	15597	23492	4300	8153	15597	19236
138240	6080	4585	9027	17009	24875	4585	8332	17009	20562
153600	5632	4760	9629	20123	27531	4760	9091	20123	22346
168960	6080	4532	10511	23295	32093	4532	9933	23295	25533
184320	4992	4648	11065	26503	32975	4648	10160	26503	28462
196608	5632	4748	11681	30801	34064	4748	10460	30801	32199
215040	5632		13699		38188		10862		34580
230400	6080		12994		39256		10888		36789
245760	4992		12117		42644		10736		39168
261120	6080		13035		44335		11572		41589
276480	5632		12337		46494		11962		43639
291840	6080		12401		49091		11918		46410
307200	4992		12935		52494		11446		48994
322560	6080		13029		54169		12357		51378
337920	5632		14001		56603		12651		53527
353280	6080		14369		60892		13021		55998
368640	5632		15683		63482		13289		58849

Figure 4.1: Execution Time Comparison between Proposed Method and A.M.Molaei's Method

Fig 4.1 above shows that the Proposed Method requires an average of 13472 milliseconds to achieve its maximum embedding capacity (450%) while A. M. Molaeis Method takes an average of 4748 milliseconds to achieve its maximum embedding capacity (150%). Although the proposed method achieves high capacity, the Previous Method runs at lower execution

time for both the embedding and extraction processes. A. M. Molaeis Method is 2 times faster than the proposed method when embedding a secret image of any size. This occurred because of the secret sharing involved in proposed method which is not there in A. M. Molaeis method. A. M. Molaeis method takes less time in extraction process than the Proposed Methods extraction process for any hidden message size within the range [480 196608] bits.

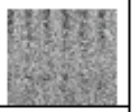
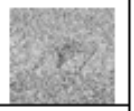
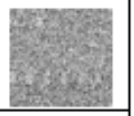
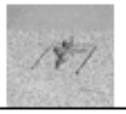
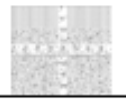
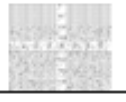
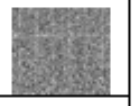
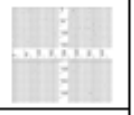
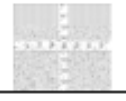
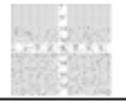
4.1.2 The Experiment result for Robustness Evaluation

The objective of the conducted experiment was to assess the robustness of both the proposed and A. M. Molaeis Method under different kind of attacks. The types of image attacks introduced are Noise, Cropping, Scratching and jpg Compression Attacks. An experiment was performed on each of these attacks and results were recorded. In the following sections an abstract of each experiment is shown and the complete experiment results are presented in APPENDIX.

Experiment Results for Noise Attacks

In this experiment a set of 20 different images were used as the cover images and another set of 20 different images as secret images. Each cover image was embedded with 33 different sizes of each secret image. The noise attacks conducted in this experiment were, Gaussian Noise, Salt and Pepper Noise, and Speckle Noise. This experiment was conducted to observe the capability of message recovery for each method. The results of the experiment is shown in the Fige 4.2

Secret Image & Size	Stego Image	Noise Attacks	Proposed Method			Previous Method		
			Attacked Stego PSNR(dB)	PSNR Secret Image(dB)	Recovered Secret Image	Attacked Stego PSNR(dB)	PSNR Secret Image(dB)	Recovered Secret Image
 15360 bits	Stego 20	Gaussian Noise: Mean 0 & Variance 0.01	20.218	9.359		20.213	9.336	
		Gaussian Noise: Mean 0 & Variance 0.1	11.464	9.713		11.466	9.815	
		Salt & Pepper: Density 0.05	18.353	100		18.271	28.390	
		Salt & Pepper: Density 0.5	8.242	12.176		8.230	11.751	
		Speckle Noise: variance 0.04	21.350	9.324		21.357	9.654	
		Speckle Noise: variance 0.4	11.879	9.409		11.878	10.337	
 15360 bits	Stego 13	Gaussian Noise: Mean 0 & Variance 0.01	20.221	9.131		20.088	8.964	
		Gaussian Noise: Mean 0 & Variance 0.1	11.457	8.829		11.378	7.995	
		Salt & Pepper: Density 0.05	18.232	100		18.105	29.193	
		Salt & Pepper: Density 0.5	8.268	12.610		8.242	11.796	
		Speckle Noise: variance 0.04	21.327	9.252		21.064	9.150	
		Speckle Noise: variance 0.4	11.876	9.220		11.064	8.792	

 196608 bits	Stego 11	Gaussian Noise: Mean 0 & Variance 0.01	20.204	17.333		20.019	9.439	
		Gaussian Noise: Mean 0 & Variance 0.1	11.453	17.176		11.354	9.276	
		Salt & Pepper: Density 0.05	18.284	100		18.095	30.082	
		Salt & Pepper: Density 0.5	8. 240	28.124		8. 216	11.505	
		Speckle Noise: variance 0.04	21.357	18.147		20.957	9.953	
		Speckle Noise: variance 0.4	11.912	17.690		11.620	9.842	
 196608 bits	Stego 10	Gaussian Noise: Mean 0 & Variance 0.01	20.208	16.449		20.030	6.562	
		Gaussian Noise: Mean 0 & Variance 0.1	11.454	16.459		11.349	6.534	
		Salt & Pepper: Density 0.05	18.206	100		18.108	31.901	
		Salt & Pepper: Density 0.5	8. 247	28.453		8. 223	9.883	
		Speckle Noise: variance 0.04	21.323	18.677		20.921	8.292	
		Speckle Noise: variance 0.4	11.872	17.368		11.349	7.101	

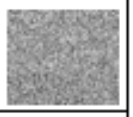
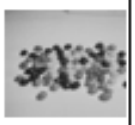
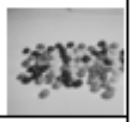
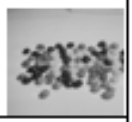
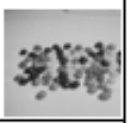
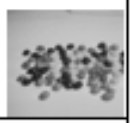
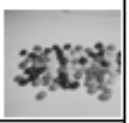
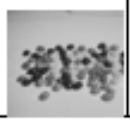
 196608 Bits	Stego 18	Gaussian Noise: Mean 0 & Variance 0.01	20.213	12.748		20.217	7.699	
		Gaussian Noise: Mean 0 & Variance 0.1	11.419	12.514		11.435	7.733	
		Salt & Pepper: Density 0.05	18.271	100		18.177	26.053	
		Salt & Pepper: Density 0.5	8. 246	22.710		8. 858	9.064	
		Speckle Noise: variance 0.04	21.330	12.469		21.305	7.653	
		Speckle Noise: variance 0.4	11.884	12.528		11.884	7.755	
 368640 bits	Any Stego	Gaussian Noise: Mean 0 & Variance 0.01	20.220	100		All Secret can successfully be recovered		
		Gaussian Noise: Mean 0 & Variance 0.1	11.454	100				
		Salt & Pepper: Density 0.05	18.166	100				
		Salt & Pepper: Density 0.5	8. 240	100				
		Speckle Noise: variance 0.04	21.335	100				
		Speckle Noise: variance 0.4	11.872	100				

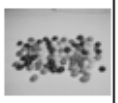
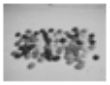
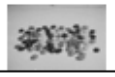
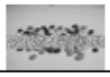
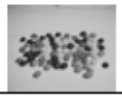
Figure 4.2: Experiment Result of robustness against noise attack

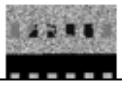
Based on Table Fig 4.2, it is shown that for any secret image size, the proposed method

retrieves better secret image quality compared to A. M. Molaeis Method. The Proposed Method successfully recovered the secret image from stego images under Salt and Pepper with Variance of 0.04 for smaller secret message sizes 1536 bits, 3072 bits, 6144 bits, 12288 bits, 24576 bits while under other attacks the secret image embedded could not be recovered. Considering the same smaller secret message sizes mentioned earlier, the proposed method cannot recover successfully the secret image. The Proposed method starts to recover successfully the whole secret under all noise attacks for larger image of sizes 1179648bits and above. Table 4.3 also shows that the A. M. Molaeis Method cannot recover successfully the secret images of any size under all Noise attacks

Experiment Results for Cropping Attacks

In this experiment a set of 20 different images were used as the cover images and another set of 20 different images as secret images. Each cover image was embedded with 33 different sizes of each secret image. Each stego image was cropped according to specific regions of interest therefore each stego image has different cropping ratios compare to others. The experiment was then performed to recover the hidden secret image in the attacked stego image. The results of the experiment is shown in the Fig 4.3

Secret Image	Stego Image	Secret Message Size (bits)	Cropping Area				Proposed Method		A. M. Molaei's Method	
			X point	Y point	Height (pixel)	Width (pixel)	PSNR Secret Image(dB)	Recovered Secret Image	PSNR Secret Image(dB)	Recovered Secret Image
	Stego 3	15360	0	0	512	360	19.898		100	
		131072	0	0	512	360	33.670		14.797	
		196608	0	0	512	360	39.357		15.863	
		368640	0	0	512	360	100			
	Stego 4	61440	150	200	220	170	8.284		8.220	
			70	80	422	422	8.376		8.443	
		196608	150	200	220	170	15.078		5.898	
			70	80	422	422	14.834		6.754	
		368640	150	200	220	170	100			
			70	80	422	422	100			

	Stego 7	15360	120	200	250	140	7.467		7.392	
			20	40	435	452	6.576		6.850	
		196608	120	200	250	140	16.031		3.947	
			20	40	435	452	16.009		4.998	
		368640	120	200	250	140	100			
			20	40	435	452	100			
	Stego 8	61440	140	190	190	155	6.540		6.546	
			20	40	435	452	6.685		6.236	
		196608	140	190	190	155	14.507		3.752	
			20	40	435	452	14.458		5.248	
		368640	140	190	190	155	100			
			20	40	435	452	100			

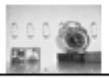
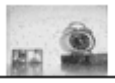
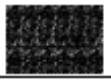
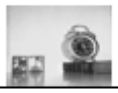
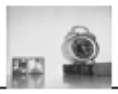
	Stego 12	30720	0	0	440	312	12.840		17.678	
			0	0	442	420	19.823		17.989	
			0	310	512	120	8.750		8.579	
		196608	0	0	440	312	27.453		12.147	
			0	0	442	420	39.455		13.708	
			0	310	512	120	14.722		6.636	
		368640	0	0	440	312	100			
			0	0	442	420	100			
			0	310	512	120	100			
	Stego 13	26880	0	30	312	512	13.721		11.793	
		131072	0	30	312	512	25.691		12.826	
		196608	0	30	312	512	28.569		9.700	
		368640	0	30	312	512	100			

Figure 4.3: Experiment Result of robustness against cropping attack

Based on Fig 4.3, the proposed method always retrieves a secret image of better PSNR values compared to the A. M. Molaeis Method. The retrieved secret image PSNR value for proposed method increases as the size of the secret image embedded increases. However A. M. Molaeis Method can successfully recover the secret message on Cropping ratio [0 0 512 360] when a smaller secret message is embedded (15360 bits). The Proposed method was not able to recover successfully the secret image for all smaller secret sizes.

Experiment Results for Scratching Attacks

In this experiment a set of 20 different images were used as the cover images and another set of 20 different images as secret images. Each cover image was embedded with 33 different sizes of each secret image. Scratching attacks were applied to each stego image on specific region of interest. There are two kinds of scratching applied to each stego, single and multiple scratching. The experiment was then conducted to observe whether the secret image could be recovered or not after scratching was applied. The results of the experiment is shown in the Fig 4.4

Secret Image	Stego Image	Secret Size(bit)	Scratches	Proposed Method		A. M. Molaei's Method	
				PSNR Secret Image(dB)	Recovered Secret Image	PSNR Secret Image(dB)	Recovered Secret Image
	Stego 4	480	Single	100		100	
			Multiple	6.167		11.012	
		26880	Single	100		100	
			Multiple	3.997		3.284	
		131072	Single	100		21.260	
			Multiple	17.717		3.390	
		196608	Single	100		21.543	
			Multiple	20.688		3.222	
		307200	Single	100			
			Multiple	31.692			
		368640	Single	100			
			Multiple	100			

	Stego 1	480	Single	100		100	
			Multiple	5.138		6.647	
		26880	Single	100		100	
			Multiple	4.944		7.133	
		131072	Single	100		19.869	
			Multiple	15.256		6.850	
		196608	Single	59.571		20.482	
			Multiple	14.375		6.468	
		307200	Single	100			
			Multiple	24.038			
		368640	Single	100			
			Multiple	100			

Figure 4.4: Experiment Result of robustness against scratching attacks

Based on Table 4.4 the Proposed Method can recover all secret image under Single Scratching attacks while A. M. Molaeis Method can only successfully recover the secret message on Singular Scratching attacks when a smaller secret message are embedded (26880 bits). As for Multiple Scratching attacks, only large secret sizes (368640 bits) can be successfully recovered using the Proposed Method while A. M. Molaeis Method cannot. Under Multiple Scratching, the recovered secret image PSNR value under proposed method increases

as the size of the secret image embedded increases while A. M. Molaeis Method cannot recover secret image of any size under Multiple Scratching attacks.

Experiment Results for JPEG Compression Attacks

In this experiment a set of 20 different images were used as the cover images and another set of 20 different images as secret images. Each cover image was embedded with 33 different sizes of each secret image. JPEG compression was applied to each stego. The experiment was then conducted to observe whether the secret image could be recovered or not after JPEG compression was applied. The results of the experiment is shown in the Fig 4.5

Secret Image	Stego Image	Secret Message Size(bits)	Proposed Method			A. M. Molaei's Method		
			PSNR of Destroyed Stego(dB)	PSNR Secret Image(dB)	Recovered Secret Image	PSNR of Destroyed Stego(dB)	PSNR Secret Image(dB)	Recovered Secret Image
	Stego 2	3840	43.436	9.590		43.609	8.870	
		30720	43.398	9.198		44.580	8.895	
		92160	43.388	11.820		44.279	8.909	
		131072	43.387	15.520		43.507	8.960	
		196608	43.391	18.565		44.598	8.779	
		261120	43.396	27.622				
		353280	43.389	41.750				
		384000	43.393	100				

	Stego 5	3840	42.714	8.738		42.321	8.875	
		30720	42.702	8.806		42.340	8.503	
		92160	42.699	11.176		42.351	8.175	
		131072	42.704	13.113		42.321	8.166	
		196608	42.696	15.382		42.340	8.157	
		261120	42.691	22.423				
		353280	42.695	31.513				
		384000	42.685	100				
	Stego 4	3840	43.161	8.533		42.381	8.515	
		30720	43.133	8.452		42.430	8.388	
		92160	43.121	11.752		42.413	8.166	
		131072	43.131	14.972		42.374	8.170	
		196608	43.132	16.848		42.429	8.211	
		261120	43.124	25.901				
		353280	43.121	32.813				
		384000	43.126	100				

Figure 4.5: Experiment Result of robustness against JPEG compression attacks

Based on Fig 4.5 the Proposed Method can recover secret image with higher PSNR values than the A. M. Molaeis Method. The recovered secret images by the proposed method are quite visible while secret images recovered by A. M. Molaeis Method are scrambled and not visible. Only large secret sizes ($\geq$ 368640 bits) can be successfully recovered using the Proposed Method while A. M. Molaeis Method cannot.

Experiment Results for Steganalysis

In this experiment a set of 20 different images were used as the cover images and another set of 20 different images as secret images. Each cover image was embedded with 33 different sizes of each secret image. Each stego image formed at each experiment was subjected to steganalysis attacks to detect and estimate the embedded secret message size. The proposed steganalysis method estimated the value of p (fraction of the secret message hidden in a given cover image). The results of the experiment is shown in the Table 4.7

Cover Image	Secret	Secret Size(bits)	Molaei's Method		Proposed Method	
			True P value	Estimated P value	True P value	Estimated P value
4.png	7.png	480	0.001831	0.001228	0.037305	0.03321
		960	0.003662	0.003414	0.040234	0.03825
		1920	0.007324	0.006838	0.046094	0.04272
		3840	0.014648	0.011239	0.057813	0.05321
		7680	0.029297	0.028534	0.08125	0.07913
		15360	0.058594	0.053251	0.128125	0.10145
		26880	0.102539	0.091232	0.198438	0.168421
		30720	0.117188	0.10044	0.221875	0.199469
		46080	0.175781	0.143821	0.315625	0.285651
		61440	0.234375	0.20115	0.402734	0.391522
		65536	0.25	0.22045	0.412109	0.409244
		76800	0.292969	0.25623	0.412109	0.410012
		92160	0.351563	0.30526	0.409375	0.399421
		107520	0.410156	0.39123	0.412109	0.408686
		122880	0.46875	0.42631	0.405469	0.39806
		131072	0.5	0.48012	0.412109	0.40969
		138240	0.527344	0.50312	0.412109	0.40870
		153600	0.585938	0.53551	0.409375	0.39380
		168960	0.644531	0.61121	0.412109	0.40187
		184320	0.703125	0.68254	0.405469	0.39467
		196608	0.75	0.7209	0.409375	0.395543
		215040			0.409375	0.39835
		230400			0.412109	0.40856
		245760			0.405469	0.39892
		261120			0.412109	0.40813
		276480			0.409375	0.39848
		291840			0.412109	0.40859
		307200			0.405469	0.39382
		322560			0.412109	0.40189
		337920			0.409375	0.39518
		353280			0.412109	0.40695
		368640			0.405469	0.39146

3.png	17.png	480	0.001831	0.00998	0.037305	0.03060
		960	0.003662	0.002831	0.040234	0.03564
		1920	0.007324	0.006130	0.046094	0.03992
		3840	0.014648	0.010012	0.057813	0.05008
		7680	0.029297	0.026141	0.08125	0.07616
		15360	0.058594	0.05150	0.128125	0.98891
		26880	0.102539	0.09514	0.198438	0.141890
		30720	0.117188	0.09804	0.221875	0.182105
		46080	0.175781	0.10014	0.315625	0.257130
		61440	0.234375	0.20019	0.402734	0.372998
		65536	0.25	0.20014	0.412109	0.400631
		76800	0.292969	0.21355	0.412109	0.402713
		92160	0.351563	0.29052	0.409375	0.378310
		107520	0.410156	0.37891	0.412109	0.400351
		122880	0.46875	0.40029	0.405469	0.38099
		131072	0.5	0.43023	0.412109	0.39310
		138240	0.527344	0.48106	0.412109	0.39110
		153600	0.585938	0.51891	0.409375	0.37660
		168960	0.644531	0.60741	0.412109	0.38113
		184320	0.703125	0.66112	0.405469	0.37408
		196608	0.75	0.69890	0.409375	0.375750
		215040			0.409375	0.37681
		230400			0.412109	0.38541
		245760			0.405469	0.37217
		261120			0.412109	0.38723
		276480			0.409375	0.37896
		291840			0.412109	0.38820
		307200			0.405469	0.38301
		322560			0.412109	0.38715
		337920			0.409375	0.37902
		353280			0.412109	0.37828
		368640			0.405469	0.37891

Figure 4.6: Experiment Result of Steganalysis

Based on Table 4.7 above, the hidden secret message is detectable under steganalysis for both proposed and A. M. Molaeis method. However, the estimated p value was more closely accurate for smaller values [480 15360 bits] for previous method than the proposed

method. The estimation accuracy of the proposed method decreased rapidly for proposed method previous method when larger secrets (¿15360 bits) are embedded.

4.2 Data Analysis

The performance of the proposed and A. M. Molaeis method are discussed in this section. This is achieved by detailed analysis of Embedding Capacity, Robustness Imperceptibility and Steganalysis of the two methods.

4.2.1 Analysis of Capacity between proposed and previous Methods

The embedding capacity of the A. M. Molaeis method was lower than the proposed method for all various cover images of size 512 x 512 as proposed.

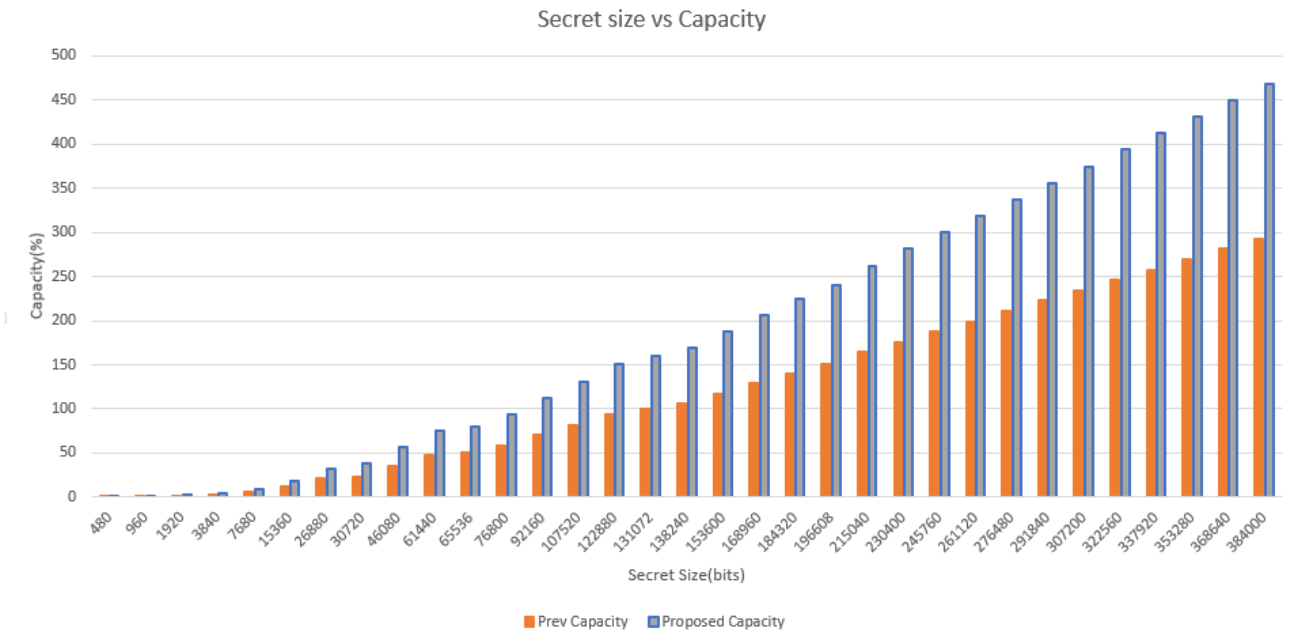


Figure 4.7: The experiment result of embedding capacity comparison between A. M. Molaeis and the Proposed Method.

According to the experiment results represented by Fig 4.7 it is shown that for each secret message size the proposed method achieves a greater embedding capacity than the previous method. This occurred because of the error correction code and the multiple embedding method adopted in the proposed method. Reed Muller code (1, 4) used in proposed method encodes a message block of size 5 into a code-word of size 16 therefore encoding expansion is $16/5 = 3.2$. The previous method implemented Reed Muller (1,3)

and Reed Muller (2,5) codes which encodes a message block of size 4 or 8 into a code-word of size 8 or 16 and its encoding expansion is 8/4 = 2. Clearly the expansion of the proposed method is higher than previous method .For any given secret message size the proposed method encodes a message to code-word of greater size than the previous method by an expansion factor of 3.2/2 = 1.6. This means that the capacity of the A. M. Molaeis method is always lower than the proposed method. The proposed method proposed multiple embedding method which only embeds message on first LSB of some pixels unlike the previous method which is limited to embedding only on the first and second LSBs of every pixels of the cover. In multiple embedding method the LSB of each pixel of the cover image is embedded more than once. This is achievable by using a Stego Key. The embedding capacity is increased because the LSB of each pixel can be embedded more than 1 message bit while the Stego key keeps track of the changes in the LSB of the pixel. The maximum of 450% embedding capacity guarantees optimal time, space complexity and maximum error correction capability as shown in Fig 4.2

As shown in the table 4.1 the A. M. Molaeis method has a maximum capacity of 150% which is (150/450) = 1/3 of the capacity of the proposed method. A. M. Molaeis method implements an error correction code with a less expansion rate (2) and embedded the first and second LSB of every pixel of the cover. The embedding capacity of A. M. Molaeis method greatly depends on the PSNR of the resulting stego image. Its maximum embedding capacity of 150% is achieved by embedding both the first and second LSB of each pixel of the cover image. The capacity is very low because as embedding proceeds from the first LSB to second LSB, the image quality degrades rapidly. Embedding in the second LSBs increases the Mean Square Error between the cover and the resulting stego image. This decreases the overall image quality (PSNR value) of the final stego to (48dB). However, the proposed method used of the pixels of each cover image and re embedded the remaining secret bits on the same pixels that were embedded before.

This guarantees that a maximum of ($\frac{3}{4}$ *512*512) = 196608 LSBs of the cover pixels will only be changed after embedding any secret message size. These small changes leads to a low Mean Square Error therefore multiple embedding method achieves high capacity accompanied with a good visual quality (¿51dB). While the previous embeds both the first and second LSBs of each covers pixels. This leads to many changes in the overall cover and changes in the second LSBs of a pixel leads to an increased Mean Square Error thereby decreasing the visual quality of the stego (¡ 48dB).

Majority of the cover images used to conduct the experiment achieved a high visual quality with a PSNR value greater than 52 for a maximum capacity of 450%. These cover images are 1, 2, 4, 6, 8, 9, 11, 12, 13, 14, 16, 17, 18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30. The remaining cover images 3, 5, 7, 10, 15, 19, 20 achieves a slightly lower PSNR value of 51 dB which is still greater than the recommended value for a standard good quality image (50dB). Among all the covers used images of similar tonal distribution as 22 achieves the

highest quality reaching up to 61 dB for a maximum capacity of 400%.

4.2.2 Analysis of Execution Time

Although the Proposed method can achieve high embedding capacity (450%) as compared to the A. M. Molaeis method (150%), the proposed method outperforms the proposed method on the execution time for both the embedding and the extraction process. This is shown in Fig 4.8 and Fig 4.9

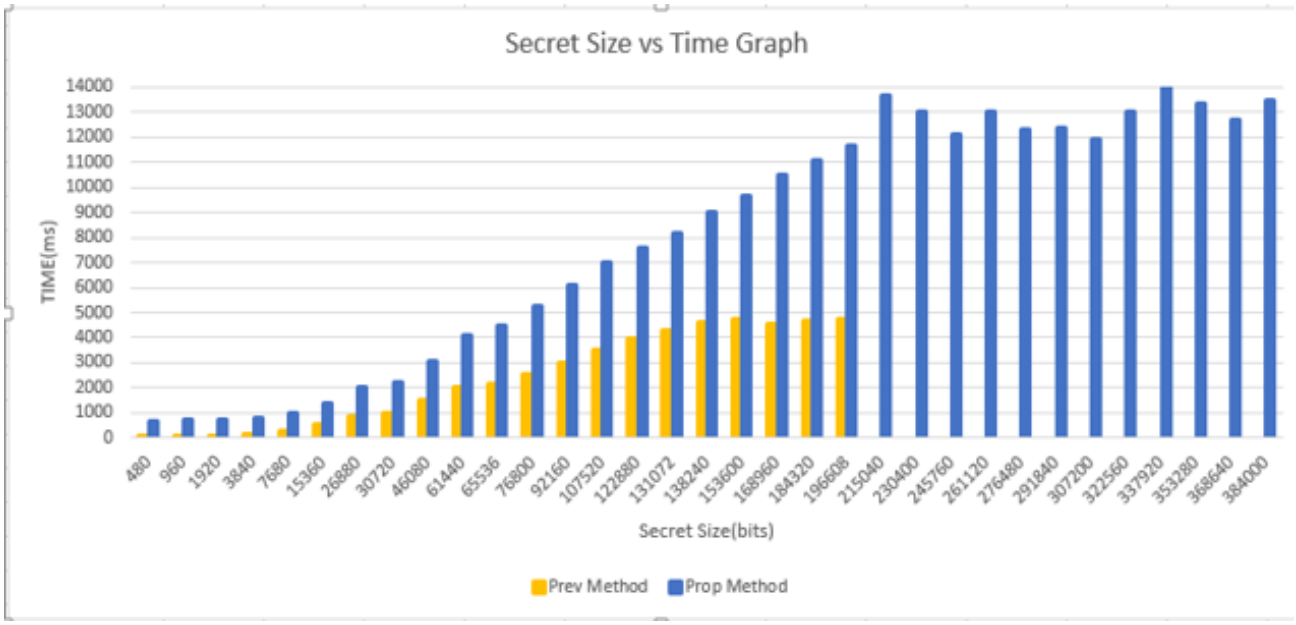


Figure 4.8: The experiment result of embedding time comparison between the A. M. Molaeis method and the Proposed Method.

Figure 4.8 shows that for every secret message size to be embedded the proposed method takes more time as compared to the previous method. For most message sizes except 107520 and 122880 bits, the execution time of the proposed method is twice or more than the proposed method. This occurred because of extra processes implemented in the proposed method although they help to achieve high embedding capacity, good visual quality and increases the robustness. The proposed method implements a PRNG with complexity O (log M) where M is the product of the two primes, M = pq. The other reason for increasing the execution time is that there is time required to embed Key trace in the Agreed Image as described in the design process.

In Figure 4.9, the diagram shows that for every secret message size embedded in any cover image the proposed method takes more extraction time as compared to the previous. The same reasons stated on the execution time for the embedding process applies for extraction

process as well.

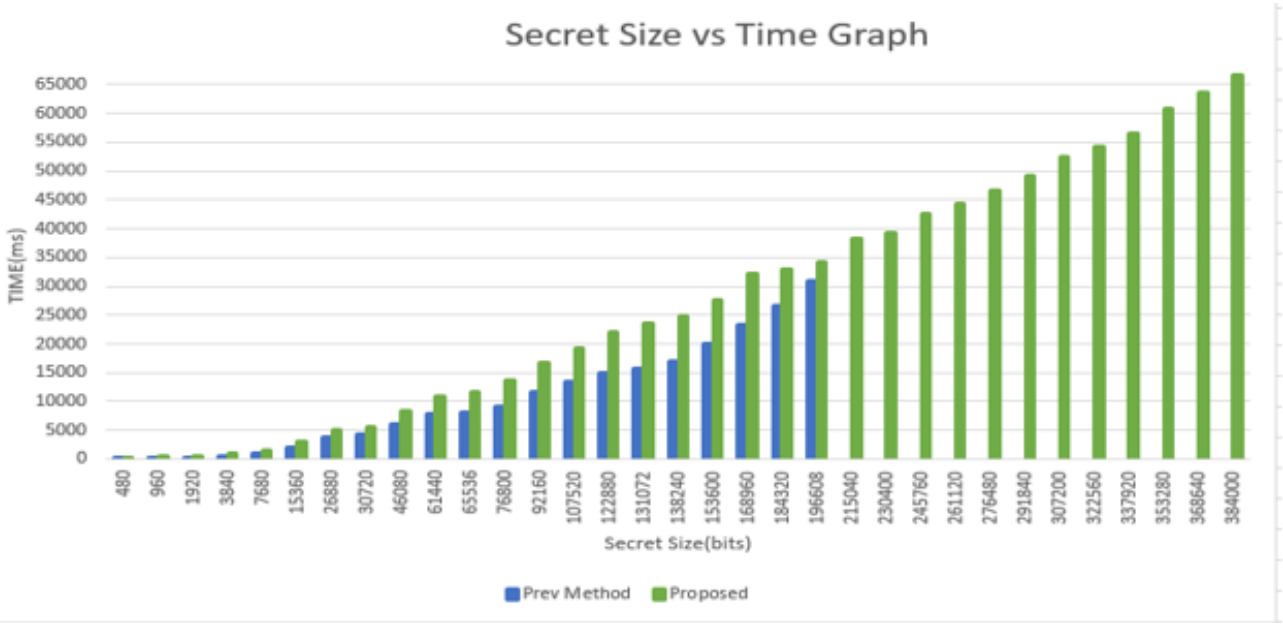


Figure 4.9: The experiment result of extraction time comparison between the A. M. Molaeis method and the Proposed Method.

4.2.3 Analysis of Robustness between proposed and A. M. Molaeis method

A set of 20 different images was used as the cover images and another set of 20 different images as secret images. Each cover image was embedded with 33 different sizes of each secret image and three kinds of attacks were introduced which include Noise, Cropping, Scratching and JPG compression. The analysis of each attack is discussed below

4.2.3.1 Analysis of Robustness against Noise Attacks between Proposed and A. M. Molaeis method

Table 4.3 shows the experiment result of robustness against Noise attacks, 20 different cover images and 20 different secret images were used. It shows that the proposed method has better error correction capability as compared to the Previous Method. This is because of the different error correction codes implemented in proposed and previous method.
By definition, Reed Muller (1,4) code implemented in proposed method encodes a message of size k is calculated as follows:

$$k = \sum_{i=0}^{r} \binom{m}{i} = \binom{4}{0} + \binom{4}{1} = 5bits$$

And a code word of size N calculated as follows:

$$N = 2^m = 2^4 = 16bits$$

The maximum number of error bits this code can correct t is calculated as follows:

$$t = \left\lfloor \frac{2^{m-r} - 1}{2} \right\rfloor = \left\lfloor \frac{2^{4-1} - 1}{2} \right\rfloor = 3bits$$

The Error correction capability ECC is calculated as follows:

$$ECC = \frac{\text{number of correctable bits}}{\text{total bits}} = \frac{3}{16} = 18.75\%$$

The A. M. Molaeis method implemented 2 types of codes, Reed Muller (1, 3) codes for first LSB embedding and Reed Muller (2, 5) codes for second LSB embedding. According to (1), each has message sizes k, 4 and 16 bits. These are encoded to codewords of sizes N, 8 and 32 bits according to (2). A. M. Molaeis method can correct a maximum of $t_1 = \left\lfloor \frac{2^{m-r}-1}{2} \right\rfloor = \left\lfloor \frac{2^{3-1}-1}{2} \right\rfloor = 1bit$ and $t_2 = \left\lfloor \frac{2^{m-r}-1}{2} \right\rfloor = \left\lfloor \frac{2^{5-2}-1}{2} \right\rfloor = 3bit$ for Reed Muller (1, 3) and Reed Muller (2, 5) respectively. The error correction capacity of Reed Muller (1, 3) and Reed Muller (2, 5) are 12.5% and 9.38% respectively according to (4). Therefore the proposed method suggested an error correction code that outperforms the one suggested in the A. M. Molaeis method.

The Secret image size and contrast are two main factors that influence the recovering of secret images. The Proposed Method out performs the A. M. Molaeis method, it successfully recovers the secret image from stego images under Salt and Pepper: Variance 0.04 for smaller secret message sizes 1536 bits, 3072 bits, 6144 bits, 12288 bits, 24576 bits and secret images (10.png, 13.png, 11.png etc) of high contrast (10, 13, 11). This occured because Salt and Pepper has a smaller variance hence makes fewer distortions on the stego such that each retrieved codeword has number of errors less than 3 which are easily correctable by proposed Reed Muller code. While the A. M. Molaeis method has poor quality recovered images because for these small hidden messages the implemented code is Reed Muller (1,3) which can correct only 1 error on each codeword block yet the retrieved code words have $\geq t$ and are not correctable.

Under Gaussian and Speckle attacks, these small secret sizes cannot be retrieved (not visible) even at any level of contrast for both A. M. Molaei and Proposed Methods. This occurred because of the large number changes on pixel values of the stego which results in retrieved codewords with a number of errors greater than t errors. However, for the secret images with low contrast (15.png, 14.png etc) and these secret sizes, the proposed method has better PSNR values than the previous method because it has a (18.75% - 12.5%) = 6.25% better error correction capability than that of the previous as shown in the definitions above. In addition, the proposed method embedded messages randomly on the

stego which increases the chances of consecutive bits to be not distorted. This contributes largely to reduce burst errors which are more common in previous method and are hardly correctable by linear block codes.

Considering secret images of size ($\geq$ 107520 bits), the Proposed method out performs the Previous method for all Noise attacks. A. M. Molaeis method can fully recover the secret message (100dB) of any level of contrast under Salt and Pepper: Variance 0.04 while the Previous cannot (¡ 40dB). However both Methods cannot recover fully the secret image under Salt and Pepper: Variance 0.4, but the proposed method retrieves secrets with better visibility of ($\geq$ 15.5dB) for low contrast secret and ($\geq$ 19.8 dB) for high contrast secret images. A. M. Molaeis method has range [9.8 11.5 dB] for both high and low contrast images and the retrieved secrets are not visible. Under Gaussian Noises and Speckle Noise, the proposed method retrieves secrets with better visibility of ($\geq$ 11.5dB) for low contrast secret and ($\geq$ 12.6 dB) for high contrast secret images.

A. M. Molaeis method recovers non visible, noisy secrets within [6.4 9.8 dB] PSNR for both high and low contrast images. This occured because for each block of the message to be re-embedded on each LSB of the pixel, only 1 bit of the block is embedded and the remaining block is represented as Key Trace and this Key Trace cannot be corrupted by any kind of attack. This boost the error correction capability of the code since fewer bits will be corrupted as the message size increases. The minimum distance between each retrieved codeword and the code reduces also as the message size increases. And this will cause t to approach $\frac{2^(m-r)-1}{2}$ rapidly, where t is number of correctable errors in a code, r and m are positive integers $0 \leq r \leq m$. The reasons why Key Trace cannot be corrupted will be added in the security reduction document.

Considering secret images of size ($\geq$ 368640 bits), the Proposed method performs best for all Noise attacks. It can successfully retrieve any secret image ($\geq$ 368640 bits) of any level of contrast. In this case the code is able to correct all corrupted blocks. At this size the number of errors in each block of code word is exactly $t = \frac{2^(m-r)-1}{2}$ or less ($\geq$ 3) and is therefore correctable successfully.

As the size of the secret image increases, the error correction capability of the proposed error correction code increases thereby increasing the retrieved secret image quality, PSNR. This is applicable to secret images of any contrast. The Key Trace embedded in the Agreed image is free from any Noise attacks and helps increasing the error correction capability during extraction process. The quality of the retrieved secret image increases as the size of secret image increases. This is because as the secret data size increases the key trace data also increases according to the equation below.

$$K_T = \frac{S * N}{k} - \frac{3}{4} * H * W \tag{4.1}$$

where S is the secret message size in bits and H and W are height and width of cover

image.

4.2.3.2 Analysis of Robustness against Cropping Attacks between Proposed and A. M. Molaeis method

Table 4.4 shows the experiment result of robustness against Cropping attacks, 20 different cover images and 20 different secret images were used. The Stego images formed were subject to cropping attacks on different regions of interest. It shows that the proposed method has better error correction capability as compared to the A. M. Molaeis method. The experiment results shows that smaller secret images ($\leq$ 30 720 bits) with low high contrast and under larger cropping regions, the previous method can achieve better PSNR values than proposed method. This is because the Previous Method embedded data serially on the cover. Therefore smaller secret data have higher probability of not being included in the cropping region depending on the image. So if the cropping region excludes or a smaller part of the embedded region then secret retrieval is more efficient. As for the proposed method, the secret data is embedded randomly across the whole image. Therefore any cropping region will definitely include the embedded image pixels. If the cropping region is larger, then the message retrieval is less efficient than the A. M. Molaeis method in most cases. The A. M. Molaeis method can also fully recover (100dB) secret images for Stegos 14, 15, 16, 9, 10, 3 etc.

However, if the cropping region is reduced, the proposed method achieves better results than the previous method. Although it cannot recover the secret images fully but the secret image is quite visible compared to the previous method. When secret images of high contrast are extracted, the proposed method scored higher PSNR values for smaller cropping regions than the A. M. Molaeis method. This is because the number of errors in most retrieved codewords caused by the cropping approaches or t 3. There are greater than the minimum number of correctable errors for each codeword, therefore cannot be fully correctable.

Considering medium sized secrets within [46 080 353 280] bits, the Proposed method retrieves better quality secret images of any contrast and cropping region size when compared to the A. M. Molaeis method. The retrieved secret images are not 100% good quality but are much visible compared to the ones retrieved in the previous method. The previous method struggles to retrieve low contrast secret images, most of which are highly noisy, not visible and contains large black regions. As the size of the secret image increases, the error correction capability of the proposed error correction code increases thereby increasing the retrieved secret image quality, PSNR. This is applicable to secret images of any contrast. The Key Trace embedded in the Agreed image is free from any cropping attacks and helps increasing the error correction capability during extraction process.

Considering large secret sizes within [368640 - 384000] bits, the Proposed Method successfully recovers the hidden secret image under and any cropping region size attack. As the

secret image size increases the retrieved secret image quality drastically increases under any of the applied cropping region attacks. The proposed method generally outperformed the A. M. Molaeis method when slightly larger images are embedded. This is because of the PRNG adopted to randomly embed and extract the secret images to avoid burst error which are more common in the A. M. Molaeis method. Furthermore, the Reed Muller code (1, 4) used by the proposed method, has a high error by 6.25% as shown in section 1.2.2.1 greater than Reed Muller (1, 3) and (2, 5) used in the A. M. Molaeis method. Most large secret images retrieved by the previous method contains large black regions on the secret image and this is caused by burst errors.

4.2.3.3 Analysis of Robustness against Scratching Attacks between Proposed and A. M. Molaeis method

Table 4.5 shows the experiment result of robustness against Cropping attacks, 20 different cover images and 20 different secret images were used. The Stego images formed were subject to scratching attacks (Single and Multiple) on different regions of interest as shown in APPENDIX 5. It shows that the proposed method has better error correction capability as compared to the A. M. Molaeis method.

Based on Table 4.5, proposed method can successfully recover the secret message on Single Scratching attacks for all high contrast secret image of any size. This because a single scratch alters few pixels on the stego and since the secret data was embedded randomly, therefore the errors will not belong to a single code word or consecutive codewords. It is because of this that burst errors are not encountered in the proposed method. The number of errors in each retrieved codeword is t $\leq$ 3 which is correctable. The A. M. Molaeis method can successfully recover any secret image of size $\leq$ 3 840) bits under Single Scratching attacks because for smaller secret size the probability that the single scratch covers the embedded region is low. Even if the smaller area of the embedded region is scratched the alterations caused are small and correctable.

It can also retrieve fully some secret images of size ($\leq$ 26 880) bits eg Stego 4 and Stego 1 depending on the position of the Single Scratch. The other remaining secret sizes cannot be fully retrieved by the A. M. Molaeis method but ranges between [18 22.5] dB. This is because as the embedded area increases, the number of alterations caused by scratching also increases. The possibility of retrieving code words with increased errors and burst errors increases. Therefore leading to retrieving code words with $t \geq \frac{2^{m-r}-1}{2}$ which are not correctable. This is quite visible although the retrieved secret messages contains some scratches also.

Considering Multiple scratch attacks, the Proposed Method cannot successfully retrieve all smaller secret image ($\leq$ 353 280) bits. This is because all of the secret data is embedded on the cover and scratching alters a larger part if not all of the data thereby increasing the value of t. And if t is greater than 3(maximum number of correctable errors of Reed

Muller code(1,4) then the code words cannot be corrected successfully. The quality of the retrieved secret image increases as the size of secret image increases. This is because as the secret data size increases the key trace data also increases.
Equation (1) clearly show that if size of the secret data increases the Key Trace also increases. The size of the secret data that is embedded on the cover image is $\leq \frac{3}{4} * H * W$. It is important to note that the key trace data is free from any attacks as mentioned earlier. Therefore as the KeyTrace increases the number of errors per each retrieved codeword decreases also approaching t or less. Thereby increasing the overall error correction capability as the secret size increases.
However, large secret sizes ($\geq$ 368640 bits) can be retrieved by the Proposed Method. This is because the size of Key trace bits in each retrieved codeword increased and $t \leq 3$ And if $t \leq 3$ then the codeword is correctable as shown in section 1.2.2.1. The Previous method cannot retrieve any secret images under Multiple Scratches because the scratches caused many alterations such that for every retrieved code word the number of errors $t_1 \geq 1$, $t_2 \geq 3$ and are not correctable. It also causes large burst errors which are not correctable with the suggested error correction code. The retrieved secret images have very low quality [3 4.5] dB and are all black in color.
The information bits (Key Trace) embedded in the Agreed image plays a key role on the performance of the error correction code. The error correction capability also increases as the embedded secret image size increases. This ensures an increase in retrieved secret image quality since the key trace is prone to attacks.

4.2.3.4 Analysis of Robustness against JPG compression Attacks between Proposed and A. M. Molaeis method

Table 4.6 shows the experiment result of robustness against jpg compression attacks, 20 different cover images and 20 different secret images were used. The Stego images formed were subject to jpg compression as shown in APPENDIX. It shows that the proposed methods error correction capability out performs the Previous Method.
The table clearly shows that the previous method cannot successfully recover any secret message size of any level of contrast under JPG compression attacks. The retrieved secret images have PSNR values within [8.5 9.9] dB. These values ensures no visibility of the retrieved image. This is caused by the large changes in pixel values caused by JPG compression which affect the embedded pixel values. They result in many alterations of the stego pixels such that for every retrieved code word the number of errors $t_1 \geq 1$, $t_2 \geq 3$ and are not correctable. It also causes burst errors which are not correctable with the suggested Reed Muller codes.
On the other hand, the proposed method can only retrieve successfully large secret images of sizes ($\geq$ 368640 bits). This is because the size of Key trace bits in each retrieved codeword increased and $t \leq 3$. And if $t \leq 3$ then the codeword is correctable. However,

the proposed method performs poorly when smaller secret images ($\leq$ 30720) bits are embedded, the retrieved secret images lack any form of visibility and PSNR ranges [8.5 - 10] dB. This is because the value of Key trace for smaller secret data is zero which means error correction will rely only on the data retrieved from the stego. And this data is highly altered or distorted by jpeg compression, therefore the number of errors $t \geq 3$ and is not correctable. The retrieved secret images of size ($\geq$ 92160) bits are quite visible and the noise on the images decreases as the size of the secret image increases. Therefore, the retrieved secret image PSNR value under proposed method increases as the size of the secret image embedded increases.

4.2.3.5 Steganalysis for Proposed and A. M. Molaeis method

In this section, steganalysis is discussed. Triples has been implemented to determine and compare how the proposed and A. M. Molaeis method perform under steganalysis. Triples was able to detect the presence of a hidden message in both the proposed and A. M. Molaeis method for every Cover image used. The margin between the estimated and true p value increases as the length of the hidden secret message increases as shown from Fig 4.10 in the range [30720 -] and [26880 -] bits.
Proportion was used to compare the performance of A. M. Molaeis method and proposed method under steganalysis. Using the results of True p value T_p and Estimated p value E_p for each secret message size recorded on Table 4.6, proportion P was calculated as follows:

$$P = \frac{E_p}{T_p} \tag{4.2}$$

A larger value of P means that the estimated p value is closer to the true p value, so the probability that the attacker can predict the true value is increased. A smaller P value means that there is a bigger difference between the estimated p value from the true value so the attacker cannot easily predict the true and steganalysis is not successful.

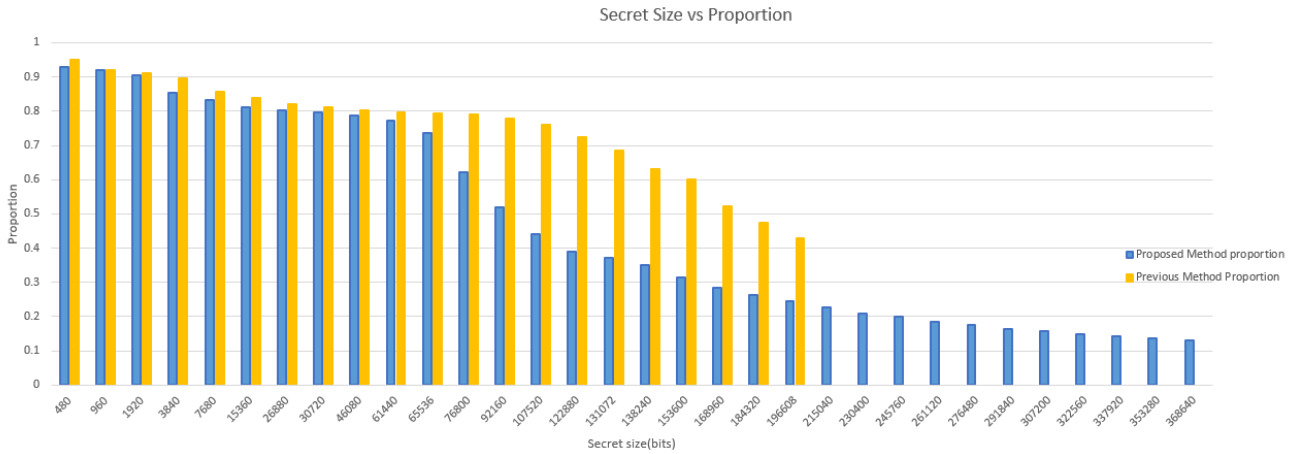


Figure 4.10: A comparison of Steganalysis results for A. M. Molaeis method and proposed method based on proportion.

Fig 4.10 above shows that for smaller secret message size [0 30720 bits], the proportion values of both proposed and A. M. Molaeis method for these secret sizes is $\geq$ 0.8. A high value of proportion means that the there is a smaller difference between the Estimated p value and the True p value. That means the Triples estimated the p value almost accurately. However it is also important to notice that for these smaller secret sizes, the proportion values of the proposed method is always smaller than the A. M. Molaeis methodd. This is caused by the high expansion rate of 3.2 in Reed Muller (1,4) code used in proposed method to encode the secret message as compared to the expansion rate of 2 in Reed Muller(1,3) code used in A. M. Molaeis method. So for any given secret message size, the proposed method results in a larger secret size to be embedded than the A. M. Molaeis method. And Triples method has been proved to be highly sensitive to smaller secret sizes than large ones, hence the proposed method performs better than the A. M. Molaeis method.

The proportion values of the proposed method decreased rapidly for secret sizes [61440 368640 bits] while the proportion values of the A. M. Molaeis method decreased steadily. The proportion values of the proposed method decreased rapidly because of the Multiple embedding method adopted. Multiple embedding re substitutes different secret bits on LSB of pixels that have already been embedded, such that at the end of the embedding process each pixel has either 1 change or same LSB as of the original/initial LSB. Triples method can only detect the message length based on the last embedding cycle. Due to this, there is an increase in True p value while the estimated value is 0.3. It is because of this that the difference between the proportion values of proposed and A. M. Molaeis method is very large. Therefore the proposed method is less detectable under steganalysis than the previous method.
The adopted triples method is very efficient and effective under LSB replacement methods.

It performs much for smaller message length and the accuracy of decreases as the True value of p approaches 1. Cover images of histogram similar to 4.png embedded with secret image of histogram similar to 7.png are highly detectable under steganalysis ie triples can easily detect the presence of hidden message and estimate almost accurately smaller length. Cover images of histogram similar to 3.png embedded with secret image of with histogram similar to 17.png are lowly detectable by steganalysis ie triples cannot estimate accurately the length of the hidden message.

4.3 Summary of Findings

The Embedding Capacity of the proposed method was 3 times higher than that of the A. M. Molaeis method. The Multiple embedding method and secret sharing adopted in the proposed method contributed largely to achieving a high embedding capacity. In multiple embedding method, the same LSBs of the covers pixels that were embedded in the first cycle are re embedded in the next cycles while keeping a Key Trace. This guaranteed a high embedding capacity of 450%.

Furthermore, the proposed method achieves high imperceptibility than A. M. Molaeis method. This is evident from the high PSNR value achieved for larger secret message sizes while the previous method achieved low PSNR value. The impact of multiple embedding contributed largely to a good visual quality of the stego in the proposed method. The changes made per cycle on the same LSBs of the of the cover pixels guaranteed a low Mean Square error between the cover and stego images. Thus, leading to a high PSNR value while the previous method has a very low PSNR value. Embedding secret bits on all first and second LSBs of the cover largely increased the Mean Square Error between the cover and stego images. Thereby achieving a very low PSNR.

In addition, the proposed method have shown to be more robust against Noise, Cropping, Scratching and jpg compression attacks than the A. M. Molaeis method. Firstly, the error correction code used by the proposed method has a higher error correction capacity than that of the previous method. In addition to that the proposed method implemented a CSPRNG to embed secret images randomly on the cover which reduced occurrences of burst errors thereby increasing the error correction capability. This led to proposed method retrieving secret images with slightly higher PSNR values than the A. M. Molaeis method. A. M. Molaeis method has low PSNR because of large occurrences of burst errors due to embedding data serially. Key Trace which is free from any of these attacks, contributed largely to achieving high quality retrieved secret images. Due to this the quality of the retrieved secret image increased as the size of the embedded secret size increased. While the proposed method has low quality retrieved secret images for any secret message size.

Finally, the proposed method also proved to be more robust against steganalysis than the A. M. Molaeis method. Since Triples is more sensitive to covers embedded with smaller secret message sizes, a larger expansion rate of the proposed method constantly results in low proportion values than the previous method which has a smaller expansion rate. Therefore the attacker can predict easily the length of the small secret in previous method than in proposed method. For larger secret sizes, the proposed method performed better under steganalysis attacks than the previous method because of the impact of multiple embedding.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusions

The proposed method was outstanding in Embedding capacity, Embedding efficiency, Robust against attacks and resistance against steganalysis and was optimal in execution time. Meanwhile A. M Molaei's method only performed better than the proposed method in execution time.

The Multiple embedding method and secret sharing adopted in the proposed method contributed largely to achieving a high embedding capacity. The limitation was only the execution time. There is a trade off between the size of the secret message to be embedded and the execution time of both the embedding and extraction process.The multiple embedding method contributed largely to high embedding efficiency as the stego is guaranteed to achieve a good visual quality. The changes made per cycle on the same LSBs of the of the cover pixels caused a smaller change in the PSNR value. Thus, leading to a high PSNR value while the A. M. Molaeis method has a very low PSNR value caused by embedding on both first and second LSBs.
Furthermore, the choice of a good error correction code combined with multiple embedding method contributed to the increased robustness against Noise, Cropping, Scratching and jpg compression attacks than in the A. M. Molaeis method. Key Trace which is free from any of these attacks, contributed largely to achieving high quality retrieved secret images The error correction code used by the proposed method has a higher error correction capacity than that of the previous method. In addition to that the proposed method implemented a CSPRNG to embed secret images randomly on the cover which reduced occurrences of burst errors thereby increasing the error correction capability. This led to proposed method retrieving secret images with slightly higher PSNR values than the A. M. Molaeis method. A. M. Molaeis method has low PSNR because of large occurrences of burst errors due to embedding data serially.
Lastly, the proposed method also proved to be more robust against steganalysis than the A. M. Molaeis method. Since Triples is more sensitive to covers embedded with smaller secret message sizes, a larger expansion rate of the proposed method constantly results in low proportion values than the previous method which has a smaller expansion rate. Therefore the attacker can predict easily the length of the small secret in A. M. Molaeis method than in proposed method. For larger secret sizes, the proposed method performed better under steganalysis attacks than the previous method because of the impact of multiple embedding.

5.2 Recommendations

High performance computers are able to embed large secret images in a short time, since there is a trade off between between the size of the secret message to be embedded and the execution time of both the proposed and A. M. Molaeis method

BIBLIOGRAPHY

[1] Self watermarking based on visual cryptography. 2019.

[2] M. V. R. CH et al. Medical image watermarking schemes against salt and pepper noise attack. *International Journal of Bio-Science and Bio-Technology*, 7(6):55–64, 2015.

[3] S. Cimato and C.-N. Yang. *Visual cryptography and secret image sharing*. CRC press, 2017.

[4] R. Crandall. Some notes on steganography. *Posted on steganography mailing list*, pages 1–6, 1998.

[5] H. V. Desai. Steganography, cryptography, watermarking: A comparitive study. *Journal of Global Research in Computer Science*, 3(12):33–35, 2013.

[6] J. C. Judge. Steganography: past, present, future. Technical report, Lawrence Livermore National Lab., CA (US), 2001.

[7] P. Junod. Cryptographic secure pseudo-random bits generation: The blum-blum-shub generator, 1999.

[8] S. Kaisar and J. A. Mahmud. Salt and pepper noise detection and removal by tolerance based selective arithmetic mean filtering technique for image restoration. *International Journal of Computer Science and Network Security*, 8(6):271–278, 2008.

[9] A. D. Ker. A general framework for structural steganalysis of lsb replacement. In *International Workshop on Information Hiding*, pages 296–311. Springer, 2005.

[10] S. Lin and D. J. Costello. *Error control coding*. Pearson Education India, 2001.

[11] M. Mansourpour, M. Rajabi, and J. Blais. Effects and performance of speckle noise reduction filters on active radar and sar images. In *Proc. ISPRS*, volume 36, page W41, 2006.

[12] A. Molaei, M. Sedaaghi, and H. Ebrahimnezhad. Steganography scheme based on reed-muller code with improving payload and ability to retrieval of destroyed data for digital images. *AUT Journal of Electrical Engineering*, 49(1):53–62, 2017.

[13] S. Raaphorst. Reed-muller codes. 2003.

[14] A. M. A.-A. Selami and A. F. Fadhil. A study of the effects of gaussian noise on image features. *kirkuk university journal for scientific studies*, 11(3):152–169, 2016.

[15] D. Taranovsky. *Data Hiding and Digital Watermarking*, pages 387–399. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-540-79567-4. doi: 10.1007/978-3-540-79567-4_31. URL https://doi.org/10.1007/978-3-540-79567-4_31.

[16] C.-C. Thien and J.-C. Lin. A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function. *Pattern recognition*, 36(12): 2875–2881, 2003.

[17] J. Van Lint and G. Van der Geer. *Introduction to coding theory and algebraic geometry*, volume 12. Birkhäuser, 2012.

[18] S. B. Wicker. *Error control systems for digital communication and storage*, volume 1. Prentice hall Englewood Cliffs, 1995.

[19] Y. Zhang, J. Jiang, Y. Zha, H. Zhang, and S. Zhao. Research on embedding capacity and efficiency of information hiding based on digital images. *International Journal of Intelligence Science*, 3(02):77, 2013.

Appendices

APPENDIX A

IMAGE DATA OF CAPACITY AND IMPERCEPTIBILITY EXPERIMENT

Table A.1: Image Data of size 512 x 512 as Cover images

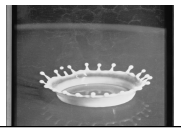
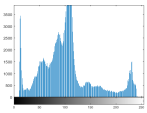
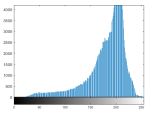
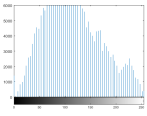
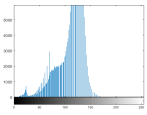
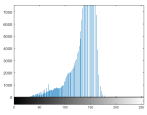
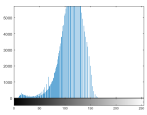
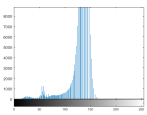
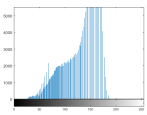
IMAGE NAME	COVER IMAGE	HISTOGRAM
1.png		
2.png		
3.png		
4.png		
5.png		
6.png		
7.png		
8.png		

Table A.1: Image Data of size 512 x 512 as Cover images

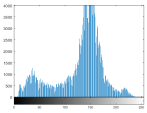
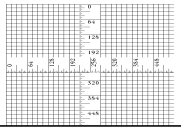
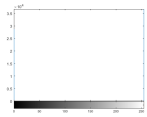
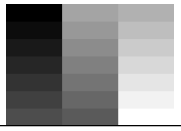
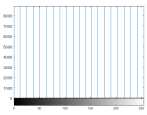
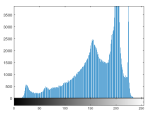
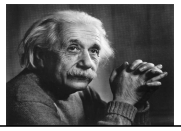
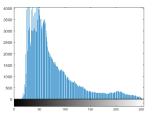
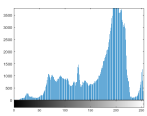
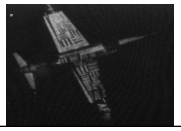
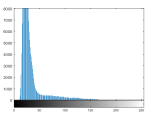
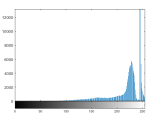
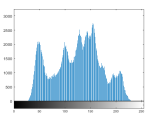
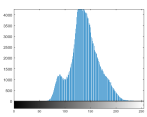
IMAGE NAME	COVER IMAGE	HISTOGRAM
9.png		
10.png		
11.png		
12.png		
13.png		
14.png		
15.png		
16.png		
17.png		
18.png		

Table A.1: Image Data of size 512 x 512 as Cover images

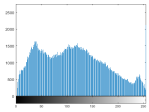
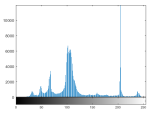
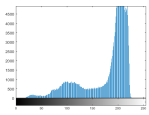
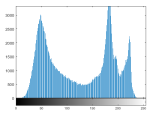
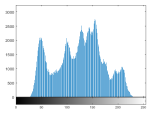
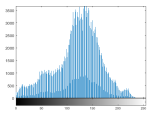
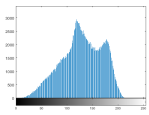
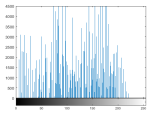
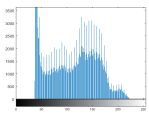
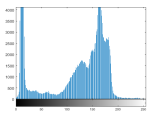
IMAGE NAME	COVER IMAGE	HISTOGRAM
19.png		
20.png		
21.png		
22.png		
23.png		
24.png		
25.png		
26.png		
27.png		
28.png		

Table A.1: Image Data of size 512 x 512 as Cover images

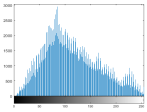
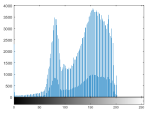
IMAGE NAME	COVER IMAGE	HISTOGRAM
29.png		
30.png		

Table A.2: Image Data set for Secret images

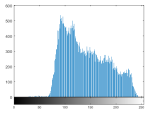
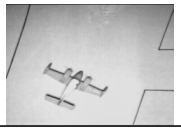
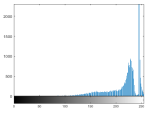
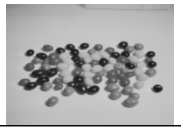
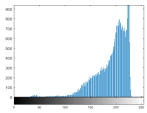
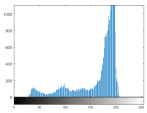
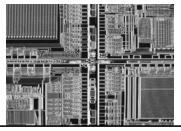
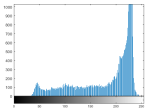
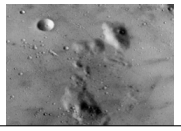
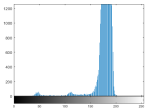
IMAGE NAME	SECRET IMAGE	HISTOGRAM
1.png		
2.png		
3.png		
4.png		
5.png		
6.png		

Table A.2: Image Data set for Secret images

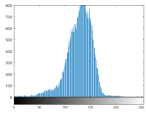
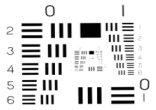
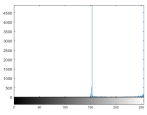
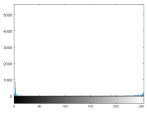
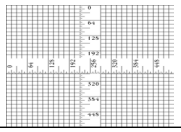
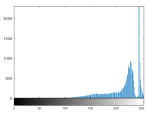
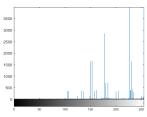
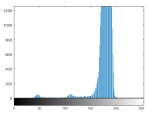
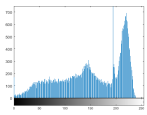
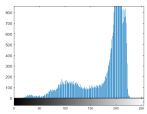
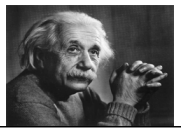
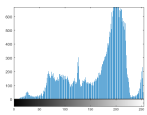
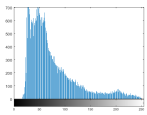
IMAGE NAME	SECRET IMAGE	HISTOGRAM
7.png		
8.png		
9.png		
10.png		
11.png		
12.png		
13.png		
14.png		
15.png		
16.png		

Table A.2: Image Data set for Secret images

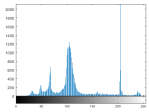
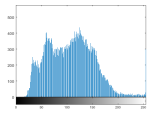
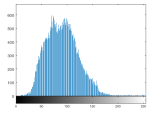
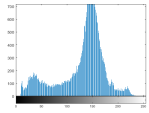
IMAGE NAME	SECRET IMAGE	HISTOGRAM
17.png		
18.png		
19.png		
20.png		

Table A.3: Image Data set for Agreed images

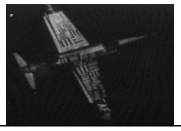
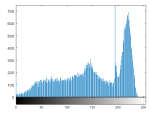
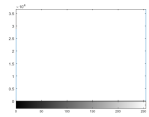
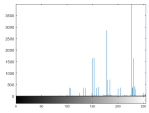
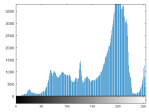
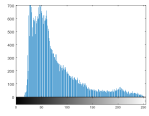
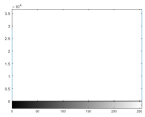
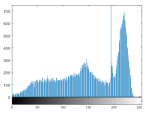
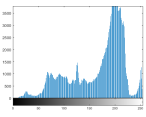
IMAGE NAME	AGREED IMAGE	HISTOGRAM
A.png		
B.png		
C.png		
D.png		
E.png		

Table A.3: Image Data set for Agreed images

IMAGE NAME	AGREED IMAGE	HISTOGRAM
F.png		
G.png		
H.png		

A.1 CAPACITY RESULTS

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
26.png	9.png	480	0.37	0.59	75.577	62.636
		960	0.73	1.17	72.246	62.242
		1920	1.46	2.34	69.408	61.548
		3840	2.93	4.69	66.435	60.573
		7680	5.86	9.38	63.464	59.342
		15360	11.72	18.75	60.467	57.379
		26880	20.51	32.81	58.012	55.736
		30720	23.44	37.5	57.419	54.406
		46080	35.16	56.25	55.688	53.579
		61440	46.88	75	54.406	52.395
		65536	50	80	54.102	51.997
		76800	58.59	93.75	53.344	51.985
		92160	70.31	112.5	52.669	52.203
		107520	82.03	131.25	52.015	51.986
		122880	93.75	150	51.386	52.070
		131072	100	160	51.126	51.988

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		138240	105.47	168.75	50.667	51.980
		153600	117.19	187.5	49.746	52.047
		168960	128.91	206.25	49.149	52.071
		184320	140.63	225	48.537	52.067
		196608	150	240	48.133	52.088
		215040		262.5	47.729	52.079
		230400		281.25	47.045	51.982
		245760		300	46.751	52.054
		261120		318.75		51.989
		276480		337.5		52.022
		291840		356.25		51.978
		307200		375		52.382
		322560		393.75		51.993
		337920		412.5		52.019
		353280		431.25		51.983
		368640		450		52.077
	15.png	480	0.37	0.59	75.614	62.285
		960	0.73	1.17	72.289	62.173
		1920	1.46	2.34	69.419	61.508
		3840	2.93	4.69	66.482	60.454
		7680	5.86	9.38	63.478	58.983
		15360	11.72	18.75	60.446	57.454
		26880	20.51	32.81	58.014	55.563
		30720	23.44	37.5	57.392	54.407
		46080	35.16	56.25	55.789	53.437
		61440	46.88	75	54.435	52.377
		65536	50	80	54.120	51.982
		76800	58.59	93.75	53.386	51.962
		92160	70.31	112.5	52.698	52.105
		107520	82.03	131.25	52.018	51.982
		122880	93.75	150	51.405	52.035
		131072	100	160	51.147	51.982
		138240	105.47	168.75	50.702	52.027
		153600	117.19	187.5	49.789	52.066

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		168960	128.91	206.25	49.170	52.028
		184320	140.63	225	48.549	52.040
		196608	150	240	48.113	52.016
		215040		262.5	47.842	51.899
		230400		281.25	47.049	52.045
		245760		300	46.746	51.989
		261120		318.75		52.012
		276480		337.5		51.983
		291840		356.25		52.039
		307200		375		51.982
		322560		393.75		52.016
		337920		412.5		51.973
		353280		431.25		52.034
		368640		450		52.077
17.png	18.png	480	0.37	0.59	75.432	62.463
		960	0.73	1.17	72.246	62.176
		1920	1.46	2.34	69.408	61.497
		3840	2.93	4.69	66.502	60.423
		7680	5.86	9.38	63.468	58.283
		15360	11.72	18.75	60.462	57.387
		26880	20.51	32.81	58.012	55.538
		30720	23.44	37.5	57.441	54.418
		46080	35.16	56.25	55.687	53.406
		61440	46.88	75	54.461	52.391
		65536	50	80	54.118	51.976
		76800	58.59	93.75	53.382	51.979
		92160	70.31	112.5	52.670	52.195
		107520	82.03	131.25	52.014	51.972
		122880	93.75	150	51.419	52.037
		131072	100	160	51.118	51.969
		138240	105.47	168.75	50.693	51.972
		153600	117.19	187.5	49.757	52.036
		168960	128.91	206.25	49.150	52.040
		184320	140.63	225	48.538	52.035

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		196608	150	240	48.123	52.040
		215040		262.5	47.838	52.066
		230400		281.25	47.041	51.975
		245760		300	46.752	52.041
		261120		318.75		51.968
		276480		337.5		52.312
		291840		356.25		51.976
		307200		375		52.039
		322560		393.75		51.975
		337920		412.5		52.014
		353280		431.25		51.039
		368640		450		52.039
	9.png	480	0.37	0.59	75.614	62.362
		960	0.73	1.17	72.242	62.163
		1920	1.46	2.34	69.401	61.490
		3840	2.93	4.69	66.534	60.410
		7680	5.86	9.38	63.461	58.277
		15360	11.72	18.75	60.471	57.381
		26880	20.51	32.81	58.017	55.516
		30720	23.44	37.5	57.424	54.405
		46080	35.16	56.25	55.687	53.390
		61440	46.88	75	54.413	52.388
		65536	50	80	54.142	51.964
		76800	58.59	93.75	53.374	51.977
		92160	70.31	112.5	52.668	52.185
		107520	82.03	131.25	52.011	51.965
		122880	93.75	150	51.411	52.038
		131072	100	160	51.114	51.966
		138240	105.47	168.75	50.683	51.962
		153600	117.19	187.5	49.752	52.027
		168960	128.91	206.25	49.175	52.037
		184320	140.63	225	48.533	52.030
		196608	150	240	48.120	52.039
		215040		262.5	47.832	52.056

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		230400		281.25	47.038	51.967
		245760		300	46.750	52.040
		261120		318.75		51.960
		276480		337.5		52.310
		291840		356.25		51.966
		307200		375		52.038
		322560		393.75		51.972
		337920		412.5		52.010
		353280		431.25		51.036
		368640		450		52.037
6.png	11.png	480	0.37	0.59	75.614	62.285
		960	0.73	1.17	72.289	62.173
		1920	1.46	2.34	69.419	61.508
		3840	2.93	4.69	66.482	60.454
		7680	5.86	9.38	63.478	58.983
		15360	11.72	18.75	60.446	57.454
		26880	20.51	32.81	58.014	55.563
		30720	23.44	37.5	57.392	54.407
		46080	35.16	56.25	55.789	53.437
		61440	46.88	75	54.435	52.377
		65536	50	80	54.120	51.982
		76800	58.59	93.75	53.386	51.962
		92160	70.31	112.5	52.698	52.105
		107520	82.03	131.25	52.018	51.982
		122880	93.75	150	51.405	52.035
		131072	100	160	51.147	51.982
		138240	105.47	168.75	50.702	52.027
		153600	117.19	187.5	49.789	52.066
		168960	128.91	206.25	49.170	52.028
		184320	140.63	225	48.549	52.040
		196608	150	240	48.113	52.016
		215040		262.5	47.842	51.899
		230400		281.25	47.049	52.045
		245760		300	46.746	51.989

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		261120		318.75		52.012
		276480		337.5		51.983
		291840		356.25		52.039
		307200		375		51.982
		322560		393.75		52.016
		337920		412.5		51.973
		353280		431.25		52.034
		368640		450		52.077
	19.png	480	0.37	0.59	75.432	62.463
		960	0.73	1.17	72.246	62.176
		1920	1.46	2.34	69.408	61.497
		3840	2.93	4.69	66.502	60.423
		7680	5.86	9.38	63.468	58.283
		15360	11.72	18.75	60.462	57.387
		26880	20.51	32.81	58.012	55.538
		30720	23.44	37.5	57.441	54.418
		46080	35.16	56.25	55.687	53.406
		61440	46.88	75	54.461	52.391
		65536	50	80	54.118	51.976
		76800	58.59	93.75	53.382	51.979
		92160	70.31	112.5	52.670	52.195
		107520	82.03	131.25	52.014	51.972
		122880	93.75	150	51.419	52.037
		131072	100	160	51.118	51.969
		138240	105.47	168.75	50.693	51.972
		153600	117.19	187.5	49.757	52.036
		168960	128.91	206.25	49.150	52.040
		184320	140.63	225	48.538	52.035
		196608	150	240	48.123	52.040
		215040		262.5	47.838	52.066
		230400		281.25	47.041	51.975
		245760		300	46.752	52.041
		261120		318.75		51.968
		276480		337.5		52.312

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		291840		356.25		51.976
		307200		375		52.039
		322560		393.75		51.975
		337920		412.5		52.014
		353280		431.25		51.039
		368640		450		52.039
4.png	8.png	480	0.37	0.59	75.614	62.362
		960	0.73	1.17	72.242	62.163
		1920	1.46	2.34	69.401	61.490
		3840	2.93	4.69	66.534	60.410
		7680	5.86	9.38	63.461	58.277
		15360	11.72	18.75	60.471	57.381
		26880	20.51	32.81	58.017	55.516
		30720	23.44	37.5	57.424	54.405
		46080	35.16	56.25	55.687	53.390
		61440	46.88	75	54.413	52.388
		65536	50	80	54.142	51.964
		76800	58.59	93.75	53.374	51.977
		92160	70.31	112.5	52.668	52.185
		107520	82.03	131.25	52.011	51.965
		122880	93.75	150	51.411	52.038
		131072	100	160	51.114	51.966
		138240	105.47	168.75	50.683	51.962
		153600	117.19	187.5	49.752	52.027
		168960	128.91	206.25	49.175	52.037
		184320	140.63	225	48.533	52.030
		196608	150	240	48.120	52.039
		215040		262.5	47.832	52.056
		230400		281.25	47.038	51.967
		245760		300	46.750	52.040
		261120		318.75		51.960
		276480		337.5		52.310
		291840		356.25		51.966
		307200		375		52.038

Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		322560		393.75		51.972
		337920		412.5		52.010
		353280		431.25		51.036
		368640		450		52.037
	10.png	480	0.37	0.59	75.432	62.463
		960	0.73	1.17	72.246	62.176
		1920	1.46	2.34	69.408	61.497
		3840	2.93	4.69	66.502	60.423
		7680	5.86	9.38	63.468	58.283
		15360	11.72	18.75	60.462	57.387
		26880	20.51	32.81	58.012	55.538
		30720	23.44	37.5	57.441	54.418
		46080	35.16	56.25	55.687	53.406
		61440	46.88	75	54.461	52.391
		65536	50	80	54.118	51.976
		76800	58.59	93.75	53.382	51.979
		92160	70.31	112.5	52.670	52.195
		107520	82.03	131.25	52.014	51.972
		122880	93.75	150	51.419	52.037
		131072	100	160	51.118	51.969
		138240	105.47	168.75	50.693	51.972
		153600	117.19	187.5	49.757	52.036
		168960	128.91	206.25	49.150	52.040
		184320	140.63	225	48.538	52.035
		196608	150	240	48.123	52.040
		215040		262.5	47.838	52.066
		230400		281.25	47.041	51.975
		245760		300	46.752	52.041
		261120		318.75		51.968
		276480		337.5		52.312
		291840		356.25		51.976
		307200		375		52.039
		322560		393.75		51.975
		337920		412.5		52.014

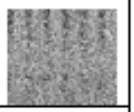
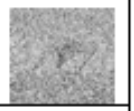
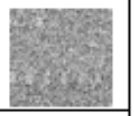
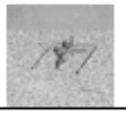
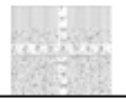
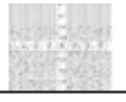
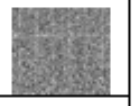
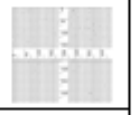
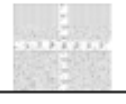
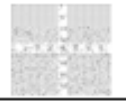
Table A.4: A Capacity Comparison between Proposed and A.M.Molaei's Method

Cover Image	Secret image	Secret Image size(bits)	Capacity(%)		PSNR(dB)	
			A.M.Molaei's Method	Proposed Method	A.M.Molaei's Method	Proposed Method
		353280		431.25		51.039
		368640		450		52.039

A.2 Experiment Results for Noise Attacks

Secret Image & Size	Stego Image	Noise Attacks	Proposed Method			Previous Method		
			Attacked Stego PSNR(dB)	PSNR Secret Image(dB)	Recovered Secret Image	Attacked Stego PSNR(dB)	PSNR Secret Image(dB)	Recovered Secret Image
 15360 bits	Stego 20	Gaussian Noise: Mean 0 & Variance 0.01	20.218	9.359		20.213	9.336	
		Gaussian Noise: Mean 0 & Variance 0.1	11.464	9.713		11.466	9.815	
		Salt & Pepper: Density 0.05	18.353	100		18.271	28.390	
		Salt & Pepper: Density 0.5	8.242	12.176		8.230	11.751	
		Speckle Noise: variance 0.04	21.350	9.324		21.357	9.654	
		Speckle Noise: variance 0.4	11.879	9.409		11.878	10.337	
 15360 bits	Stego 13	Gaussian Noise: Mean 0 & Variance 0.01	20.221	9.131		20.088	8.964	
		Gaussian Noise: Mean 0 & Variance 0.1	11.457	8.829		11.378	7.995	
		Salt & Pepper: Density 0.05	18.232	100		18.105	29.193	
		Salt & Pepper: Density 0.5	8.268	12.610		8.242	11.796	
		Speckle Noise: variance 0.04	21.327	9.252		21.064	9.150	
		Speckle Noise: variance 0.4	11.876	9.220		11.064	8.792	

 15360 bits	Stego 15	Gaussian Noise: Mean 0 & Variance 0.01	20.224	6.463		20.011	9.485	
		Gaussian Noise: Mean 0 & Variance 0.1	11.443	7.385		11.372	8.727	
		Salt & Pepper: Density 0.05	18.342	33.006		18.119	28.790	
		Salt & Pepper: Density 0.5	8.248	8.864		8.246	11.881	
		Speckle Noise: variance 0.04	21.348	6.505		20.956	9.439	
		Speckle Noise: variance 0.4	11.877	6.540		11.622	9.364	
 15360 bits	Stego 14	Gaussian Noise: Mean 0 & Variance 0.01	20.246	8.880		20.020	8.647	
		Gaussian Noise: Mean 0 & Variance 0.1	11.470	8.777		11.363	8.986	
		Salt & Pepper: Density 0.05	18.216	46.644		18.194	28.466	
		Salt & Pepper: Density 0.5	8.232	11.632		8.223	10.991	
		Speckle Noise: variance 0.04	21. 351	8.747		20. 943	8.882	
		Speckle Noise: variance 0.4	11.870	8.514		11.640	9.273	

 196608 bits	Stego 11	Gaussian Noise: Mean 0 & Variance 0.01	20.204	17.333		20.019	9.439	
		Gaussian Noise: Mean 0 & Variance 0.1	11.453	17.176		11.354	9.276	
		Salt & Pepper: Density 0.05	18.284	100		18.095	30.082	
		Salt & Pepper: Density 0.5	8. 240	28.124		8. 216	11.505	
		Speckle Noise: variance 0.04	21.357	18.147		20.957	9.953	
		Speckle Noise: variance 0.4	11.912	17.690		11.620	9.842	
 196608 bits	Stego 10	Gaussian Noise: Mean 0 & Variance 0.01	20.208	16.449		20.030	6.562	
		Gaussian Noise: Mean 0 & Variance 0.1	11.454	16.459		11.349	6.534	
		Salt & Pepper: Density 0.05	18.206	100		18.108	31.901	
		Salt & Pepper: Density 0.5	8. 247	28.453		8. 223	9.883	
		Speckle Noise: variance 0.04	21.323	18.677		20.921	8.292	
		Speckle Noise: variance 0.4	11.872	17.368		11.349	7.101	

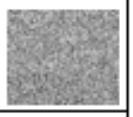
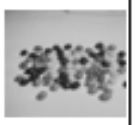
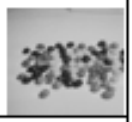
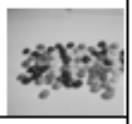
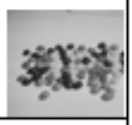
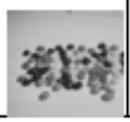
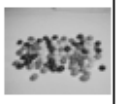
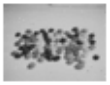
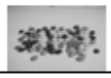
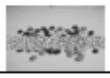
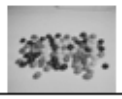
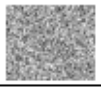
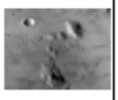
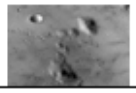
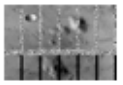
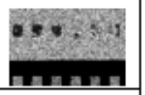
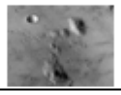
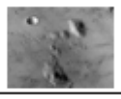
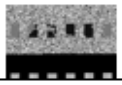
 196608 Bits	Stego 18	Gaussian Noise: Mean 0 & Variance 0.01	20.213	12.748		20.217	7.699	
		Gaussian Noise: Mean 0 & Variance 0.1	11.419	12.514		11.435	7.733	
		Salt & Pepper: Density 0.05	18.271	100		18.177	26.053	
		Salt & Pepper: Density 0.5	8. 246	22.710		8. 858	9.064	
		Speckle Noise: variance 0.04	21.330	12.469		21.305	7.653	
		Speckle Noise: variance 0.4	11.884	12.528		11.884	7.755	
 368640 bits	Any Stego	Gaussian Noise: Mean 0 & Variance 0.01	20.220	100		All Secret can successfully be recovered		
		Gaussian Noise: Mean 0 & Variance 0.1	11.454	100				
		Salt & Pepper: Density 0.05	18.166	100				
		Salt & Pepper: Density 0.5	8. 240	100				
		Speckle Noise: variance 0.04	21.335	100				
		Speckle Noise: variance 0.4	11.872	100				

Figure A.1: Experiment Result of robustness against noise attack

A.3 Experiment Results for Cropping Attacks

Secret Image	Stego Image	Secret Message Size (bits)	Cropping Area				Proposed Method		A. M. Molaei's Method	
			X point	Y point	Height (pixel)	Width (pixel)	PSNR Secret Image(dB)	Recovered Secret Image	PSNR Secret Image(dB)	Recovered Secret Image
	Stego 3	15360	0	0	512	360	19.898		100	
		131072	0	0	512	360	33.670		14.797	
		196608	0	0	512	360	39.357		15.863	
		368640	0	0	512	360	100			
	Stego 4	61440	150	200	220	170	8.284		8.220	
			70	80	422	422	8.376		8.443	
		196608	150	200	220	170	15.078		5.898	
			70	80	422	422	14.834		6.754	
		368640	150	200	220	170	100			
			70	80	422	422	100			

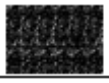
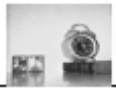
	Stego 5	61440	20	80	412	432	8.661		8.055	
			90	160	260	290	8.606		8.167	
		196608	20	80	412	432	13.434		7.643	
			90	160	260	290	13.359		7.328	
		368640	20	80	412	432	100			
			90	160	260	290	100			
	Stego 6	61440	20	80	412	432	21.878		17.721	
			90	160	260	290	9.603		9.204	
		196608	20	80	412	432	45.563		15.634	
			90	160	260	290	14.659		8.581	
		368640	20	80	412	432	100			
			90	160	260	290	100			

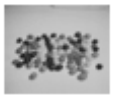
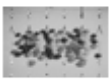
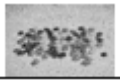
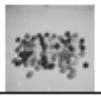
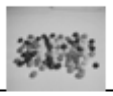
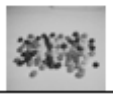
	Stego 7	15360	120	200	250	140	7.467		7.392	
			20	40	435	452	6.576		6.850	
		196608	120	200	250	140	16.031		3.947	
			20	40	435	452	16.009		4.998	
		368640	120	200	250	140	100			
			20	40	435	452	100			
	Stego 8	61440	140	190	190	155	6.540		6.546	
			20	40	435	452	6.685		6.236	
		196608	140	190	190	155	14.507		3.752	
			20	40	435	452	14.458		5.248	
		368640	140	190	190	155	100			
			20	40	435	452	100			

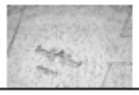
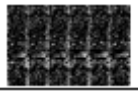
	Stego 12	30720	0	0	440	312	12.840		17.678	
			0	0	442	420	19.823		17.989	
			0	310	512	120	8.750		8.579	
		196608	0	0	440	312	27.453		12.147	
			0	0	442	420	39.455		13.708	
			0	310	512	120	14.722		6.636	
		368640	0	0	440	312	100			
			0	0	442	420	100			
			0	310	512	120	100			
	Stego 13	26880	0	30	312	512	13.721		11.793	
		131072	0	30	312	512	25.691		12.826	
		196608	0	30	312	512	28.569		9.700	
		368640	0	30	312	512	100			

Figure A.2: Experiment Result of robustness against cropping attack

A.4 Experiment Results for Scratching Attacks

Secret Image	Stego Image	Secret Size(bit)	Scratches	Proposed Method		A. M. Molaei's Method	
				PSNR Secret Image(dB)	Recovered Secret Image	PSNR Secret Image(dB)	Recovered Secret Image
	Stego 4	480	Single	100		100	
			Multiple	6.167		11.012	
		26880	Single	100		100	
			Multiple	3.997		3.284	
		131072	Single	100		21.260	
			Multiple	17.717		3.390	
		196608	Single	100		21.543	
			Multiple	20.688		3.222	
		307200	Single	100			
			Multiple	31.692			
		368640	Single	100			
			Multiple	100			

	Stego 3	480	Single	100		29.755	
			Multiple	5.160		11.166	
		26880	Single	100		17.485	
			Multiple	4.613		4.428	
		131072	Single	100		20.453	
			Multiple	17.789		4.193	
		196608	Single	100		22.386	
			Multiple	19.241		4.331	
		307200	Single	100			
			Multiple	31.220			
		368640	Single	100			
			Multiple	100			

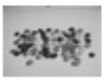
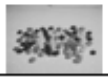
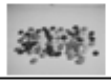
	Stego 2	480	Single	100		100	
			Multiple	2.372		7.910	
		26880	Single	100		15.102	
			Multiple	2.391		3.638	
		131072	Single	100		18.488	
			Multiple	20.957		3.478	
		196608	Single	100		18.243	
			Multiple	16.365		3.514	
		307200	Single	100			
			Multiple	30.295			
		368640	Single	100			
			Multiple	100			

	Stego 1	480	Single	100		100	
			Multiple	5.138		6.647	
		26880	Single	100		100	
			Multiple	4.944		7.133	
		131072	Single	100		19.869	
			Multiple	15.256		6.850	
		196608	Single	59.571		20.482	
			Multiple	14.375		6.468	
		307200	Single	100			
			Multiple	24.038			
		368640	Single	100			
			Multiple	100			

Figure A.3: Experiment Result of robustness against scratching attacks

A.5 Experiment Results for JPEG Compression Attacks

Secret Image	Stego Image	Secret Message Size(bits)	Proposed Method			A. M. Molaci's Method		
			PSNR of Destroyed Stego(dB)	PSNR Secret Image(dB)	Recovered Secret Image	PSNR of Destroyed Stego(dB)	PSNR Secret Image(dB)	Recovered Secret Image
	Stego 2	3840	43.436	9.590		43.609	8.870	
		30720	43.398	9.198		44.580	8.895	
		92160	43.388	11.820		44.279	8.909	
		131072	43.387	15.520		43.507	8.960	
		196608	43.391	18.565		44.598	8.779	
		261120	43.396	27.622				
		353280	43.389	41.750				
		384000	43.393	100				

	Stego 3	3840	42.693	10.043		42.444	9.397	
		30720	42.674	9.367		42.468	9.197	
		92160	42.680	12.119		42.455	9.156	
		131072	42.686	14.853		42.463	9.150	
		196608	42.682	17.831		42.465	9.108	
		261120	42.679	25.225				
		353280	42.688	37.828				
		384000	42.675	100				
	Stego 1	3840	44.965	10.105		43.609	9.981	
		30720	44.895	9.544		44.580	9.239	
		92160	44.870	11.790		44.279	9.133	
		131072	44.862	14.030		43.507	9.079	
		196608	44.863	16.964		44.598	9.099	
		261120	44.874	24.061				
		353280	44.870	33.136				
		384000	44.869	100				

	Stego 5	3840	42.714	8.738		42.321	8.875	
		30720	42.702	8.806		42.340	8.503	
		92160	42.699	11.176		42.351	8.175	
		131072	42.704	13.113		42.321	8.166	
		196608	42.696	15.382		42.340	8.157	
		261120	42.691	22.423				
		353280	42.695	31.513				
		384000	42.685	100				
	Stego 4	3840	43.161	8.533		42.381	8.515	
		30720	43.133	8.452		42.430	8.388	
		92160	43.121	11.752		42.413	8.166	
		131072	43.131	14.972		42.374	8.170	
		196608	43.132	16.848		42.429	8.211	
		261120	43.124	25.901				
		353280	43.121	32.813				
		384000	43.126	100				

Figure A.4: Experiment Result of robustness against JPEG compression attacks

A.6 Experiment Results for Steganalysis Attacks

Cover Image	Secret	Secret Size(bits)	Molaei's Method		Proposed Method	
			True P value	Estimated P value	True P value	Estimated P value
4.png	7.png	480	0.001831	0.001228	0.037305	0.03321
		960	0.003662	0.003414	0.040234	0.03825
		1920	0.007324	0.006838	0.046094	0.04272
		3840	0.014648	0.011239	0.057813	0.05321
		7680	0.029297	0.028534	0.08125	0.07913
		15360	0.058594	0.053251	0.128125	0.10145
		26880	0.102539	0.091232	0.198438	0.168421
		30720	0.117188	0.10044	0.221875	0.199469
		46080	0.175781	0.143821	0.315625	0.285651
		61440	0.234375	0.20115	0.402734	0.391522
		65536	0.25	0.22045	0.412109	0.409244
		76800	0.292969	0.25623	0.412109	0.410012
		92160	0.351563	0.30526	0.409375	0.399421
		107520	0.410156	0.39123	0.412109	0.408686
		122880	0.46875	0.42631	0.405469	0.39806
		131072	0.5	0.48012	0.412109	0.40969
		138240	0.527344	0.50312	0.412109	0.40870
		153600	0.585938	0.53551	0.409375	0.39380
		168960	0.644531	0.61121	0.412109	0.40187
		184320	0.703125	0.68254	0.405469	0.39467
		196608	0.75	0.7209	0.409375	0.395543
		215040			0.409375	0.39835
		230400			0.412109	0.40856
		245760			0.405469	0.39892
		261120			0.412109	0.40813
		276480			0.409375	0.39848
		291840			0.412109	0.40859
		307200			0.405469	0.39382
		322560			0.412109	0.40189
		337920			0.409375	0.39518
		353280			0.412109	0.40695
		368640			0.405469	0.39146

2.png	11.png	480	0.001831	0.001211	0.037305	0.03282
		960	0.003662	0.003252	0.040234	0.03780
		1920	0.007324	0.006546	0.046094	0.04198
		3840	0.014648	0.011426	0.057813	0.05211
		7680	0.029297	0.028534	0.08125	0.07819
		15360	0.058594	0.053862	0.128125	0.10089
		26880	0.102539	0.097472	0.198438	0.157310
		30720	0.117188	0.103244	0.221875	0.198210
		46080	0.175781	0.14614	0.315625	0.271820
		61440	0.234375	0.20893	0.402734	0.384210
		65536	0.25	0.21963	0.412109	0.407131
		76800	0.292969	0.25214	0.412109	0.407813
		92160	0.351563	0.30326	0.409375	0.392112
		107520	0.410156	0.39118	0.412109	0.403121
		122880	0.46875	0.41876	0.405469	0.39213
		131072	0.5	0.47089	0.412109	0.40031
		138240	0.527344	0.50112	0.412109	0.40321
		153600	0.585938	0.53120	0.409375	0.39130
		168960	0.644531	0.61001	0.412109	0.39987
		184320	0.703125	0.67834	0.405469	0.38821
		196608	0.75	0.7199	0.409375	0.392145
		215040			0.409375	0.39510
		230400			0.412109	0.40213
		245760			0.405469	0.39432
		261120			0.412109	0.40321
		276480			0.409375	0.39312
		291840			0.412109	0.40131
		307200			0.405469	0.39013
		322560			0.412109	0.40024
		337920			0.409375	0.39202
		353280			0.412109	0.39989
		368640			0.405469	0.38146

1.png	15.png	480	0.001831	0.001089	0.037305	0.03170
		960	0.003662	0.002991	0.040234	0.03679
		1920	0.007324	0.006213	0.046094	0.04042
		3840	0.014648	0.010112	0.057813	0.05101
		7680	0.029297	0.027211	0.08125	0.07708
		15360	0.058594	0.052611	0.128125	0.99089
		26880	0.102539	0.096241	0.198438	0.14389
		30720	0.117188	0.10004	0.221875	0.195100
		46080	0.175781	0.12104	0.315625	0.268110
		61440	0.234375	0.20211	0.402734	0.381190
		65536	0.25	0.21123	0.412109	0.402891
		76800	0.292969	0.24185	0.412109	0.402713
		92160	0.351563	0.30014	0.409375	0.387821
		107520	0.410156	0.38001	0.412109	0.401691
		122880	0.46875	0.41321	0.405469	0.39021
		131072	0.5	0.46063	0.412109	0.39981
		138240	0.527344	0.49009	0.412109	0.40119
		153600	0.585938	0.52872	0.409375	0.38890
		168960	0.644531	0.60972	0.412109	0.39321
		184320	0.703125	0.67212	0.405469	0.38519
		196608	0.75	0.7012	0.409375	0.387895
		215040			0.409375	0.38890
		230400			0.412109	0.39643
		245760			0.405469	0.38389
		261120			0.412109	0.39881
		276480			0.409375	0.38152
		291840			0.412109	0.39531
		307200			0.405469	0.38913
		322560			0.412109	0.39018
		337920			0.409375	0.38102
		353280			0.412109	0.38718
		368640			0.405469	0.38001

3.png	17.png	480	0.001831	0.00998	0.037305	0.03060
		960	0.003662	0.002831	0.040234	0.03564
		1920	0.007324	0.006130	0.046094	0.03992
		3840	0.014648	0.010012	0.057813	0.05008
		7680	0.029297	0.026141	0.08125	0.07616
		15360	0.058594	0.05150	0.128125	0.98891
		26880	0.102539	0.09514	0.198438	0.141890
		30720	0.117188	0.09804	0.221875	0.182105
		46080	0.175781	0.10014	0.315625	0.257130
		61440	0.234375	0.20019	0.402734	0.372998
		65536	0.25	0.20014	0.412109	0.400631
		76800	0.292969	0.21355	0.412109	0.402713
		92160	0.351563	0.29052	0.409375	0.378310
		107520	0.410156	0.37891	0.412109	0.400351
		122880	0.46875	0.40029	0.405469	0.38099
		131072	0.5	0.43023	0.412109	0.39310
		138240	0.527344	0.48106	0.412109	0.39110
		153600	0.585938	0.51891	0.409375	0.37660
		168960	0.644531	0.60741	0.412109	0.38113
		184320	0.703125	0.66112	0.405469	0.37408
		196608	0.75	0.69890	0.409375	0.375750
		215040			0.409375	0.37681
		230400			0.412109	0.38541
		245760			0.405469	0.37217
		261120			0.412109	0.38723
		276480			0.409375	0.37896
		291840			0.412109	0.38820
		307200			0.405469	0.38301
		322560			0.412109	0.38715
		337920			0.409375	0.37902
		353280			0.412109	0.37828
		368640			0.405469	0.37891

Figure A.5: Experiment Result of Steganalysis

APPENDIX B

Curriculum Vitae # Example

PERSONAL INFORMATION

Name	:	Mr. Handsome
Date and Place of Birth	:	May 04, 2015 - Bandung, West Java
Address	:	...
City and Province	:	...
Postal Code	:	99999
Sex	:	Male
Nationality	:	Indonesia
Religion	:	...
Marital Status	:	Single
Language Skill 1	:	Indonesian (Mother Tongue)
Language Skill 2	:	English (Sufficient)
Language Skill 3	:	Minang (Sufficient)
Language Skill 4	:	Java (Intermediate)
Language Skill 5	:	Malay (Basic Knowledge)
Language Skill 6	:	Arabic (Basic Knowledge)
Phone 1	:	+62 851 xxxx xxxx
Phone 2	:	+62 852 xxxx xxxx
Fax	:	-
Email	:	youremail@students.telkomuniversity.ac.id

EDUCATION

Education Degree	**Institution**	**Period of Education**	**Year of Graduation**
Master	Telkom University	2013 - 2015	2015
Bachelor	...	2008 - 2012	2012
High School	...	2007 - 2008	2008
Middle School	...	2003 - 2005	2005