



저작자표시 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.
- 이 저작물을 영리 목적으로 이용할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#) 

February 2015

PhD's Degree Thesis

Highly Undetectable Modular Steganography using Error Images

Graduate School of Chosun University

Department of Computer Engineering

Afrakhteh Masoud

압축 이미지와 원본 이미지의 차이를
이용한 고도의 모듈러 정보 은닉 기법
Highly Undetectable Modular Steganography using Error Images

February 25, 2015

Graduate School of Chosun University

Department of Computer Engineering

Afrakhteh Masoud

Highly Undetectable Modular Steganography using Error Images

Advisor: Prof. Jeong-A Lee

This Thesis is submitted to Graduate School of Chosun
University in partial fulfillment of the requirements for
a PhD's degree






October 2014

Graduate School of Chosun University

Department of Computer Engineering

Afrakhteh Masoud

마수드 아프라크테 박사학위논문을 인준함

위원장	조선 대학교 교수	<u>이상웅</u>	
위 원	조선 대학교 교수	<u>문인규</u>	
위 원	조선 대학교 교수	<u>권구락</u>	
위 원	전남 대학교 교수	<u>김철홍</u>	
위 원	조선 대학교 교수	<u>이정아</u>	

2014 년 12 월

조선 대학교 대학원

I dedicate this thesis to my beloved wife and parents, my dad for all his love and support, and to my teachers, especially Prof. Lee Jeong-A for her trust in me.

Table of Contents

Table of Contents	i
List of Figures	iii
List of Tables	iv
List of Acronyms	v
초 록	vi
I. Introduction	1
A. Research Motivation.....	1
B. Research Objectives	2
C. Thesis Contributions.....	2
D. Thesis Organization.....	3
II. Overview and Related Work	4
A. Introduction to Steganography.....	5
B. Steganography and cryptography.....	7
C. Performance and Evaluation Measures	8
D. Steganography in Spatial Domain.....	10
E. Steganography in Transform Domain	11
III. Proposed Methods	14
A. Adaptive LSBM-R using Error Images	16
B. Modular Steganography with the help of error images.....	20
C. Double Phase Modular Steganography with the help of error images.....	27
D. Parallel modular steganography using error images	33
1. Secret partition size estimation	37
2. Secret partition addressing	41
3. Parallel pre-processing phase.....	42
4. Block-wise parallel algorithm	43
IV. Experimental Results	44
A. Adaptive LSBM-R using Error Images.....	44
B. Modular Steganography with the help of error images.....	49
C. Double Phase Modular Steganography with the help of error images.....	56
D. Parallel modular steganography using error images	62

1. Speedup analysis	66
2. Pre-processing phase analysis	66
3. Block-wise speedup analysis	67
V. Conclusion	70
Bibliography	72
ABSTRACT	76
ACKNOWLEDGMENT	78

List of Figures

Fig. 2.1 Information encoded using Morse code along with the riverside onto the grass [7].....	5
Fig. 2.2 Graphical Representation of the Prisoner's Problem [14].....	6
Fig. 2.3 Competing factors in Steganographic systems [16].....	7
Fig. 3.1 Process of calculating Base matrix for 4×4 pixels of a typical cover image	17
Fig. 3.2 Structure of the proposed embedding method 1	18
Fig. 3.3 Structure of the proposed embedding method 2	24
Fig. 3.4 Structure of the proposed embedding method 3	29
Fig. 3.5 Embedding algorithm for a pair of pixels [5].....	37
Fig. 3.6 Embedding example	40
Fig. 3.7 Relative Bases and Addressing matrices	41
Fig. 4.1 The cover image from image database called ‘Lena’	47
Fig. 4.2 The error image (left) and the Base matrix (right) (IQF 40, payload 0.4 bpp, 56.33 dB) ...	48
Fig. 4.3 Third cover image from BOSS 1.01, “3.pgm”	52
Fig. 4.4 Error image (a) and <i>Base</i> matrix (IQF: 8, Payload: 1 bpp, 52.69 dB) (b)	53
Fig. 4.5 Error image (a), and <i>Base</i> matrix (IQF: 1, Payload: 1 bpp, 51.54 dB) (b)	53
Fig. 4.6 Standard grey-scale images frequently used in classical steganography.....	56
Fig. 4.7 Chi-square attack on two stego images embedded with 1 and 4 bpp by Thien and Lin [19]	59
Fig. 4.8 Chi-square attack on two stego images embedded with 1 and 4 bpp by DPMS-E.....	59
Fig. 4.9 First cover image from image database BOSS 1.01 called “1.pgm”	60
Fig. 4.10 Left, the error image, and right, the <i>Base</i> matrix (IQF 83, Payload 0.4 bpp, 57. 42 dB) ..	61
Fig. 4.11 Left, the Error image, and right, the <i>Base</i> matrix (IQF 20, Payload 0.4 bpp, 57. 28 dB) ..	61
Fig. 4.12 Maximization point shown at number of PUs	69

List of Tables

TABLE 3-1. A NUMERIC ILLUSTRATION OF EMBEDDING AND EXTRACTING.....	19
TABLE 3-2. PARTITION SIZES AND PSNR VALUES; PAYLOAD: 0.2 BPP; IQF 70; FIRST IMAGE OF BOSS 1.01	22
TABLE 3-3. NUMERIC EXAMPLE OF EMBEDDING AND EXTRACTION.....	26
TABLE 3-4. A NUMERIC ILLUSTRATION OF EMBEDDING AND EXTRACTING.....	32
TABLE 4-1. DETECTABILITY COMPARISON BETWEEN THE PROPOSED METHOD AND THE HUGO (T = 90) [55]	45
TABLE 4-2. DETECTABILITY COMPARISON BETWEEN THE PROPOSED METHOD, HUGO, AND EXTENDED HUGO	46
TABLE 4-3. THE GLOBAL P_E OF EMBEDDING METHODS RESISTING 2ND-ORDER SPAM AND SQUARE+ H^X	46
TABLE 4-4. EMBEDDING TIME (IN SECONDS) COMPARISON.....	47
TABLE 4-5. DETECTABILITY COMPARISON BETWEEN MS-E AND HUGO (T = 90) [55].....	50
TABLE 4-6. THE GLOBAL pe OF FOUR EMBEDDING METHODS RESISTING 2ND ORDER SPAM AND SQUARE+ H^X	50
TABLE 4-7. THE GLOBAL pe OF ADAPTIVE ± 1 STEGANOGRAPHY IN EXTENDED NOISY REGIONS [62] AND MS-E.....	51
TABLE 4-8. EMBEDDING TIME (IN SECONDS) OF HUGO, EXTENDED HUGO AND MS-E USING “LENA”	51
TABLE 4-9. MS-E EXPERIMENT RESULTS WITH DIFFERENT IQFS FOR THE SAME PAYLOAD	55
TABLE 4-10. DETECTABILITY COMPARISON BETWEEN DPMS-E AND HUGO AND EA APPROACHES	57
TABLE 4-11. A COMPARISON TO THE CLASSICAL STEGANOGRAPHIC SCHEMES IN TERMS OF PSNR VALUE	58
TABLE 4-12. DPMS-E EXPERIMENT RESULTS WITH DIFFERENT IQFS FOR THE SAME PAYLOAD	61
TABLE 4-13. AVERAGE TESTING ERROR OVER 10 SPLITS (pe) COMPARISON ON BOSS 1.01 DATABASE.....	63
TABLE 4-14. GLOBAL pe WHILE RESISTING 2ND ORDER SPAM AND SQUARE+ H^X FEATURES.....	64
TABLE 4-15. GLOBAL pe OF ADAPTIVE ± 1 STEGANOGRAPHY IN EXTENDED NOISY REGIONS [62] AND MS-E.....	64
TABLE 4-16. TIME PERFORMANCE COMPARISON (IN SECONDS)	65
TABLE 4-17. PRE-PROCESSING PHASE INSTRUCTION-TYPE ANALYSIS.....	67
TABLE 4-18. BLOCK-WISE INSTRUCTION-TYPE ANALYSIS	68
TABLE 4-19. MAXIMIZATION LOOK-UP TABLE.....	68

List of Acronyms

BPP	Bits Per Pixel
PSNR	Peak Signal to Noise Ratio
IQF	Image Quality Factor
dB	Decibel
BOSS	Break Our Steganographic System
SPAM	Subtractive Pixel Adjacency Model
LSB	Least Significant Bit
LSBM-R	LSB Matching - Revisited
HUGO	Highly Undetectable Stego
EA	Edge Adaptive
JPEG	Joint Photographic Experts Group
PRNG	Pseudo Random Number Generator
PVD	Pixel Value Differencing
BPCS	Bit-Plane Complexity Segmentation
POV	Pair of Values
MBNS	Multiple Base Notational System
OEF	Optimal Extension Fields (Overflow Embedding Factor)
MSE	Modular Steganography using Error Images
DPMSE	Double Phase Modular Steganography using Error Images

초 록

압축 이미지와 원본 이미지의 차이를 이용한 고도의 모듈러 정보 은닉 기법

아프라크테 마수드
 지도교수: 이정아 교수, Ph.D.
 컴퓨터공학과
 조선대학교 대학원

HUGO, 확장된 HUGO, Edge Adaptive(EA), embedding in extended noisy regions and LSB-Matching revisited 기법 등 대부분의 최신 스테가노그래피 기법은 더 나은 은닉성을 유지하기 위해 점점 복잡해지고 있다. 이 기술들은 목표가 되는 화소 안에 숨길 수 있는 비밀 비트(Secret Bits)들의 수를 추정하기 위해 주변 픽셀을 사용한다. 그러나 주변 픽셀들과 커버이미지의 가장자리를 고려하는 연산은 시간이 많이 소모되는 복잡한 연산이다. 그 결과로 Stego-image 는 SPAM 과 Square+ h^x 같이 추출된 Steganalysis 의 특징 설정에 의해 적게 탐지되는 결과가 도출되었다. 고도의 정보은닉 수준을 달성하기 위하여 연산 실행시간은 더 필요하게 된다. 최신기술들은 순차적으로 픽셀에 정보를 은닉하며, 변경된 픽셀정보는 다음 픽셀의 정보은닉을 위한 계산에 이용된다. 이러한 데이터의 종속성은 병렬 실행을 어렵게 한다.

본 연구의 주요 목표는 고도의 정보은닉을 달성할 뿐만 아니라 지금까지 개발된 최신기술들의 접근법보다 수행시간 면에서 더 뛰어난 특징을 가지고 있는 효율적인 스테가노그래피 기법을 찾아내는 것이다. 이것을 달성하기 위해 커버 이미지에서 8×8 화소마다의 비 중첩 블록에 대해 그 각 각의 JPEG 압축 된 화상과 원 화상의 차이인 오류이미지를 이용한다. 이 오류 이미지는 어느 픽셀 위치에 얼마만큼의 비밀 비트들이

은닉될 수 있는지를 가이드할 수 있는 맵이 된다. 이와 더불어 비밀 비트들(페이로드)의 사용량은 매립 처리를 초기화하기 전에 발신자에게 미리 알려 질 수 있다. 그리하여 화소들의 개별블록을 각 각 처리할 여러 장치는 해당 장치에서 은닉하여야 할 정보의 미리 할당된 부분인 시작과 끝을 쉽게 판별할 수 있다. 따라서 모든 처리 장치들은 정보은닉 작업을 동시에 작동시킬 수 있으며, 수신부에서 이러한 은닉정보의 추출도 용이하다.

본 논문에서 제안된 방법은 수준 높은 정보 은닉기법이라는 것이 증명되었다. 왜냐하면 우리의 제안된 방법들은 산출된 Base 행렬을 가이드로 이용하여 커버 이미지에 Secret Bit 를 적응적으로 분배하기 때문이다. 이러한 가이드는 커버이미지의 질감을 이용하는 것으로, 비밀 비트들이 적절하게 배분되어 해당 픽셀에 은닉된다. 그리고, 모든 32-bit 단위의 기밀 블록들의 10 진수 값은 커버 이미지 픽셀에 할당된 Base 값으로 나누어 은닉할 값을 작게 만들 수 있다. 모든 32-bit 기밀 블록들은 어떤 임의의 숫자로 나누어지기도 하고, 나머지 값을 이용하여 더 작게 표현된다. 나머지를 이용하여 이렇게 은닉할 정보값을 작게 하면, 화소값의 변형 또한 작아지게 되며, 결과적으로 커버이미지의 변형도 작아진다. 따라서, 정보은닉은 작은 변형으로 가능하게 되어, 높은 PSNR 을 달성할 수 있고, 고도의 정보은닉도 가능하게 된다.

지금까지 제시된 스테가노그래피의 최신기술들이 정보은닉을 위한 계산에서 픽셀정보의 데이터 종속성 때문에 병렬로 실행하는 것이 힘든 것과 달리, 본 논문에서 제안한 방법은 병렬화가 용이함을 “Parallel Modular Steganography Using Error Images (P-MSE)”을 이용하여 보였다. P-MSE 에서는 처리해야 하는 해당 은닉정보들을 모든 처리장치들에게 배분하여 동시 실행할 수 있으며, 수신부에서 추출될 때 비밀비트들의 혼합도 피할 수 있게 된다. P-MSE 를 110 개의 처리장치(PU)에 실행하면 실행속도는 55 배 향상된다.

I. Introduction

A. Research Motivation

Steganography detectability has become a major concern as the secret message size increases, so that most of the state-of-the-art methods have become more complex to maintain a greater undetectability such as HUGO (highly undetectable steganography), Extended HUGO, Edge Adaptive (EA), ± 1 embedding in extended noisy regions and LSB-Matching revisited. They employ surrounding pixels to estimate the number of secret bits to conceal in a typical target pixel; however, they endure complex, time-consuming calculations considering the neighboring pixels and edge regions of a cover image so that the resulted stego-image might become less detectable by extracting steganalysis feature sets such as SPAM (Subtractive Pixel Adjacency Model). SPAM features are extracted from the neighboring pixels of every pixel of the cover image and will be input to ensemble classifiers which is proven to be the most powerful steganalysis attack. Among state-of-the-art methods, HUGO is shown to be less detectable; however, it is more detectable using other extracted features such as Square+ h^x . With regard to this issue, extended-HUGO simply tries to be less detectable against 2nd order SPAM features by extending its calculations for square features. For the same reason, it becomes more complex and takes longer time to run compared to original HUGO. In terms of maximum payload size in bits, all of the state-of-the-art can embed up to 1 bpp while tending to embed in edges rather than regions with less textural complexity so that their resulted stego-images would become more difficult to detect by ensemble classifiers while they take a longer execution time. Hence, the consequence of increasing complexity in the functionality of Steganography applications accelerated the demand of less complex systems while detectability level is less.

For efficient implementation of steganographic techniques with higher time performance, it is important to propose a method which has the potential to be implemented in parallel preserving higher undetectability level. The problem is that the above mentioned methods are reliable on the surrounding pixels and their stego-images will be more detectable if they are bound to use only a block of pixels to make the proper decision on amount of embedding. Therefore, they might not be estimating the right payload unless the secret bits are totally embedded once which is not

reasonable in parallel implementation. To achieve higher time performance in a possible parallel implementation, each processing has to be notified about the amount of secret bits that can be embedded in one individual block of pixels prior to beginning the embedding process and it has to avoid data embedding dependency so that every block of pixels from the original (cover) image can be embedded at the same time while other blocks simultaneously embed their pre-assigned number of secret bits.

B. Research Objectives

The main goal of this research is to come up with a steganography method with a less detectability level compared to the state-of-the-art methods while we can embed not only up to 1 bpp but also 4 bpp. Furthermore, we expect the to-be-proposed method to have the potential to be parallelized so that we could save more execution time compared to the state-of-the-art techniques.

C. Thesis Contributions

In order to achieve our goals, we formulated and designed a method using an error image by differentiating an original image from its respective JPEG compressed image. The error image will be simply considered an embedding map implying where and to what extent secret bits should be embedded. Furthermore, the amount of secret bits (payload) must be known to the sender prior to starting the embedding procedure so that each processing unit assigned to every individual block of pixel will easily select its respective partition of secret bits to embed so that all processing units will be run in parallel while there will be no scrambling of the secret bits after extraction. There are four approaches proposed in this work as follows:

1. “Adaptive least significant bit matching revisited with the help of error images” has shown how ensemble classifiers would become confused by the JPEG IQF and JPEG decompression artifacts. It is also proven that how undetectable a stego-image will become if we distribute the secret bits through the cover image more adaptively and select the right pixels to hold secret bits using a calculated Base matrix as a guide. The guide exploits the texture of the cover image so that the secret data bits are distributed through the white target pixels while trying to affect the cover image the least by utilizing LSBM-R method for every pair of target pixels.

The execution time of the proposed method is proven to be around two times faster than that of the HUGO methods with two different settings.

2. MS-E which stands for “Modular steganography with the help of error images” and simplifies DPMS-E, imposes lesser changes on the cover image pixel values because the decimal value of every 32-bit secret block becomes smaller if it is divided into multiple base numbers. Hence, the decimal value of every 32-bit secret block is broken down into much smaller remainders and a small change in pixel value is required, consequently. Thus, less change leads to a greater PSNR value and lower detectability while more secret bits are embedded utilizing either LSBM-R or MBNS method. Unlike the original MBNS method, the greatest change to be enforced on each pixel is computed from an error image and the smallest impact of the change is minimized using LSB matching.
3. DPMS-E stands for “Double phase modular steganography with the help of error images” and it applies a second phase of implicit compression using pseudo random numbers. Thus, unlike LSB matching-based methods, EA and HUGO methods that can only embed up to 1 bpp, this scheme could even embed up to 4 bpp.
4. We proved that ensemble classifiers have more probability of error to detect the resulted stego-images. Furthermore, the execution time of the proposed method in serial was verified to be approximately 2.32 times faster than HUGO. Moreover, we explained that owing to their embedding dependency, EA and HUGO cannot be implemented in parallel while the proposed “Parallel Modular Steganography Using Error Images (P-MSE)” has a speedup of over 55 times using 110 PUs.

D. Thesis Organization

The rest of this thesis is organized as follows: Theoretical background of Steganography schemes are introduced in Chapter I. The overview of previous related works is given in chapter II. The overall block diagrams of four proposed methods are discussed in chapter III. Experimental results related to every one of the four proposed methods are discussed and evaluated in Chapter IV. Finally, the conclusion with future goals are discussed in the last chapter.

II. Overview and Related Work

Since the early days, individuals send messages as a form of communication. In some situations, it was supposed to be sent by means of a pre-secured method hoping that no other person except the desired receiver could get the meaning of the content. So, people always hide their favorite messages by a variety of methods [1] [2]. For instance, ancient Greeks put the message on the underlying wood of a tablet and then covered it with some wax to be considered as a kind of useless thing. Another method is that a messenger shaved the head and wrote something on his head. After his hair grew back, he was sent to somewhere and nobody could ever detect or guess that if he had any message embedded on his head [1]. Another secure method is to use invisible inks and it was widely used in World War II [1]. With invisible ink, a seemingly innocent letter could include another written message between the lines [3]. In addition, there were some innovative methods to transfer these secret messages by means of an ordinary message, like what a German spy did in World War II. He transferred null-cipher message that included one sentence. The receiver could extract the second letters of each word to find another sentence. The secret message was the statement: “Pershing sails from NY June 1”. Here is the long cover message sent: “Apparently neutral’s protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suits and vegetable oils.” [1]

In comparison with today’s technologies, lots of new interesting methods are appeared and computers play the most important role in this section. It is worth mentioning that a great deal of information hiding systems deal with media files such as video, sound and different types of images formats. So, digital media can be transmitted over networks. A secured transmission is mostly wanted in such an area in order to protect secret messages. Hence, computers with a connection to internet are considered the major communication media forming a kind of huge virtual world including many kinds of people. These people could be of any type holding different jobs; however there are many precious works shared online holding some author property rights that must be preserved. The problem is among all of these people, there are always sniffers watching the network so that they might get a chance to either change or remove these copyrights in favor of their illegal usage. Thus, it is important to have such materials protected in such a way

that the existence of copyrights or any form of secret information becomes imperceptible to such potential attackers.

A. Introduction to Steganography

As in the real world, regardless of adding security locks protected by passwords or putting them in a safe, one of the first ways of keeping anything intact is to hide it. The key to hide any binary information has led to the development of a science called Steganography. The word “Steganography” is derived from the Greek word, ‘Steganos’ meaning hidden or concealed. Thus, “Steganography” stands for “concealed writing”. It is about hiding the existence of any information in such a way that it does not attract any attention and the media keeps the same innocent look as the original does. Steganography is a powerful security tool that provides a high level of security and it becomes more reliable if it is combined with encryption [4]. In steganography, the media with and without hidden information are called stego-media and cover media [5] [6].



Figure 2.1 Information encoded using Morse code along with the riverside onto the grass [7]

As stated earlier, the word “steganography” is basically a Greek word meaning “covered writing”. There are many cases, however the earliest might be in the 5th century when the Greek tyrant Histiaeus tattooed a message on his slave’s shaved head. As soon as the slave’s hair grew, he let him go with the message to his son-in-law, Aristagoras, who was in Miletus [7]. Additionally, there was a wax covered tablet that they had to remove the wax in order to write a secret message. Later on, for reading the message, the receiver had to peel the wax off to read the message. They also had different methods to write a message between lines using invisible ink which could be made of substances like fruit juice, urine, or milk [8]. Invisible inks were also widely used in World War II. The Germans innovated a micro dot way of writing during the same time [9]. In 1550, an Italian mathematician, Jerome Cardan, proposed a secret way of writing using a paper mask in which holes had to be filled out with the message characters. For the unused space between the characters, they would make some sentences including the same characters to convey an innocent meaning without the mask. In Saudi Arabia at the King Abdulaziz City of science and technology, a project was initiated in order to translate the secret writings into English from Arabic manuscripts as old as 12 centuries [10]. Some other manuscripts were discovered in Turkey and Germany [11].

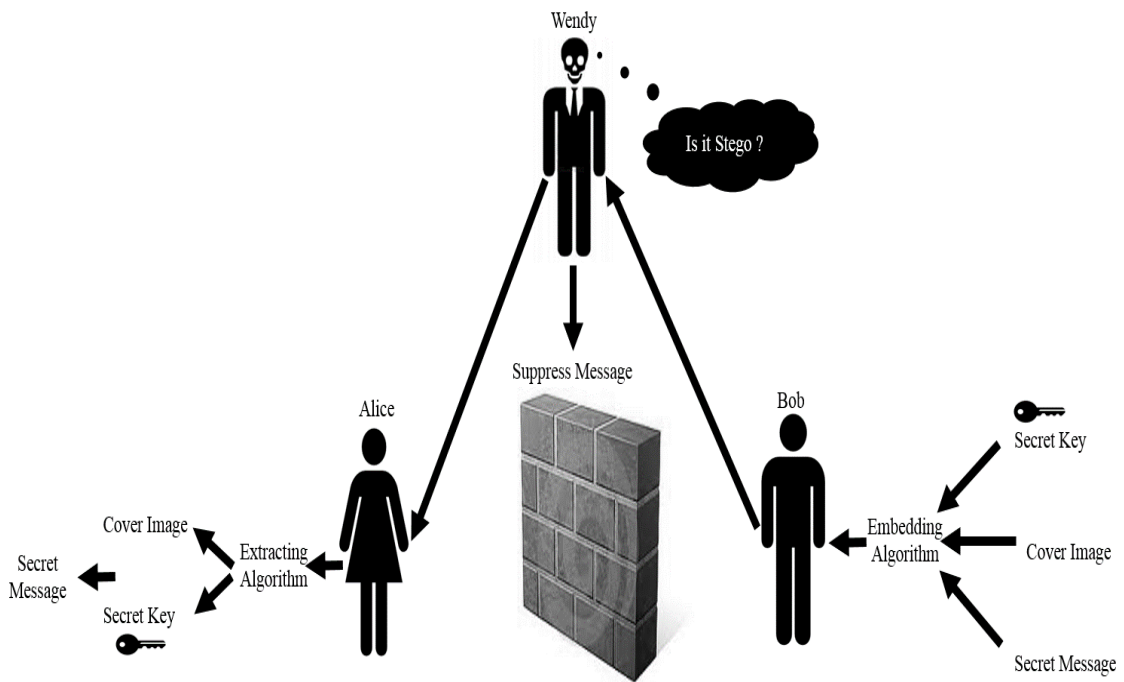


Figure 2.1 Graphical Representation of the Prisoner's Problem [14]

In 1945, Morse code was concealed and encoded in a drawing (Figure 2.1) onto the stretch of grass along the side of the river. The long and short grasses would be denoted as line and point.

B. Steganography and cryptography

Steganography and cryptography are frequently mistaken with each other. Steganography differs from cryptography. Cryptography tries to secure communications by changing the data into a form that an unwanted attacker would not understand its meaning. On the other hand, Steganography conceals the existence of the secret message itself, which makes it difficult for human vision system (eyes) to figure out where the message is located. Encrypted information may sometimes draw attention, while imperceptible information will not. Accordingly, cryptography cannot guarantee the most secure communication, however it can only be part of the solution [8] [12].

As a typical Steganographic example presented in [13], Alice and Bob are defined as two prisoners who have to send some information in a hidden way that Wendy does not understand. They must communicate in a way that no one can reveal the message by looking at their objects

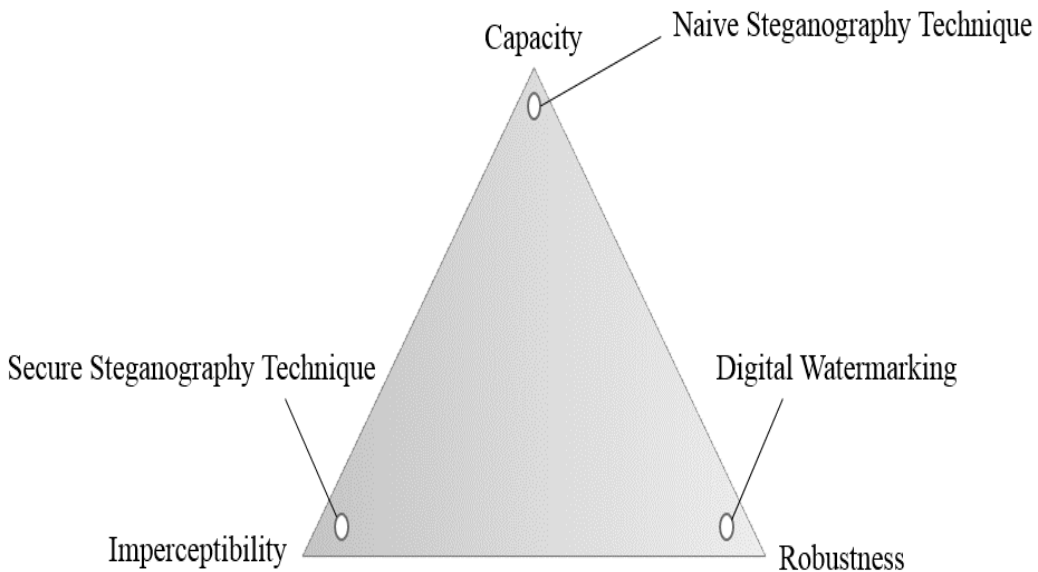


Figure 2.2 Competing factors in Steganographic systems [16]

(Fig. 2.2). Thus, it can be said that the major purpose in Steganographic is to be hidden from human vision system and any algorithm would be considered imperceptible enough as long as it has more invisibility to human eyes. This imperceptibility would be a result of high image quality. In other words, if the image quality is degraded, it would be meant as some data has been put into the cover image. The more quality drops then, the more data should have been embedded.

C. Performance and Evaluation Measures

There is a trade-off between imperceptibility and capacity payload. The more a Steganographic method embeds, the lesser imperceptibility for stego-image would be achieved. As in watermarking, unlike steganography, degree of robustness to either visual attacks or intentional added geometric noise is much more of concern rather than capacity and imperceptibility. Furthermore, the original watermark (like secret bits in steganography) has to be embedded in a cover image. The watermark size is far smaller than the maximum capacity applied in steganography. Since any probable attack is always expected for a watermarked image, the main purpose in watermarking is to protect the hidden watermark namely owner's property right or a load of confidential data from sniffers [14]. This difference is well shown in Fig. 2.3.

Generally, there are three competing factors in Steganographic systems as follows:

Imperceptibility (Undetectability) is considered as the main evaluation factor for steganography. There is a common way to measure the quality of an image of size $M \times N$ pixels by using a commonly used factor called PSNR (less commonly used metrics such as Watson [15] and universal Q index [16]). PSNR stands for Peak Signal to Noise Ratio. First, we need to calculate the error which is resulted from the intentional change made by a steganography scheme. MSE stands for Mean Squared Error and it is computed between a cover image and a stego image using the following equation:

$$MSE = \left(\frac{1}{M \times N}\right) \sum_{i=1}^M \sum_{j=1}^N (a_{i \times j} - b_{i \times j})^2 \quad (1)$$

Where variable $a_{i \times j}$ refers to the value of a pixel at (i, j) coordinates of the cover image and similarly $b_{i \times j}$ is the corresponding pixel value in the stego image. Pixels cannot get a value greater than 255 (2^8 , 8-bit color depth). Finally, PSNR is calculated using the computed MSE below:

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

Where the result of PSNR is said in dB and experimentally a stego image with a PSNR value over 25 dB should be imperceptible enough to human vision system (HVS). However, this value might differ from person to person. In addition, there are two kinds of watermarking techniques namely visible or invisible. Unlike visible watermarked images that the watermark can be seen with one's eyes, Steganographic schemes can never be visible. Hence, undetectability is the first concern in steganography [14] [10].

1. Robustness measures the ability of any proposed scheme to survive any visual or statistical attacks (Steganalysis) to either stego image or watermarked image. The difference is that in steganography, higher PSNR is by itself depicting the robustness unless the existence of embedded information is revealed by any attack. On the other hand in watermarking, usually more sophisticated attacks would be applied to watermarked image in order to either change or remove the embedded watermark as it is already clear to the sniffer that there exists hidden information in a watermarked image. In other words, steganography fails when it is detected and watermarking fails as soon as the watermark is either removed or replaced [10].

Steganalysis is an analysis to reveal the existence of the secret bits in a typical stego-image. A typical Steganalysis would try to find either strange or similar steps namely in histogram of the stego-image. One of the preliminary steganalysis attacks is called Chi-squared attack [17] looking for pair of values in the stego-image histogram so that it will reveal the existence of secret bits as well as providing a probable embedding rate. However, the steganalysis results indicate that the performance of steganalysis would be affected by the JPEG quality factor and JPEG recompression artifacts. They make it more confusing to steganalysis schemes [18]. Due

to this fact, secret data in most of the Steganographic systems in transform domain seems to be embedded into the non- zero discrete cosine transform (DCT) coefficients of JPEG images [19].

2. Payload capacity is the other difference between watermarking and steganography. It refers to the size of secret message in bits. Maximum Payload capacity in Steganographic schemes is greater than payload capacity in watermarking methods.

To be added, there are two types of steganography techniques, namely blind and non-blind steganography. Blind steganography schemes are capable of extracting the secret message without access to the original image on the receiver side. On the contrary, non-blind Steganographic schemes require the original image in order to extract any data [14]. It is also supposed for non-blind schemes that the original photo could be easily found with respect to its similarity to the stego-image. Hence, non-blind schemes have some limitation on choosing the cover image and the receiver has to find the original cover image which is similar to the stego-image. The original image is not necessarily sent to the receiver.

D. Steganography in Spatial Domain

There are two domains available for steganography. The first domain is called Transform (Frequency) domain, such as Discrete Cosine Transform (DCT), Fourier Transform (FT), and Discrete Wavelet Transform (DWT). The second domain is called Spatial. The former includes mostly complicated algorithms along with indirect embedding of secret information in a transform space of the signal (e.g. in the frequency domain) [20]. While the latter usually has less complexity and it is easier to implement, it uses a direct least significant bit (LSB) replacement technique [10]. According to a list of methods presented in [21], data hiding by LSB is the most conventional type of steganography in spatial domain. It is based on manipulating the least significant bits of target pixels by directly replacing them with new bits [22] [23] [24] [25] [26]. Data hiding by MBPIS (Multi bit Plane Image steganography) is another Steganographic technique proposed to be robust to RS steganalysis methods [27]. Data hiding by QIM (Quantization index modulation) is quantizing an input signal x to the output y using a set of quantizers. It is applicable to both steganography and watermarking [28]. Data hiding by PVD (pixel value differencing) owns a high imperceptibility, as well as high payload capacity. It segments the cover image into non

overlapping blocks containing two connecting pixels and modifies the pixel difference in each block (pair) for data embedding [29]. The shape of the different histograms of the stego images is changed a great deal, so that it is vulnerable to some steganalyzers [30]. Thus, two methods are proposed to improve PVD. The first tries to exploit the concept of PVD and uses a patch reference table (PRT) as a guide to preserving the different histogram shape [31]. The second scheme also claims that there exists a loophole in the PVD method. Unusual steps in the histogram of pixel differences reveal either the presence or even the amount of a secret message. This modified PVD scheme avoids occurrence of such steps while [32] preserving the advantage of low visual distortion of the PVD. Another data hiding is based on GLM (Gray level modification) which is mapping data by modifying the gray level of the cover image pixels. It is kind of one-to-one mapping between secret data and selected pixels while using an odd and even concept [33]. Also, [34] works by dividing the cover image into blocks of same size and then embedding the message on the edges of the blocks depends on the number of ones in the most significant four bits of the pixel. Another work [35] proposed a scheme to embed the data. It first encrypts the secret data using a secret key and compresses it afterwards. The same key is used to apply encoding in the cover image, too. There are some theoretic models published, too. As [36] proposes a blind universal stego system. This stego model needs no knowledge of the cover media distribution, except that it is generated from independently repeated experiments.

As it can be seen from the results achieved by such methods, the PSNR values are compared between them to show to what extent the image quality is preserved for each stego-image. That is to say that the most powerful steganalysis methods are rarely applied to the stego images holding maximum payload (secret bits capacity).

E. Steganography in Transform Domain

There is an approach [37] in transform domain. It applies DWT to the gray-scale cover image. The secret message is also compressed by Huffman. The Huffman code bits are embedded where wavelet coefficients have high frequency in order to preserve the image quality. This method has proved a contribution in higher quality of stego image over [14] in which A DWT based approach is proposed in frequency domain and based on the different users' demands on the embedding capacity and image quality, their algorithm is divided into two modes and 5 cases. The secret

message also goes through a pre-processing and a well-designed mapping Table keeps the messages away from stealing. Finally, pre-processed secret messages are embedded in high frequency coefficients and lower coefficients are left intact. The scheme in [37] also claims a contribution over [38] in such a way that it tries to minimize the image distortion by exploiting the correlation between neighboring pixels to estimate the degree of smoothness or contrast of pixels. Those pixels located in edge areas, since they have higher variation, would be thought as tolerating larger changes so that more of secret data would be embedded with lesser change in the quality of stego image. This method is a blind Steganographic and does not refer to original cover photo in extraction time.

In spatial domain, data hiding by MBNS [39] segments the secret bits into several partitions of the same size of 32 bits each. The decimal value of each partition would be represented using multiple bases calculated in advance using the same variance of neighboring pixels as used in [37]. Therefore, every pixel in the cover image represents one base which is calculated already. By estimating the high variation around busy areas, particularly edges, the decimal values of each 32-bit block of secret data would be divided over bigger bases. The remainder of each division would be the amount of change to be made into the current pixel. However, this amount of degradation is so small that it makes less change in pixel values in smooth areas. Thus, human vision system (HVS) would be unable to see them. MBNS method has proved its approach to be more imperceptible in comparison to either PVD method [40] or Bit-plane Complexity Segmentation method called BPCS [41]. BPCS method computes the complexity of decomposed bit-planes and applies the idea in which the embedding is based on the fact that areas of low complexity such as, homogeneous color or simple shapes, would tolerate less change compared to complex areas. Another method [42] processes a secret image by splitting it into n shares to become hidden in n user selected camouflage images. By embedding fragile watermarks and use of parity-bit checking the fidelity of each processed camouflage image, called a stego-image, would be authenticated. In [43], it is shown that there is a problem regarding those n participants and that dishonest participants can easily manipulate the stego-image for successful authentication, but cannot recover the secret image. With this regard, the authors have presented another scheme to improve authentication ability that prevents such dishonest participants from cheating. Furthermore, as in [44], there are two data hiding schemes incorporating both run-length encoding and modular arithmetic. One is good to be applied for embedding simple data with long streams of repeating bits

and the other is good for embedding complicated data with short streams of repeating bits. The stego-image quality is controlled by arithmetic modular operation and the number of repeating bits in the secret data and the bit itself (run-length encoding) are recorded as well. Another scheme [45] has contributed with a better quality of stego-image over [44] [23]. It begins with dividing the cover image into non-overlapping two-pixel blocks and embeds using some modulus operations. In [46], a simple LSB based scheme is proposed to hide either a 256×256 or 256×512 image in a 512×512 host image based on modulus operation. In [47], another module-based LSB substitution method is discussed in which a lossless secret data compression is applied, too. This method also tries to conceal smoother areas of secret image by modifying fewer pixels thanks to the compressions made. Another scheme [48] was proposed to see whether it would be a right decision to embed the maximum amount of data in a host image or not in such a way that the degradation of host image would be the smallest. In this approach the host image is partitioned into non-overlapping blocks and passing an imaginary plane in some three critical pixels. The characteristics of this plane should not be changed after embedding the message. In another approach [49], the secret data is broken down into data blocks of variable length and each block is embedded in intermediate significant bits in addition to least significant bits of cover image. The embedding is in such a way that highest length data vector is embedded in lower order bit plane and vice versa. There is another approach [50] which is to apply two different ways of embedding in two different categories of a host image. Two component based LSB technique embeds in edge areas and an adaptive LSB Steganographic scheme embeds in smooth areas. The main principle in their work is that edge areas can tolerate more change compared to smooth areas.

The current work embeds in spatial domain due to simplicity in the algorithmic nature and ease of mathematical analysis. Furthermore, as stated and proved in [18], spatial domain techniques could carry the largest messages (embedding rate) compared to transform domains namely DCT-based embedding techniques [10]. The reason is that transform domain techniques can only embed in nonzero coefficients while all pixels can be utilized in spatial domain. Finally, in a comparison which is made using the same data set, a method is proposed that is able to embed the maximum reported size of embedded secret message while the change of pixels is less perceptible to human vision system and the most powerful steganalysis methods. Furthermore, it is proven that the proposed method is more robust, too.

III. Proposed Methods

Steganography conceals the very existence of any secret information in terms of bits. It usually embeds secret data into any type of digital cover media, such as an image, video, audio, and so on. The manipulated image or video looks innocent, and the message cannot be detected with the human eye. On the other hand, exposing the existence of any hidden information in a cover image is what steganalysis does. Such steganalytic algorithms are able to estimate the probable existence of secret bits in different ways. If steganalysis detects the hidden information with a minimum probability of testing error, the steganographic scheme has been broken.

There are two factors that have to be considered in designing a modern steganographic scheme, namely embedding rate and undetectability, with a trade-off between them. The higher the embedding rate, the greater the detectability. Some approaches are more concerned about embedding capacity, with higher imperceptibility levels provided by greater peak signal-to-noise ratio (PSNR) values, but there are many that try to be more undetectable rather than having higher PSNR values. Least significant bit (LSB) replacement [26] embeds the information in the LSB of a pixel, independent of its value. The LSB is directly replaced by the secret bit. This adds some unwanted statistical artifacts, by which the existence of secret bits can be exposed. Such artifacts are paired with values in a histogram of the stego image made by the LSB replacement method. This makes detection easier for a chi-square attack [51]. LSB matching (LSBM) [52] applies minor changes after LSB replacement because it randomly increments or decrements the LSB of a pixel according to a pseudo-random number generator if the secret bit does not match the pixel's LSB. It is also called non-adaptive ± 1 embedding. Unlike LSB replacement and LSBM, which deal with the pixel values independently (non-adaptively), LSBM revisited (LSBMR) [53] is another approach that modifies the LSBM algorithm in such a way that the choice of incrementing or decrementing the pixel value is no longer random. It performs the operation by using a pair of pixels as a unit. The first pixel value changes in such a way that the first secret bit is saved in its LSB and the second secret bit equals a function of the two modified pixel values. Both LSBM and LSBMR are undetectable with a chi-square attack because, statistically, the probability of change is the same as the increment/decrement performed, either randomly or by using a function. Although the asymmetry artifacts of LSB replacement are almost completely avoided, they can still be detectable using stronger steganalytic attacks. These LSB-based approaches do not consider the

difference between the pixel and its neighbors. Edge adaptive (EA) image steganography [54] embeds secret bits based on the LSBMR method. It begins embedding from the edge regions as far as possible, while keeping other smooth areas as they are. The maximum embedding capacity of this approach is limited to 1 bpp while the visual quality and security of stego images are proving to be better than those of LSB-based and edge-based methods. Another approach is to use high-dimensional image models to perform highly undetectable stego (HUGO) [55]. The source code is available at Break Our Steganographic System (BOSS) website [56]. This method calculates distortions corresponding to modification of each pixel by ± 1 and sets the stego image pixel value as the minimum of these numbers. The best embedding order starts from pixels with the high cost of embedding to the lowest, which is ascertained by an additive distortion function. The default parameters of the distortion function were $\sigma = 1$ and $\gamma = 1$, and the switch $-T 90$, which means that the distortion function was computed with threshold $T = 90$ used in the BOSS challenge [56]. Security of HUGO is evaluated by training support vector machine (SVM)-based steganalyzers utilizing second-order subtractive pixel adjacency model (SPAM) features [57]. A filter suppresses the stego image content and exposes the added noise in the stego image. Dependencies between neighboring pixels are modeled as a higher-order Markov chain. The resulting sample transition probability matrix is a vector feature that is a SPAM of covers. The second-order Markov chain results in a second-order SPAM including 686 features for a typical stego image. In this work, the undetectability level of the mentioned methods is benchmarked utilizing the second-order SPAM as input features to state-of-the-art ensemble classifiers [58]. They proved to have better performance compared with SVM-based steganalyzers in terms of both time and accuracy. The classifier has to be trained with a database of pictures to detect the information more accurately, so BOSS [56] version 1.01 was used to create sufficient stego images. The BOSS database consists of 10 000 8-bit grayscale images at 512×512 pixels. Kodovský et al. [59] use $T = 255$ in order to remove a weakness of HUGO with original threshold value $T = 90$ that makes the algorithm vulnerable to first-order attacks because of an artifact present in the histogram of pixel differences. Thus, they have compared the detection error for six different payloads (0.05, 0.1, 0.2, 0.3, 0.4, and 0.5 bpp) and two settings of HUGO when using the histogram features (dim 4), the SQUARE feature (338) [60], and a combination of both (SQUARE+ h^x), which is equal to 342 features. HUGO embeds in those places of the cover image where it is hard to model, and that is why they are more secure and less detectable compared with ± 1 embedding [60].

The first proposed method embeds in the spatial domain because of the simplicity of the algorithmic nature and ease of mathematical analysis. Also, spatial-domain techniques can carry the largest messages (embedding rate) compared with transform domains, namely discrete cosine transform (DCT)-based embedding techniques and LSB-based approaches [10]. The reason is that transformation domain techniques can only embed in nonzero coefficients, whereas all pixels can be utilized in the spatial domain. Modern steganographic schemes are supposed to be undetectable, rather than stressing the PSNR value, so the current scheme also shows the undetectability level. The detectability level is shown by ensemble classifiers using SQUARE+ h_x feature (dim 342) and second-order SPAM feature (dim 686). The algorithm uses LSBMR for embedding the secret bits; however, unlike LSBMR method, the target pixels are adaptively chosen based on a preprocessing phase.

A. Adaptive LSBM-R using Error Images

State-of-the-art steganographic schemes, such as highly undetectable stego (HUGO) and its extended version, aim at least significant bit (LSB)-based approaches embedding up to 1 bpp and are more concerned about the undetectability level of the stego image rather than the peak signal-to-noise ratio. The complexity of such methods is quite high, too. In this work, a steganographic scheme is proposed in a spatial domain that takes advantage of error images resulting from applying an image quality factor (the same as the ones used in JPEG compression) in order to find the pixels where a slight change could be made. The amount of change is adaptively embedded using LSB matching revisited. We show that our proposed method is less detectable than HUGO and almost as undetectable as the extended HUGO while it has a greater time performance.

The proposed method pre-processes the cover image sized $M \times N$ pixels in order to create an error image. The required error image is computed by applying a suitable image quality factor (IQF), like the one suggested in JPEG compression, as follows:

- 1) Partition the cover image into non-overlapping blocks sized 8×8 pixels.
- 2) Apply DCT to every block using the JPEG standard quantization table. The cover image can be transformed using a JPEG IQF, which can be any float number between

- 0 (the least expected quality or full substitution of pixel values) and 100 (identical image without any change).
- 3) Apply inverse DCT to the matrix of coefficients resulting from DCT of each block.
 - 4) The compressed image resembles the original image. It consists of some imperceptible added noise. The amount of noise has a direct relation to compression level and embedding rate. The larger the embedding rate, the more noise.
 - 5) The noise becomes greater if a smaller IQF is employed. In this regard, for every pixel_{i,j} from CoverImg_{M×N} where $i \leq M$ and $j \leq N$:

$$\text{ErrorImg}_{i,j} = \text{CoverImg}_{i,j} - \text{CompressedImg}_{i,j} \quad (1)$$

According to the pixel values from the error image, multiple bases are calculated for every corresponding pixel. Base $M \times N$ represents the matrix of multiple bases:

$$\text{Base}_{i,j} = \log_2(|\text{Errorimg}_{i,j}| + 1) \quad (2)$$

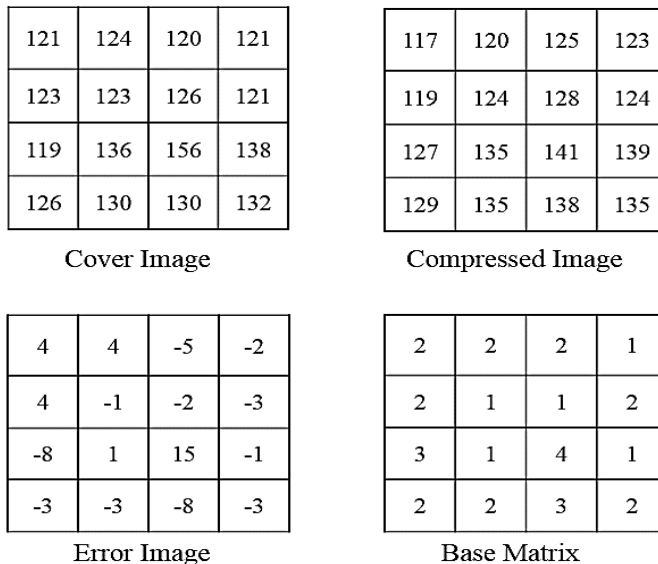


Figure 3.1 Process of calculating Base matrix for 4×4 pixels of a typical cover image

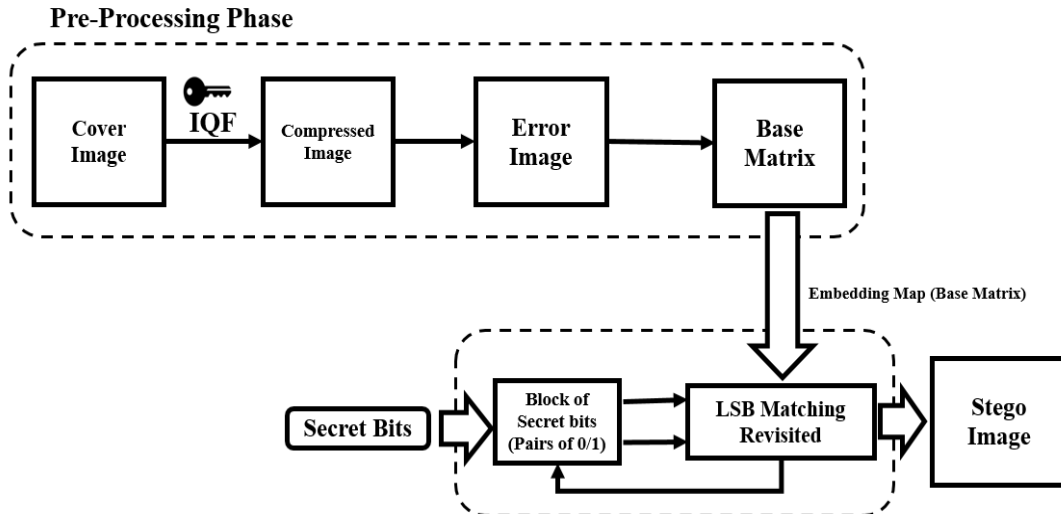


Figure 3.2 Structure of the proposed embedding method 1

The whole procedure of embedding is illustrated in Fig. 3.1. According to the embedding order, which is line by line from top to bottom of the cover image, the algorithm reads the corresponding Base from Base $M \times N$. Pixels with a Base value less than 2 are skipped and left the way they are. Thus, only pixels with a base greater than or equal to 2 are allowed to be embedded. The secret bits are partitioned into non-overlapping 2-bit blocks. We applied 2-bit blocks to embed secret bits as described by LSBMR method. Thus, for every 2-bit block, LSBMR [53] is applied to minimize the embedding effect for two pixel units. Note that, unlike the LSBM concept, there might be some pixels between two such pixels to which LSBM is not applied.

The structure of the proposed scheme is modeled in Fig. 3.2. The embedding is repeated until every 2-bit block of secret bits is processed. If all pixels from the cover image have been processed and some of the secret bits have not been embedded yet, IQF has to be decreased toward 0.01. If the secret bits are still partially embedded, the cover image cannot take more secret bits, and that would be the maximum payload of the current method (up to 1 bpp).

The algorithm extracts the secret bits from the stego image pixels using the same IQF and Base matrix. The stego pixels with a corresponding base greater than 2 implies where secret bits are embedded. Using the embedding map (Base matrix), every pair of secret bits can be easily extracted using LSBMR algorithm.

Table 3-1. A numeric illustration of embedding and extracting

<i>Secret bits</i>	: Only two bits can be embedded									
	-	0	-	-	-	0	-	-	-	-
<i>Pre-Processing Phase</i>	Cover Image pixel values	80	87	101	157	165	213			
	Corresponding Base numbers	0	3	1	0	2	0			
<i>Embed using LSBMR</i>	Stego Image Pixel Values	No Embedding	86	No Embedding	No Embedding	165	No Embedding	No Embedding	No Embedding	No Embedding
<i>Extraction</i>	Using LSBMR	No Extracting	LSB(86)= 0	No Extracting	No Embedding	LSB($\lfloor \frac{165}{2} \rfloor + 165$) =0	No Embedding	No Embedding	No Embedding	No Embedding

The embedding procedure is illustrated in Table 3-1. Four pixels with grayscale values 80, 101, 157, and 213 are excluded because their corresponding Base values are either 0 or 1. Pixels with values of 87 and 165 have formed a pair of pixel units to be given to LSBMR [53] for embedding (zero in both of them). The pixel with a value of 165 is identical to the original pixel value. In this case, only one pixel is spared from being changed, yet the message bits are still extractable. Finally, the extracted values are shown in boldface. According to the LSBMR algorithm, in the worst case, only one of the two pixels is incremented or decremented.

B. Modular Steganography with the help of error images

In this section, another spatial domain-based modular steganographic scheme is proposed that also uses the error image that results from applying an image quality factor to the cover image similar to the one used in JPEG compression—in order to discover the pixels where a few secret bits could be adaptively inserted. Similar to the first proposed method, the amount of change in a pixel is based on the initial error image in such a way that the processed partitions of secret bits could be embedded using not only LSB matching revisited (first proposed method), but also a modular function for two different embedding cases. This helped the new scheme embed more than 1 bpp and up to around 4 bpp. We also show that our proposed method is less detectable than three state-of-the-arts methods, edge adaptive (EA), highly undetectable steganography (HUGO, with threshold parameters of either 90 or 255), and Adaptive ± 1 Steganography in Extended Noisy Regions. The detectability level is evaluated by the most recent state-of-the-art steganalysis attack, the ensemble classifier method, with a second-order subtractive pixel adjacency model (SPAM, dim 686) and SQUARE + hx features (dim 342) given as their input.

It partitions the cover image into 8×8 pixel blocks and applies an image quality factor (greater than zero and less than 100) to every partition. The discrete cosine transform (DCT) of every partition is quantized using the image quality factor, and the compressed cover image is reconstructed by applying the inverse DCT to every quantized partition. Finally, the error image differentiates the cover image from the compressed cover image. A matrix is computed according to the values of error image, called the Base matrix. The Base matrix consists of multiple bases and reveals the places and number of secret bits that can be embedded in the cover image pixels. Every 32-bit partition of secret bits is represented in multiple-base notational system [39] in the base

matrix and is embedded adaptively using either a modular function or LSB matching revisited [53]. Modern steganography schemes associate greater importance to undetectability rather than high PSNR value. Hence, the current scheme is also shown to be very undetectable to modern steganalysis attacks while maintaining high PSNR values for the stego images.

When an image is compressed using an image quality factor (IQF) similar to the one used in JPEG compression, some of the pixels in the original image will be slightly altered. This slight change is considered a noise, which is invisible to the human vision system. Thus, the places where such noise occurs could be the best candidate for holding secret data. This implies that the noise that occurs in the error image suggests where and to what extent we can insert secret bits. Unlike other methods described earlier, the current method does not require a noisy function or the calculation of many features to render the stego image more undetectable. Further, it can embed in any image adaptively (sensitively to texture) in such a way that less value changes are tolerated by pixels of the cover image.

As explained in the first proposed method, this scheme also pre-processes the $M \times N$ cover image to create an error image. The required error image is computed by applying a suitable image quality factor (IQF), similar to the one suggested for JPEG compression, as follows:

- 1) The cover image is partitioned into non-overlapping 8×8 pixel blocks.
- 2) The discrete cosine transform is applied to every block using the JPEG standard quantization table. The cover image can be transformed using a JPEG IQF, which can be any floating point number between zero (the least expected quality or full substitution of pixel values) and 100 (an identical image without any change).
- 3) The inverse DCT is applied to the matrix of coefficients resulting from the DCT of each block.
- 4) The resulted inverse DCT is the compressed cover image that resembles the original image, but now has some imperceptible added noise. The amount of noise has a direct relation to the compression level and embedding rate; the larger the embedding rate, the

Table 3-2. Partition sizes and PSNR values; payload: 0.2 bpp; IQF 70; first image of BOSS 1.01

Partition length (bit)	Achieved PSNR (dB)
52	59.67
32	59.62
16	59.59
8	59.41

more noise has been added. The noise becomes greater if a smaller IQF is employed. In this regard, for every pixel (i,j) from $CoverImg_{(M \times N)}$ where $i \leq M$ and $j \leq N$:

$$ErrorImg_{i,j} = CoverImg_{i,j} - CompressedImg_{i,j} \quad (3)$$

- 5) According to the pixel values of the error image, multiple bases are calculated for every corresponding pixel. $Base_{M \times N}$ represents the matrix of multiple bases:

$$Base_{i,j} = \log_2(|Errorimg_{i,j}| + 1) \quad (4)$$

The overall procedure is already shown in Fig. 3.1 (Similar to the first scheme). According to the embedding order, which is line-by-line from the top to bottom of the cover image, the algorithm reads the corresponding $Base$ from $Base_{M \times N}$. Pixels with a $Base$ value less than two are skipped and are retained unchanged. Thus, only pixels with a base greater than or equal to two are allowed to be embedded. The secret bits are partitioned into non-overlapping 32-bit blocks. It has been shown that the larger the partition, the greater the PSNR (Table 3-2). Hence, we applied 32-bit blocks because MBNS method can implicitly compress the secret bits while embedding them and compresses more if partition size is bigger. For the same reason, PSNR value shown in Table 3-2 increases as partition length becomes more.

Using the decimal value D of every block, one can convert D into a multiple-base notational system [39]. Thus, Vector D^{\wedge} is formed from the conversion of every decimal value D into n multiple bases.

The embedding is applied in two different methods, as follows:

Case 1: For every element of vector D' , say $D'_{l \leq n}$, greater than or equal to two, the pixel values are directly changed using the following pseudo code, proposed by Thien and Lin [46]:

```

Set d to  $D'_l \bmod (CoverImg_{i,j}, Base_{i,j})$ 
If  $-\lfloor \frac{Base_{i,j}-1}{2} \rfloor \leq d \leq \lfloor \frac{Base_{i,j}-1}{2} \rfloor$  then
    Set Change to d
Else If  $1-Base_{i,j} \leq d < -\lfloor \frac{Base_{i,j}-1}{2} \rfloor$  then
    Set Change to  $d+Base_{i,j}$ 
    Else If  $\lfloor \frac{Base_{i,j}-1}{2} \rfloor < d < Base_{i,j}$  then
        Set Change to  $d - Base_{i,j}$ 
 $StegoImg_{i,j} = CoverImg_{i,j} + Change$ 
If  $StegoImg_{i,j} < 0$  then
     $StegoImg_{i,j} = StegoImg_{i,j} + Base_{i,j}$ 
Else if  $StegoImg_{i,j} > 255$  then
     $StegoImg_{i,j} = StegoImg_{i,j} - Base_{i,j}$ 
    
```

where D'_l can easily be extracted by dividing the stego image pixel value into the corresponding Base value of the same pixel. The above algorithm merely searches for the best candidate that is closest to the original pixel value. Thien and Lin [46] applied a fixed value of Base for every pixel, while we have assigned multiple bases denoted by $Base_{i,j}$ for every pixel of the cover image. By using multiple bases and not embedding in every pixel (in contrast to [41]), our proposed method achieves more invisibility and less detectability, as shown in experimental results.

Case 2: For every D'_l where the value is either zero or one, LSBMR [53] is applied to minimize the embedding effect for two pixel units. Note that, unlike the LSB matching concept, there could be some pixels between the two pixels to which LSB matching is not applied. LSBMR [53] embeds two bits in a pair of pixels and applies a function that is dependent on the LSBs of the pixel values. In the worst case, only one of the pixel values must be either incremented or decremented. The best case involves retaining the pixel values intact and unaltered

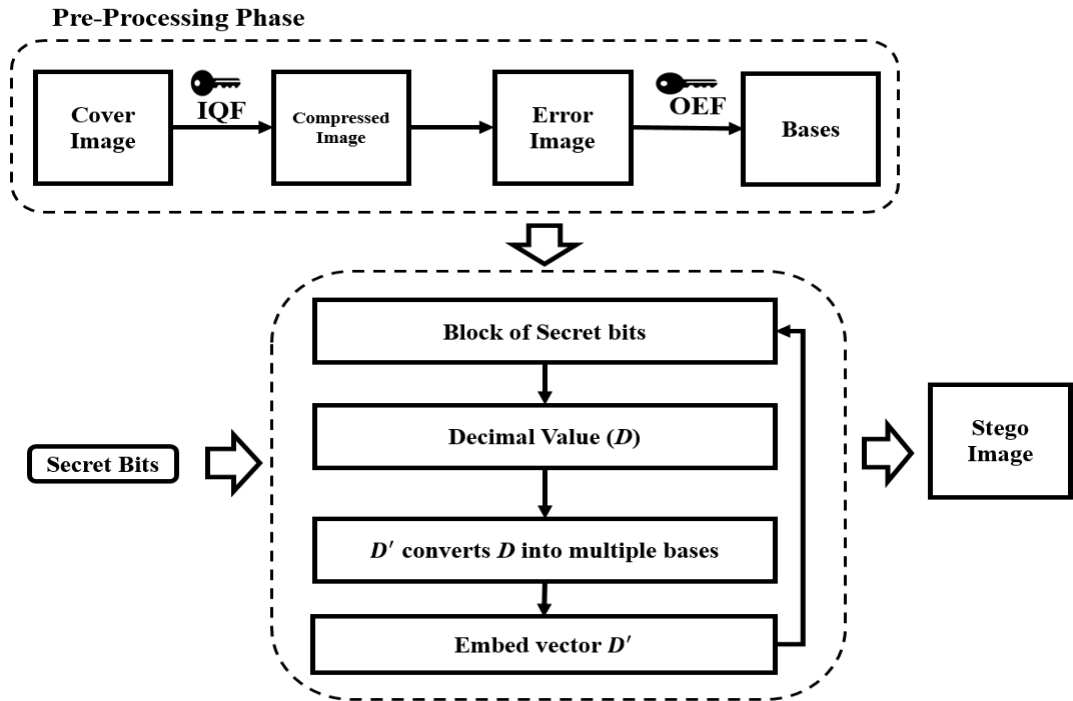


Figure 3.3 Structure of the proposed embedding method 2

The structure of the proposed scheme is shown in Fig. 3.3. The embedding is repeated until every block of secret bits is processed. If all pixels from the cover image have been processed and some of the secret bits have not been embedded yet, IQF must be decreased closer to 0.01. If the secret bits are still partially embedded, the cover image cannot take more secret bits, and that is the maximum payload of this method for that particular cover image.

The algorithm extracts the secret bits from the stego image as follows:

```

Get Base matrix, Payload, and IQF
Set NextBlock to True
Initialize SumBases to 0
Initialize MulBases to 1
Repeat for every pixel from the stego image row-by-row from top to bottom
  If Corresponding Base value for the current pixel is greater than or equal to 2 then
    If NextBlock is true then
      Set NextBlock to False
      Initialize Bases to 1
      Initialize SumBases to 0
    Let M save the remainder for division of the stego pixel value into its corresponding Base
    If M is either 0 or 1 then
      Make two pixel units
      Set M1 to the LSB of the 1st pixel
      Set M2 to the function of 1st and 2nd pixels as proposed in LSBMR
      Let M be M1 and M2 in order
    Increment SumBases with  $M \times \text{MulBases}$  value
    Add  $\text{MulBases} \times \text{Base}$  to MulBases
    If  $\text{SumBases} + \text{MulBases}$  is greater than  $2^{32}$  then
      Set NextBlock to True
      Monitor binary form of SumBases as part of the secret bits extracted
      Initialize SumBases to 0
      Initialize MulBases to 1
  Until payload is extracted completely
  
```

The embedding procedure is illustrated in Table 3-3. The decimal value of the first 32-bit block is calculated as 105 and converted into multiple bases of 3, 4, 2, and 5, which is $(4030)_{5,2,4,3}$ equal to 105. Values 4, 0, 3, and 0 must be embedded in two different methods, as stated in the algorithm. The extraction is also performed with two different methods. The final extracted values are shown in boldface. Two cover image pixels with grayscale values 80 and 101 are excluded because their corresponding Base values are zero and one, respectively. The pixels with values of 87 and 165 form a pair of pixel units to be passed to LSBMR [53] for embedding (both get the value zero to embed). The stego image pixel with value 165 is identical to the original pixel value. In this case, one pixel is spared from change, yet the message bits are still extractable. According to the LSBMR algorithm, in the worst case, only one of the two pixels are incremented or decremented.

Table 3-3. Numeric example of embedding and extraction

Initialization	1 st 32-bit block					0...00000001101001					
Values to Embed	Decimal Value					$105 = (4030)_{5,2,4,3} = 0 + 3 \times 3 + 0 \times 4 \times 3 + 4 \times 2 \times 4 \times 3$					
	$D_i = \{0, 3, 0, 4\}$					-	0	-	3	0	4
Pre-Processing Phase	Cover Image	80	87	101	157	165	213				
	Corresponding Base numbers	0	3	1	4	2	5				
Embed If Base ≥ 2	Stego Image	Case 1: if $D_i \geq 2, Thien[19]$		Case 2: if $D_i < 2, LSBMR[3]$							
		No Embedding	-	No Embedding	159	-	214				
		No Embedding	86	No Embedding	-	165	-				
Extraction	If Case 1: Modular Function		If Case 2: LSBMR								
	Extraction	No	-	No	$\text{Mod}(159,4) = 3$	-	$\text{Mod}(214,5) = 4$				
	Extraction	No	$\text{LSB}(86) = 0$	No	-	$\text{LSB}(\lfloor \frac{86}{2} \rfloor + 165) = 0$	-				

C. Double Phase Modular Steganography with the help of error images

As we know, many steganographic schemes based on popular least significant bit (LSB) embedding have been proposed. The embedding procedure is totally independent of pixel values due to the fact that only the LSB has to be altered. Other methods either randomly increment/decrement the pixel value or use a function to do so. Because of this, the maximum embedding capacity cannot exceed 1 bit per pixel (bpp). State-of-the-art steganographic schemes, such as edge adaptive (EA) and highly undetectable stego (HUGO), aim at such LSB-based approaches with the help of LSB-matching algorithms and, unlike classical steganography, are more concerned about the undetectability level of the stego image rather than the peak signal-to-noise ratio (PSNR) of the stego images. There is a trade-off between detectability and the embedding rate of the stego images. The larger the embedding rate, the higher the chance of detectability, and the opposite is also true. On the other hand, classical steganographic approaches aim for maximum embedding capacity (say 4 bpp) while their main ability is to provide greater PSNR values rather than undetectability. In this work, the second scheme is extended using double phases in such a way that can embed more secret bits while it also takes advantage of error images resulting from applying an image quality factor (the same as the ones used in JPEG compression) in order to find the pixels where the secret data could be embedded. We show that our proposed method is still less detectable compared to the EA method. The proposed method is almost as undetectable as HUGO. As for the previous two proposed methods, the detectability level is again evaluated by ensemble classifiers with second-order subtractive pixel adjacency model features given as their input. In addition and as a result of using an extra phase, the proposed method can easily embed up to 4 bpp, yet maintain a high PSNR value.

As stated earlier, classical steganography provides larger embedding rates, which are usually more than 1 bpp. Such steganographic methods try to embed a large payload into a cover image with high PSNR values (imperceptibility), in which more data changes the values of pixels from busy areas. On the other hand, modern steganographic schemes are supposed to be undetectable, rather than stressing the PSNR value, so the current scheme is also shown to be more undetectable while it can even embed more than 1 bpp. With classical steganography, the proposed method embeds up to the maximum reported embedding capacity, which is equal to 4 bpp [46].

The proposed method also pre-processes the cover image sized $M \times N$ pixels in order to create an error image. The required error image is computed by applying a suitable image quality factor (IQF) like the one suggested in Joint Photographic Experts Group (JPEG) compression:

1. Partition cover image into non-overlapping blocks sized 8×8 pixels.
2. Apply discrete cosine transform to every block using the JPEG standard quantization table. The cover image can be transformed using a JPEG IQF, which can be any float number between 0 (the least expected quality or full substitution of pixel values) and 100 (identical image without any change).
3. Apply inverse DCT to the matrix of coefficients resulted from DCT of each block.
4. The compressed image resembles the original image. It consists of some imperceptible added noise. The amount of noise has a direct relation to compression level and embedding rate. The larger the embedding rate, the more noise.
5. The noise becomes greater if a smaller IQF is employed. In this regard, for every pixel $_{i,j}$ from $CoverImg_{M \times N}$ where $i \leq M$ and $j \leq N$:

$$ErrorImg_{i,j} = CoverImg_{i,j} - CompressedImg_{i,j} \quad (5)$$

According to the pixel values from the error image, multiple bases are calculated for every corresponding pixel. $Base_{M \times N}$ represents the matrix of multiple bases:

$$Base_{i,j} = \log_2(|Errorimg_{i,j}| + 1) + OEF \quad (6)$$

For embedding rates up to 1 bpp, the optimal extension fields (OEF) factor is equal to zero. It does not increment unless the embedding rate (EmbRate) is between 2 bpp and 4 bpp. Similarly, the whole procedure is illustrated in Figure 1, while OEF is zero. The structure of the proposed scheme is modeled in Figure 3.4.

- Phase 1:** the secret bits are partitioned into non-overlapping 32-bit blocks. Decimal value D of every block is represented using a corresponding pseudo random number RND . Vector D' converts D into the base of RND and includes a list of integer values. RND is experimentally

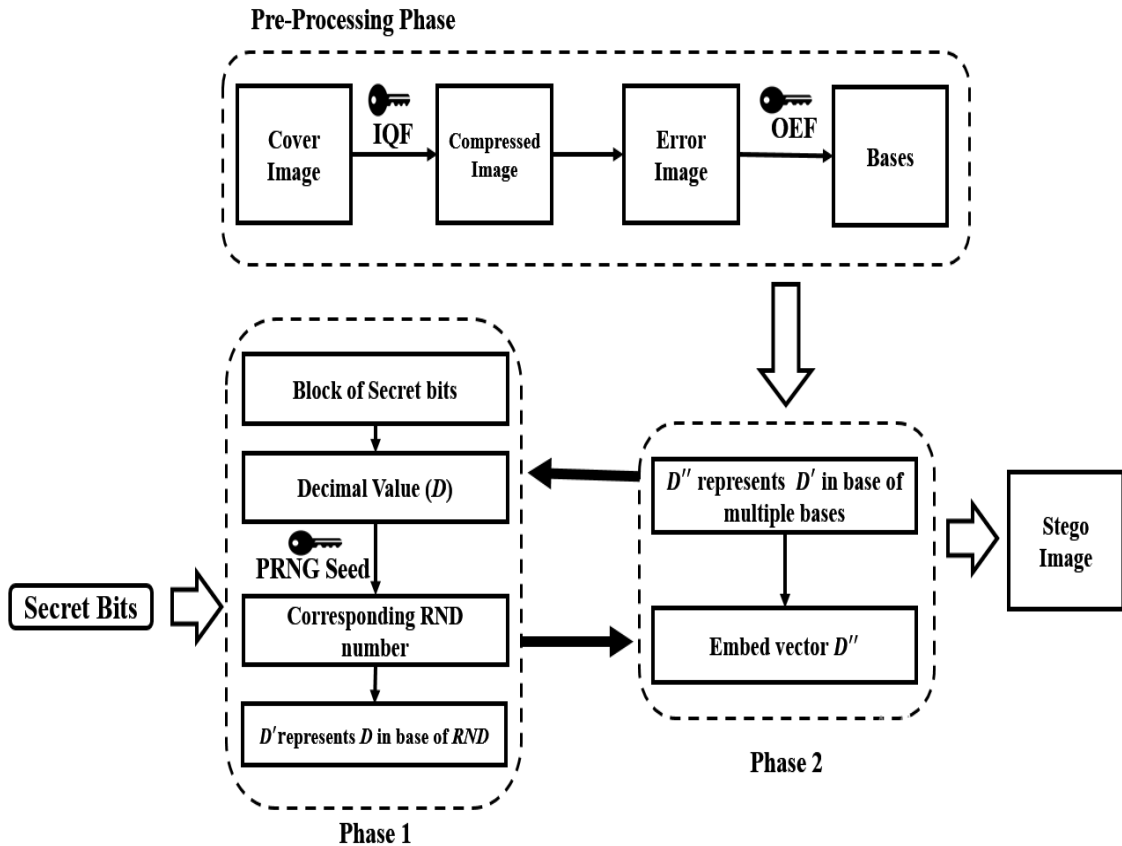


Figure 3.4 Structure of the proposed embedding method 3

computed utilizing a PRNG and the desired embedding rate, which is depicted in bpp. It becomes larger for higher embedding rates because larger secret bits require larger random numbers to divide them in smaller partitions:

$$RND = \left\lfloor \left(\frac{EmbRate}{8} + PRNG + 1 \right) \times 10 \right\rfloor + \left\lfloor \frac{EmbRate}{4} \right\rfloor \times 2^{32} \quad (7)$$

- **Phase 2:** According to the embedding order, which is line by line from top to bottom of the cover image, the algorithm reads the corresponding Base from Base_{M×N}. Pixels with a Base value less than two are skipped and left the way they are. Thus, only pixels with a base greater than or equal to 2 are allowed to be embedded. Using the decimal values of every element of vector D' , one can convert every decimal value into the multiple-base notational system [39]. Thus, Vector D'' keeps conversion(s) of every decimal value from vector D' into n number of multiple bases. The embedding is applied in two different ways:
 - **Case 1:** For every element of vector D'' , say $D_i'' \leq n$, which is greater than or equal to 2, directly change pixel value using the following reversible formulation:

$$\text{StegoImg}_{i,j} = \left\lfloor \frac{\text{CoverImg}_{i,j} - D_i''}{\text{Base}_{i,j}} \right\rfloor \times \text{Base}_{i,j} + D_i'' \quad (8)$$

Where D_i'' can easily be extracted by dividing the stego image pixel value into the corresponding Base value of the same pixel.

- **Case 2:** For every D_i'' where the value is either 0 or 1, LSBMR [53] is applied to minimize the embedding effect for two pixel units. Note that, unlike the LSB matching concept, there might be some pixels between two such pixels to which LSB matching is not applied.

Phases 1 and 2 are repeated for every pixel from the cover image. If some of the secret bits have not been embedded yet, IQF has to be decreased towards 0.01. If the secret bits are still embedded partially, the OEF has to be incremented until the secret bits are embedded completely.

The algorithm extracts the secret bits from the stego-image as follows:

```

Get Base matrix, Payload, PRNG seed number, IQF and OEF
Set NextBlock to True
Initialize SumRndBases and SumBases to 0
Initialize RndBases and MulBases to 1
Repeat for every pixel from the stego image row by row from top to bottom
  If Corresponding Base value for the current pixel is greater than or equal to 2 then
    If NextBlock is true then
      Calculate RND according to the given payload size and PRNG
      Set NextBlock to False
      Set Bases and RndBases to 1
      Set SumBases and SumRndBases to 0
    Let M save the remainder for division of the stego pixel value into its corresponding Base
    If M is either 0 or 1 then
      Make two pixel units
      Set M1 to the LSB of the 1st pixel
      Set M2 to the function of 1st and 2nd pixels as proposed in LSBMR approach
      Let M be M1 and M2 in order
    Increment SumBases with  $M \times \text{MulBases}$ 
    Add  $\text{MulBases} \times \text{Base}$  to MulBases
    If SumBases plus MulBases is greater than RND then
      Increment SumRndBases with SumBases times RndBases
      Add  $\text{RndBases} \times \text{RND}$  to RndBases
      If SumRndBases plus RndBases > the maximum decimal value of block of 32 bits then
        Set NextBlock to True
        Monitor binary form of SumRndBases as part of the secret bits extracted
        Initialize SumRndBases to 0
        Initialize RndBases to 1
      else
        Initialize SumBases to 0
        Initialize MulBases to 1
  Until payload is extracted completely

```

The embedding procedure is illustrated in Table 3-4. The decimal value of the first 32-bit block is calculated as 105 and converted into the base of RND 100, which is $(15)_{100}$ equal to $5+1 \times 100$. Number 5 and 1 are again converted using those pixels with their corresponding Base greater than or equal to 2. Thus, 5 is represented as $(12)_{4,3}$. Numbers 1 and 2 have to be embedded in two different ways, as stated in Phase 2. The extraction is also performed in two different ways. Finally, the extracted values are shown in boldface. Two pixels with gray-scale values 80 and 101 are excluded because their corresponding Base values are 0 and 1, respectively.

Table 3-4. A numeric illustration of embedding and extracting

<i>Initialization</i>	1 st 32-bit block <u>0...00000001101001</u>	1 st block Corresponding RVD Number: 100
	Decimal Value $105 = (15)_{100} = 5 + 1 \times 100$	
<i>Phase 1</i>	$D' = \{5, 1\}$	$5 = (12)_{43} = 2 + 1 \times 3$ $1 = (1)_2 = 1$
<i>Phase 2</i>	$D'' = \{2, 1, 1\}$	2 1
<i>Pre-Processing Phase</i>	Cover Image Corresponding Base numbers	80 87 101 157 165 0 3 1 4 2
<i>Embed</i> If Base ≥ 2	Stego Image if Case 1: $D'_i \geq 2$ if Case 2: $D'_i < 2$	No Embedding No Embedding No Embedding No Embedding No Embedding
		86 - 157 - 165
<i>Extraction</i>	If Case 1: Modular function If Case 2: LSBMR[4]	No Extracting No Extracting Mod(86,3)=2 No Extracting LSB(157)=1 LSB($\lfloor \frac{157}{2} \rfloor + 165$)=1

Pixels with values of 157 and 165 have formed a pair of pixel units to be given to LSBMR [53] for embedding (one in both of them). Their values are identical to the original pixel values. In this case, two pixels are spared from being changed, yet the message bits are still extractable. According to the LSBMR algorithm, in the worst case, only one of the two pixels are incremented or decremented.

D. Parallel modular steganography using error images

As described earlier, the embedding process for most of the state-of-the-art adaptive steganographic methods (not easily detectable) such as highly undetectable steganography (HUGO), edge adaptive image steganography (EA), and Adaptive ± 1 Steganography in Extended Noisy Regions, is contingent upon the neighboring pixel values to prevent discovery by steganalysis methods. Hence, the calculation of the possible embedding rate must be suspended until the embedding process is performed pixel by pixel, in sequence. If, however, the cover image is JPEG-compressed, an error image can be considered as an embedding map implying where and to what extent secret bits can be embedded.

Although the quality and undetectability level of the embedded image must indeed be maintained after embedding, it is equally important that the embedding process be sufficiently fast and time efficient in practical use. Thus, some researchers employ customized IC-chips or coprocessors to improve the time performance of their complex methods. Regardless of the approach implemented, the time efficiency of a method will be improved if the method can be executed in parallel. In practical terms, not all algorithms can be implemented in parallel owing to data dependence during processing. Frequently, the calculations for the next step must be suspended until the results are obtained from the previous running step. Therefore, an algorithm that has the potential to be executed in parallel is of substantial value.

Regarding parallel execution, simple LSB insertion and LSBMR methods can easily embed secret bits in either a pixel or a group of pixels simultaneously. These methods do not consider any distortion based on the other pixel values surrounding a pixel where secret bits are embedded. However, the stego images of such methods can be broken easily using either a Chi-square attack or ensemble classifiers. Conversely, the embedding process for the state-of-the-art adaptive

steganographic methods (not easily detectable), such as HUGO [55] and EA [54], depends on neighboring pixel values, allowing them to embed secret bits adaptively. They adjust every target pixel to hold the secret bits in a manner that the embedding creates less distortion based on their neighboring pixels. Thus, it is important to have the same original values before and after embedding pixel values; the distortion was previously calculated for every pixel. If manipulated, pixel values are used instead of the original values to calculate the distortion; the same secret bits cannot be extracted. Hence, the calculation of the possible embedding rate must be suspended until the embedding process is performed pixel by pixel, in sequence. Consequently, not all undetectable embedding methods can be executed in parallel.

Afrakhteh, Moon, and Lee [61] propose double-phase modular steganography using error images (DPMSE). This method pre-processes JPEG compression error images created by applying a desired Image Quality Factor ($0 < IQF < 100$). It then computes a base value for every pixel according to the error image before it begins the embedding. DPMSE segments the secret bits into non-overlapping partitions of the same size (32 bits) with corresponding pseudorandom numbers. In the first phase, it represents every secret partition in the corresponding pseudorandom number. In the second phase, every element of this representation is again represented in multiple bases [39]. This method embeds only in those pixels from which multiple bases greater than or equal to two are read; it leaves the remaining pixels unchanged. This method can embed up to 4 bpp using an over-embedding factor (OEF) and has been verified to be undetectable by Chi-square attack [51]. It has also been confirmed that ensemble classifiers using second-order SPAM features can detect an embedding rate of 1 bpp with an average testing error less than that of EA.

In this section, DPMSE [61] is revised such that the maximum embedding rate is up to 1 bpp (OEF is removed) and every 32-bit secret partition is directly represented in multiple bases (pseudorandom numbers are removed). The revised method simplifies DPMSE to facilitate understanding of the proposed parallel method. In this work we have the following:

1. It is confirmed that the revised version is still less detectable compared to EA [54], HUGO [55], extended HUGO [59], and Adaptive ± 1 Steganography [62].
2. It is demonstrated that the proposed method runs faster than extended HUGO in serial.
3. The revised version is re-written, implementing parallel processing mainly to improve the time performance. A block-wise parallel algorithm is presented where a single thread is

assigned to each block of pixels. Because of the independent behavior of the embedding method and unlike the other methods, it is proved that each block of pixels can be embedded independently. The effectiveness of this parallel approach is analyzed and verified to execute approximately 55 times faster than serial execution.

MS-E is a simplified version of DPMSE that operates without OEF or pseudorandom numbers. The error image is calculated by applying JPEG compression and the desired IQF to each 8×8 block of pixels. Multiple bases are computed based on the corresponding error image blocks and saved in the bases matrix. The secret bits are segmented into non-overlapping 32-bit secret partitions. Finally, every secret partition is represented in multiple bases. The pseudo-code for MS-E can be written as follows:

```

1. Read Cover_image (m, n)
2. Read Secret_bits (k)
3. Read Image Quality Factor (IQF)
4. mb = nb =8                                //block size
5. Secret_Partition_size=32                  //in bits
6. Dec_partition=0
7. For i=1 to n, step i+nb // Block-by-block calculation of error image & Bases
    7.1. For j=1 to m, step j+mb
        7.1.1. Image_block=Cover_image (i: i+mb-1, j: j+nb-1)
        7.1.2. Compressed_block =JPEG (Image_block, IQF)
        7.1.3. Error_image_block =|Compressed_block – Image_block|
        7.1.4. Bases_block =  $\lfloor \log_2(1 + \text{Error\_image\_block}) \rfloor$ 
        7.1.5. Bases (i: i+mb-1, j: j+nb-1)=Bases_block
    7.2. EndFor
8. EndFor
9. For i=1 to n // line-by-line browsing of the cover image pixels
    9.1. For j=1 to m
        9.1.1. If Dec_partition = 0 then // reading the next secret bits
            9.1.1.1. Read Secret_partition (Secret_bits, Secret_partition_size)
            9.1.1.2. Dec_partition= Decimal_Conversion(Secret_partition)
        9.1.2. If Bases(i, j)  $\geq 2$ 
            9.1.2.1. Remainder =mod (Dec_partition, Bases (i, j))
            9.1.2.2. If Remainder < 2 Embed Remainder by LSBMR [53]
                    Else Embed Remainder by Thien [46]
            9.1.2.3. Dec_partition=(Dec_partition – Remainder)/Bases(i, j)
        9.2. EndFor
10. EndFor
    
```


Case1: For every Remainder whose value is either zero or one, LSBMR [53] is applied to minimize the embedding effect for two pixel units. Note that in contrast to the LSB matching concept, there could be pixels between two pixels to which LSB matching is not applied. LSBMR [53] embeds two bits in a pair of pixels and applies a function that is dependent on the LSBs of the pixel values. In the worst case, only one of the pixel values must be either incremented or decremented. The best case involves retaining the pixel values intact and unaltered (Fig. 3.5).

Case 2: For every Remainder greater than or equal to two, the pixel values are changed directly using the following pseudo-code proposed by Thien and Lin [46]:

```

Set d to Remainder  $\text{-mod}(\text{CoverImg}_{i,j}, \text{Bases}_{i,j})$ 
If  $-\left\lfloor \frac{\text{Bases}_{i,j}-1}{2} \right\rfloor \leq d \leq \left\lfloor \frac{\text{Bases}_{i,j}-1}{2} \right\rfloor$  then
    Set Change to d
Else If  $1-\text{Bases}_{i,j} \leq d < -\left\lfloor \frac{\text{Bases}_{i,j}-1}{2} \right\rfloor$  then
    Set Change to  $d+\text{Bases}_{i,j}$ 
Else If  $\left\lfloor \frac{\text{Bases}_{i,j}-1}{2} \right\rfloor < d < \text{Bases}_{i,j}$  then
    Set Change to  $d - \text{Bases}_{i,j}$ 
StegoImgi,j = CoverImgi,j + Change
If StegoImgi,j < 0 then
    StegoImgi,j = StegoImgi,j + Basesi,j
Else if StegoImgi,j > 255 then
    StegoImgi,j = StegoImgi,j - Basesi,j
    
```

where Remainder can be extracted easily by dividing the stego image pixel value into the corresponding Bases value of the same pixel. This algorithm searches for the best candidate closest to the original pixel value. Thien and Lin [46] applied a fixed value of Bases for every pixel. We have assigned multiple bases, denoted by Bases_{*i,j*} for every pixel of the cover image. By using multiple bases and not embedding in every pixel (in contrast to [46]), the proposed method achieves more invisibility and less detectability, as will be shown in experimental results.

The pseudo-code can be understood as implying that the decimal conversion of every 32-bit partition is divided into bases greater than or equal to two. Simultaneously, the remainder of every division is directly embedded in two different manners. If the remainders are less than two, the pair must be embedded using LSBMR [53]; otherwise, the remainders are embedded as described by Thien [46]. The secret partition is fully embedded if the decimal value of the Dec_partition becomes zero (line 9.1.2.3). The next secret partition must be read unless either the secret bits are fully embedded or the image pixels are thoroughly browsed.

```

Input: a pair of cover image pixels  $x_i, x_{i+1}$ 
          Two message bits  $m_i, m_{i+1}$ 
Output: a pair of stego image pixels  $y_i, y_{i+1}$ 

If  $m_i = LSB(x_i)$ 
    If  $m_{i+1} \neq f(x_i, x_{i+1})$ 
         $y_{i+1} = x_{i+1} \pm 1$ 
    Else
         $y_{i+1} = x_{i+1}$ 
    End
     $y_i = x_i$ 
Else
    If  $m_{i+1} \neq f(x_i - 1, x_{i+1})$ 
         $y_i = x_i - 1$ 
    Else
         $y_i = x_i + 1$ 
    End
     $y_{i+1} = x_{i+1}$ 
End
    
```

Figure 3.5 Embedding algorithm for a pair of pixels [53]

1. Secret partition size estimation

The number of secret bits to embed depends on the IQF value, which can be calculated easily. The estimation method is similar to binary representation. In binary representation, the maximum number of states that can be represented using n bits is equal to the product of n instances of two (i.e., 2^n). The maximum numerical value in binary using n bits is $2^n - 1$. Every bit can show only zero or one because the remainder of the division of any decimal value by two is always less than two. That is, binary representation uses a base of two exclusively, with the result that the

remainders are always either zero or one. The question arises, “What if we represent the numerical value in multiple bases equal to or greater than two?” The new representation is easily proven to be shorter than the binary version.

Suppose that we have a cover image with only 10 pixels having 10 multiple bases greater than or equal to two (saved in the Bases matrix). In binary representation, the maximum decimal value cannot exceed the product of all of the multiple bases, as follows:

$$\begin{aligned} \text{MaxDecValue} &= 2 \times 3 \times 3 \times 4 \times 3 \times 4 \times 5 \times 2 \times 2 \times 5 \\ &= 86,400 \end{aligned}$$

The decimal value 86,400 requires 17 bits in binary representation; however, it can be represented using only ten digits using multiple bases. Whereas $2^n - 1$ is the maximal value that can be represented using n bits, the numerical value of MaxDecValue can be represented similarly using N bits, as follows:

$$\begin{aligned} N = \text{Secret_partition_size} &= \lceil \log_2 \text{MaxDecValue} \rceil \\ &= \lceil \log_2 86'400 \rceil \\ &= 16 \end{aligned}$$

In this manner, we are able to estimate the number of secret bits embeddable using multiple bases, greater than or equal to two, before beginning the embedding process.

The proposed method reads non-overlapping 32-bit partitions of secret bits and the pixels serially, line by line. However, it has been proven that the number of secret bits can be partially estimated for every group of pixels. Therefore, the proposed method has the potential for parallel implementation, because each processing unit (PU) can embed a portion of secret bits in a group of pixels independently.

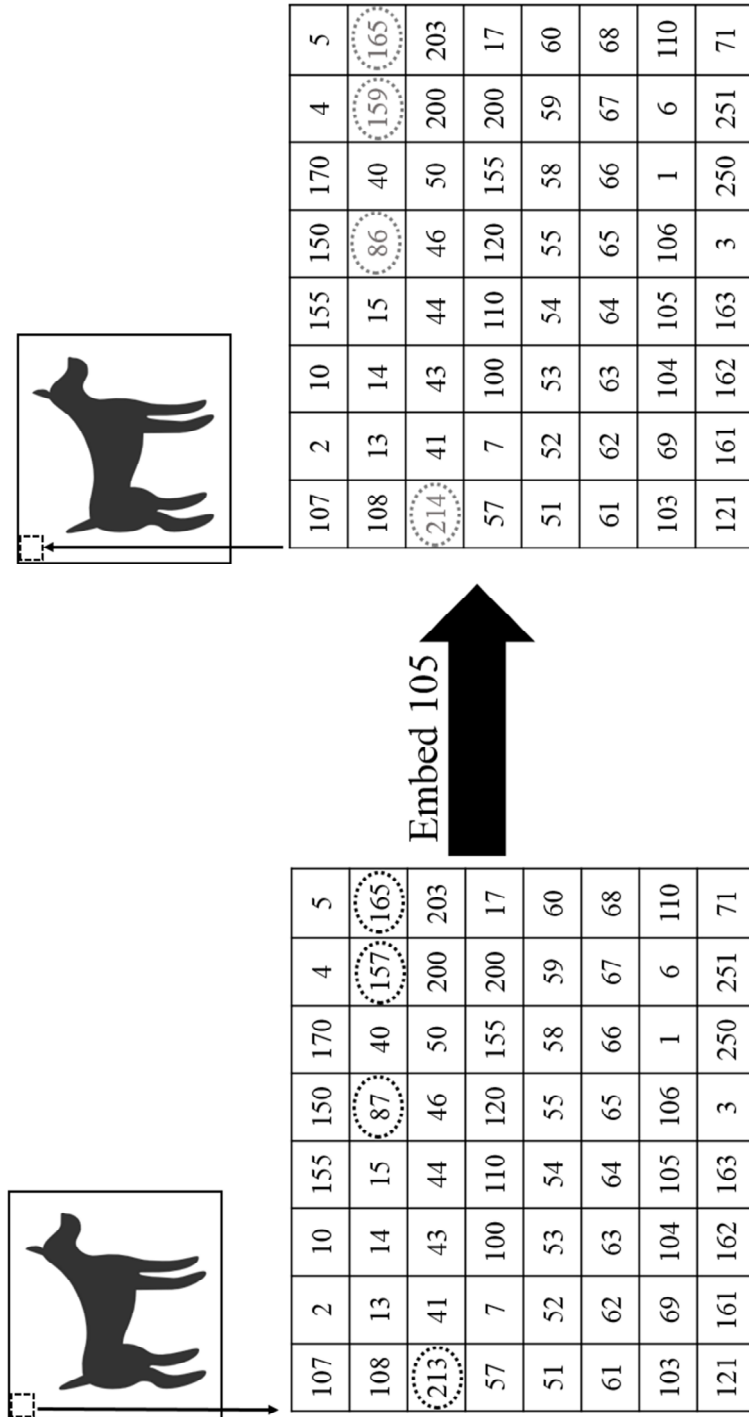
In contrast, HUGO, EA, and Adaptive ± 1 Steganography in Extended Noisy Regions cannot pre-estimate the quantity of the secret bits for each thread, because the embedding of each pixel or block depends on the value of the immediately neighboring pixels, deferring knowledge of the

amount of change, unless the secret bits are fully embedded and counted sequentially. Hence, pre-calculation of the embedding rate becomes practically impossible, and the embedding process cannot be implemented in parallel without knowing the size of the secret data for an individual thread.

The embedding procedure is illustrated in Table 3-3 and Fig. 3.6. The decimal value of the first 32-bit block is calculated as 105 and converted into multiple bases of three, four, two, and five, thereby becoming $(4030)_5, (2,4)_3$ equal to 105. The numbers four, zero, three, and zero must be embedded in two different manners, as stated in the algorithm. The extraction is also performed in two different manners.

Finally, the extracted values are shown in boldface. Two pixels with grayscale values 80 and 101 are excluded, because their corresponding Bases values are zero and one, respectively. Pixels with values of 87 and 165 comprise a pair of pixel units to be provided to LSBMR [53] for embedding (zero in both). The pixel with value 165 is identical to the original pixel value. In this example, only one pixel is unchanged, yet the message bits remain extractable. For LSBMR, only one of the two pixels is incremented or decremented, in the worst case. Finally, every secret partition will be extracted completely if the product of all the used Bases exceeds 232.

Figure 3.6 Embedding example



2. Secret partition addressing

Because it is important to read the secret partitions in the same order as they appear in the secret bits stream, avoiding scrambling, it is necessary to address the secret partitions for each individual thread. This can be achieved easily by saving the product of the multiple bases in another matrix similar in size to the cover image and initialized with zero. Suppose that we have two neighboring blocks of 3×3 pixels. The addressing matrix must be calculated according to the respective Bases matrices, as shown in Fig. 3.7. It is important that the first cell of addressing matrix two be multiplied by the last product of addressing matrix one to avoid secret-bit reading conflicts.

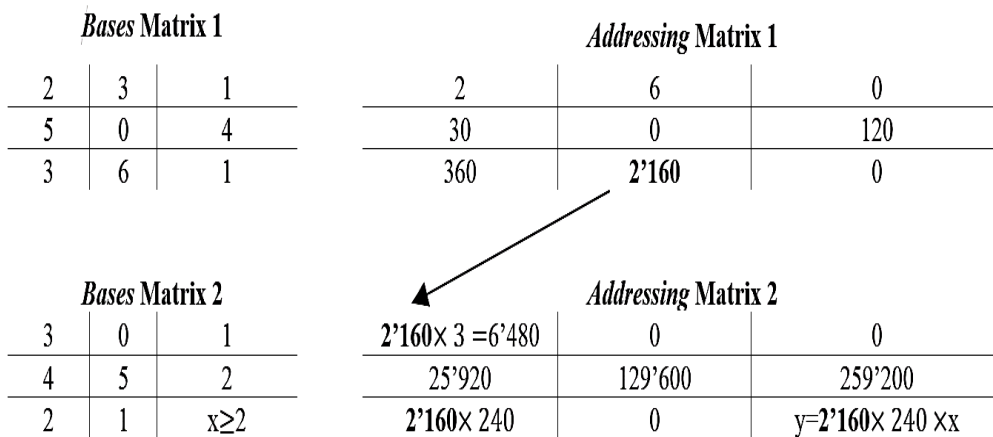


Figure 3.7 Relative Bases and Addressing matrices

A block-wise reading of Bases having been considered; each cell of the addressing matrix saves the frequent product of the previous bases in the current Bases at the corresponding coordinates. Thus, if we assign a thread to every block, the first thread for the upper block must read from the $[\log_2 2]_{th}$ to the $[\log_2 2,160]_{th}$ bit of the secret bits, that is, from the first bit to the 11_{th}. The second block must read from the $[\log_2 2,160 \times 3]_{th}$, i.e., the 12_{th}, of the secret bits to the $[\log_2(240 \times 2,160 \times x)]_{th}$. Hence, the two threads can be executed in parallel on the condition that the second thread knows the last product of the first block, 2160. Through a simple synchronization, they can locate their respective secret partitions according to the addressing matrix. Furthermore, the secret partition sizes of the two blocks are different, as the first thread reads 11 bits, and if $x = 2$, the second thread reads only eight bits from the 12_{th} bit to the $[\log_2(518'400 \times 2)]_{th}$, i.e., the 19_{th}.

3. Parallel pre-processing phase

To accelerate the algorithm, we must also perform the error image calculation in parallel, such that each thread or processing unit can calculate the error image of one block of pixels. However, conflicts occur in reading the secret bits from the secret stream if we do not know the last product of the neighboring block (e.g., 2160). Thus, *Blocks_Product* (size 64×64) saves the greatest product of each block of size 8×8 from a cover image of size 512×512 .

Once all the threads have reached their greatest product values, a serial synchronization of all the products in *Blocks_Product* is necessary (starting at line nine in the following pseudo-code). With this step, each thread reads the correct partition from the secret stream without conflict or overlapping with other threads.

This is addressed in the Pre-processing phase as follows:

1. Read Cover_image (m, n)
2. Read Secret_bits (k)
3. Read Image Quality Factor (IQF)
4. mb = nb = 8 *//block size*
5. Initialize Addressing_Matrix(m,n) cells to zero
6. Initialize Blocks_Product($\frac{n}{nb}, \frac{m}{mb}$) cells to one *//64*64 blocks*
7. For i=1 to n, step i+nb *// Block-by-block calculation of error image & bases*
 - 7.1. For j=1 to m, step j+mb
 - 7.1.1. Image_block=Cover_image (i: i+mb-1, j: j+nb-1)
 - 7.1.2. Compressed_block =JPEG (Image_block, IQF)
 - 7.1.3. Error_image_block =|Compressed_block – Image_block|
 - 7.1.4. Error_image(i: i+mb-1, j: j+nb-1)= Error_image_block
 - 7.1.5. Initialize Product to 1
 - 7.1.6. For ip=i to i+mb-1
 - 7.1.6.1. For ip=j to j+nb-1
 - 7.1.6.1.1. Bases (ip, jp)= $\lfloor \log_2(1 + \text{Error_image}(ip, jp)) \rfloor$
 - 7.1.6.1.2. If Bases (ip, jp) $\geq p$ then
 - 7.1.6.1.2.1. Product=Product \times Bases (ip, jp)
 - 7.1.6.1.2.2. Addressing_Matrix(ip, jp)=Product
 - 7.1.6.2. EndFor
 - 7.1.7. EndFor
 - 7.1.8. Update Blocks_Product($\frac{i}{nb}, \frac{j}{mb}$) with Product
 - 7.2. EndFor

```

8.EndFor
9.Total_Product=1 //Synchronizing the products calculation in Serial
10.For i=1 to  $\frac{n}{nb}$ 
    10.1.For j=1 to  $\frac{m}{mb}$ 
        10.1.1. Temp=Blocks_Product(i,j)
        10.1.2. Blocks_Product=Total_Product
        10.1.3. Total_Product=Temp×Total_Product
    10.2.EndFor
11.EndFor

```

4. Block-wise parallel algorithm

This algorithm considers only one thread for every 8×8 block of pixels. Thus, every thread reads a secret partition from the secret bits stream according to the method described in Sections ‘1’ and ‘2’. The embedding can be performed independently in parallel as follows:

```

1. Get Addressing_Matrix and Blocks_Product using Preprocessing_phase
2. For i=1 to n, step i+nb // Block by block Browsing the cover image
    2.1. For j=1 to m, step j+mb //One thread for a block
        2.1.1. Addressing_block= Addressing_Matrix (i: i+mb-1, j: j+mb-1)
        2.1.2. S=the first non-zero value of Addressing_block
        2.1.3. E=the last non-zero value of Addressing_block
        2.1.4. Partition_start= $\left\lfloor \log_2(S \times \text{Blocks\_Product}(\frac{i}{nb}, \frac{j}{mb})) \right\rfloor$ 
        2.1.5. Partition_end= $\left\lfloor \log_2(E \times \text{Blocks\_Product}(\frac{i}{nb}, \frac{j}{mb})) \right\rfloor$ 
        2.1.6. Read Secret_partition (Secret_bits, Partition_start ,Partition_end)
        2.1.7. Dec_partition= Decimal_Conversion (Secret_partition)
        2.1.8. For ip=i to i+mb-1
            2.1.8.1. For ip=j to j+nb-1 // Block-wise embedding of secret bits
                2.1.8.1.1. If Bases(ip, jp) ≥ then
                    2.1.8.1.1.1. Remainder =mod (Dec_partition, Bases (ip, jp))
                    2.1.8.1.1.2. If Remainder < 2 then Embed remainder by LSBMR [53]
                        Else Embed remainder by: Thien [46]
                    2.1.8.1.1.3. Dec_partition=(Dec_partition – Remainder)/Bases(ip, jp)
                2.1.8.2. EndFor
            2.1.9. EndFor
        2.2. EndFor
    3. EndFor

```

Because the *Base matrix* and its respective products are calculated on the receiver side, we can easily extract the secret bits.

IV. Experimental Results

The results of each of four proposed methods will be given separately. The first method, adaptive LSB matching-revisited, mainly focuses on the importance of LSBM_R method while using the proposed embedding map where we can find the pixels holding secret bits. Unlike LSBM-R that embeds secret bits in random pixels while trying to lower down the unwanted noise by either 50% or 100%, the proposed method knows where to embed and then it applies LSBM-R so that the first contribution becomes more clearly as it is less detectable compared to modern steganographic methods. Both methods can only embed up to 1 bpp.

On the contrary, the second proposed method, modular steganography with the help of error images, shows that it is feasible to embed even up to 4 bpp while being less detectable compared to other methods.

The third proposed method, double phase modular steganography with the help of error images, represents a method to make an implicit partial compression of each partition of secret bits while making stego images even less detectable compared to the second proposed method. This method has double phases and the first phase compresses the secret bits while the second phase embeds.

Finally, the last proposed method, parallel modular steganography using error images, reveals the fact that although all of the proposed methods are originally faster than the state-of-the-art methods, they can even become far faster if it is parallelized.

A. Adaptive LSBM-R using Error Images

One of the most important aspects of any performance evaluation is to use a standard data set with a variety of image textures. The proposed scheme employs the image database of BOSS version 1.01—it consists of 10 000 grayscale images sized 512×512 pixels—which is also used to evaluate modern steganographic schemes with embedding rates less than or equal to 1 bpp. The proposed method was implemented and executed using MATLAB R2012a (MathWorks, California Office 970 West 190th Street Suite 530 Torrance, CA 90502, UNITED STATES) on an Intel Core i5-2500, 3.3–3.6 GHz, with 8 GB RAM.

Modern steganography is more concerned with undetectability levels than imperceptibility, so the PSNR value is always supposed to be high. In this regard, HUGO is a modern steganography method in which the LSBM concept is applied to manipulate the LSB of the pixels. The undetectability level is shown using a probability of error provided by ensemble classifiers using second-order SPAM features and SQUARE+h^x features (dim 342). When the value of the probability error goes down, there is a greater chance of detection. HUGO [55] has been proven to have the greatest probability of error compared with the edge adaptive method using SPAM features [60].

Table 4-1 shows the average testing error over 10 splits of BOSS image database, which is calculated by ensemble classifier using second-order SPAM features (dim 686). It can be seen that HUGO, T = 90 [55], has greater detectability error compared with the proposed method. In Table 4-2, it is shown that HUGO [55] is more vulnerable to SQUARE+ h^x features (dim 342), because it has smaller detection errors compared with both adaptive LSBMR (A-LSBMR) with the help of error images and extended HUGO [59]. Furthermore, our proposed method, A-LSBMR has a contribution beyond HUGO with T = 90 in terms of lesser detectability (higher average testing error). Meanwhile, it is almost as undetectable as extended HUGO with T = 255.

Table 4-1. Detectability comparison between the proposed method and the HUGO (T = 90) [55]

BOSS 1.01 database (10,000 images)		Average testing error over 10 splits (p_e) using second-order SPAM features (dim 686)	
Payload (bpp)	Capacity (bits)	HUGO – T=90 [55]	A-LSBMR (PSNR, IQF)
0.05	13,101	0.5000	0.3782 (65.28 dB, 60.0)
0.1	26,214	0.4844	0.3224 (00.00 dB, 55.0)
0.2	52,428	0.4469	0.2665 (59.25 dB, 50.0)
0.3	78,643	0.4010	0.1963 (57.49 dB, 45.0)
0.4	104,857	0.3600	0.1870 (56.24 dB, 40.0)

Table 4-2. Detectability comparison between the proposed method, HUGO, and Extended HUGO

BOSS 1.01 database (10,000 images)		Average testing error over 10 splits (p_e) using SQUARE+ h^x features (dim 342)		
Payload (bpp)	Capacity (bits)	HUGO [55]	HUGO [59]	A-LSBMR (PSNR, IQF)
0.05	13,101	0.3233	0.4432	0.3399 (65.28 dB, 60.0)
0.1	26,214	0.2911	0.3993	0.2841 (62.27 dB, 55.0)
0.2	52,428	0.2254	0.3262	0.2282 (59.25 dB, 50.0)
0.3	78,643	0.1648	0.2630	0.1937 (57.49 dB, 45.0)
0.4	104,857	0.1284	0.2008	0.1844 (56.24 dB, 40.0)

Table 4-3 shows the global p_e , which is the minimum average classification error for every embedding method. Generally, SQUARE features provide the minimum detectability error for any of HUGO methods with two settings and A-LSBMR. The reason is that SQUARE features are resulted from third-order residuals and have a better detection accuracy than first-order and second-order residuals (SPAM) for every method [60].

Table 4-3. The global p_e of embedding methods resisting 2nd-order SPAM and SQUARE+ h^x

BOSS 1.01 database (10,000 images)		Average testing error over 10 splits (p_e) using second-order SPAM features (dim 686) and SQUARE+ h^x features (dim 342)		
Payload (bpp)	Capacity (bits)	HUGO – T=90 [55]	HUGO – T=255 [59]	A-LSBMR
0.05	13,101	0.3233	0.4432	0.3399
0.1	26,214	0.2911	0.3993	0.2841
0.2	52,428	0.2254	0.3262	0.2282
0.3	78,643	0.1648	0.2630	0.1937
0.4	104,857	0.1284	0.2008	0.1844

In Table 4-4, time performance of the proposed algorithm is about two times greater than that of the HUGO methods [55] [59], whereas A-LSBMR executes a MATLAB source code, which is actually slower than a source code in C language. That is to say that the proposed method could have been executed faster if it was implemented in C language. Furthermore, the complexity of A-LSBMR code is similar to classical schemes and can be implemented more easily. HUGO with threshold value $T = 255$ has to compute more features so that it results in a longer computation time, which is around two times slower than A-LSBMR method.

Table 4-4. Embedding time (in seconds) comparison

Embedding Rates	0.05	0.1	0.2	0.3	0.4	0.5
HUGO [55]	4.32	4.37	5.21	5.27	5.63	5.83
HUGO [59]	4.75	4.80	5.72	5.79	6.19	6.41
A-LSBMR	2.02	2.04	2.43	2.72	2.92	3.02



Figure 4.1 The cover image from image database called 'Lena'

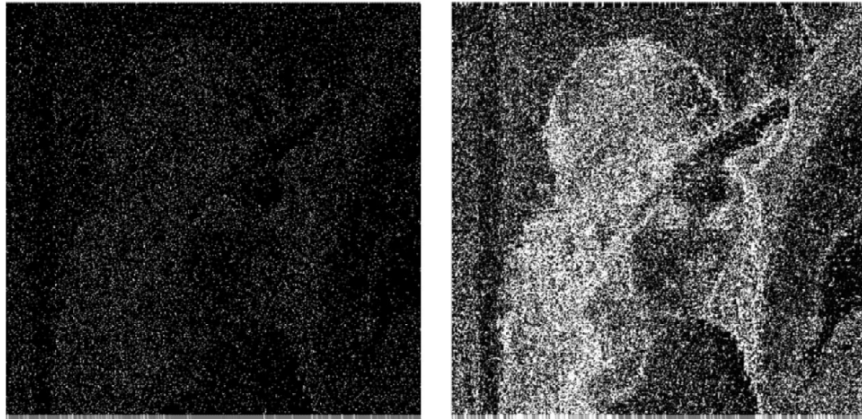


Figure 4.2 The error image (left) and the Base matrix (right) (IQF 40, payload 0.4 bpp, 56.33 dB)

Fig. 4.1 shows the Lena image, which is one of the most commonly known standard test images in steganography. In a simple experiment, we attempted to prove that higher IQF values would result in more undetectability of the stego image. Theoretically, PSNR values will be greater if we employ a higher IQF, and as it is shown in our experiments, the detection error has a direct relation to the value of PSNR. Note that there are two error images introduced in the proposed method. One is computed in pre-processing phase of the algorithm. As with Fig. 4.2, another error image is calculated between the stego images and the original image to show where, and to what extent, the impact of embedding has happened. In Fig. 4.2, the experiment is performed to embed a payload of 0.4 bpp using an IQF of 40. The Base matrix (Fig. 4.2) implies where to embed with a white pixel for a Base value greater than 2. The black pixels are not meant to hold any secret data because they have a Base value less than 2. The white area represents where to embed more. As can be seen, the secret bits are scattered through the stego image (PSNR 56.33 dB), and we can hardly see the detailed texture of the image (Fig. 4.1) in the resulted error image which is calculated from differentiating stego image from the original image. The black area of the Base matrix has become wider in the error image, and that is due to the fact that LSBMR will leave 50% of its pairs intact. The same fact is true for the white areas. Those white pixels in Base matrix are changed with the same probability of embedding change. Thus, the number of white pixels is halved, consequently.

The proposed method showed how ensemble classifiers would be affected by the JPEG IQF and JPEG decompression artifacts. They make it more confusing to steganalysis schemes. It is also proven that the proposed method guarantees less detectability compared with the HUGO (T = 90) method and is close to the detectability level of the HUGO (T = 255) method because, unlike a

conventional LSBMR method, the current algorithm distributes the secret bits through the cover image more adaptively and selects the right pixels to hold secret bits using a calculated Base matrix as a guide. The guide exploits the texture of the cover image so that the secret data bits are distributed through the white target pixels using LSBMR method. The execution time of the proposed method is proven to be around two times faster than that of the HUGO methods with two different settings. The proposed scheme requires a Base matrix to extract the secret information successfully. Hence, there could be a solution to obtain the same Base matrix calculated from the stego image so that the existence of a cover image is not necessary. This can be investigated in future work.

B. Modular Steganography with the help of error images

Modern steganography is more concerned with undetectability levels than imperceptibility; hence, PSNR values are generally high. In this regard, HUGO and EA are modern steganography methods in which the LSB matching concept is applied to manipulate the LSB of the pixels. The undetectability level is shown using second-order SPAM features (dim 686) and SQUARE + h^x features (dim 342). When the value of the probability error decreases, there is a greater chance of detection. HUGO has been proven to have the greatest probability of error compared to EA and LSB matching [60].

Table 4-5 lists the average testing error for the two features over ten splits (P_e) of the BOSS images for the proposed method and other state-of-the-art embedding methods. Table 4-6 shows the global P_e , the minimum average classification error, for every embedding method compared in Table 4-5. Generally, SQUARE features provide the minimum detectability error for both HUGO methods and the proposed method. This is because the SQUARE features result from third order residuals and have a better detection accuracy than the first and second order residuals (SPAM) for every method [60]. As depicted, our proposed method, modular steganography with the help of error images (MS-E), performs better than EA [54], HUGO (T = 90) [55], and HUGO (T = 255) [59] in terms of lower detectability (higher average testing error). It has been proven in [59] that HUGO (T = 90) is vulnerable to other features, particularly the SQUARE + h^x combination. Table 4-6, shows the weakness of HUGO (T = 90) [55] to SQUARE + h^x features in such a way that the detectability error is almost halved, i.e., the stego images created by HUGO (T = 90) are more

detectable by using these features in the ensemble classifiers. Thus, [59] improves HUGO ($T = 90$) [55] by increasing T to 255, called HUGO ($T = 255$). However, HUGO for either setting is still more detectable than MS-E.

Table 4-5. Detectability comparison between MS-E and HUGO ($T = 90$) [55]

BOSS 1.01 database (10,000 images)		Average testing error over ten splits (p_e)								
Payload (bpp)	Capacity (bits)	EA [54]		HUGO $T = 90$ [55]		HUGO $T = 255$ [59]	MS-E			
		SPAM	Squ.+h ^x	SPAM	Squ.+h ^x	Squ.+h ^x	SPAM	Squ.+h ^x	IQF	PSNR (dB)
0.05	13,101	0.47	0.42	0.50	0.32	0.44	0.47	0.47	96	65.19
0.1	26,214	0.43	0.36	0.48	0.29	0.39	0.45	0.49	94	62.29
0.2	52,428	0.33	0.27	0.44	0.22	0.32	0.40	0.37	90	59.46
0.3	78,643	0.25	0.21	0.40	0.16	0.26	0.35	0.34	85	57.83
0.4	104,857	0.19	0.16	0.36	0.12	0.20	0.24	0.28	75	56.63
0.5	131,072	0.15	0.14	0.31	0.09	0.14	0.14	0.23	55	55.57

Table 4-6. The global p_e of four embedding methods resisting 2nd order SPAM and SQUARE+h^x

BOSS 1.01 database (10,000 images)		Minimum average testing error over ten splits (global p_e)			
Payload (bpp)	Capacity (bits)	EA [54]	HUGO $T = 90$ [55]	HUGO $T = 255$ [59]	MS-E
0.05	13,101	0.4292	0.3233	0.4432	0.4779
0.1	26,214	0.3668	0.2911	0.3993	0.4504
0.2	52,428	0.2795	0.2254	0.3262	0.3798
0.3	78,643	0.2112	0.1648	0.2630	0.3401
0.4	104,857	0.1644	0.1284	0.2008	0.2477
0.5	131,072	0.1421	0.0926	0.1468	0.1428

Table 4-7. The global p_e of Adaptive ± 1 Steganography in Extended Noisy Regions [62] and MS-E

BOSS 1.01 database (1000 random images)		Average testing error over ten splits (p_e)					
Payload (bpp)	Capacity (bits)	Adaptive ± 1 Steganography in Extended Noisy Region [62]			MS-E		
		SPAM (dim 686)	SQUARE+h ^x (dim 342)	global P_e	SPAM (dim 686)	SQUARE+h ^x (dim 342)	global P_e
0.05	13,101	0.4917	0.4647	0.4647	0.4920	0.4932	0.4920
0.1	26,214	0.4711	0.4207	0.4207	0.4585	0.4681	0.4585
0.2	52,428	0.4382	0.3441	0.3441	0.4228	0.4207	0.4207
0.3	78,643	0.3898	0.2743	0.2743	0.3937	0.4073	0.3937
0.4	104,857	0.3303	0.2110	0.2110	0.3206	0.3665	0.3206
0.5	131,072	0.2833	0.1672	0.1672	0.1697	0.2983	0.1697

Table 4-8. Embedding time (in seconds) of HUGO, Extended HUGO and MS-E using “Lena”

Embedding Rates	0.05	0.1	0.2	0.3	0.4	0.5
HUGO [55]	4.37	4.43	4.52	4.63	4.69	4.81
HUGO [59]	4.74	4.79	4.97	5.07	5.16	5.33
MS-E	1.78	1.89	2.13	2.27	2.46	2.63

Table 4-7 presents another comparison in terms of the global P_e , the minimum average classification error for MS-E and Adaptive ± 1 Steganography in Extended Noisy Regions [62]. The experiment applies both methods on 1000 randomly chosen images from the BOSS database and the secret data is generated randomly for both methods. It is clear that the minimum testing error has greater values for the proposed method. This implies that MS-E can embed more adaptively in noisy regions without applying a noisy function. The error image by itself simply conveys where and to what extent the secret data can be embedded.

Table 4-8 shows that the execution time of the proposed algorithm is around 2.32 times faster than the extended HUGO method [59]. The code is less complex than HUGO because MS-E only computes the error image for the cover image and there is no need to calculate more than 107 features, as computed by HUGO. HUGO with $T = 255$ has to compute more features, resulting in longer computation times that are about two times slower than MS-E.

It is also worth mentioning that Adaptive ± 1 steganography in extended noisy region [62] could be run about 3.93 times faster than the extended HUGO [59] using source code written in C, while it can be detected more easily than MS-E (Table 4-7). Furthermore, MS-E does not require noisy functions and only relies on an interpretation of the error image. In terms of time performance, MS-E could have been executed faster if it had been implemented in C, even though it executes in MATLAB, which is slower than source code in C, because C/C++ is a lower-level language and generates relatively optimized machine code when compared to high-level languages like MATLAB.

Fig. 4.3 shows the third grayscale image of the BOSS database. In a simple experiment, we attempted to prove that higher IQF values would result in more undetectability of the stego image. Theoretically, PSNR values will be greater if we employ a higher IQF.



Figure 4.3 Third cover image from BOSS 1.01, “3.pgm”

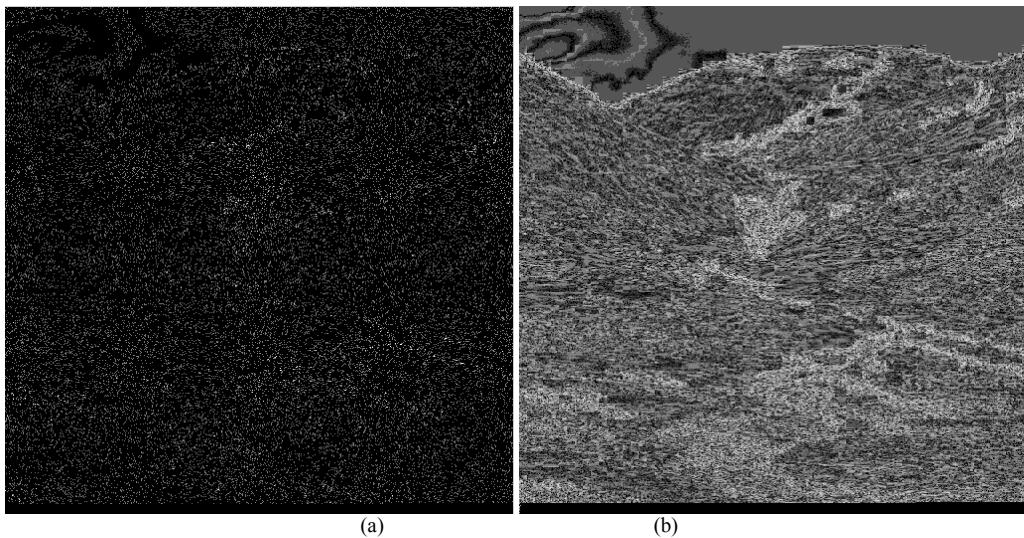


Figure 4.4 Error image (a) and *Base* matrix (IQF: 8, Payload: 1 bpp, 52.69 dB) (b)

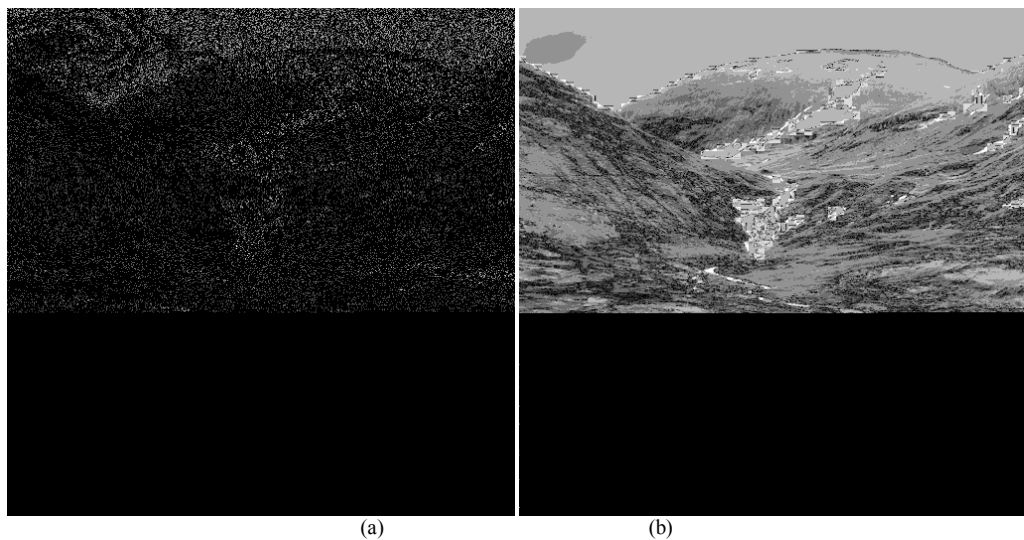


Figure 4.5 Error image (a), and *Base* matrix (IQF: 1, Payload: 1 bpp, 51.54 dB) (b)

Note that there are two error images introduced in the proposed method. One is computed in pre-processing phase of the algorithm. With both Figs. 4.4 and 4.5, another error image is calculated between the stego image and the original image to show where, and to what extent, the embedding has happened. In Fig. 4.4, the experiment is performed to embed a payload of 1 bpp using an IQF of eight. As can be seen, the secret bits are scattered through the cover image (PSNR = 52.69 dB), and it is difficult to see the detailed texture of the image (Fig. 4.3).

On the other hand, Fig. 4.5 shows the effect on the image texture when the secret bits are not well scattered but concentrated in the middle of the image (PSNR = 51.54 dB, IQF = 1). The imperceptibility level is given by the PSNR values. Because the second experiment in Fig. 4.5, has a smaller PSNR, it is considered more visible, and the detected probability of error can also be less than the first experiment in Fig. 4.4 according to the Table 4.9. This is because when IQF is one, the method has greater computed Base values, and hence the secret bits can be embedded using fewer pixels, although the change is significantly larger. If IQF = 8, the Base values are smaller, and the secret bits are depicted using more pixels, however, the change is well distributed among more pixels and is hence considerably smaller.

To explain further, the Base matrix corresponds to every pixel of the cover image and each cell in Base has a value ranging from zero (black pixels) to 255 (white). In the areas located above the hills in Fig. 4.4 and 4.5, there is a clear difference. The Base matrix resulting from an IQF of eight has notably darker areas compared to the one resulting from an IQF of one. An IQF of one creates significantly larger base values so that the pixel values tend to white, while those of IQF = 8 are smaller and the pixels tend to black. For the same reason, the secret bits affect the stego image of IQF = 8 less and the texture of the cover image is less changed compared to the one shown in Fig. 4.5. The lower amount of change in the texture brings about greater PSNR values and the detectability is consequently decreased, as feature extraction methods are more sensitive to the busy areas that form the texture of the stego image. The stego image would be broken if the method embedded more information in busy regions. The method with IQF = 8 tries to embed in smooth areas, whereas if IQF = 1, it concentrates more on busy areas. This is revealed in the error image, where the texture is clearer when using IQF = 1.

Table 4-9. MS-E experiment results with different IQFs for the same payload

BOSS 1.01 database (10,000 images)		Experiment 1		Input Parameters	Experiment 2		Input Parameters
Payload (bpp)	Capacity (bits)	P_e	PSNR (dB)	IQF	P_e	PSNR (dB)	IQF
0.05	13,101	0.4401	65.41	90.0	0.4782	65.19	96.0
0.1	26,214	0.3822	62.55	80.0	0.4504	62.29	94.0
0.2	52,428	0.3072	59.57	70.0	0.4066	59.46	90.0
0.3	78,643	0.2303	57.81	60.0	0.3506	57.83	85.0
0.4	104,857	0.1690	56.55	50.0	0.2477	56.63	75.0

Table 4-9 summarizes the experiments using all 10,000 images from the BOSS database. It is clear that Experiment 2 is less detectable because of the greater values of error and a more complete scattering of the secret bits. Smaller values of IQF increase the probability of detection with less average testing error (P_e) by ensemble classifiers. We can see that PSNR values are slightly greater than those of Experiment 2, whereas Experiment 1 is more detectable according to the smaller P_e . Hence, imperceptibility (greater PSNR values) does not necessarily guarantee less detectability by ensemble classifiers.

The proposed method imposes lesser changes on the cover image pixel values because the decimal value of every 32-bit secret block becomes smaller if it is divided into multiple base numbers assigned to cover image pixels. Hence, the decimal value of a 32-bit secret block is broken down into much smaller remainders. For embedding smaller remainders, a smaller change in pixel value is required; hence, smaller changes are applied to the cover image. Less change leads to a greater PSNR value and lower detectability. In this regard, more secret bits are embedded, despite smaller changes being made to the cover image. Unlike the MBNS method, the greatest change to be enforced on each pixel is computed from an error image. Furthermore, the impact of the change is minimized using LSB matching. We have also shown how varying the JPEG image quality factor and JPEG recompression artifacts affect steganalysis. It decreased the chances of detection by steganalysis schemes. We also showed that the larger the embedding rate, the higher the chance of detectability, and vice versa. Furthermore, the results prove that the proposed method guarantees less detectability compared to EA [54], HUGO ($T = 90$) [55], HUGO ($T = 255$) [59], and Adaptive ± 1 Steganography in Extended Noisy Regions [62] methods, because the algorithm

distributes the secret bits throughout the cover image more uniformly and selects the appropriate pixels to hold secret bits using the calculated Base matrix as a guide. The source code is written in MATLAB and executes about twice as fast as the HUGO methods that were coded in Visual C. Further, the proposed scheme could be executed significantly faster if it was coded in Visual C because this language is appreciably quicker than MATLAB.

C. Double Phase Modular Steganography with the help of error images

Besides using BOSS image database which is also used in modern steganographic schemes, for classical steganographic schemes with embedding rates over 1 bpp, the results from the experiments are given using the same message size and standard cover images, such as Lena, Baboon, Peppers, and so on (Fig. 4.6).

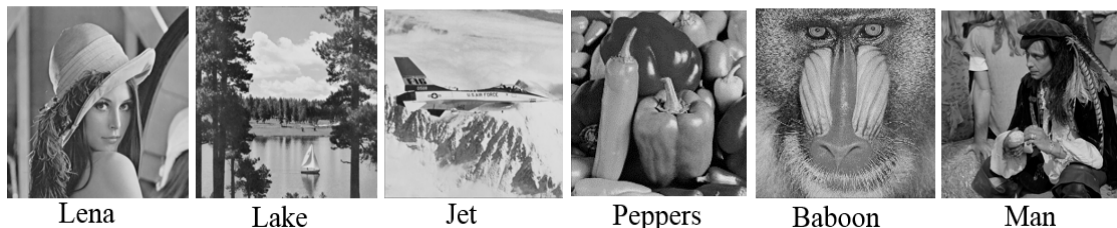


Figure 4.6 Standard grey-scale images frequently used in classical steganography

On the other hand, modern steganography is more concerned with undetectability levels than imperceptibility, so PSNR value is always supposed to be high on a huge image database BOSS. In this regard, HUGO and EA are modern steganography methods in which the LSB matching concept is applied to manipulate the LSB of the pixels. The undetectability level is shown using a probability of error provided by ensemble classifiers using second-order SPAM features. When the value of the probability error goes down, there is a greater chance of detection. HUGO has been proven to have the greatest probability of error compared to the EA method and LSB matching [60]. As depicted in Table 4-10, our proposed method, double phase modular steganography with the help of error images (DPMS-E), has a contribution beyond EA in terms of lesser detectability (higher average testing error). It is almost as undetectable as HUGO. On the other hand, DPMS-E is able to embed up to 4 bpp, whereas HUGO and EA can only embed up to 1 bpp. The execution time of the HUGO algorithm is about 5.47 s for 0.4 bpp, whereas DPMS-E executes in only 3.5 s using the same processor. Furthermore, the complexity of DPMS-E code is similar to classical schemes and can be implemented easier.

Table 4-10. Detectability comparison between DPMS-E and HUGO and EA approaches

BOSS 1.01 database (10,000 images)		Average testing error over 10 splits (p_e) using 2nd SPAM features		
Payload (bpp)	Capacity (bits)	EA [54]	HUGO [55]	DPMS-E (PSNR, IQF, OEF)
0.05	13,101	0.4717	0.5000	0.4814 (68.79 dB, 96.0 , 0)
0.1	26,214	0.4309	0.4844	0.4571 (65.43 dB, 94.0, 0)
0.2	52,428	0.3381	0.4469	0.4089 (62.57 dB, 90.0, 0)
0.3	78,643	0.2549	0.4010	0.3605 (60.89 dB, 85.0, 0)
0.4	104,857	0.1920	0.3600	0.2697 (59.49 dB, 75.0, 0)

Classical steganography usually considers the PSNR value when the payload exceeds 1 bpp. The highest PSNR value was reported by Thien and Lin [46], which was smaller than the achieved PSNR value by the proposed method, whereas DPMS-E was still undetected by the Chi-square attack. Hence, it was proven that the proposed method is able to embed up to 4 bpp with a greater PSNR while being less detectable. The achieved PSNR results are shown in Table 4-11. Our proposed method was applied to the same images and embedded with the same number of bits. It was also shown that the proposed method performs better compared to other classical steganographic schemes in terms of PSNR values. Thien and Lin had a smaller PSNR value, and their method is detectable even with a simple Chi-square attack that is completely outdated (Fig. 4.7 and 4.8). We can see that roughly 40 % of the pixels are changed with a probability of 1, while this value equals 1 for 100 % of the pixels if 4 bpp is embedded.

As described earlier for classical steganography, the MBNS method [39] contributed by presenting higher imperceptibility values (i.e., PSNR) over PVD [30] and BPCS [41]. Table 4-11 shows that the proposed method performs better than other methods in terms of reported PSNR values. The reason is that the decimal value of a block gets smaller if it is divided into a generated random number. Because of this, the decimal value of a 32-bit block would be broken down into much smaller remainders. For embedding smaller remainders, a lower number of bases is required,

so that less change is applied to the cover image. Less change leads us to a greater PSNR value. In this regard, more secret bits are embedded, even though less change is made to the cover image. Larger embedding rates need larger RNDs, too. Unlike the MBNS method, the greatest change to be enforced on each pixel is computed based on an error image, as explained. Furthermore, the impact of the change is minimized using LSB matching.

Table 4-11. A comparison to the classical steganographic schemes in terms of PSNR value

Payload (bpp)	Capacity (bits)	Cover Image	Embedding Method	PSNR (dB)
1.68	440,000	<i>Baboon</i>	Modified PVD	40.07
			DPMS-E (IQF=4, OEF=0)	50.10
1.68	440,000	<i>Lena</i>	BPCS	34.60
			PVD	41.70
2.52	660,725	<i>Baboon</i>	DPMS-E (IQF=2, OEF=0)	50.37
			Side-Match	33.53
2.5	655,360	<i>Lake</i>	DPMS-E (IQF=2, OEF=0)	44.98
			Zeki et al.	27.52
2.82	740,000	<i>Man</i>	Parah et al.	34.59
			DPMS-E (IQF=0.01, OEF=0)	45.29
3.0	786,432	<i>Peppers</i>	MBNS	38.10
			DPMS-E (IQF=0.01, OEF=5)	40.84
4.0	1,048,576	<i>Lake</i>	K-LSB	37.92
			Soleimani and Niazi Torshiz	40.27
			DPMS-E (IQF=0.01, OEF=5)	40.35
			Simple LSB replacement	31.83
4.0	1,048,576	<i>Lake</i>	Chang et al.	31.89
			Chen	34.34
			Thien and Lin	34.80
			DPMS-E (IQF=0.01, OEF=10)	36.91

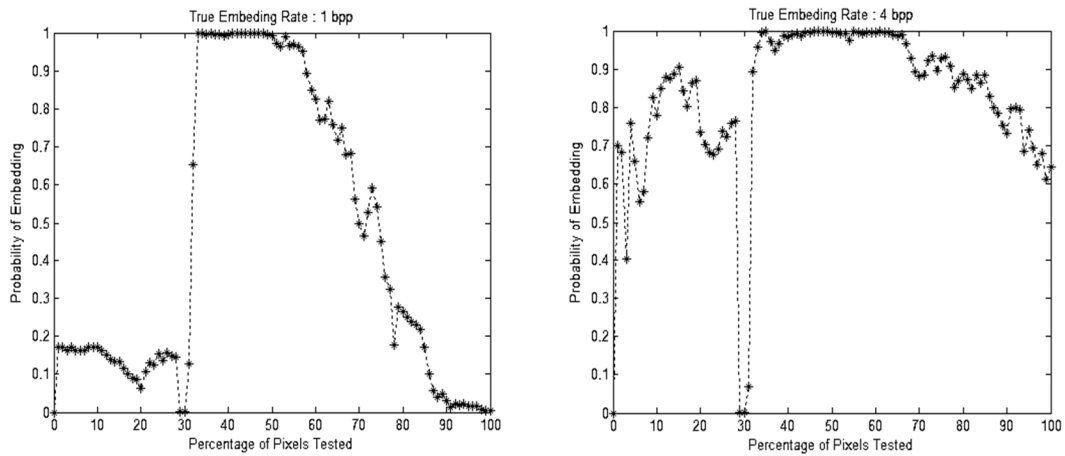


Figure 4.7 Chi-square attack on two stego images embedded with 1 and 4 bpp by Thien and Lin [46]

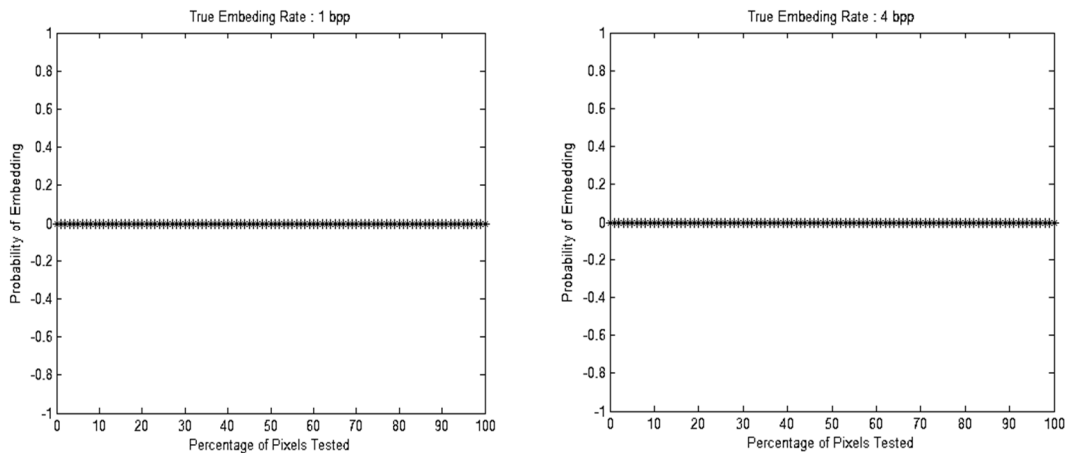


Figure 4.8 Chi-square attack on two stego images embedded with 1 and 4 bpp by DPMS-E

Fig. 4.9 shows the first grayscale image out of 10,000 images provided by the image database of BOSS version 1.01. In a simple experiment, we attempted to prove that higher IQF values would result in more undetectability of the stego image. Theoretically, PSNR values will be greater if we employ a higher IQF. Note that there are two error images introduced in the proposed method. One is computed in Phase 1 of the algorithm. As with either of Figs. 4.10 or 4.11, another error image is calculated between the stego image and the original image to show where, and to what extent, the impact of embedding has happened. In Fig. 4.10, the experiment is performed to embed a payload of 0.4 bpp using an IQF of 83. As can be seen, the secret bits are scattered through the cover image (PSNR 57.42 dB), and we can hardly see the detailed texture of the image (Fig. 4.9). On the other

hand, Fig. 4.11 represents the detail of the image texture when the secret bits are not well scattered but lumped up in the middle of the image (PSNR 57.28 dB, IQF 20). The imperceptibility level is shown by the PSNR values, and the second experiment shows a smaller PSNR, so it is considered more visible, such that the detected probability of error is also less than the one with IQF 83. The reason is that IQF 20 has bigger computed Base values, so the secret bits can be embedded using fewer pixels, while the change is much bigger. IQF 83 has smaller Base values, and the secret bits are depicted using more pixels, while the change is well distributed among more pixels and is much smaller, too. Feature extraction methods are more sensitive on the busy areas, and the stego image would be broken as soon as it embeds more in busy regions. IQF 83 tries to embed in smooth areas, whereas IQF 20 is more concentrated on busy areas. This is revealed in the error image because the texture is clearer when using IQF 20.



Figure 4.9 First cover image from image database BOSS 1.01 called “1.pgm”

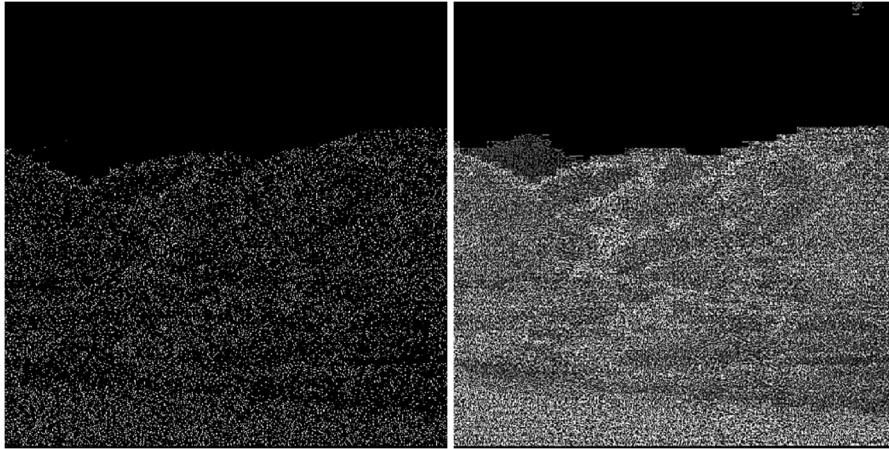


Figure 4.10 Left, the error image, and right, the *Base* matrix (IQF 83, Payload 0.4 bpp, 57.42 dB)

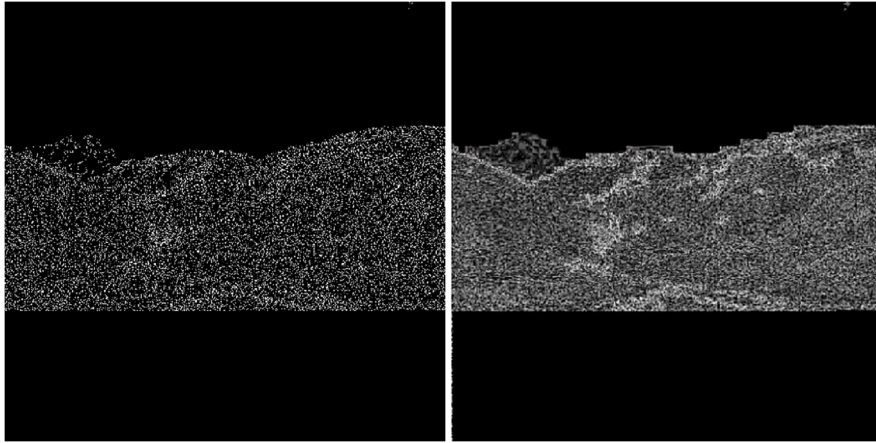


Figure 4.11 Left, the Error image, and right, the *Base* matrix (IQF 20, Payload 0.4 bpp, 57.28 dB)

Table 4-12. DPMS-E experiment results with different IQFs for the same payload

BOSS 1.01 database (10,000 images)		Experiment 1		Input Parameters		Experiment 2		Input Parameters	
Payload (bpp)	Capacity (bits)	P_e	PSNR (dB)	IQF	OEF	P_e	PSNR (dB)	IQF	OEF
0.05	13,101	0.4505	67.29	90.0	0	0.4814	68.79	96.0	0
0.1	26,214	0.3975	63.86	80.0	0	0.4571	65.43	94.0	0
0.2	52,428	0.3351	61.14	70.0	0	0.4089	62.57	90.0	0
0.3	78,643	0.2745	59.43	60.0	0	0.3605	60.89	85.0	0
0.4	104,857	0.1838	58.39	50.0	0	0.2697	59.49	75.0	0

Table 4-12 benchmarks the experiment using 10,000 images from the BOSS database. It is clear that experiment “2” is less detectable due to the greater values of error and a complete scattering of the secret bits. Smaller values of IQF increase the probability of being detected with less average testing error (P_e) by ensemble classifiers.

The proposed method shows how steganalysis would be affected by the JPEG image quality factor and JPEG recompression artifacts. They make it more confusing to steganalysis schemes. It is also proven that a higher IQF number guarantees less detectability compared to the edge adaptive method, and is close to HUGO’s detectability level because the algorithm distributes the secret bits through the cover image more evenly and selects the right pixels to hold secret bits using a calculated Base matrix as a guide. The execution time of the proposed method is considered more than 2 times faster than HUGO. In addition to this, unlike LSB matching-based methods, EA and HUGO embed at 1 bpp, whereas this scheme can embed at 4 bpp.

D. Parallel modular steganography using error images

The experiments were conducted with two objectives. The first was to compare the undetectability level achieved by MS-E to state-of-the-art methods such as EA and HUGO. Then, an effectiveness and processing speed analysis of the block-wise parallel approach was calculated mathematically to demonstrate the throughput gained by parallel processing.

We used both SQUARE and SPAM features because SQUARE features generally provide less detectability error. This is because SQUARE features result from third-order residuals and have superior detection accuracy to first or second-order residuals (SPAM) for all methods [60].

Table 4-13 presents the average testing error (P_e) using second order SPAM and SQUARE+h^x features over ten splits of 10,000 images in the BOSS database. According to Table 4-13 results, Table 4-14 confirms that the global P_e of MS-E, the minimum average classification error for every embedding method, contributes more than EA and both of HUGO methods in terms of a lower detectability level through the use of ensemble classifiers using different features (higher average testing error).

Table 4-13. Average testing error over 10 splits (p_e) comparison on BOSS 1.01 database

Payload (bpp)	Capacity (bits)	EA		HUGO T=90		HUGO T=255		MS-E		
		SPAM	SQUARE+ h^x	SPAM	SQUARE+ h^x	SPAM	SQUARE+ h^x	SQUARE+ h^x	IQF	PSNR (dB)
0.05	13,101	0.47	0.42	0.50	0.32	0.44	0.47	0.47	96	65.2
0.1	26,214	0.43	0.36	0.48	0.29	0.39	0.45	0.49	94	62.3
0.2	52,428	0.33	0.27	0.44	0.22	0.32	0.40	0.37	90	59.5
0.3	78,643	0.25	0.21	0.40	0.16	0.26	0.35	0.34	85	57.8
0.4	104,857	0.19	0.16	0.36	0.12	0.20	0.24	0.28	75	56.6
0.5	131,072	0.15	0.14	0.31	0.09	0.14	0.16	0.27	65	55.9

Table 4-14. Global p_e while resisting 2nd order SPAM and SQUARE+h^x features

Payload (bpp)	Capacity (bits)	EA	HUGO T=90	HUGO T=255	MS-E
0.05	13,101	0.4292	0.3233	0.4432	0.4779
0.1	26,214	0.3668	0.2911	0.3993	0.4504
0.2	52,428	0.2795	0.2254	0.3262	0.3798
0.3	78,643	0.2112	0.1648	0.2630	0.3401
0.4	104,857	0.1644	0.1284	0.2008	0.2477
0.5	131,072	0.1421	0.0926	0.1468	0.1697

Table 4-15. Global p_e of Adaptive ± 1 Steganography in Extended Noisy Regions [62] and MS-E

Payload (bpp)	Capacity (bits)	Adaptive ± 1 Steganography in Extended Noisy Regions [11]			MS-E		
		SPAM (dim 686)	SQUARE+h ^x (dim 342)	global P_e	SPAM (dim 686)	SQUARE+h ^x (dim 342)	global P_e
0.05	13,101	0.4917	0.4647	0.4647	0.4920	0.4932	0.4920
0.1	26,214	0.4711	0.4207	0.4207	0.4585	0.4681	0.4585
0.2	52,428	0.4382	0.3441	0.3441	0.4228	0.4207	0.4207
0.3	78,643	0.3898	0.2743	0.2743	0.3937	0.4073	0.3937
0.4	104,857	0.3303	0.2110	0.2110	0.3206	0.3665	0.3206
0.5	131,072	0.2833	0.1672	0.1672	0.1697	0.2983	0.1697

Table 4-15 presents a comparison in terms of the global P_e , the minimum average classification error for MS-E and Adaptive ± 1 Steganography in Extended Noisy Regions [62]. The experiment applied both methods on 1,000 randomly chosen images from the BOSS database and the secret data was generated randomly for both methods. It is clear that the minimum testing error has greater values for the proposed method. This implies that MS-E can embed more adaptively in noisy regions without applying a noisy function. The error image by itself simply conveys where and to what extent the secret data can be embedded.

Table 4-16. Time performance comparison (in seconds)

Embedding Rates(bpp)	0.05	0.1	0.2	0.3	0.4	0.5
HUGO	4.37	4.43	4.52	4.63	4.69	4.81
Extended HUGO	4.74	4.79	4.97	5.07	5.16	5.33
MS-E	1.78	1.89	2.13	2.27	2.46	2.63

Table 4-16 presents the embedding processing times using the 512×512 grayscale image “Lena” as the cover. The execution time of the proposed algorithm is approximately 2.32 times faster than the extended HUGO method [59] in serial mode. Adaptive ± 1 Steganography in Extended Noisy Regions [62] is reported to be 3.93 times faster than the extended HUGO [59] using a C implementation. However, it can be detected more easily than MS-E (Table 4-15). Regarding time performance, MS-E could have executed faster if it had been implemented in C. It executed in MATLAB, which is slower than C code, a lower-level language that generates relatively optimized machine code when compared to high-level languages such as MATLAB. In the next section, we will prove that, unlike [62] that has block dependency in noisy regions, the proposed method can be executed more quickly because it has the ability to be parallelized. The current method is a semi-blind steganography method. This implies that the receiver has the original image in advance or the compressed “Base matrix” is provided before the stego image is sent. Knowing the original image and the sender’s image quality factor (shared key), the receiver can easily create the base matrix without applying noisy functions to the neighboring blocks. Hence, the dependency rate between the blocks of pixels tends to zero and speedup is possible. Therefore, every block can be embedded independently from the processing of the neighboring blocks of pixels.

The proposed code is less complex than that of HUGO and Adaptive ± 1 Steganography in Extended Noisy Regions [62] because MS-E does not require noisy functions and relies only on an interpretation of the error image. There is no requirement to calculate more than 107 features, as required by HUGO. Extended HUGO must compute additional features resulting in longer computation times, approximately two times slower than MS-E.

1. Speedup analysis

As discussed by Wu Lee [63], there are many factors influencing the performance of any parallel algorithm, such as concurrency, grain size, speedup, efficiency, and effectiveness. An algorithm is concurrent if a significant portion of its instructions can be run in parallel. The parameter T_p is the total number of instructions. The grain size parameter (G_p) depends on the number of instructions executable in parallel, using numerous PUs. Speedup is the measure calculated as below:

$$\text{Speedup}_{up} = \frac{\text{Sequential Computation Time}}{\text{Parallel Computation Time}} = \frac{T_p}{G_p}$$

The algorithm efficiency parameter (E_p) is calculated by dividing Sp into PU . Finally, a parallel algorithm is considered effective if it can maximize the product $Sp \times E_p$, as explained in [64] [65] [66].

The first serial method proposed in Section III.D is not considered concurrent, even though it is less detectable and executes more rapidly than others, because all the instructions must be run serially. Hence, block-wise and pixel-based algorithms are proposed to demonstrate that the first algorithm also has the potential to run in parallel.

First, the total number of either serial or parallel instructions must be recognized for every algorithm. Then, the effectiveness of the algorithm must be calculated based on the total number of instructions.

2. Pre-processing phase analysis

Table 4-17 indicates the total number of serial and parallel instructions according to the line numbers of the parallel pre-processing phase defined in Section III.D.3 and used at the beginning of the block-wise parallel algorithm. If-condition clauses of the pre-processing phase have the probability True for 0.5, which is equal to the applied embedding rate in this example. T_p is calculated as the sum of the right-most column of Table 4-17; its value is 561,159, including 12,295 instructions that can be run only serially.

Table 4-17. Pre-processing phase instruction-type analysis

From	To	Type	P (Ins #)	Repetition	P × Repetition
1	6	Serial	6	1	6
7.1.1	7.1.5	Parallel	5	64×64	20'480
7.1.6.1.1	7.1.6.1.2.2	Parallel	1+0.5× 2	8×8×64×64	524'288
7.1.8	7.1.8	Parallel	1	64×64	4'096
9	9	Serial	1	1	1
10.1.1	10.1.3	Serial	3	64×64	12'288

3. Block-wise speedup analysis

Instruction types are identified and shown in Table 4-18. Lines 2.1.2 and 2.1.3 are run in a nested For-loop; this is omitted herein to increase readability. The first and last non-zero cells of a matrix must be found. Therefore, 64 pixels must be read to determine these values (Repetition 8×8). In this situation, T_p equals 1,368,071, including 12,295 instructions that can be run only serially.

As images of size 512×512 are used, we have 262,144 pixels, each represented by one byte. The block size for the image compression is 8×8 . Considering this, we calculate other factors as below:

$$\begin{aligned}
 \text{Cover image size in pixels} &= 512 \times 512 = 262'144 \text{ pixels} \\
 \text{Block Size in pixels} &= 8 \times 8 = 64 \\
 \text{Total number of blocks} &= 256 \text{ KB} / 64 = 4096 \\
 \text{Maximum threads} &= 4096 \text{ (as many as block numbers)} \\
 \text{Desired embedding rate of 0.5 bpp} &= \frac{1 \text{ bit}}{2 \text{ pixel}} = \frac{32 \text{ bits}}{64 \text{ pixels}} \\
 \text{Average secret partition size for 64 pixels} &= 32 \text{ bits} \\
 \text{Total number of symbols} &= 4096 \times 32 \text{ bits} = 16'384 \text{ bytes} \\
 \text{Assumed message size} &= 131'072 \text{ bits} = 16'384 \text{ bytes}
 \end{aligned}$$

The proposed method could embed more or fewer than 32 bits in a block of 64 pixels, depending on the texture of each block. The embedding rate 0.5 bpp implies that approximately 32 bits can be embedded from 64 pixels if half of the bases equal two. The secret partition size increases if the bases are significantly greater than two. The opposite is also true. We can calculate the effectiveness of the block-wise parallel algorithm as depicted in Table 4-19.

Table 4-18. Block-wise instruction-type analysis

From	To	Type	P (Ins. #)	Repetition	P × Repetition
1	1	Serial	12'295	1	12'295
1	1	Parallel	548'864	1	548'864
2.1.1	2.1.1	Parallel	1	64×64	4'096
2.1.2	2.1.3	Parallel	1	8×8×64×64	262'144
2.1.4	2.1.7	Parallel	4	64×64	16'384
2.1.1	2.1.1	Parallel	2	8×8×64×64	524'288

Table 4-19. Maximization look-up table

PU	Gp	Sp	Ep	Sp×Ep
1	1'368'071	1	1	1
50	39'410.52	34.71	0.6943	24.1
80	29'242.2	46.78	0.5848	27.36
100	25'852.76	52.92	0.5292	28.003
110	24'620.76	55.57	0.5052	28.069
120	23'593.13	57.99	0.4832	28.019
400	15'684.44	87.23	0.2181	19.02
10'000	12'430.58	110.06	0.011	1.21
500'000	12'297.71	111.25	0.000222	0.0248
1'355'776	12296	111.26	0.000082	0.0091

As Fig. 4.12 indicates, the value of $S_p \times E_p$ is maximized at processing unit 110 (pixel in red). We can conclude that the algorithm is effective and that for an embedding rate of 0.5 bpp, 110 is the optimal number of PUs.

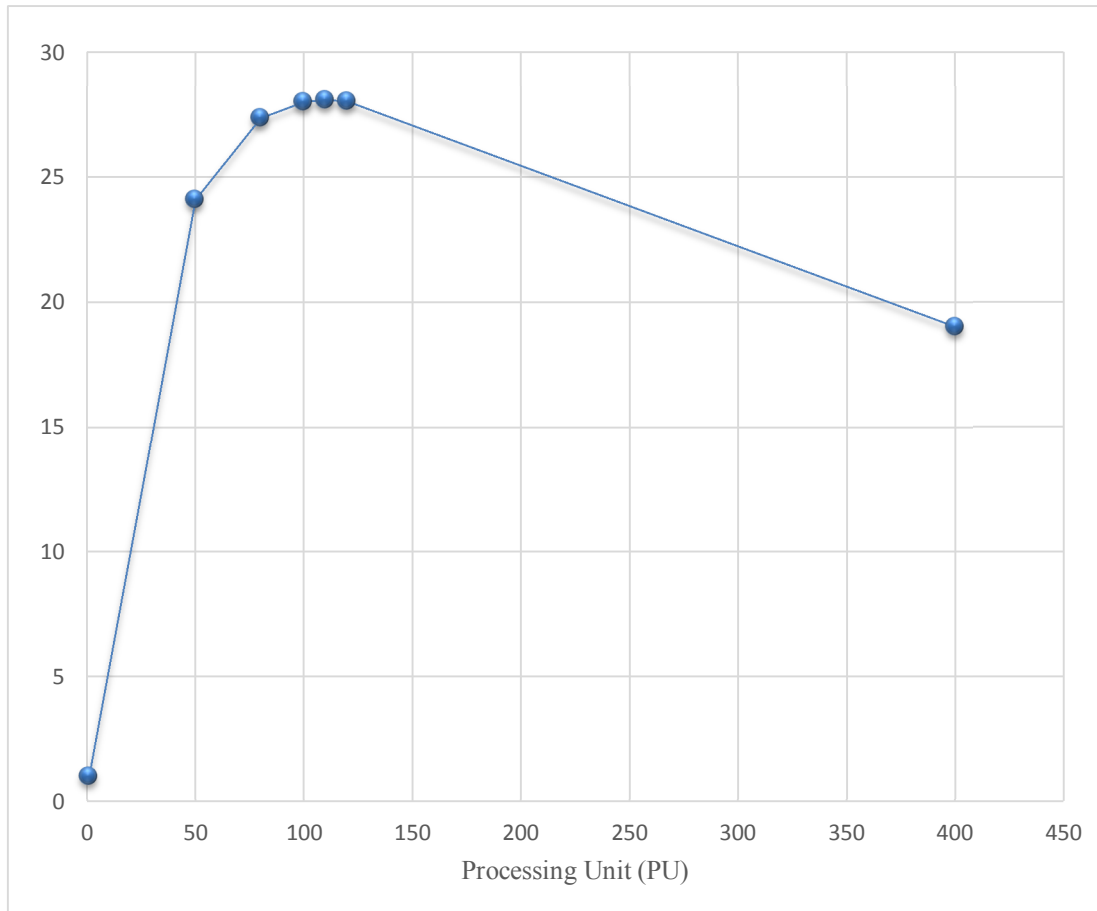


Figure 4.12 Maximization point shown at number of PUs

V. Conclusion

We proposed an efficient steganographic method which is not only highly undetectable, but also one whose time performance is greater than the state-of-the-art approaches. In addition, the proposed method has the potential to be implemented in parallel while it is highly less detectable than the state-of-the-art methods.

“Adaptive least significant bit matching revisited with the help of error images” has shown how ensemble classifiers would become confused by the JPEG IQF and JPEG decompression artifacts. It is also proven how undetectable a stego-image will become if we distribute the secret bits through the cover image more adaptively and select the right pixels to hold secret bits using a calculated Base matrix as a guide. The guide exploits the texture of the cover image so that the secret data bits are distributed through the white target pixels while trying to affect the cover image the least by utilizing LSBM-R method for every pair of target pixels. The execution time of the proposed method is proven to be around two times faster than that of the HUGO methods with two different settings.

MS-E which stands for “Modular steganography with the help of error images” simplifies DPMS-E and imposes lesser changes on the cover image pixel values because the decimal value of every 32-bit secret block becomes smaller if it is divided into multiple base numbers. Hence, the decimal value of every 32-bit secret block is broken down into much smaller remainders and a small change in pixel value is required, consequently. Thus, less change leads to a greater PSNR value and lower detectability while more secret bits are embedded utilizing either LSBM-R or MBNS method. Unlike the original MBNS method, the greatest change to be enforced on each pixel is computed from an error image and the smallest impact of the change is minimized using LSB matching.

DPMS-E stands for “Double phase modular steganography with the help of error images” and it applies a second phase of implicit compression using pseudo random numbers. Thus, unlike LSB matching-based methods, EA and HUGO methods that can only embed up to 1 bpp, this scheme could even embed up to 4 bpp. DPMS-E is less detectable compared to the MS-E due to the application of an extra phase using pseudo random numbers.

We proved that ensemble classifiers have a greater probability of error when detecting the resulting stego-images. Furthermore, the execution time of the proposed method in serial was verified to be approximately 2.32 times faster than HUGO. Moreover, we explained that owing to their embedding dependency, EA and HUGO cannot be implemented in parallel while the proposed “Parallel Modular Steganography Using Error Images (P-MSE)” has a speedup of over 55 times using 110 PUs. Thus, the block-wise parallel algorithm can be an effective, highly undetectable parallel implementation of MS-E in real-time processing. It can even be more undetectable if we use DPMS-E.

The drawback of the proposed scheme is that it requires a Base matrix to extract the secret information successfully so that the existence of a cover image would not be necessary. For future work, a CUDA (Compute Unified Device Architecture) implementation of the block-wise algorithm can be investigated with NVIDIA graphic cards or field-programmable gate array boards.

Bibliography

- [1] K. David, *The Code breakers*, Weidenfeld and Nicolson, 1974.
- [2] E. T. Lin and E. J. Delp., A review of data hiding in digital images, 1999.
- [3] H. S. Zim, *Codes and secret writing*, W. Morrow, 1948.
- [4] M. F. A. S. Halim Suhaila Abd, "Embedding using spread spectrum image steganography with GF," in *IMT-GT-ICMSA*, Kuala Lumpur, 2010.
- [5] P. Birgit, "Information Hiding Terminology- results of an informal plenary meeting and additional proposals," *Springer*, vol. 1174, pp. 347-350, 1996.
- [6] S. C. A. S. N. C. D. Roy Ratnakirti, "Evaluating image steganography techniques: Future research challenges.," in *International Conference on*, 2013.
- [7] N. Johnson, "Steganography; technical report," Available: http://www.jjtc.com/pub/tr_95_11_nfj/, 1995.
- [8] H. Nagham, A. Yahya, R. B. Ahmad and O. M. Al-Qershi, "Image Steganography Techniques: An Overview," *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 3, pp. 168-187, 2012.
- [9] T. Jamil, "Steganography: the art of hiding information in plain sight.," *IEEE Potentials*, vol. 18, no. 1, pp. 10-12, 1999.
- [10] C. Abbas, J. Condell, K. Curran and P. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing*, vol. 90, no. 3, pp. 727-752, 2010.
- [11] S. B. Sadkhan, "Cryptography: Current status and future trends," *International Conference IEEE Proceedings on Information and Communication Technologies: From Theory to Applications*, pp. 417-418, 2004.
- [12] K. S. K. B. R. R. K. C. a. S. P. Kumar, "Performance comparison of robust steganography based on multiple transformation techniques," *International Journal of Computer Technology and Applications*, vol. 2, no. 4, pp. 1035-1047, 2011.
- [13] R. Ratnakirti, S. Changder, A. Sarkar and N. C. Debnath, "Evaluating image steganography techniques: Future research challenges.," in *International Conference on In Computing, Management and Telecommunications (ComManTel)*, 2013.
- [14] C. Po-Yueh and H.-J. Lin, "A DWT based approach for image steganography," *International Journal of Applied Science and Engineering*, vol. 4, no. 3, pp. 275-290, 2006.
- [15] M. Abdelhallim, T. Eude and H. Cherifi, "A comparison of image quality models and metrics based on human visual sensitivity," in *Image Processing, 1998. ICIP 98. Proceedings.*, 1998.
- [16] Z. a. A. C. B. Wang, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81-84, 2002.
- [17] C. A. Stanley, *Pairs of Values and the Chi-squared Attack*, Department of Mathematics, Iowa State University, 2005.
- [18] M. H. T. S. a. N. M. Kharrazi, "Performance study of common image steganography and steganalysis techniques," *Journal of Electronic Imaging*, vol. 15, no. 4, pp. 041104-041104, 2006.
- [19] N. Hamid, A. Yahya, R. B. Ahmad, D. Najim and L. Kanaan, "Steganography in image files: A survey," *Australian Journal of Basic and Applied Sciences*, vol. 7, no. 1, pp. 35-55, 2013.

- [20] S. G.S., S. D. B. and R. S.M., "A Spatial Domain Image Steganography Technique Based on Plane Bit Substitution Method," *Global Journal of Computer Science and Technology*, vol. 12, no. 15, 2012.
- [21] B. Souvik, "A survey of steganography and steganalysis technique in image, text, audio and video as cover carrier," *Journal of global research in computer science*, vol. 2, no. 4, 2011.
- [22] C. Chin-Chen, J.-Y. Hsiao and C.-S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy," *Pattern Recognition*, vol. 36, no. 7, pp. 1583-1595, 2003.
- [23] C. Chi-Kwong and L.-M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern recognition*, vol. 37, no. 3, pp. 469-474, 2004.
- [24] L. Yeuan-Kuen and L.-H. Chen, "High capacity image steganographic model," *IEE Proceedings-Vision*, vol. 147, no. 3, pp. 288-294, 2000.
- [25] W. Ran-Zan, C.-F. Lin and J.-C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern recognition*, vol. 34, no. 3, pp. 671-683, 2001.
- [26] B. Walter, "Techniques for data hiding," *IBM systems journal*, vol. 35, no. 3.4, pp. 313-336, 1996.
- [27] N. B. Cong, S. M. Yoon and a. H.-K. Lee, "Multi bit plane image steganography," *Digital Watermarking. Springer Berlin Heidelberg*, pp. 61-70, 2006.
- [28] C. Brian and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding.," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1423-1443, 2001.
- [29] W. Da-Chun and W.-H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1613-1626, 2003.
- [30] L. Xiaolong, B. Li, X. Luo, B. Yang and R. Zhu, "Steganalysis of a PVD-based content adaptive image steganography," *Signal Processing*, vol. 93, no. 9, pp. 2529-2538, 2013.
- [31] H. Wien, "Adaptive image data hiding in edges using patched reference table and pair-wise embedding technique," *Information Sciences*, vol. 221, pp. 473-489, 2013.
- [32] Z. Xinpeng and S. Wang, "Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security," *Pattern Recognition Letters*, vol. 25, no. 3, pp. 331-339, 2004.
- [33] P. V.M and C. E, "Grey level modification steganography for secret communication," in *2nd IEEE International Conference on industrial Informatics.*, Berlin, 2004.
- [34] Al-Taani, A. T. and A. M. Al-Issa, "A Novel Steganographic Method for Gray-Level Images," in *International Journal of Computer, Information, and Systems Science, and Engineering*, 2009.
- [35] O. T. Joseph and A. O. Adetunmbi, "Development of multi-level system of steganography," *Journal of Computer Science & Technology*, vol. 13, 2013.
- [36] C. Christian, "An information-theoretic model for steganography," *Springer Berlin Heidelberg - Information Hiding*, pp. 306-318, 1998.
- [37] N. Amitava, S. Biswas, D. Sarkar and P. P. Sarkar, "A novel technique for image steganography based on Block-DCT and Huffman Encoding," *arXiv preprint arXiv*, pp. 1006-1186, 2010.
- [38] C. Chin-Chen and H.-W. Tseng, "A steganographic method for digital images using side

- match," *Pattern Recognition Letters*, vol. 25, no. 12, pp. 1431-1437, 2004.
- [39] Z. Xinpeng and S. Wang, "Steganography using multiple-base notational system and human vision sensitivity," *IEEE Signal Processing Letters*, vol. 12, no. 1, pp. 67-70, 2005.
- [40] W. Da-Chun and W.-H. Tsai, "A steganographic method for images by pixel value differencing," *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1613-1626, 2003.
- [41] N. Hideki, J. Spaulding, M. N. Shirazi and E. Kawaguchi, "Application of bit-plane decomposition steganography to JPEG2000 encoded images," *IEEE Signal Processing Letters*, vol. 9, no. 12, pp. 410-413, 2002.
- [42] L. Chang-Chou and W.-H. Tsai, "Secret image sharing with steganography and authentication," *Journal of Systems and Software*, vol. 73, no. 3, pp. 405-414, 2004.
- [43] Y. Ching-Nung, T.-S. Chen, K. H. Yu and C.-C. Wang, "Improvements of image sharing with steganography and authentication," *Journal of Systems and software*, vol. 80, no. 7, pp. 1070-1076, 2007.
- [44] C. Chin-Chen, C.-Y. Lin and Y.-Z. Wang, "New image steganographic methods using run-length approach," *Information Sciences*, vol. 176, no. 22, pp. 3393-3408, 2006.
- [45] S. S. Rahman and M. N. Torshiz., "A new high quality vision non-adaptive steganographic method, using module and combined functions.," *Int J Emerg Trends in Signal Process*, pp. 2319-9784, 2013.
- [46] T. Chih-Ching 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*, vol. 36, no. 12, pp. 2875-2881, 2003.
- [47] C. Shang-Kuan, "A module-based LSB substitution method with lossless secret data compression," *Computer Standards & Interfaces*, vol. 33, no. 4, pp. 367-371, 2011.
- [48] Z. A. M., A. A. Manaf and S. S. Mahmod, "High watermarking capacity based on spatial domain technique. Information technology journal," *Information technology journal*, vol. 10, no. 7, pp. 1367-1373, 2011.
- [49] P. S. A., J. A. Sheikh and G. M. Bhat, "High Capacity Data Embedding using joint Intermediate Significant Bit (ISB) and Least Significant Bit (LSB) Technique," *Journal of Information Engineering and Applications*, vol. 2, no. 11, pp. 1-11, 2012.
- [50] J. Mamta and P. S. Sandhu, "Two Components based LSB and Adaptive LSB Steganography based on Hybrid Feature Detection for Color Images with improved PSNR and Capacity," *International Journal of Computer Science and Electronics Engineering (IJCSSEE)*, vol. 1, no. 2, 2013.
- [51] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," *Information Hiding*, 2000.
- [52] K. AD., "Improved detection of LSB steganography in grayscale images," *Information Hiding*, 2005.
- [53] M. J., "LSB matching revisited," *Signal Processing Letters*, vol. 13, no. 5, pp. 285-287, 2006.
- [54] L. WW., H. F. and H. J., "Edge adaptive image steganography based on LSB matching revisited," *Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 201-214, 3 March 2010.
- [55] B. P., F. T. and P. T., "Break Our Steganographic System": the ins and outs of organizing BOSS," *Information Hiding*, 2011.

- [56] F. T., P. T. and B. P. , "BOSS(Break Our Steganography System)," [Online]. Available: <http://boss.gipsa-lab.grenoble-inp.fr>. [Accessed July 2010].
- [57] P. T., B. P. and F. J., "Steganalysis by subtractive pixel adjacency matrix," *Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215-224, 2010.
- [58] K. J., F. J. and H. V., "Ensemble classifiers for steganalysis of digital media," *Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 432-444, 2012.
- [59] K. J., F. J. and H. V., "On dangers of overtraining steganography to incomplete cover model," *Proceedings of the 13th ACM Multimedia & Security Workshop*, 29-30 September 2011.
- [60] F. J. and K. J., "Rich models for steganalysis of digital images," *Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868-882, 2012.
- [61] A. M., M. IY. and L. JA, "Double phase modular steganography with the help of error images," *Multimedia Tools and Applications*, vol. 72, no. 3, pp. 1573-7721, 2014.
- [62] H. T., Z. W., W. C., Y. N. and Z. Y., "Adaptive±1 Steganography in Extended Noisy Region," *The Computer Journal*, vol. 57, no. 4, pp. 557-566, 2013.
- [63] G. S., M. E., A. K. MU., C. C., C. S. and Wu-Lee, "Steganographic Algorithm on Binary Images Processed in Parallel," *IJVIPNS: International Journal of Video & Image Processing*, vol. 12, no. 3, pp. 1-4, 2012.
- [64] F. H., *Elemente te njehsimit parallel*, Tirana: SHBLU, 2004.
- [65] K. V., G. A., G. A. and K. G., "Introduction to parallel computing: design and analysis of algorithms," *Redwood City: Benjamin/Cummings*, vol. 2, no. 2, p. 110, 1994.
- [66] H. NJ., L. J. and V. A. L., "Provably secure steganography. Advances in Cryptology - CRYPTO," *Lecture Notes in Computer Science*, vol. 2002, no. 2442, pp. 77-92, 2002.
- [67] J. Delahaye, "information cach," *Information noyee, Pour La Science* , vol. 229, no. French, pp. 142-146, 1996.

ABSTRACT

Highly Undetectable Modular Steganography using Error Images

Afrakhteh Masoud

Advisor: Prof. Jeong-A Lee, Ph. D.

**Department of Computer Engineering
Graduate School of Chosun University**

Steganography detectability has become a major concern as the secret message size increases. As such, it has become more complex to maintain a greater level of undetectability in most state-of-the-art methods such as HUGO (highly undetectable steganography), Extended HUGO, Edge Adaptive (EA), ± 1 embedding in extended noisy regions and LSB-Matching revisited. They employ surrounding pixels to estimate the number of secret bits to conceal in a typical target pixel; however, they endure complex, time-consuming calculations considering the neighboring pixels and edge regions of a cover image. As a result, the stego-image might become less detectable by extracting steganalysis feature sets such as SPAM and Square+ h^x . Accordingly, their time performance level degrades as they try to gain a lower level of undetectability. The state-of-the-art methods have to embed pixel by pixel in serial while depending on the previous changed pixel values for the next target pixels and the presence of such data dependency does not make parallel execution feasible.

The main goal of this research is to come up with an efficient steganographic method which is not only highly undetectable, but one whose time performance is greater than the state-of-the-art approaches. In order to achieve this, we formulated and designed a method using an error image by differentiating an original image from its respective JPEG compressed image for every non-overlapping block of 8×8 pixels from a cover image. The error image can be simply considered as an embedding map implying where and to what extent secret bits are to be embedded. In addition, the amount of secret bits (payload) will be known to the sender prior to initializing the embedding process so that each processing unit for an individual block of pixels will easily select a pre-

assigned partition of secret bits to embed in the same block. Hence, all processing units can be run in parallel while there will be no scrambling of the secret bits after extraction.

The proposed method is proven to be highly undetectable because our proposed method basically distributes the secret bits through the cover image more adaptively and selects the right pixels to hold secret bits using a calculated Base matrix as a guide. The guide exploits the texture of the cover image so that the secret data bits are distributed through the proper target pixels. Secondly, the decimal value of every 32-bit secret block becomes smaller if it is divided into multiple base numbers assigned to cover image pixels. The decimal value of a 32-bit secret block was divided into some pseudo random number and broken down into much smaller remainders. For embedding smaller remainders, a smaller change in pixel value is required; hence, smaller changes are applied to the cover image. As a result, less change led to a greater PSNR value and lower detectability.

We also explained that owing to the existing embedding dependency of the state-of-the-art methods, they cannot be implemented in parallel while in the proposed “Parallel Modular Steganography Using Error Images (P-MSE)”, every processing unit could embed a part of secret partition in a pre-defined block of pixels. Accordingly, scrambling of the secret bits was avoided when they are extracted at the receiver side. The proposed method results in a speedup of over 55 times using 110 PUs.

ACKNOWLEDGMENT

I would first of all thank God Almighty who enabled me to carry out this project with full devotion and consistency. It is only because of His blessings that I could find my way up to the completion of this task.

Next, I would like to express my immense regards and honest gratitude to my supervisor, Prof. Jeong-A Lee. Her valuable support, guidance, appreciation and supervision, throughout the course of this novel research, motivationally steered me to accomplish this task successfully.

I would like to thank my father for all his support till his last moments of his life. Also my lab mates Olufemi Adeluyi and Hossein Moradian for their absolute help.

Finally, I also pay my esteem regards to NIIED (National Institute for International Education) of South Korea, under the Korean Government Scholarship Program (KGSP), for its financial support during the period of my PhD studies.

I would love to dedicate this thesis to my lovely wife, Sujeong Lee, who has been always supporting me with her true love in the difficulties so that I could get over the pain of losing my dad and other problems.