國 立 交 通 大 學

資 訊 學 院

資 訊 科 學 與 工 程 研 究 所

博士論文

# 影像隱藏、驗證與修復之研究
# A Study on Image Hiding, Authentication, and Recovery

研 究 生：張 御 傑

指導教授：林 志 青 博士

中華民國九十八年二月

# 影像隱藏、驗證與修復之研究
# A Study on Image Hiding, Authentication, and Recovery

| | |
|---|---|
| 研 究 生：張 御 傑 | Student：Yu-Jie Chang |
| 指導教授：林 志 青 博士 | Advisor：Dr. Ja-Chen Lin |

國 立 交 通 大 學
資 訊 學 院
資 訊 科 學 與 工 程 研 究 所
博 士 論 文

A Dissertation Submitted to
Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in
Computer and Information Science
February 2009
Hsinchu, Taiwan, Republic of China

中華民國九十八年二月

# 國 立 交 通 大 學

## 博碩士論文全文電子檔著作權授權書

（提供授權人裝訂於紙本論文書名頁之次頁用）

本授權書所授權之學位論文，為本人於國立交通大學資訊科學與工程研究所 __________組， 97 學年度第 二 學期取得博士學位之論文。

論文題目：影像隱藏、驗證與修復之研究
指導教授：林志青

■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學與台灣聯合大學系統圖書館：基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學及台灣聯合大學系統圖書館得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：

| | |
|---|---|
| 本校及台灣聯合大學系統區域網路 | ■ 立即公開 |
| 校外網際網路 | ■ 中華民國 99 年 3 月 4 日公開 |

■ **全文電子檔送交國家圖書館**

**授 權 人：張御傑**

**親筆簽名：** 張御傑

**中華民國** 98 **年** 3 **月** 5 **日**

# 國　立　交　通　大　學

## 博碩士紙本論文著作權授權書

(提供授權人裝訂於全文電子檔授權書之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學資訊科學與工程研究所 ______________組， 97 學年度第__二__學期取得博士學位之論文。

論文題目：影像隱藏、驗證與修復之研究
指導教授：林志青

■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行閱覽或列印。

本論文為本人向經濟部智慧局申請專利(未申請者本條款請不予理會)的附件之一，申請文號為：________________________，請將論文延至_____年_____月_____日再公開。

**授　權　人：張御傑**

**親筆簽名：**__張御傑__

**中華民國**　98　**年**　3　**月**　5　**日**

# 國家圖書館
# 博碩士論文電子檔案上網授權書

（提供授權人裝訂於紙本論文本校授權書之後）

ID:GT009023816

本授權書所授權之論文為授權人在國立交通大學資訊科學與工程研究所 97 學年度第二學期取得博士學位之論文。

論文題目：影像隱藏、驗證與修復之研究
指導教授：林志青

茲同意將授權人擁有著作權之上列論文全文（含摘要），非專屬、無償授權國家圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

※ 讀者基於非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

**授權人：張御傑**

**親筆簽名：**張御傑

**民國**98**年**3**月**5**日**

# 國立交通大學

## 資訊科學與工程研究所博士班

## 論文口試委員會審定書

本校　資　訊　科　學　與　工　程　研　究　所　　張　御　傑　君

所提論文　影　像　隱　藏　、　驗　證　與　修　復　之　研　究

合於博士資格水準、業經本委員會評審認可。

口試委員：[illegible]　[illegible]

黃仲陵　莊仁輝

范國清　傅楸善

[illegible]

指導教授：[illegible]

所　　長：[illegible]

中華民國 九十八 年 二 月 十六 日

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

Date: Feb. 16st, 2009

*We have carefully read the dissertation entitled* **A Study on Image Hiding, Authentication, and Recovery** *submitted by* **Yu-Jie Chang** *in partial fulfillment of the requirements of the degree of* Doctor *of* Philosophy *and recommend its acceptance.*

[illegible] [illegible]

[illegible] [illegible]

Kuo-Chin Fan [illegible]

[illegible]

Thesis Advisor: [illegible]

Director: [illegible]

# 影像隱藏、驗證與修復之研究

研究生：張御傑　　　　　　　　　　　指導教授：林志青　博士

國立交通大學　　　資訊學院　　　資訊科學與工程研究所

## 摘要

本論文提出了數種在網路或儲存系統上保護重要或隱私的影像之方法。在保護機密影像或資料方面，我們提出兩個資料隱藏的方法來隱匿隱藏之資料。在傳輸影像前，人們常將影像先進行壓縮以減少接收端的等待時間。有鑑於此，我們所提出的第一種方法是利用搜尋順序編碼法，將機密資料藏於利用向量量化壓縮法所產生的索引檔中，不僅不會對索引檔產生任何的失真，而且接收端在解壓縮的同時亦可以獲得所藏匿的機密資料。為了要隱匿更大的機密影像，第二種方法則是利用改良式搜尋順序編碼法(一種利用周圍像素值常有高度相似性的關係的編碼法)對機密影像進行編碼，並提供一個可調整的門檻值來控制抽取出的影像品質。在嵌入步驟中，利用周圍像素的變異數來決定掩護影像上每個像素的隱藏量，並搭配模運算將上述的MSOC 碼嵌入。從實驗結果得知，所產生的偽裝影像與抽取出的影像的視覺品質，與一些已發表的方法相比具有競爭性。

在保護重要影像方面，我們設計兩種具有不同修復能力的影像驗證系統。第一種是應用於保護單張重要影像。我們利用某些相關於影像區塊的性質產生驗證資料，並嵌入影像區塊中，可用來檢測影像的完整性。同時，我們利用向量量化編碼產生關於影像的修復資料，並搭配$(r, n)$門檻式分享方法，分散地藏到影像本身中。使得影像本身除了能偵測是否遭到惡意的竄改之外，還具有自我修復被破壞區域的能力。第二種是應用於多張影像傳輸或分散式儲存時，可能會有因網路或儲存系統不穩定而導致某些影像遺失的情形。為了解決這個問題，我們提出了一個具有交互修復能力的影像驗證系統，亦即設計一套雙層式分享機制來保存所有影像的修復資訊，使得影像本身除了原有的驗證功能之外，還能藉由其它倖存的影像互相合作，來修復那些遺失影像。

# A Study on Image Hiding, Authentication, and Recovery

Student: Yu-Jie Chang Advisor: Dr. Ja-Chen Lin

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

## Abstract

In this dissertation, we propose several techniques to protect important data and private images in a transmission or storage system. For a confidential image or secret data, two hiding methods are proposed to conceal the existence of the hidden data in the cover images. Because people often compress digital images to reduce the waiting time of the receiver before transmitting the images, the first method uses the search-order coding to embed secret data in the index file of the vector quantization compression result. The proposed method causes no distortion to the VQ version of the image, and the receiver end can obtain both the hidden data and the VQ image. To embed a bigger confidential image, the MSOC scheme in the second method utilizes the feature of high correlation among adjacent pixels (i.e. neighboring pixels are often with similar gray-values) to encode the important image. An adjustable threshold *T* is used in the MSOC; and this *T* directly controls the quality of the extracted image. In the embedding part, we use a variance-based criterion to estimate the hiding capacity of each pixel in the cover image. Then the MSOC code is embedded in the cover image using two sets of modulus function. Experimental results show that the quality of both the stego-images and extracted important images are competitive to those obtained in many existing steganography methods reported recently.

To protect important images in public environment, we develop two image authentication methods along with different recovery abilities. The first method is used to protect a single image. In the method, the authentication data for each block is generated using some related information within the block, and then embedding it into the block to serve as the attestation for the integrity of the image. Meanwhile, the recovery data obtained by vector quantization technique are shared by using an (*r*, *n*)-threshold sharing method, and then scattered all over the image. The proposed method can not only detect whether malicious manipulations have occurred, but also self-recover the tampered parts. In the transmission of multiple images, it is possible that the network connection is unstable; and hence, some images at the receiver end are lost. To solve this problem, we develop an image authentication method with cross-recovery ability to protect a group of images. In the method, a two-layer sharing scheme is designed to preserve the recovery data of all images. The proposed method can not only verify the integrity of each member of the image group, but also reconstruct those lost images by the mutual support of the surviving members.

# Acknowledgements

I would like to express my sincere appreciation to my advisor, Professor Ja-Chen Lin for his kind guidance and patience throughout the course of this dissertation as well as the invaluable training during my study.

Thanks are also given to Dr. Ran-Zan Wang, Dr. Chih-Ching Thien, Dr. Shang-Kuan Chen, and Mr. Sian-Jheng Lin for their helpful discussion and suggestions. Appreciations are also extended to all colleagues in the Computer Vision Laboratory at National Chiao Tung University for their assistance and helps.

Finally, I earnestly thank my family and friends for their love, encouragement and support. The thesis is dedicated them.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

In this chapter, the motivation for the dissertation is introduced in Section 1.1; and a brief review of the related researches is presented in Section 1.2. The overview of the proposed methods is in Section 1.3; and the organization of this dissertation is in Section 1.4.

## 1.1 Motivation

With the flourish of the Internet and the development of computer technologies, the transmission of digital media, such as text, image, audio, and video, via networks becomes more convenient. Especially, digital images are ubiquitous in individual or commercial activities. Some of these images are quite important or confidential, such as a close picture with lover, a secret agreement between two companies, or an image list of the new products in this year, and the loss or misappropriation of them may cause a large loss of money, time, or fame. Since Internet is a public environment, the transmitted material is exposed to high risk of illegal access and unauthorized tampering. As a result, the major goal of the dissertation is to develop several techniques to protect important or private images during a transmission through Internet.

The first research topic of this dissertation is about confidential images which is private and can not be shown publicly. To safely transmit this kind of image without causing an eavesdropper's perception, numerous techniques have been proposed. One of the famous solutions for protecting important images against illegal access is using encryption techniques [1-8]. An encryption scheme encodes an important image in such a way that it is nearly impossible for any person who does not possess the secret key to decode the cipher-image. The cipher-image might appear noisy, and so assure the content of the important image is not readable before decoding. However, looking noisy might thus attract the attention of attackers during the transmission activity; and hence increases the risk of getting attacks. Steganography [16-50] is an art of covert communication which aims to send the important image silently under the cover of some carrier signals [16], and thus compensates the aforementioned drawback of encryption technique. Unlike encryption techniques which make the content of the important image unreadable, steganography tries to cheat the hackers by concealing the existence of the content. (After hiding the important image in a cover image, they become a so-called stego-image, which still looks like the cover image, although the important image can be extracted from the stego-image.) Usually, if encryption is combined with a steganography scheme, the protection becomes more powerful. In this dissertation, we propose in Chapter 2 a steganogragphic scheme to embed secret data in the vector quantization (VQ) index file of a cover image such that the eavesdroppers can not notice the existence of secret data. In Chapter 3, we develop a hiding method for the transmission of a top-secret image via a cover image whose size is economic.

Another topic of this dissertation is about the public images whose contents reveal some meaningful information in public environment. In recent years, applications of digital images on Internet, such as digital library, digital bookstore, and network album, become very popular. Because an intrinsic property of digital images is the easy duplication and modification, they are exposed to high risk of being pirated and tampered. More specifically, the pervasive and powerful image manipulation tools developed nowadays have made imperceptible modification very easy. As a result, it is no longer true saying that to see is to believe. A well-know example is shown in Fig. 1.1. It is a forged image of Abraham Lincoln arm-in-arm with Marilyn Monroe on the 1994 February cover of *Scientific American*. Common knowledge indicates that Lincoln and Monroe are separated in birth by over 100 years, and it is easy to conclude that the image is forged. However, the image appears visually genuine. Therefore, an efficient method that can verify automatically the integrity of the contents of a digital image is urgently required. In the past decade, many methods [54-86] have been proposed to solve this problem. One of them is the technique of image authentication whose main function is to check whether the content of an image is tampered by illicit attackers or destroyed due to the unstable network connection. Most of them have the ability to detect whether malicious manipulations have occurred on the test image, and some are even have the position-locating ability to identify the image areas that have been tampered or manipulated. We believe that the recovery work for the tamper parts has to be taken immediately to reduce the possible damage and side effects caused by the manipulation. A simple method to recover a corrupted image is to replace the image with its backup copy. However, the extra storage space is a waste. Therefore, a mechanism of recovery that owns the ability to automatically correct the distorted contents of an image without accessing the backup copy is preferred. Therefore, we develop in Chapters 5 a technique which can not only detect-and-locate the tampered portion of an image, but also recover the tampered portion using the non-tampered part of the image itself.

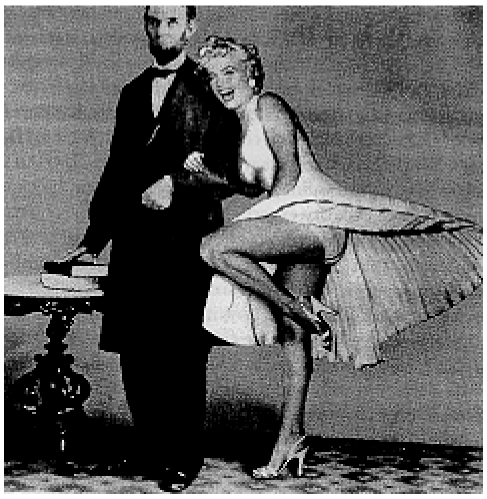


Fig. 1.1. A forged image of the 1994 February cover of *Scientific American*.

In daily life, transmission of multiple images via networks is a quite common activity. However, the network connection is sometimes unstable and causes some images missing. A general remedy is to request the sender end to resend those lost images, but the transmission time has been wasted. Furthermore, if the sender end does not immediately handle the send-again request or the network environment is still unstable, then the waiting time may influence the decision of an important business seriously when a cooperative company is eager to know what the content of the images about new products look like. Though the method proposed in Chapter 5 has the self-recovery capability to recover the lost portions of the protected image, if the whole image is lost, then this image is unable to be recovered. To solve this problem, in which some member images are lost during the transmission of a group of images, we develop an image authentication method with cross-recovery ability in Chapter 6. The word "cross-recovery" means that if some of the images in the group are lost during transmission, the lost images can be reconstructed by the mutual support of the surviving members at the receiver end.

## 1.2 Related Studies

### 1.2.1 Data Hiding

Data hiding is a technique which often emphasizes hiding important data of large size and yet keeping the fidelity of the cover image. A very simple approach is the least-significant-bits (LSB) substitution scheme [18-21]. This kind of method replaces directly certain bits of the cover image's pixels with the pixels' bits of the important image; and a random permutation process [9-10] to the important data is often conducted to enhance the security level of the method. In 2001, Wang et al. [18] used a genetic algorithm to approximate a theoretically-optimal solution of the simple LSB substitution method; however, due to the nature of genetic algorithms, the computation time is quite huge. Chang et al. [19] also used another algorithm to search for optimal substitution for the simple LSB substitution method. Although their PSNR performance is similar to Wang et al.'s method [18] (to hide a 256×256 important image Jet, the PSNR of the 512×512 Lena stego-images in Ref. [18] and Ref. [19] are, respectively, 44.172 and 44.169 dB); their embedding takes only about 1/7 of the time needed by Wang et al.'s method [18]. Chan and Cheng [20] used another pixel adjustment process to enhance the quality of the stego-image obtained by the simple LSB substitution method, and its extra computational cost is relatively small compared with Ref. [18].

To explore more steganography techniques, other studies based on vector quantization (VQ) have also been introduced [22-27]. In Ref. [22], Chung et al. proposed to hide an image using singular value decomposition (SVD) and VQ techniques. By using partial SVD procedure on both the important image and cover image, they hid some quantized singular values of a diagonal matrix corresponding to the important image in the less significant SVD components of the cover image. They can hide an important image whose size is as large as that of the cover image, but the degradation of the extracted important image is not very small. In Ref. [23], before the embedding process, Hu and Lin first encoded the important image according to a given codebook. Then, the obtained indices and related parameters are encrypted by a cipher scheme such as the Data Encryption Standard (DES) [1]. Finally, the encrypted data is embedded in the cover image using the LSB substitution. The hiding capacity of their scheme can be large, and the scheme can even embed multiple important images by using a codebook of smaller size and increasing the number of LSBs used in each pixel of the cover image. However, the quality of the extracted important images usually degraded. Some other hiding methods based on vector quantization (VQ) can be found in [24-27].

To reveal the important image without any loss, some studies based on difference have also been introduced [28-30]. Wu and Tsai [28] utilized the difference among two adjacent pixels in the cover image to hide the important data. However, the hiding capacities of their so-called PVD steganographic scheme depend on the nature of the cover image (smooth or noisy), and are usually smaller comparing with the aforementioned VQ-based method. In order to improve the hiding capacity, Wu et al. [29] utilized the simple LSB substitution method in smooth area of the cover image, and still used the PVD method [28] in edge area. Compared with the PVD method [29], this mixed version [29] obtains not only larger hiding capacity, but also better stego-image quality. To securely transmit a secret image whose size is half of the cover image, Li et al. [30] proposed a novel image hiding method based on the block difference. By exploring the block similarity between the cover image and the secret image, the stego-image quality can be significantly improved.

Some data hiding methods based on the modulus function can be found in the studies [31-33], and other data hiding methods for various applications are reported in

[34-50, 103-104].

### 1.2.2 Image Authentication

Image authentication is a technique for maintaining the integrity of an image; the goal is to detect and locate the positions where the image was tampered. According to integrity criteria, these schemes can be classified into hard authentication and soft authentication [54]. A hard authentication rejects any modification to the image content; but soft authentication passes certain content modifying, called incidental manipulations and rejects all the rest, called malicious manipulations. Generally, image authentication is divided into two common approaches, digital-signature-based and watermarking-based [54]. A digital-signature-based scheme [61-64] stores a so-called digital signature as the second file. The digital signature is a set of important features extracted from the original image, and can be used for authentication. The digital signature approach can tolerate some slight manipulations of the original image, unless the important features of the original image are changed. However, the separation of the authentication data from the images may increase management overhead for transmission and storage. Additionally, the digital signature usually does not locate where the image is tampered, and thus is not directly applicable to image recovery. Unlike digital-signature-based authentication schemes [61-64] which store the authentication data separately from the image, watermarking-based authentication schemes [65-78] embed the authentication data (watermark) directly into the original image. The direct embedding reduces the overhead caused by storing a separate file (digital signature), thus facilitating transmission. However, the embedded watermark is usually very sensitive to any manipulation to the watermarked image.

Many watermarking-based schemes can identify the image areas that have been tampered or manipulated by checking the presence and integrity of the local fragile watermarks. For instance, Wong [68] proposed a block-based authentication method. A hash function and XOR operations are applied to each block to create a corresponding watermarked block. Later in verification procedure, the embedding watermark is extracted from the least significant bits (LSBs) of the query image, decrypted using a public key, and finally applied an XOR operation with the hash value calculated from the query image block. Due to the property of the hash function, any tampering with a block generates an undesired binary output for that block. This scheme elegantly integrates cryptography with watermarking, and works well in detecting cropping attacks. However, because Wong's approach [68] is block-wise independent, it is vulnerable to the collage attack [88] (also known as the "vector quantization" (VQ) attack [87]). The attacker constructs a vector quantization codebook with codewords taken directly from the blocks of a set of watermarked images. The image to be counterfeited is then approximated by this codebook. Since each block is already authenticated by itself, the counterfeit image appears authentic to the watermarking algorithm. To solve this problem, Wong and Memon [69] proposed two more input parameters to the hash function of Wong's previous method [68]. These new input parameters are the block index *k* and the image identification number *ID*, both of which increase the difficulty of performing a collage attack. Their scheme not only makes the collage attack infeasible, but also maintains the tampered-area locating ability of the original version. However, the issue of the recovery of the tampered area is not yet provided in their scheme.

Several detecting-locating (authentication) methods with tamper recovery ability have recently been presented. Lin et al. [71] in 2005 proposed an attractive block-based digital watermarking scheme for images' tampering detection, location, and recovery. They used parity check and comparison to generate the authentication watermark of each block, and then added the recovery information, which recorded the six most significant bits (MSBs) of the mean value of another block, to form the embedding watermark. Finally, the embedding watermark was embedded using the simple LSB substitution method. Their verification procedure uses a four-layer hierarchical inspection system to increase the accuracy of locating the tampered area. Although their method can recover a tampered area by storing the mean of each block in the LSBs of another block (the backup block), it cannot recover a block if both this block and its single backup block are tampered at the same time (e.g. in a cropping attack of an extensive area). Hence, the tampered-area recovery ability is not strong enough in some situations. Luo et al. [72] in 2008 proposed a pixel-wise and block-wise complementary watermarking scheme based on digital halftoning and inverse halftoning techniques. Their method transforms the original image into its halftone version, which is then treated as the watermark, and embedded in the original image using the simple LSB substitution method. In tampering recovery, inverse halftoning is performed to repair the tampered areas. Wu and Chang [70] in 2002 developed a method based on the JPEG compression technique. Their method uses an edge detection technique to identify the edges of the image before the JPEG compression, then embeds the resulting edge characteristic into some AC coefficients of the frequency domain after the JPEG compression. If the image is tampered, then the embedded edge characteristic can be used to detect the tampered areas and cooperate the interpolation method to reconstruct it. Wang and Tsai [73] proposed a block-based digital watermarking scheme for image authentication and recovery in 2008. Their method selects the region of importance (ROI) in the image, and applies the fractal image coding for the blocks of ROI to generate the recovery data. If the tampered area is not the ROI, then damaged region is restored by image inpainting. Other image authentication methods with tampering recovery ability can also be found in the studies [74-78], and some image authentication schemes for verifying the integrity of a palette-based image can be found in [84-86].

### 1.2.3 Secret Sharing

The concept of secret sharing was introduced independently by Shamir [3] and Blakley [4] in 1979. Their ($r$, $n$) threshold scheme divides the secret data into $n$ shares. If any $r$ of these shares are available ($r$ is a predefined number and $2 \leq r \leq n$), then the secret data can be reconstructed, but fewer than $r$ shares cannot. One major advantage of secret sharing is fault-tolerant, because it allows $n - r$ shares to be absent due to network delay problem or storage damage. Consequently, the survival rate of secret data increases. Several secret sharing methods based on the ($r$, $n$) threshold scheme have been proposed [89-97]. Among them, Chang and Huang [89] applied the VQ technique [15] to encode the secret image, then the generated codebook is shared among the $n$ participants by applying the ($r$, $n$) threshold scheme. Thien and Lin [90] proposed an ($r$, $n$) scheme for sharing images. In their scheme, a secret image was shared among $n$ participants, and each participant held a generated shadow image whose size was only 1/$r$ of that of the secret image. The secret image could be reconstructed if at least $r$ of the $n$ shadow images were received. The smaller size of their shadow images ($r$ times smaller than the shadow images generated by ordinary sharing methods) is an advantage in transmission and storage. They further developed a method [91] that made the shadow images looked like portraits of the original secret image, and thus provided a user-friendly interface to facilitate the management of the shadow images.

As for other variations, Lin and Tsai [92] integrated the ($r$, $n$) threshold scheme with watermarking, thus ensuring that each share has authentication capability to verify its integrity before reconstructing the secret image. Chen and Lin [93] applied a secret image sharing scheme [90] to transmit an image in a progressive way. Wang and Shyu [94] also proposed a scalable secret image sharing scheme with three sharing modes: 1) multi-secrets, 2) priority, and 3) progressive, according to the spatial and depth information to increase the potential application for secret image sharing. Combination of Shamir's masterpiece [3] with visual cryptography (VC) [5] can be found in Ref. [95]. Even the visual cryptography [5] itself has many extensions. For example, Wu and Chang [100] employed circular shape of shares and improve the amount of the embedded message in traditional visual cryptography [5] that uses rectangular shapes, without sacrificing the clarity quality of the stacking result.

Besides the aforementioned spatial-domain methods, Lin and Tsai [96] proposed a frequency domain method to transform the secret image into the frequency domain, and then utilized a sequence of random numbers to record the lower frequency coefficients (the AC values) except the DC value. The DC value of each block is regarded as the secret key and is shared among the *n* participants by applying the (*r*, *n*) threshold scheme. Some other secret image sharing methods can also be found in [98-102].

## 1.3 Overview of the Dissertation

In the dissertation, several techniques to protect digital images during a transmission through the Internet are proposed for different kinds of applications. The proposed methods contain three hiding techniques for transmission of a confidential image, and two image authentication methods with self or cross-recovery ability abilities. The framework of the dissertation is depicted in Fig. 1.2, and a brief overview of each of the proposed methods is given in the following subsections (A-D).

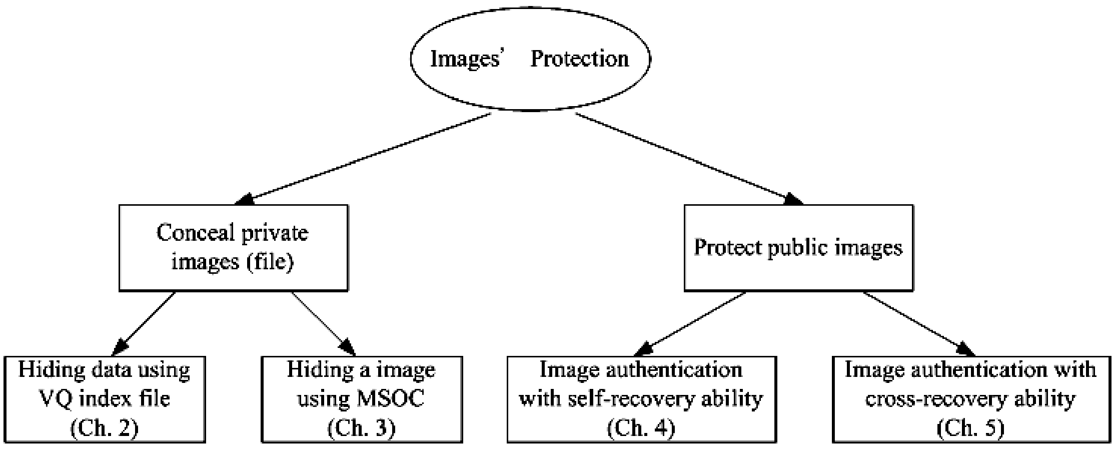

Images' Protection
Conceal private images (file)
Protect public images
Hiding data using VQ index file (Ch. 2)
Hiding a image using MSOC (Ch. 3)
Image authentication with self-recovery ability (Ch. 4)
Image authentication with cross-recovery ability (Ch. 5)

Fig. 1.2. The framework of the dissertation

## A. Hiding Data Using VQ Index File

People usually compress digital images before the transmission through Internet, for the compression result has smaller size and hence, can reduce the waiting time of the receiver end. Vector quantization (VQ) [15] is an easy technique for compressing, and its decoding is very fast. In Chapter 2, we propose a method to hide secret data in the VQ index file of a cover image. In the proposed method, the secret data are embedded in the compression result of a cover image by employing the search-order coding (SOC) algorithm [13]. In the embedding, there is no further distortion to the compression codes of the cover image, so the receiver can recover both the hidden data and the VQ-version of the cover image. The major goal of Chapter 2 is to improve the compression rate and hiding capacity of Chang et al.'s method [17]. Both our hiding method and Chang et al.'s hiding method [17] use the search-order coding (SOC) [13] to error-freely compress VQ indices.

However, our cover-image compression rate (the b.p.p., i.e. the bit-per-pixel) is better. In fact, even if our method hides a secret data set which is twice as big as their data set, our cover-image compression rate can still be competitive to Chang et al.'s method [17, 104].

## B. Hiding Images Using Modified Search-Order Coding and Modulus Function

In Chapter 3, a steganographic method based on the modified search-order coding (MSOC) and a variance-based modulus function is proposed. The MSOC scheme utilizes the feature of high correlation among adjacency pixels (i.e. many neighboring pixels are with similar gray-values) to encode the important image. An adjustable threshold *T* is used in the MSOC; and this *T* directly controls the quality of the extracted image. Finally, to embed the MSOC output code in a cover image, at each pixel of the cover image, its {Northwest, North, West} three-neighbor variance is evaluated to estimate the hiding capacity of the pixel, and the MSOC output code is embedded in the cover image using two sets of modulus function. Notably, in order to have the ability of extracting the important image in the future, the evaluation of the aforementioned variance uses the stego-pixel-values (rather than the original cover-pixel-values) of the three neighbors. This is because the {Northwest, North, West} three-neighboring-pixels are already modified to hide data in them, and we do not keep the cover image after embedding; therefore, in the future decoding-phase, the evaluation of the variance can only use stego-image neighbors, not the original cover-image neighbors.

## C. An Sharing-Based Authentication and Self-Recovery Method against Image Tampering

In Chapter 4, we propose a competitive image authentication and tampered-area recovery method. The authentication data for each block are calculated using a cryptographic hash function, in which the input includes the MSBs information within the block, the user's secret key, the block address information, and some private information about the image. The recovery data are VQ-index value of the block, which can be generated by any vector quantization (VQ) technique (e.g. Ref. [15]). To increase the recovery ability, the VQ-index value is shared by Thien and Lin's secret image sharing method [90], which is reviewed briefly in the next section. Finally, the recovery information is combined with the authentication data to form the watermark. To improve security, when a part of the watermark is embedded into a block's LSBs, the embedding locations of the bits are arranged according to a pseudo-random permutation, based on the Mersenne Twister (MT) pseudo-random number generator [9]. Experimental results show that the quality of the watermarked image is acceptable, and that the positions of the tampered parts are located accurately. The recovery of large-area corruption is also good.

## D. Authentication and Cross-Recovery for Multiple Images

In Chapter 5, we design an image authentication scheme that deals with a group of images simultaneously instead of a single image, and the recovery of any member images in this group can be done through the mutual support of $r$ of the $n-1$ remaining member images, as long as these $r$ images pass the authentication tests. More specifically, give a group of images, which the number of images is $n$, and $r$ is a pre-specified threshold value not less than $n/2$. In the first place, the recovery data for each image of the group is obtained by the compression techniques. Then, to achieve the cross-recovery purpose, the recovery data are shared by the proposed two-layer sharing method, which is derived from secret image sharing method [90], to generate the final (embedding) shadows. Next, these shadows are embedded into the $n$ member images of the group using the simple LSB substitution scheme and obtain $n$ stego-images, which still look like their original versions. For each stego-image, we utilize some private information about it to generate the authentication data (watermark) for verification purpose by using a cryptographic hash function. Finally, the watermark is embedded into the stego-image to form the watermarked image. The experimental results show that the quality of each watermarked image is acceptable, and even if ($n-r$) of the watermarked images get lost or damaged, we could still recover the corrupt watermarked images using $r$ of the authentic watermarked images. Moreover, the visual quality of the recovery images is good to maintain their usage.

## 1.4 Dissertation Organization

In the remainder of this dissertation, the two hiding methods to conceal secret data or a confidential image in a cover image are proposed in Chapter 2 and 3, respectively. The proposed image authentication methods with self-recovery and cross-recovery ability are described in Chapter 4 and 5, respectively. Finally, the conclusions and future works are in Chapter 6.

# Chapter 2
# Hiding Data Using VQ Index File

In this chapter, we propose a method for hiding secret data in the vector quantization (VQ) index file of a cover image. In the embedding procedure, a search-order coding (SOC) technique is employed to hide the secret data and compress the index file simultaneously. The recovered cover image is identical to the one recovered by traditional VQ; and the compression rate is competitive to traditional VQ's (often better than traditional VQ's if we use *e* = 1 mode); while the hidden binary data is revealed in the decompression procedure as a bonus. The compression rate and hiding capacity are also better than those in Chang et al.'s method [17].

The rest of this chapter is organized as follows. Section 2.1 takes a brief review of the SOC algorithm. The details of the proposed method are described in Section 2.2. Experimental results are shown in Section 2.3, and a brief summary is made finally in Section 2.4.

## 2.1 Review of the Search-Ordering Coding (SOC) Tool

Vector quantization (VQ) [15] is a simple technique to compress images. According to a given codebook, an index file is generated as the compression result for each given image. To reduce the size of the index file further, Hsieh and Tsai [13] proposed the use of the search-order coding (SOC). The SOC algorithm in Ref. [13] encodes traditional VQ indices with fewer bits by utilizing the fact that there exists high correlation among adjacent indices, i.e. there exist many blocks whose VQ indices also appear in their neighborhood blocks. Recall that in traditional VQ, each block of the given image is represented by an index. So, the whole image is represented by an index file in which the number of indices (counting repetition) equals the number of blocks. The SOC algorithm encodes the index file of an image in an index by index manner (or equivalently, block by block). All of the indices that appear in a predefined search path, which just cover a small area of the neighborhood blocks, are called search points (*SP*), and the non-search points are just those indices whose block location are beyond the range covered by the predefined search path. To encode the current not-yet-processed index, people begin a search along a predefined path. The search is in order to find whether a nearby block also has the same VQ index used by the current block. If a match is found in nearby area, then the original index value (OIV) of the current block is replaced by a search-order code (SOC) that indicates the position of the matched block in the search path; the SOC uses fewer bits than the OIV. On the other hand, if no match can be found in nearby area, then Ref. [13] still uses the OIV of the current block. In general, a SOC is defined as an order in which the indices of the already-processed neighborhood blocks are compared with the index of the current block. Of course, to let the decoder distinguish between the SOC and OIV, an extra indicator bit (the flag) is added in front of the resulting compression code for each index.

An example of the SOC is shown in Fig. 2.1. The current block is at location (3, 3), and the index value for the current block is 76. A predefined search path is shown with arrows. In this example, assume the starting search point is the neighboring block (3, 2), and a search-order code of *N*=2 bits is used, where *N* is the number of bits used to record the order of the matched position in the SOC searching. Since $2^N=2^2=4$, there are at most four *SP*s are used for comparison excluding the repetition points. The path 75-75-74-74 is the level 1 search. Since there is no value that matches 76, we begin the level 2 search 75-76-… in the outer loop; and stop at the first matched value 76 located at (2, 1). Since the position of block (2, 1) is “10” (the 3rd kind of value) in the search code, the block (3, 3) is encoded as the search-order code “10”. Of course, a flag bit is also needed. Therefore, block (3, 3) is coded using $1+N=1+2=3$ bits, which is more economic than, for example, the 8-bits index needed in traditional VQ if there are $256=2^8$ distinct code-blocks in the codebook. (In traditional VQ, each block of the given image is represented by a code-block found in this VQ codebook, and only the code-block’s index [rather than the code-block itself] is transmitted. So, 8-bits for an index if there are $256=2^8$ distinct code-blocks.)

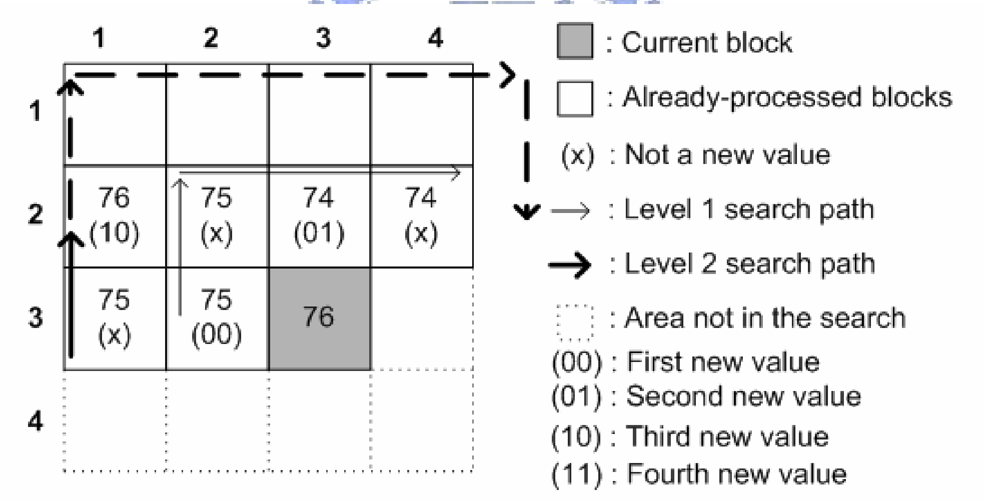


Fig. 2.1. An example of the SOC algorithm.

## 2.2 The Proposed Method

In this section, we introduce our image hiding method. Each index in the index file is encoded either by search-order codes (SOC) or original index values (OIV). The experiment results of Ref. [13] suggested that $N = 2$, so we also use $N = 2$ bits to record the match position in the search area. From Table 2.1, we can see that the selection of the *SSP* is not critical. Therefore, we utilize *SSP* to hide the secret data without significantly increasing the total size of index file. Each VQ index value can hide $e$ bits of the secret data; and $e$ is set to 1 (or 2). Without the loss of generality, assume $e = 1$. Then each time, for the current index, we take $e = 1$ bit of the secret data, then use Fig. 2.2(a) to determine the *SSP* of the search. For example, use *SSP* = West when the secret data bit is 0; and use *SSP* = North when the secret data bit is 1. After that, we begin the search starting from that *SSP*. If we can find a match in a nearly area of blocks that has at most $2^N = 2^2 = 4$ distinct index values, then we record the matched position using $N = 2$ more bits. If no such match can be found in nearly area, then we record the OIV of the current index.

Notably, to enhance the scheme's security, the secret data are encrypted first by some traditional cryptography methods such as DES [1] before the embedding process, and then treated as a binary string. Only the people who own the encryption key can decrypt the secret data correctly. The details of our hiding algorithm are as follows:

**Input:** (1) The VQ index file of a cover image;

(2) A secret data set which is encrypted by DES [1] and then transformed into a binary string.

**Parameter settings:** Let $N = 2$ be the number of bits needed to record the matched position in the SOC searching. Also set $e = 1$ (or $e = 2$) where $e$ is the number of bits to be embedded in each index.

**Output:** The cover image's compression code, which also contains the secret data.

**Step 1:** Take the next not-yet-processed index value $I$ from the index file.

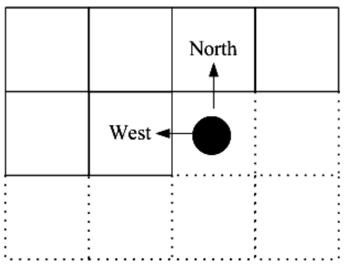
**Step 2:** Take next $e$ bits from the secret binary string, and use these $e$ bits as the *SSP* value. In the predefined search path starting from *SSP*, try to find a block whose index value matches the index value $I$. There are two possible cases.

Either a matched one is found; or, no match in the searching area covered by the search codes {00, 01, 10, 11}, i.e. in the area in which at most four distinct index values appear.

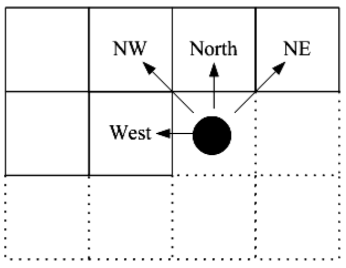
**Step 3:** If a match is found, the current block's index will be encoded with a 1-bit indicator, followed by an *e*-bits *SSP* record (which equals to the *e* bits of secret data); then followed by the corresponding code indicating the matched position. If no match is found, then encode the current block with a 1-bit indicator, followed by the *e* bits of secret data; then followed by the block's original index value. In a word, the SOC-case is coded using (1+*e* +*N*) bits, and the OIV-case is coded using (1+*e* +8) bits (if the codebook size is 256).

**Step 4:** Go to step 1.

In Step 3 above, for example, as shown in Fig. 2.2, the block (3, 3) is encoded with the compression code "($\mathbf{0}\underline{0}10)_2$". The bold-faced bit **0** is the indicator to indicate SOC-case, and the underlined bit $\underline{0}$ is the secret data bit (assume *e* = 1). The index value can be found using a previous block at position 10. On the other hand, if no match of the block (3, 3) is found, then it is encoded with the OIV compression code "($\mathbf{1}\underline{0}01001100)_2$" where $(01001100)_2 = (76)_{10}$ is the OIV. The decoder will see the bold-faced **1** and underlined $\underline{0}$ as the secret and 01001100 as the index value.



(a)



(b)

Fig. 2.2. Starting Search Points (*SSP*). (a) The case of using *e* = 1 bit to record *SSP* (West vs. North); (b) the case of using *e* = 2 bits.

## 2.3 Experimental Results

In the first experiment, each of the five *gray-value* cover images {Jet, Lena, Peppers, Boat, Toys} has 256×256 pixels before VQ compression. The block size is 4×4, and the codebook size is 256. So, for each image, each block of 4×4 = 16 gray-value pixels are replaced by an index value in the range 0-255. Each image is thus replaced by an index file consisting of (256×256)/(4×4) = 64×64 index values. Therefore, the bit rate of the traditional VQ is 8/16 = 0.5 bit per pixel (bpp). Notably, the bit rate is 8 bpp before VQ compression.

After VQ compression, for each of the five images, we try to use SOC technique to re-code each index value, and also hide a binary secret data in the image's index file. The bpp value 0.5 is thus changed. In Chang et al.'s scheme [17], each index value hid only one bit of the secret data. Since there are only 64×64 index values, the hidden secret data using Ref. [17] has 64×64 bits. However, with our method, each index value can hide either $e$ = 1 or $e$ = 2 bits of the secret data. Therefore, in Table 2.2, the 64×64 secret binary image Lena shown in Fig. 2.3(a) is used as the secret data for Chang et al.'s scheme, and also for the 1-bit mode of our approach (only $e$ = 1 bit of the secret data is hidden in each index). On the other hand, the 64×128 binary secret image "Lena and Mark" shown in Fig. 2.3(b) is used as the secret data for the 2-bit mode of our approach. In the second experiment (Table 2.3), both the binary secret image and the cover images are 2×2 = 4 times larger.

Tables 2.2 and 2.3 indicate that the bpp compression rate of our 1-bit mode is better than Chang et al.'s. Also, although the bpp compression rate of our 2-bit mode is similar to Chang et al.'s, our hiding capacity is twice as big as theirs. Notably, no matter in Ref. [17] or in the proposed method, when there are more cover image blocks encoded by the SOC format rather than OIV format, then the bpp compression rate is better. Smooth images, for instance, the images “Toys” and “House”, seem to have this tendency.

Figure 2.4 shows the compression rate of some published methods and our scheme. It indicates that the compression rate of our 1-bit mode is better than that of all other listed methods, and the compression rate of our 2-bit mode is competitive to theirs. Figure 2.5 shows the PSNR value of the reconstructed images of the published methods and our scheme. From Fig. 2.5, we can see that Chang et al.’s methods [17, 104] and our scheme are reversible information hiding methods for VQ compressed images (i.e. the reconstructed image is identical to the one decompressed by traditional VQ). Figure 2.6 shows the embedding rate of the published methods and our scheme. The embedding rate (bit per index) indicates how many secret bits can be embedded into a cover image. The embedding rate is defined by

$$\mathrm{ER} = \frac{\|S\|}{N_{\mathrm{IDX}}}\,(\mathrm{bpi}), \tag{2.1}$$

where $\|S\|$ represents the total number of secret bits embedded into the cover image and $N_{\mathrm{IDX}}$ represents the total number of indices in the index file of the VQ-compressed image. From Fig. 2.6, we can see that the embedding rate of our 2-bit mode is double that of all other listed methods.

As for the processing time, the average encoding and decoding time of our method are shown in Tables 2.4. Note that, all programs in the current dissertation were implemented by using the Borland C++ Builder 6.0, and ran on a notebook with Intel Pentium Processor M-740 (1.73 GHz) and 512MB RAM under the operation system of Microsoft Windows XP Professional.

## 2.4 Summary

In this chapter, a modified data hiding scheme is proposed to embed the secret data into VQ index file of an image. Both ours and Chang et al.'s [17] use search-order to reduce code length. But our approach has better bpp compression rate than [17] if hiding capacity is the same. Also, ours has better hiding capacity when the bpp compression rates are similar. Although some secret data are hidden in indices, the decompressed cover image is identical to the one decompressed by traditional VQ; moreover, the compression rate is often better than VQ's if we use *e* = 1 mode (see Tables 2.2-2.3 and Figs. 2.4-2.6).

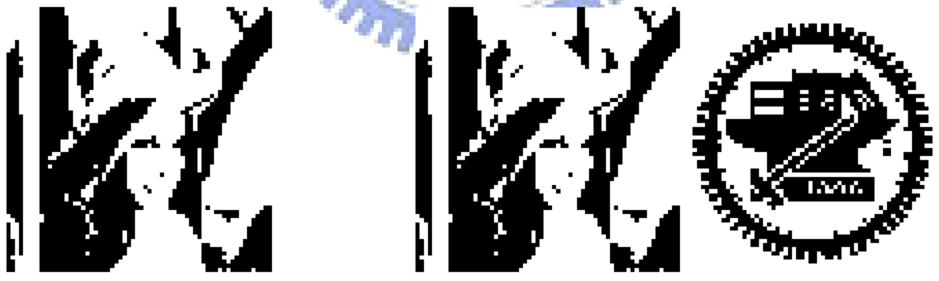


Fig. 2.3. Secret "binary" images: (a) "Lena" of 64×64 bits, (b) "Lena and Mark" of 64×128 bits.

Table 2.1. The b.p.p. compression rate of selecting different SSP's for 256×256 images in the search-order coding.

| SSP | Jet | Lena | Peppers | Boat | House |
|---|---|---|---|---|---|
| West | 0.335 | 0.379 | 0.386 | 0.390 | 0.281 |
| NW | 0.337 | 0.380 | 0.389 | 0.393 | 0.282 |
| North | 0.336 | 0.378 | 0.388 | 0.391 | 0.280 |
| NE | 0.337 | 0.379 | 0.387 | 0.390 | 0.282 |

Table 2.2. The b.p.p. compression rate (i.e. how many bits to represent a gray-value pixel of the compressed images). Fig. 2.3(a) is not only hidden by Chang et al.'s ("*" means "quoted from [17]"), but also hidden by ours[1] (using *e* = 1 mode). Fig. 2.3(b) is only hidden by ours[2] (using *e* = 2 mode). Notably, traditional VQ would be of 0.5 bpp and with no hiding feature.

| Methods | Cover images (256×256) quantized by VQ | | | | |
|---|---|---|---|---|---|
| | Jet | Lena | Peppers | Boat | Toys |
| Chang et al.'s [17] | 0.4962* | 0.5418* | 0.5196* | 0.5025* | 0.4789* |
| Ours[1]. (Hide 1 bit per VQ index) | 0.4312 | 0.4617 | 0.4700 | 0.4886 | 0.3370 |
| Ours[2]. (Hide 2 bit per VQ index) | 0.4937 | 0.5242 | 0.5325 | 0.5455 | 0.3995 |

Table 2.3. Analogous to Table 2.2, but with larger hidden secret images (a 128×128 binary image "Lena" for [17] and for ours[1]; while a 128×*256* binary image "Lena and Mark" for ours[2]). The cover images are also larger now (512×512).

| Methods | Cover images (512×512) quantized by VQ | | | | |
|---|---|---|---|---|---|
| | Jet | Lena | Peppers | Boat | House |
| Chang et al.'s [17] | 0.4797 | 0.4900 | 0.4824 | 0.5166 | 0.4526 |
| Ours[1]. (Hide 1 bit per VQ index) | 0.4016 | 0.4199 | 0.4108 | 0.4608 | 0.3626 |
| Ours[2]. (Hide 2 bit per VQ index) | 0.4641 | 0.4824 | 0.4733 | 0.5233 | 0.4251 |

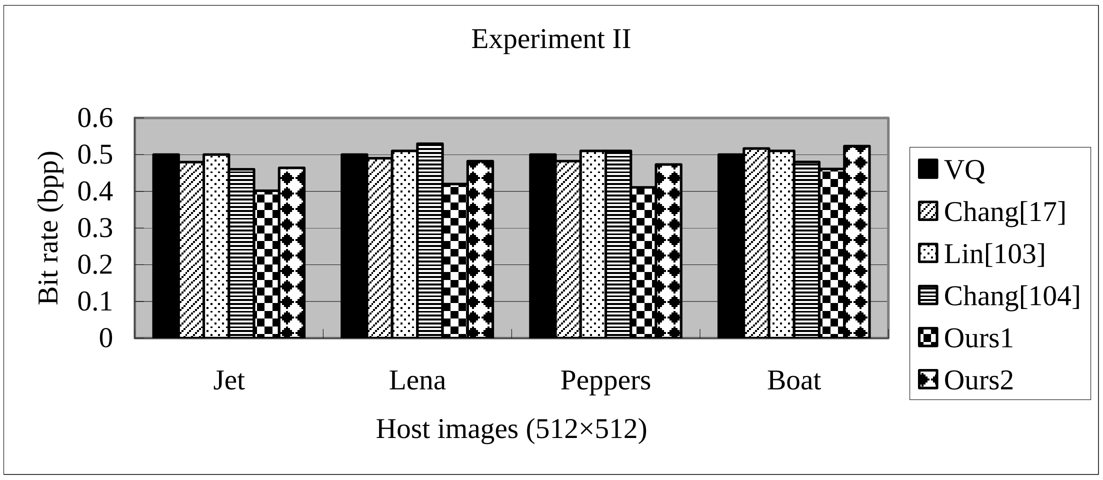


Fig. 2.4. The compression rate results of some published methods and our scheme.

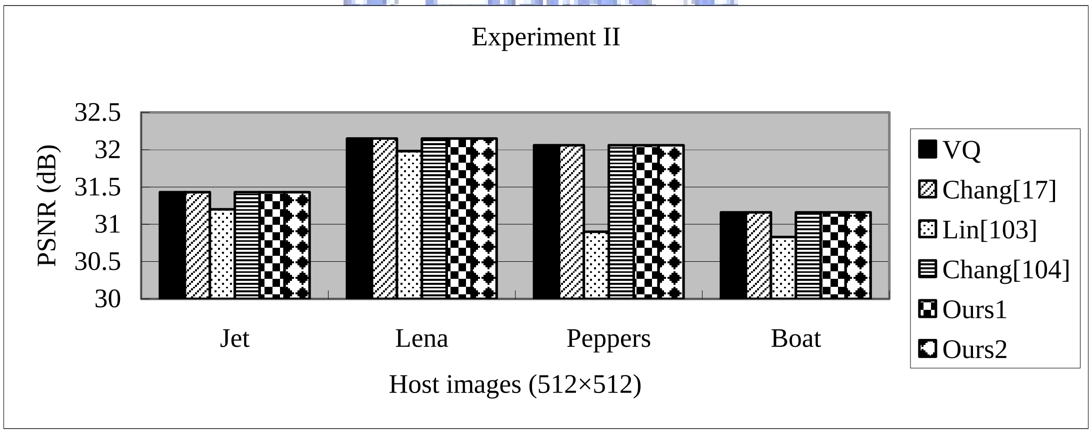


Fig. 2.5. The visual quality of the reconstructed images of some published methods and our scheme.

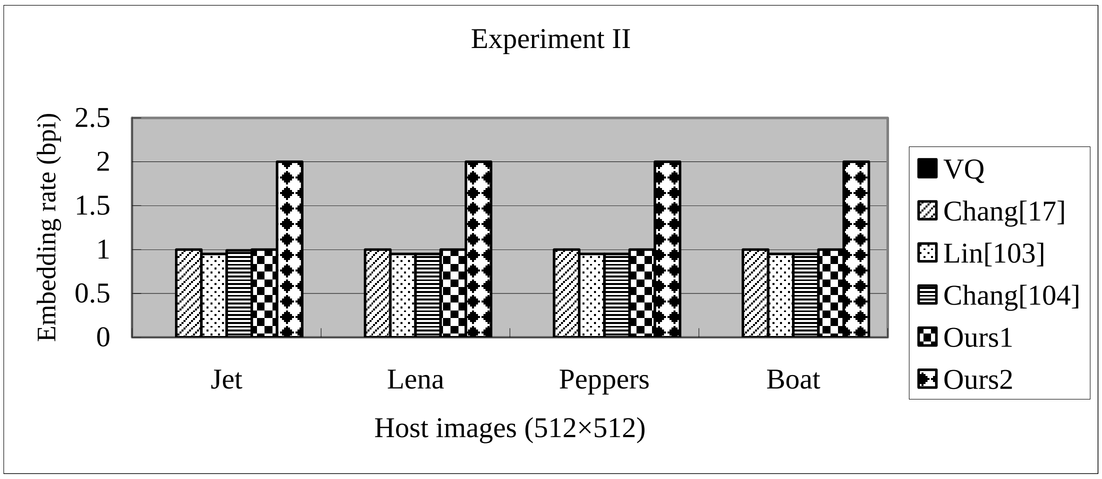


Fig. 2.6. The embedding rate results of some published methods and our scheme.

Table 2.4. The processing time of our method. (unit: second)

| Size of cover image | Encoding | Decoding |
|---|---|---|
| 256×256 | 0.643 | 0.216 |
| 512×512 | 1.311 | 0.412 |

# Chapter 3
# Hiding Images Using Modified Search-Order Coding and Modulus Function

In this chapter, we propose a method to hide an important image in a cover image whose size is economic. In the method, in order to save space, a modified search-order coding (MSOC) technique first transforms the important image. Then, a randomization procedure permutes the transformed image to increase further the security. Finally, a modulus function embeds the permuted code in a cover image; notably, in the modules function, the modulus base used for a pixel is determined according to the variance of its neighboring pixels.

The rest of this chapter is organized as follows. Section 3.1 takes a brief review of the modulus embedding function. The details of the proposed method are described in Section 3.2. Experimental results are shown in Section 3.3. The discussions are in Section 3.4, and the summary is in Section 3.5.

## 3.1 Review of the Modulus Embedding Function

In this section, the embedding methods using modulus function [31-32] are briefly reviewed to provide some necessary background knowledge.

In 2003, Thien and Lin [31] applied a modulus function to embed data in still images. Their scheme can hide data efficiently and the base value is not necessarily in {2, 4, 8, 16, …}. Their scheme is simple and fast, and their experimental results show that the qualities of the obtained stego-images are much better than that of the simple LSB substitution method. In 2005, Wang [32] observed that the quality of the stego-image in Ref. [31] may degrade too much if a large number of bits are embedded in a pixel of the cover image whose pixel value is small. Therefore, Wang extended further the modulus embedding method to a more flexible extent. In his method, the pixels of the cover image are classified into two groups: one group is $G_U$ which contains the pixels whose values are greater than a pre-determined $V$, and the other group $G_L$ which contains the pixels whose value are at most $V$. Then, a modulus function with a large modulus base $M_U$ is applied to embed data in $G_U$, and another modulus function with a small modulus base $M_L$ is preformed to embed data in $G_L$. This indicates that more data bits are embedded in the pixels of $G_U$, and fewer data bits are embedded in the pixels of $G_L$.

The modulus embedding procedure [32] are reviewed below. We first start with a system using a single modulus base. Let the integer parameter $M_O$ denote a modulus base. To embed a numerical data $x$ ($0 \le x < M_O$) in a pixel value $y_{ij}$ ($0 \le y_{ij} \le 255$) at position ($i$, $j$) of the cover image, we show below how to construct the new value $\hat{y}_{ij}$ which can be utilized to extract $x$. First, calculate the plain difference value

$$d_{ij} = x - (y_{ij} \bmod M_O). \tag{3.1}$$

Then, evaluate the modified difference value $d'_{ij}$ by the rule

$$d'_{ij} = \begin{cases} d_{ij} & \text{if } \left(-\left\lfloor \frac{M_O - 1}{2} \right\rfloor\right) \le d_{ij} \le \left\lceil \frac{M_O - 1}{2} \right\rceil; \\ d_{ij} + M_O & \text{if } (-M_O + 1) \le d_{ij} < \left(-\left\lfloor \frac{M_O - 1}{2} \right\rfloor\right); \\ d_{ij} - M_O & \text{if } \left\lceil \frac{M_O - 1}{2} \right\rceil < d_{ij} < M_O. \end{cases} \tag{3.2}$$

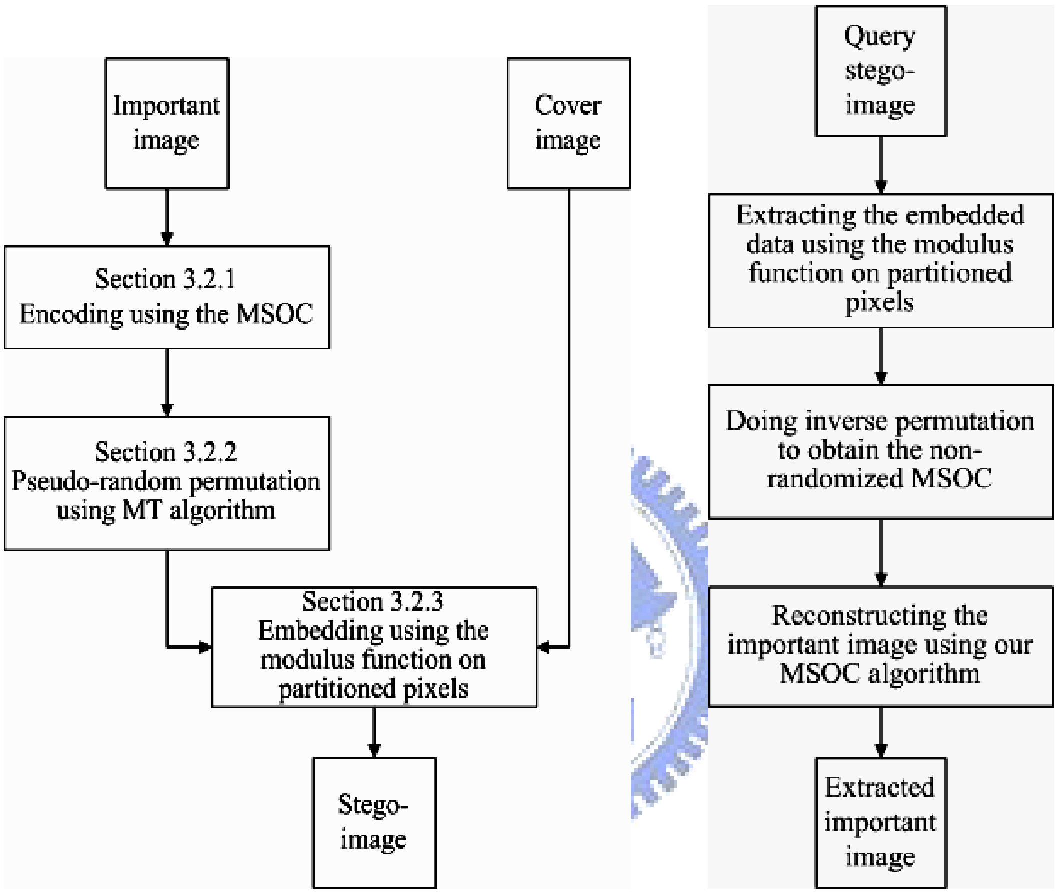
Then, replace the old pixel value $y_{ij}$ by the new value

$$\hat{y}_{ij} = d'_{ij} + y_{ij}. \tag{3.3}$$

Of course, a boundary checking procedure is needed to ensure that the gray value $\hat{y}_{ij}$ falls in a valid range between $V_b$ and $V_t$. Do the following adjustment if necessary:

$$\hat{y}_{ij} = \begin{cases} \hat{y}_{ij} + M_O & \text{if } \hat{y}_{ij} < V_b; \\ \hat{y}_{ij} - M_O & \text{if } \hat{y}_{ij} > V_t. \end{cases} \tag{3.4}$$

Note that in this method of Ref. [32], the pixel values $y_{ij}$ of the cover image are classified into two groups $G_U$ and $G_L$, and each group has its own modulus bases values: $M_U$ for $G_U$, and $M_L$ for $G_L$. If a pixel value $y_{ij}$ of the cover image belongs to $G_L$, then the value of $V_b$ is set to 0 and the value of $V_t$ is set to $V$. On the other hand, if $y_{ij}$ belongs to $G_U$, then the value of $V_b$ is $V$ and the value of $V_t$ is 255.


Important image
Cover image
Section 3.2.1 Encoding using the MSOC
Section 3.2.2 Pseudo-random permutation using MT algorithm
Section 3.2.3 Embedding using the modulus function on partitioned pixels
Stego-image
Query stego-image
Extracting the embedded data using the modulus function on partitioned pixels
Doing inverse permutation to obtain the non-randomized MSOC
Reconstructing the important image using our MSOC algorithm
Extracted important image

(a) (b)

Fig. 3.1. Two flowcharts showing the proposed method: (a) the encoding procedure; (b) the decoding procedure.

## 3.2 The Proposed Method

In this section, we introduce our image hiding method. The flowchart of the proposed method is depicted in Fig. 3.1. First, the MSOC is used to encode the important image, and the output code is then permuted using a pseudo-random method. Finally, the randomized code is embedded in the cover image by using the modulus embedding function. Therefore, there are three major parts in our scheme: (a) the MSOC scheme, (b) the pseudo-random permutation process, and (c) the modulus embedding process. The details of these three parts are described in Sec. 3.2.1–3.2.3, respectively. The extraction procedure to unveil the embedded important image from stego-image is presented in Sec. 3.2.4.

### 3.2.1 The Modified Search-Order Coding (MSOC)

Instead of processing the vector-quantization (VQ) indices of an image as the traditional SOC scheme [13] did (see Sec. 2.1), our MSOC scheme processes the pixel data directly. Also, due to the fact that adjacent pixels of an image often have similar gray values, we modify the match condition used in traditional SOC method [13], and use a pre-determined threshold $T$ to encode the important image. The purpose of these modifications is that, when compress a set of numbers, the MSOC can yield more compact code than SOC does, and hence produce a better-quality stego-image in the embedding stage later. The notation and details of MSOC are stated below:

**Notation**

$N$: the length of the code for MSOC position, i.e. the number of bits used to record a matched position in the searching path.

$T$: a threshold indicating the tolerance level in the MSOC matching equations

(3.5)-(3.6). (Set $T$=1 will make the extracted important image free of error.)

$SP$: the pixels that appear in the predefined search path.

$P_c$: the currently-processed pixel (of the important image).

$S_c$: the set of candidate $SP$s for $P_c$ (see Eq. (3.5)), in which the gray values of all candidates must not differ too much from the gray value of $P_c$.

$V_{cut}$: the threshold to decide which modulus base is to be used in the embedding phase (Sec. 3.2.3).

**The MSOC encoding algorithm**

**Input:** The important image, and two positive integers $N$ and $T$. (Set $T$ to 1 if the extracted important image is required to be error-free.)

**Output:** The MSOC code of the important image.

**Step 1:** According to raster scan order, take the next not-yet-processed "pixel" $P_c$ from the important image. Note that $P_c$ is a pixel in MSOC algorithm, rather than a block.

**Step 2:** Without the loss of generality, assume that the left adjacent pixel of $P_c$ is the starting search point. Generate $2^N$ $SP$s from a predefined search path. Note that a pixel whose value equals to a value already appear in some previous $SP$s (of $P_c$) of the current search path shall be skipped, and no $SP$ value will be assigned to a skipped pixel. From the set of $SP$s, we will be only interested in the $SP$s whose values are similar to the value of $P_c$ (up to a threshold $T$), i.e.

$$S_c = \{SP_j : |\,\text{value}(SP_j) - \text{value}(P_c)\,| < T, 0 \le j \le 2^N - 1\}\,. \quad (3.5)$$

**Step 3:** If $S_c$ is not empty, then the original pixel value of $P_c$ is replaced by MSOC($P_c$), which is defined as the element in $S_c$ whose value is most similar to the value of $P_c$, i.e.

$$MSOC(P_c) = \arg_j \min_{SP_j \in S_c} \{|\,\mathrm{value}(SP_j) - \mathrm{value}(P_c)\,|\}, \qquad (3.6)$$

where the arg(·) operator returns the position-code (a $N$-bits binary string since the predefined search path in Step 2 has $2^N$ $SP$s) of the point $SP_j*$ in $S_c$ satisfying $|\,\mathrm{value}(SP_j*) - \mathrm{value}(P_c)\,| = \min_{SP_j \in S_c} \{|\,\mathrm{value}(SP_j) - \mathrm{value}(P_c)\,|\}$. However, if $S_c$ is empty, then the original pixel value (OPV) of $P_c$ is used as the output. As in traditional SOC scheme, we also add an extra indication bit (the flag) in order to distinguish between an MSOC and an OPV.

**Step 4:** Repeat Steps 1 to 3 until all pixels of the important image is processed.

Below we use Fig. 2.1 of Chapter 2 again to explain MSOC. Note that every cell in Fig. 2.1 should now be explained as a pixel, rather than an image block or a VQ-index. Assume the threshold $T$ is set to 2, and the value of the pixel (1, 1) is 77. Let $P_c$ = (3, 3) be the current pixel. By Step 2 of the MSOC algorithm mentioned above, the gray values in the candidate set $S_c$ of pixel (3, 3) is {75 (00), 76 (10)}. Since the value 76 is the closest (in fact, identical) to the value of the current pixel (3, 3), the value of (3, 3) will be encoded using the code "($\underline{\mathbf{0}}10)_2$" by the Step 3 above. The underlined bit $\underline{\mathbf{0}}$ is the indicator bit, and the following two bits "10" indicates the pixel value can be found using a previous pixel at position $(10)_2$ in the search-path.

In the other example, assume that the values of the pixels (2, 2), (3, 1), and (3, 2) are all 78 rather than 75, and the value of the pixel (2, 1) is 79 rather than 76. Then, since none of the values {78, 79, 74} is so close to value($P_c$)=76 that the difference is less than the threshold $T$=2; Step 2 of the MSOC algorithm implies that the candidate set $S_c$ of the pixel (3, 3) is an empty set. In other words, no tolerable match of the pixel (3, 3) can be found. Therefore, the pixel (3, 3) is encoded as "($\underline{\mathbf{1}}01001100)_2$"

where **1** is the flag bit, and $(01001100)_2 = (76)_{10}$ is the original pixel value (OPV) of the pixel (3, 3).

### 3.2.2 The Pseudo-Random Permutation of Location

Before embedding the produced MSOC code in cover image, to increase the level of security, we will permute the MSOC code by using a pseudo-random process. Ref. [20] had adopted a mono-alphabetic substitution cipher algorithm [10] to randomize their important image before doing their LSB-related embedding process. However, we do not intend to use the mono-alphabetic substitution cipher algorithm in the current approach, for the reason stated below. Fig. 3.2(b) shows the result of applying the mono-alphabetic substitution cipher algorithm to the image shown in Fig. 3.2(a), and it is obvious that there are still some regular patterns in Fig. 3.2(b). In other words, the result is not quite random. Therefore, we design here another pseudo-random permutation method based on the Mersenne Twister (MT) pseudo-random number generator [9].

Assume that the output code of the MSOC is treated as a binary string, and the bit locations in the binary string are numbered sequentially from 0 to $S-1$, where $S$ is the size of the binary string. Our pseudo-random permutation is to transform each bit location $i$ of the binary string to a new location $f(i)$ .

**The pseudo-random permutation algorithm**

**Input:** A binary stream which is the MSOC output code of an important image.

**Parameter settings:** Randomly choose a 32-bit integer greater than 0. This is the so-called "seed" for the MT pseudo-random number generator. With this seed, a series of pseudo-random real numbers $R_j$ ($j = 0,1,2,\ldots,2^{19937}-1$), which are uniformly distributed on [0, 1]-interval, can be generated by the Mersenne Twister (MT) pseudo-random number generator [9].

**Output:** The randomized form of the MSOC output code.

**Step 0:** Initially, set $i = -1$.

**Step 1:** Increase $i$ by 1. The $i^{th}$ bit of the MSOC binary string will be permuted to a new location *f(i),* as computed by Steps 2-4 below.

**Step 2:** Get next not-yet-taken fraction-number $R_j$ from the MT series.

**Step 3:** The new location $f(i)$ is computed by

$$f(i) = \text{int}(R_j \times (S-1)), \tag{3.7}$$

where the $\text{int}(\cdot)$ operator means getting the integer part of a real number.

**Step 4:** If the new location $f(i)$ is a repetition, return to Step 2 to get the next $R_j$.

**Step 5:** Repeat Steps 1 to 4 until the entire binary strings are processed ($i = S-1$).

By the above pseudo-random permutation algorithm, the output code of the MSOC procedure mentioned in Sec. 3.2.1 is permuted into a pseudo-random code to enhance the security property of our scheme. The seed used in the MT algorithm can be regarded as a secret key; and only the authorized extractor who owns the same secret key can obtain the same sequence of pseudo-random real numbers to recovery the MSOC code in the decoding phase.

Of course, the above pseudo-random permutation algorithm can also be used to permute pixels of any image. Just replace the term “bit location” by “pixel location”, and treat the whole input image as a sequence of $S$ pixels. When we use this “pixel” version to randomize the Pepper image in Fig. 3.2(a), the result is shown in Fig. 3.2(c). Compare Figs. 3.2(b) and 3.2(c), it is quite clear that our pseudo-random permutation algorithm has better randomness. In this example, the seed of the MT algorithm is set to 4357, and $S$=512×512=262144, because Fig. 3.2(a) has 512×512 pixels. When the seed is 4357, the MT generator creates a series of fraction numbers whose first five numbers $R_0 - R_4$ are {0.817330, 0.999061, 0.510354, 0.131533, 0.035416}. Hence, by Step 3 of the above pseudo-random permutation algorithm, the new locations $f(0) - f(4)$ are {214257, 261896, 133785, 34480, 9284}. It means that the location 0 is transformed to the new location 214257, location 1 is transformed to the new location 261896, and so forth.

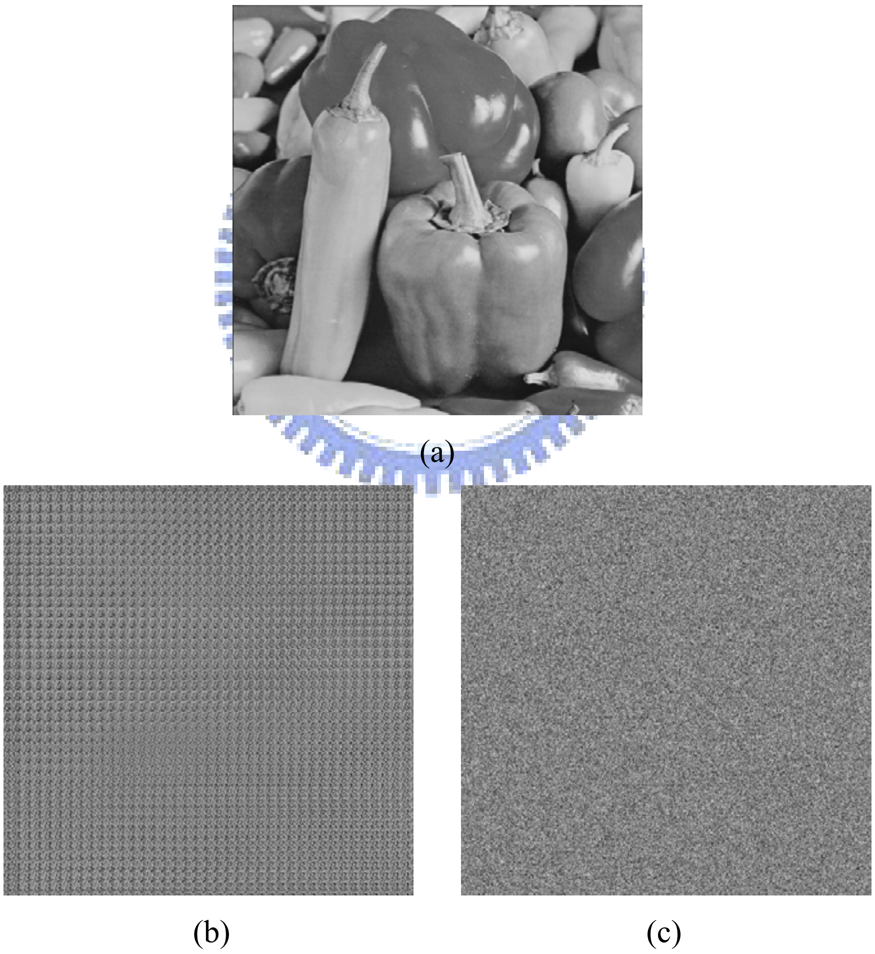


Fig. 3.2. The results of randomize the image "Pepper" shown in (a). Here, (b) the result of using the mono-alphabetic substitution cipher algorithm [10], and (c) the result of using our process described in Sec. 3.2.2.

### 3.2.3 The Modulus Embedding Phase on Partitioned Pixels

After the aforementioned pseudo-random permutation phase, the randomized code, which is a binary string, is ready to be embedded in a cover image. To obtain high hiding-capacity and yet still keep good image quality of the stego-image, we need to estimate the hiding capacity pixel-by-pixel in the cover image. We will use a phenomenon found in human visual system (HVS), namely, hiding more data in the area where the gray values change much.

As defined in Section 3.1, the symbol $y_{ij}$ denotes the gray value of the pixel at position ($i$, $j$) in the cover image. Apparently, $0 \le y_{ij} \le 255$, and the cover image is $\{y_{ij}\text{: } 0 \le i \le h-1, \text{and } 0 \le j \le w-1\}$ if the size of the cover image is $h \times w$. Analogously, let $\hat{y}_{ij}$ denote the gray value of the pixel at position ($i$, $j$) in the stego-image. Then, for each position ($i$, $j$), to create pixel ($i$, $j$) of the stego-image, the variance $var_{ij}$ is defined according to the values of the three already-obtained adjacent "stego" pixels ($i$, $j-1$), ($i-1$, $j-1$), and ($i-1$, $j$), which lead pixel ($i$, $j$) when the image is transformed from cover to stego. Without the loss of generality, we may just define

$$var_{ij} = \begin{cases} \infty & \text{if } i = 0 \text{ or } j = 0; \\ \dfrac{(\hat{y}_{ij-1} - \bar{y})^2 + (\hat{y}_{i-1j-1} - \bar{y})^2 + (\hat{y}_{i-1j} - \bar{y})^2}{3} & \text{otherwise,} \end{cases} \tag{3.8}$$

where $\bar{y}$ denotes the mean of $\hat{y}_{ij-1}, \hat{y}_{i-1j-1}, \text{and } \hat{y}_{i-1j}$. Note that, for simplicity, the $var_{ij}$ value of the pixels which are either in the first row or in the first column of the cover image are set to infinity. After calculating the $var_{ij}$ of each pixel in the cover image, the pixels of the cover image are classified into two groups: one is the edge group $G_E$ which contains those pixels whose $var_{ij}$ values are larger than a pre-determined threshold $V_{cut}$, and the other group is the smooth group $G_S$ which contains pixels whose $var_{ij}$ values are not more than $V_{cut}$. Apparently, according to

Human Visual System, we can hide more bits in the pixels in edge group than in smooth group. Therefore, a modulus function with a large modulus base $M_E$ will be applied to embed data in $G_E$, and another modulus function with a small modulus base $M_S$ is utilized to embed data in $G_S$.

The embedding procedure using variance-based modulus function is as follows. According to the raster scan-order, we sequentially take a pixel $(i, j)$ from the cover image, and call its pixel value $y_{ij}$ $(0 \le y_{ij} \le 255)$. Then, if the $var_{ij}$ value of this pixel is greater than $V_{cut}$, we set the modulus base $M_O$ to $M_E$; else set $M_O$ to $M_S$. Now, sequentially grab next $\lfloor \log_2 Mo \rfloor$ not-yet-embedded bits from the randomized code to form a short length data $x$ $(0 \le x < M_O)$. Then we embed $x$ in pixel $(i, j)$ by replacing its gray value from $y_{ij}$ to the resulting pixel value $\hat{y}_{ij}$. The embedding equation is

$$\hat{y}_{ij} = x + M_O \times \text{rounding}(\frac{y_{ij} - x}{M_O}), \tag{3.9}$$

where the rounding(·) operator means rounding its content to the nearest integer. Of course, we need to check whether $\hat{y}_{ij}$ falls in the valid range $0 \le \hat{y}_{ij} \le 255$. If it is out of the range, then $\hat{y}_{ij}$ should be adjusted by

$$\hat{y}_{ij} = \begin{cases} \hat{y}_{ij} + M_O & \text{if } \hat{y}_{ij} < 0; \\ \hat{y}_{ij} - M_O & \text{if } \hat{y}_{ij} > 255. \end{cases} \tag{3.10}$$

By sequentially processing each pixel $(i, j)$ of the cover image using above the proposed embedding procedure introduced here, the randomized code produced from Sec. 3.2.2 can be embedded in the pixels of the cover image.

Notably, in the above, about whether $M_E$ or $M_S$ should be used, the decision rule is according to the variance rather than the pixel value itself. Below we explain why. Figure 3.3 is the stego-image obtained when our embedding process is replaced by the embedding method introduced in Ref. [32], which are quite similar to ours, except that they used pixel value directly (rather than using variance) in the decision rule. To obtain Fig. 3.3(a), the 256×512 important image "Jet" shown in Fig. 3.5(a) is embedded in Fig. 3.4(a). The threshold $V$ (see Sec. 3.1) is set to 160, and two modulus bases are set to $M_U$=32 and $M_L$=16, the same as suggested in Ref. [32]. In other words, when the gray value of a cover image pixel is larger than 160, five bits of the hidden data are embedded in this pixel. On the other hand, when the gray value is not more than 160, then only four bits of the hidden data are embedded in this pixel. Obviously, there are some pockmarks on the shoulder and face of Lena in Fig. 3.3(a). Artificial distortion appears on the shoulder and forehead of Lena where pixel values vary smoothly and yet the gray values are often larger than 160. Hence, the pixel value might not be a suitable criterion to estimate the hiding capacity of a pixel in the cover image.

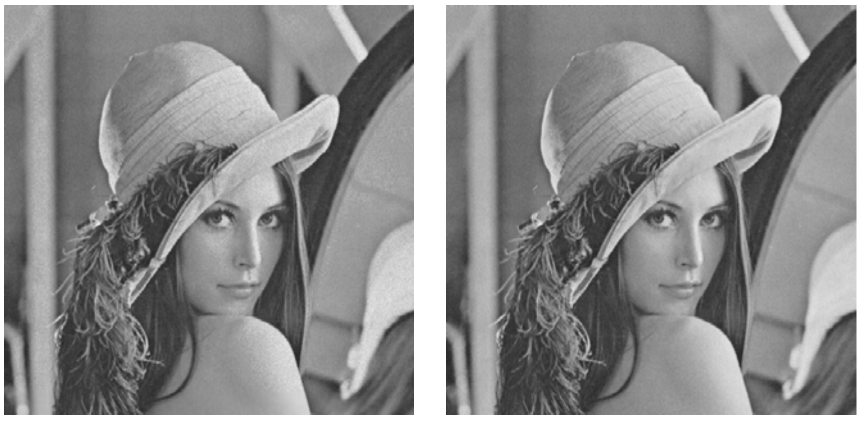


Fig. 3.3. The stego-images obtained by embedding a 256×512 important image Jet (Fig. 3.5(a)) in a 512×512 Lena (Fig. 3.4(a)). (a) The stego-image obtained by the pixel-based method [32] (PSNR=31.05 dB); (b) the stego-image using our variance-based method (PSNR=34.76 dB).

### 3.2.4 The Extraction Procedure for the Decoding

The extraction steps for revealing the important image are as follows.

(a) For a given stego-image, we first use Eq. (3.8) to compute the $var_{ij}$ value of each pixel $(i, j)$ of the stego-image, and then the modulus base $M_O$ of each pixel can be determined by comparing $var_{ij}$ with the threshold $V_{cut}$.

(b) Extract the randomized MSOC code (a binary stream) from the stego-image. This is achieved by sequentially process each pixel of the stego-image, with the raster scan order, and the extraction equation

$$x = \hat{y}_{ij} \pmod{M_O} \tag{3.11}$$

to obtain the hidden data $x$ from pixel $(i, j)$. Of course, convert each $x$ $(0 \le x < M_O)$ to its binary equivalent before processing next pixel of the stego-image.

(c) Obtain the same series of pseudo-random fraction numbers, which are used in the pseudo-random permutation phase (Sec. 3.2.2), by running the MT pseudo-random number generator [9] with the same seed. Then, do inverse permutation to obtain the non-randomized MSOC code stream.

(d) Fetch a bit sequentially from the (remaining) MSOC code stream; it is an indicator bit in our MSOC scheme.

(e) If the indicator bit obtained in step (d) is 0, fetch $N$ bits from the remaining MSOC code stream. Then, according to these $N$ bits and our MSOC pre-determined search path, locate the corresponding search position in the partially reconstructed important image. Use the pixel value at the pointed pixel position to paint the gray value of the current pixel position in the reconstructed image.

(f) If the flag bit obtained in step (d) is 1, fetch 8 bits from the remaining MSOC code stream, and directly assign these 8 bits to the pixel value of the current position in the reconstructed image.

(g) Repeat steps (d) through (f) until all of the data in the MSOC code stream are processed.

The generated image is the revealed important image.

## 3.3 Experimental Results

All images in the experiments are 8-bit gray-scaled. In each experiment, one of the important images is embedded in the cover image. As suggested in Ref. [13], we set $N=2$, i.e. we also use 2-bit string to record each MSOC position. As for the value of our threshold $V_{cut}$ and the values of the two modulus bases $M_E$ and $M_S$, they are all dynamic according to the total size of MSOC code. (Notably, the total size of MSOC code is affected by the threshold value $T$ in turn.)

In the MSOC encoding algorithm, set $T$ to 1 if the extracted important image is required to be error-free. On the other hand, use a larger value of $T$ if the cover image is not much bigger than the important image; for example, when both images are of 512×512.

In the first experiment we hide an important image in a cover image of the same size. Figure 3.4 shows an example of the hiding result in this experiment, where Fig. 3.4(a) is the cover image "Lena" of size 512×512, and Fig. 3.4(b) is the important image "Jet" of size 512×512. The obtained stego-images using thresholds $T$=7 and 9 are as shown in Figs. 3.4(c) and (d), respectively. The corresponding lossy version of the important images extracted from Figs. 3.4(c) and (d) are shown in Figs. 3.4(e) and (f), respectively. Finally, the settings of the experimental parameters, and the PSNR values of the stego-images and the extracted versions of the important image, are summarized in Table 3.1. From Table 3.1, we can see that the qualities of the stego-images are acceptable (the PSNRs between the stego-images and the original cover image are all greater than 36.0 dB). It is also hard to distinguish between Fig. 3.4(b) and those in Figs. 3.4(e) and 3.4(f) using naked eyes, and the PSNRs between the extracted lossy versions and the original version of the important image are all greater than 37.0 dB. Note that the extraction is lossy because we try to embed a 512×512 image in another 512×512 image of the same size, which is just impossible for us to set $T$=1.

In the second experiment we hide an important image whose size is half of the cover image. Figure 3.5 shows some examples of the hiding result in this experiment. The settings of the experimental parameters, and the PSNRs of the stego-images and the extracted important images, are listed in Table 3.2. From the PSNRs in Table 3.2, we can see that the qualities of the stego-images are good, for the PSNRs between the stego-images and the original cover image are all between 40.0 dB and 46.8 dB. The qualities of the extracted important images are also high, for the PSNRs between the extracted important images and the original important image are between lossless (∞) and 44.4 dB. Notably, the recovered important image is error-free when we use the most strict value $T$=1. As a remark, when $T$=3, the $M_E$ and $M_S$ computed by Step (3) of Sec 3.4.1 are ($M_E$=8, $M_S$=4) for image Jet, different from the ($M_E$=4, $M_S$=2) for image Tiffany; this should be of no surprise because the total number of bits in the produced MSOC codes are different for these two images.

To know the performance of our scheme, the PSNRs of the stego-image obtained in our scheme are compared in Table 3.3 with those elegant methods reported in literature. Note that in Ref. [23], the important image is first compressed using the VQ technique, and then the VQ indices and the encoded codebook are embedded in a cover image by the LSB substitution method. Therefore, in the comparison with Ref.

[23], we also use the VQ technique to compress the important image before hiding, and then encode the VQ index file using the MSOC algorithm with the threshold $T$=1 before using the pseudo-random permutation or the modulus embedding procedure. Of course, both the MSOC output code and the encoded codebook are randomized and embedded in the cover image. In Table 3.3, the symbol α means that the experiment used this special VQ approach. Notably, Refs. [18,20,29,31,32] did not process any important image of size as big as the 512×512 cover image, so we only list in Table 3.3 the experimental data copied from Refs. [22-23] when the important image is 512×512.

In order to make a fair comparison, we show the results of our two versions in Table 3.3. One version is without MSOC compression, the other version is with MSOC compression. The MSOC version has a PSNR much better than the PSNRs of all other listed methods; and this should be of no surprise because MSOC reduces the size of the important image. Below we focus on the without-MSOC version.

Since Ref. [20] outperformed the remaining reported methods listed in Table 3.3, we compare our without-MSOC version with Ref. [20], as follows. When the bpp (bits per pixel, here interpreted as the number of bits in the important image over the number of pixels in the cover image) is an integer, then the PSNR of the stego-image in our “without-MSOC" version is less than, but very close to, that of Ref. [20]. The phenomenon can be seen in Table 3.3. For instance, when we embed the 256×256 Tiffany (or 256×512 Tiffany) in a 512×512 Lena, the bpp 256×256×8/512×512=2 (or 256×512×8/512×512=4) is an integer; then the stego-image’s PSNR is 46.33 dB (or 34.74 dB) in our method, and 46.37 dB (or 34.84 dB) in Ref. [20]. (The difference is only 0.04 dB (0.1 dB)).

However, when the bpp is not an integer, then our stego-image’s PSNR (in the without-MSOC version) is higher than that of Ref. [20]. To show this, we embed the important image Jet of various sizes into the 512×512 cover image Lena (Fig. 3.4(a)), without using MSOC. The experimental results are shown in Fig. 3.6(a). One of such examples is to embed a 320×320 Jet into the 512×512 Lena, which means the bpp is 320×320×8/ (512×512)= 3.125; and it is found that our 39.32 dB stego-image is better than the 35.90 dB stego-image of Ref. [20]. Figure 3.6(b) is the average of 30 tests (30 tests for each bpp value), in which each test uses one of the 30 important images {Lena, Jet, Tiffany, Baboon, Barbara, Boat, Bridge, Couple, Elaine, Family, Gold, House, Milk, Painting, Pepper, Scene, Tank, Toys, Woman, Zelda, Girl, Cameraman, Beach, Car, Iran, Logo, Map, Mickey, Satellite image, X-ray image} of a specified size. In general, as shown in Fig. 3.6, if the bpp is not an integer, then our stego-images' PSNR are better than the stego-images' PSNR of Ref. [20]. This is because Ref. [20] was originally designed to use *t*-bits LSB in a finer manner, and their *t* were integers; as a result, when the number of bits in the important image was, for example, 3.125 times larger than the number of pixels in the cover image, then they could not use *t*=3 bits LSB; instead, they needed to use *t*=4. Therefore, their distortion curve has a sudden jump (while our PSNR curve has no such jump).

In conclusion, if no MSOC is allowed in our method, then as shown in Fig. 3.6, when the bpp is an integer (i.e. when the number of "bits" in the important image is an integer multiple of the number of "pixels" in the cover image), then Ref. [20] is a little better than ours (the PSNR difference is an amount between 0.04 dB and 0.1 dB in Table 3). However, when the bpp is not an integer, then ours is better (the PSNR difference could be as big as 3 dB).

As for the experiment done by Wu and Tsai [28], they used a Word-format file, rather than an image, as the hidden important data. To compare with Ref. [28], we try to embed a Word-format file with 50,960 bytes in an extra experiment. If the cover image is the 512×512 Lena, the PSNR of the stego-image Lena obtained by Ref. [28]

is 41.79 dB. On the other hand, our stego-image's PSNR is 47.65 dB even if we do not use the MSOC compression (the PSNR would have been better than 47.65dB if MSOC had been used). Analogously, when trying to embed a Word-format file of 25,940 bytes, their PSNR is 48.43 dB and ours is 51.89 dB when MSOC is not used. In summary, our method can also compete with Ref. [28].

About the (net) embedding rate of our scheme, the MSOC reduced the 256×256 Jet (and 256×256 Tiffany) from 256×256=65,536 bytes to 51,817 bytes (and 53,467 bytes) with threshold *T*=1 before embedding. Similarly, the MSOC reduced the 256×512 Jet (and 256×512 Tiffany from 256×512=131,072 bytes to 100,989 bytes (and 109,054 bytes) with threshold *T*=1 before embedding. So, the (pure) embedding rate for the 512×512 cover image Lena mentioned in the experiments and Table 3.3 is 51817×8/(512×512)= 1.58 bits per pixel (bpp) for the compressed code of 256×256 Jet (and 53467×8/(512×512)=1.63 bpp for 256×256 Tiffany), and 100989×8/(512×512)= 3.08 bpp for the compressed code of 256×512 Jet (and 109054×8/(512×512)=3.33 bpp for 256×512 Tiffany).

In summary, to hide Jet's MSOC code, according to Table 3.3, we obtained 47.76 dB stego-image Lena when the (net) hiding rate is 1.58 bits per pixel; and obtained 39.80 dB stego-image Lena when the (net) hiding rate is 3.08 bits per pixel. Analogously, to hide the source data of image Jet directly without using compression, we obtained 46.33 dB stego-image Lena when the hiding rate is 256×256×8/(512×512)=2 bits per pixel ; and obtained 34.76 dB stego-image Lena when the hiding rate is 256×512×8/(512×512)=4 bits per pixel.

To ensure that the proposed method can work well for most of the images, we test thirty important images {Lena, Jet, Tiffany, Baboon, Barbara, Boat, Bridge, Couple, Elaine, Family, Gold, House, Milk, Painting, Pepper, Scene, Tank, Toys, Woman, Zelda, Girl, Cameraman, Beach, Car, Iran, Logo, Map, Mickey, Satellite image, X-ray image}. Each image has three kinds of size: 256×256, 256×512, and 512×512. So there are in fact 30×3=90 images. Each time one of these 90 important images is hidden in the 512×512 cover image Lena shown in Fig. 3.4(a). The results are shown in Table 3.4, from where we can see that, to hide a 256×512 important image using the threshold $T$=3, the PSNR of the extracted image is about 48.89 dB and the PSNR of the stego-image is about 43.97 dB. So both the qualities of the extracted and stego-image are acceptable. Analogously, when we use $T$=8 to hide one of the 30 important images whose sizes are all as large as the 512×512 cover image, the PSNR of the extracted image is about 39.88 dB in average, and the PSNR of the stego-image is about 37.24 dB in average. Therefore, the proposed method can still work for most of the large images.

We also do another thirty tests; and the cover image in each test is one of the thirty 512×512 images {Lena, Jet, Tiffany, Baboon,…., Mickey, Satellite image, X-ray image} mentioned above. Table 3.5 summarizes these thirty tests. From Table 3.5, we can see that the proposed method is stable (has a narrow range) for using different images as the cover image.

As for the processing time, the average encoding/decoding time of our method are shown in Tables 3.6 and 3.7. From Tables 3.6 and 3.7, we can see that the larger the size of the important image, the higher is the encoding/decoding time. Note that, all programs in the current dissertation were implemented by using the Borland C++ Builder 6.0, and ran on a notebook with Intel Pentium Processor M-740 (1.73 GHz) and 512MB RAM under the operation system of Microsoft Windows XP Professional.

## 3.4 Discussions

### 3.4.1 Parameter Setting

Because the extracted image is the compression result of the important image with the threshold $T$, the quality of the extracted image is determined by the threshold $T$. (The larger the threshold $T$, the higher is the compression ratio; i.e. the larger the threshold $T$, the lower is the quality of the extracted "important image".)

On the other hand, once the MSOC code (the compression result) is generated, the MSOC code is fixed (so the quality of the important image recovered in the future is also fixed). To hide this fixed MSOC file, we determine the suitable $V_{cut}$ value so that we can avoid causing too much distortion to the cover image (as long as the "whole" MSOC code file can be hidden in the cover image completely). So, the $V_{cut}$ value determines the quality of the "stego-image". In general, the larger the $V_{cut}$ value, the better is the quality of the stego-image. However, the $V_{cut}$ value cannot be too large; otherwise, there will not be enough space in the cover image to hide the whole MSOC code file.

Therefore, about parameters setting, we proceed as follows

(1) We first determine a value of $T$ according to the size of the important image. (For example, Table 3.1 uses a larger $T$ (between 7－11) because its important image size 512×512 is larger than the important image size 256×512 in Table 3.2, which uses a smaller $T$ (between 1－5).

(2) Then get the MSOC code corresponding to this $T$.

(3) Since we want to hide the MSOC code in the cover image, the bits per pixel (bpp) used to measure the (net) embedding rate is

$$\text{bpp} = \frac{\text{The total number of bits in the MSOC code}}{\text{The total pixels of the cover image}}. \tag{3.12}$$

Then, let $M_S = 2^{\lfloor \text{bpp} \rfloor}$ and $M_E = 2^{\lceil \text{bpp} \rceil}$, where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the ceiling and floor function to get two closest integers of bpp.

(4) Then, to avoid damaging the cover image too much, we start from a large value of $V_{cut}$ .

(5) If the MSOC code cannot be embedded completely in the cover image using this $V_{cut}$ value, then try to hide again using a smaller $V_{cut}$ value. Repeat reducing the $V_{cut}$ value if necessary. □

As a final remark of this subsection 3.4.1, note that in our method, since the input to the embedding algorithm (Sec 3.2.3), is in fact the MSOC code, and the MSOC code can be embedded in cover image and extracted from cover image without any error; therefore, BER=0 always holds (BER stands for bit-error-rate). The fact that BER is always 0 has nothing to do with parameter setting.

### 3.4.2 Using Pseudo-Random Process to Increase Security Level

As in other image-hiding techniques, our stego-images are utilized to reduce the attention of hackers. However, because the parameters $V_{cut}$, $M_S$, and $M_E$ are not necessarily constant, we transmit the values of these parameters along with the steog-image to the receiver. If a hacker happens to monitor the transmission of a stego-image, and if he also knows the number of bits ($N$) used to record MSOC, then the MSOC code of the important image might be extracted completely, and the important image is then exposed. (For example, if the hacker has got a MSOC code (**1**01110110**0**00**1**01111001)$_2$, he can decode the code according the MSOC algorithm described in Sec. 3.2.1 to obtain that the three pixels are (OPV(118), MSOC(0), OPV(121)), and then know the three pixel values are (118, 118, 121). ) Therefore, the pseudo-random process is needed to prevent the important image from being revealed. The protection is through randomizing the MSOC code before embedding. To randomize a binary string of length $S$, there are $C(S,g)=\dfrac{S!}{(S-g)!g!}$ possible combinations where $g$ is the number of 1 in the binary string. The possibility of guessing the right solution is only $\dfrac{1}{C(S,g)}$. In the above example, the possibility is $\dfrac{1}{C(21,12)}=\dfrac{1}{293{,}930}$. In our experiment which embed 256×256 Jet into 512×512 Lena with threshold $T$=3, the size of the MSOC code is 285762, and $g$ is 106090, so the possibility is 1/$C$(285762,106090). Obviously, there is a very small chance for the hacker to guess the right combination to obtain the important image. Hence, the pseudo-random process is utilized to prevent the important image from being divulged. Note that the seed of the pseudo-random number generator [9] can be a private key known to the sender and the receiver. The hacker can hardly know the important image without the private key, even if he monitors the whole process of transmission, and picks the right stego-images from a bunch of coy images sent along with these stego-images.

## 3.5 Summary

In this dissertation, a steganography scheme is proposed. The scheme contains three parts: 1) MSOC encoding; 2) pseudo-random permutation; and 3) an embedding process using variance-based modulus function. The MSOC method makes the important image more compact and hence more suitable for the embedding procedure later. A user-specified non-negative integer threshold $T$ is introduced to control the length of the MSOC code, which in turn affects later the quality of the extracted important image. If the size of important image is not small, then we might need to use a larger value of $T$ to handle the case. However, if the size of important image is small, then the readers may just use $T$=1 so that the extracted important image is error-free. Therefore, the use of $T$ provides more flexibility for practical applications. To increase the security level, a pseudo-random permutation algorithm has been utilized by applying the MT pseudo-random number generator [9]. In the embedding part, we use a variance-based criterion to estimate the hiding capacity of a pixel in the cover image. The criterion is based on human visual system (HVS): more bits can be hidden in a pixel of busy area. From the experimental results, it can be seen that the variance-based estimation is more suitable than its counterpart in Ref. [32] which uses the pixel value to estimate the hiding capacity of a pixel. Experimental results show that the quality of both the stego-images and extracted important images are competitive to those obtained in many existing steganography methods reported recently. From Table 3.3, it can be seen that the proposed method can create low-profile stego-images to protect the important image (because stego-images are with competitive qualities), and yet preserve the fidelity of the important image. Notably, the cover images are not necessarily bigger than the important images in our approach.

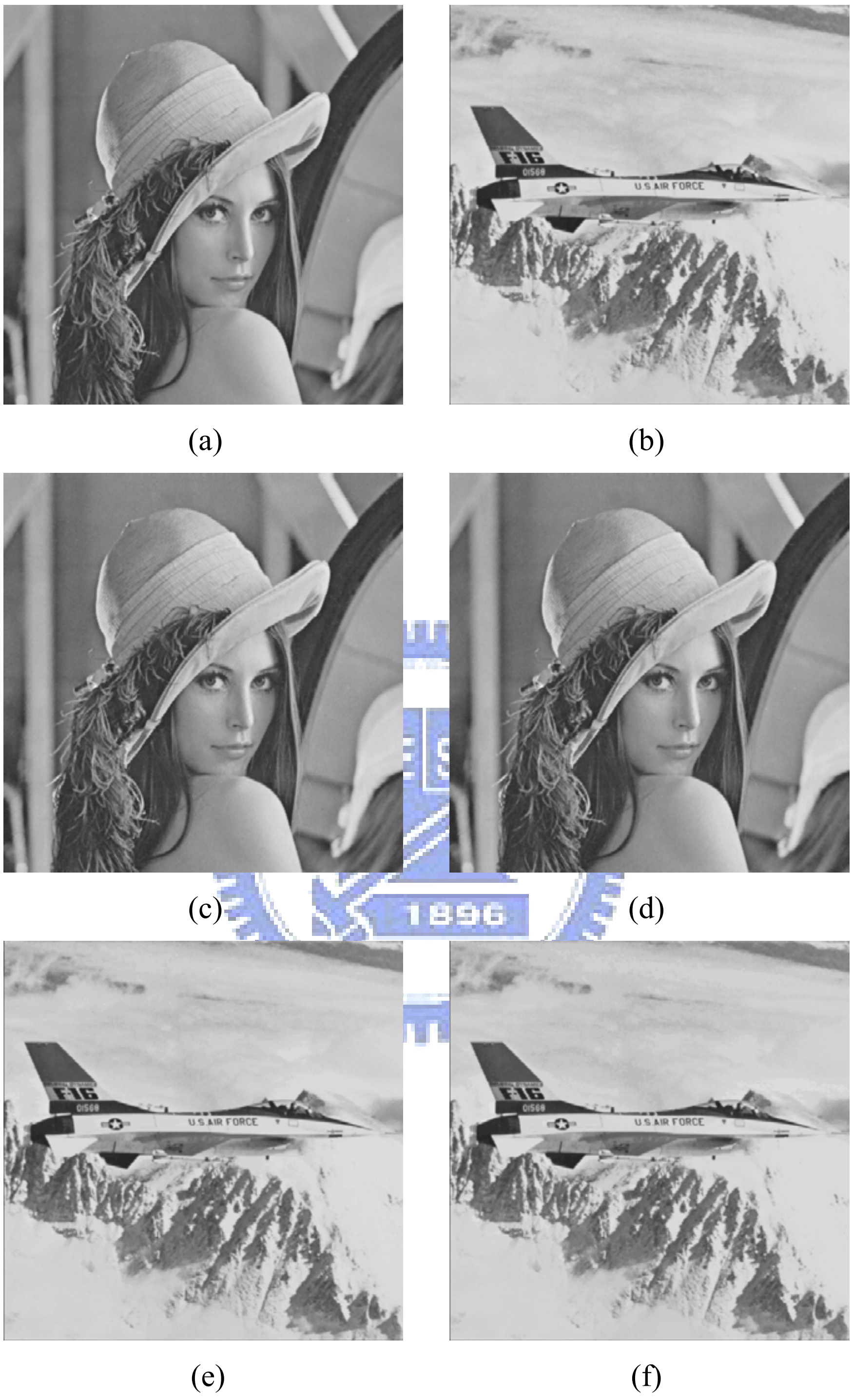


Fig. 3.4. The first experiment. (a) the cover image “Lena” with 512×512 pixels; (b) the important image “Jet” with 512×512 pixels; (c-d) the stego-images obtained by embedding Fig. 3.4(b) in Fig. 3.4(a) with threshold values $T = 7$ and 9, respectively; (e-f) the extracted important images from Fig. 3.4(c) and (d), respectively.

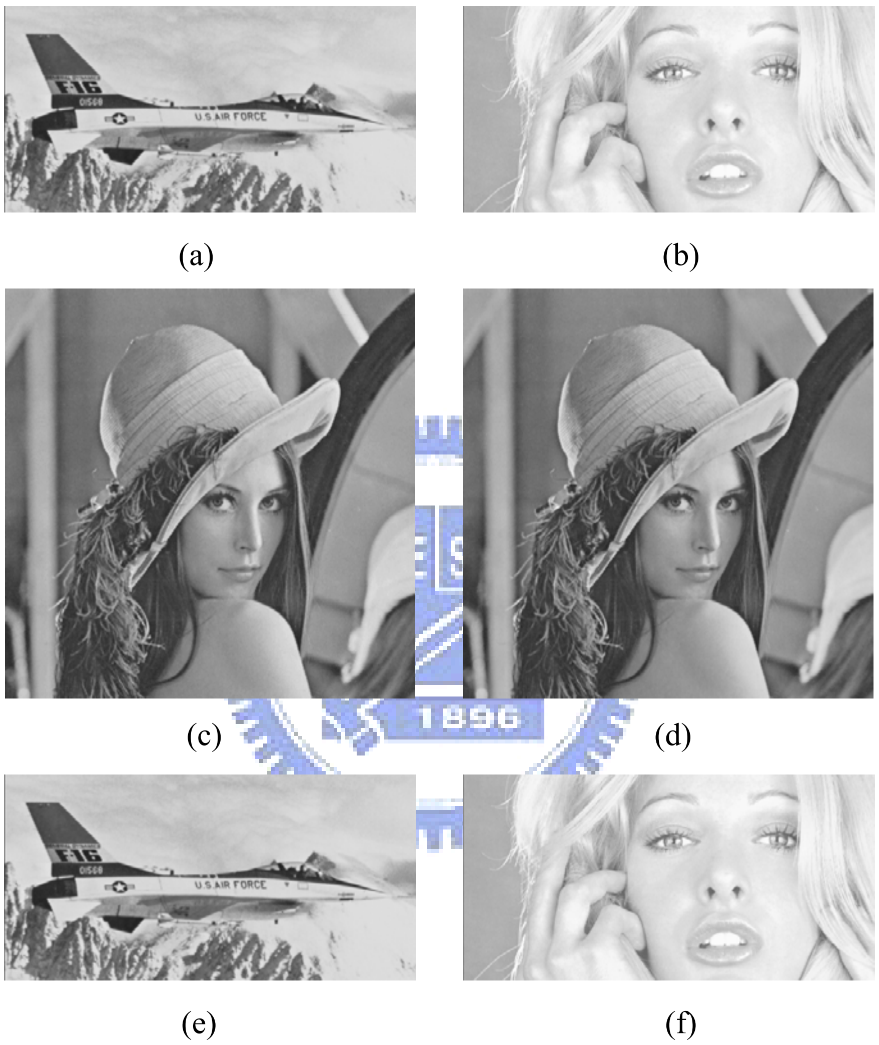
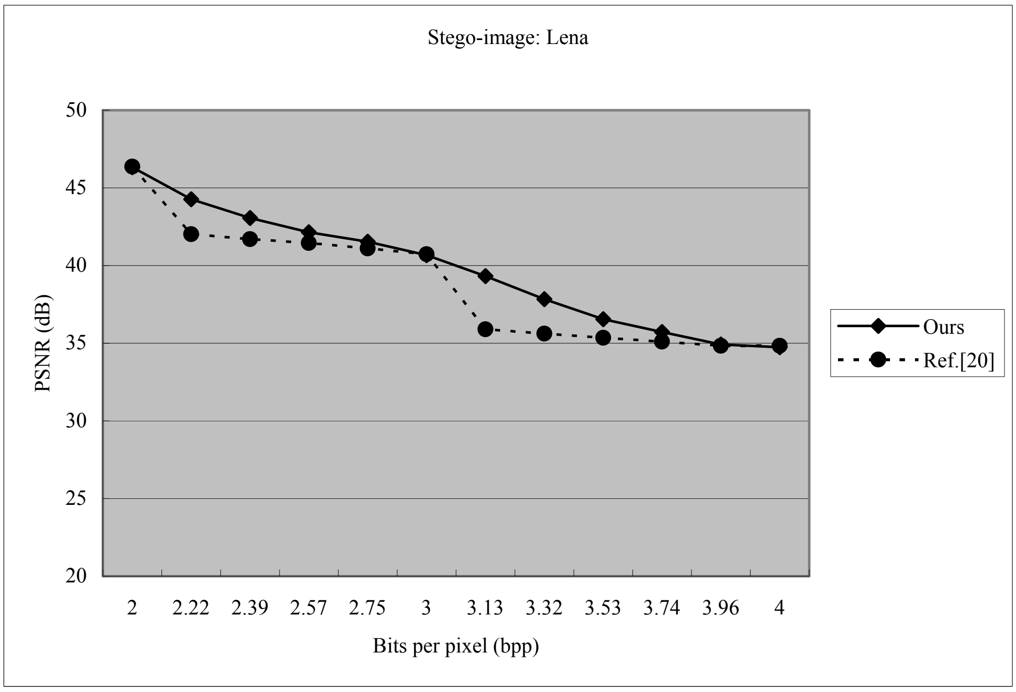
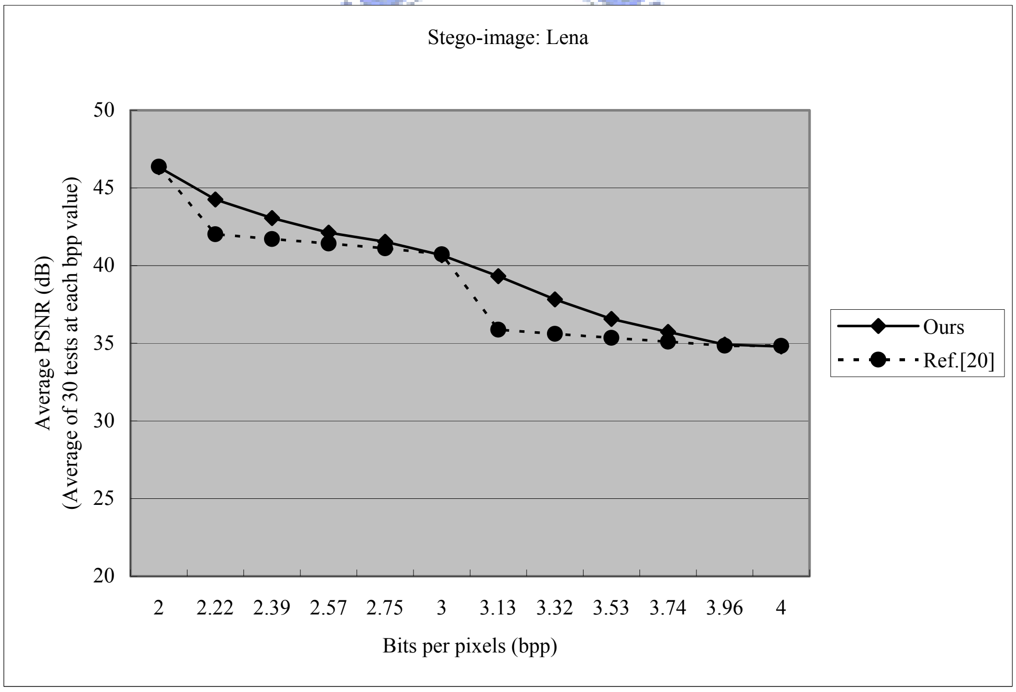


Fig. 3.5. The second experiment about hiding the middle-size (256×512 pixels) important image. (a) the important image "Jet"; (b) the important image "Tiffany"; (c-d) the stego-images obtained by embedding Figs. 3.5(a) and 3.5(b) in Fig. 3.4(a) with threshold values $T$ = 1 and 3, respectively; (e-f) the extracted important images from Figs. 3.4(c) and 3.4(d), respectively. Note that (e) is identical to Fig. 3.5(a) without any loss.



(a)



(b)

Fig. 3.6. The comparison of stego-image's quality between Ref. [20] and our no-MSOC version, when the cover (stego) image is Lena. (a) The result of using Jet as the hidden important image; (b) the average of testing 30 important images (so, 30 tests for each bpp value).

Table 3.1. The parameters used in our experiments to embed an important image of size 512×512. The PSNR values (marked as "stego") are between the stego-image and cover image, and the PSNR values (marked as "extracted") are between the extracted important image and original important image.

| Cover image | Important image | Parameters and PSNR | $M_E = 16, M_S = 8$ | | | | |
|---|---|---|---|---|---|---|---|
| | | | $T$=7 | $T$=8 | $T$=9 | $T$=10 | $T$=11 |
| Lena | Jet (512×512) | Threshold ($V_{cut}$) | 14 | 16 | 20 | 24 | 32 |
| | | PSNR (stego) | 37.12 | 37.56 | 38.05 | 38.31 | 38.70 |
| | | PSNR (extracted) | 41.77 | 40.55 | 39.50 | 38.77 | 37.87 |
| | Tiffany (512×512) | Threshold ($V_{cut}$) | 29 | 42 | 55 | 70 | 98 |
| | | PSNR (stego) | 38.62 | 38.96 | 39.40 | 39.63 | 39.83 |
| | | PSNR (extracted) | 41.43 | 40.15 | 39.36 | 38.66 | 37.66 |

Table 3.2. The parameters used in our experiments to embed the important image of size 256×512. The PSNR values (marked as "stego") are between the stego-image and cover image, and the PSNR values (marked as "extracted") are between the extracted important image and original important image.

| Cover image | Important image | Parameters and PSNR | The tolerance parameter $T$ | | | | |
|---|---|---|---|---|---|---|---|
| | | | $T$=1 | $T$=2 | $T$=3 | $T$=4 | $T$=5 |
| Lena | Jet (256×512) | $M_E$ / $M_S$ | $M_E$=16, $M_S$=8 | $M_E$=8, $M_S$=4 | | $M_E$=4, $M_S$=2 | |
| | | Threshold ($V_{cut}$) | 90 | 12 | 130 | 0.6 | 0.8 |
| | | PSNR (stego) | 39.80 | 43.68 | 45.75 | 46.61 | 46.83 |
| | | PSNR (extracted) | Error-free | 52.83 | 49.12 | 46.22 | 44.40 |
| | Tiffany (256×512) | $M_E$ / $M_S$ | $M_E$=16, $M_S$=8 | $M_E$=8, $M_S$=4 | $M_E$=4, $M_S$=2 | | |
| | | Threshold ($V_{cut}$) | 16 | 11 | 0.2 | 1 | 1.5 |
| | | PSNR (stego) | 37.54 | 43.57 | 46.43 | 46.94 | 47.18 |
| | | PSNR (extracted) | Error-free | 52.18 | 48.40 | 45.68 | 43.80 |

Table 3.3. A comparison between some published methods and our scheme. ("*" means "quoted from the reported papers", and "α" means "the input of the MSOC is the VQ index file of the important image, rather than the important image itself")

| Cover image | Scheme | Size of the important image | PSNR of extracted | PSNR of stego image |
|---|---|---|---|---|
| Lena (Baboon, when PSNR is enclosed by parentheses in the final column of this table) | [18] | 256×256 Tiffany | Error-free | 44.90*(44.90*) |
| | [18] | 256×256 Jet | Error-free | 44.53*(44.54*) |
| | [20] | 256×256 Tiffany | Error-free | 46.37*(46.38*) |
| | [29] | 256×256 Jet | Error-free | 38.85* |
| | [32] | 256×256 Jet | Error-free | 44.08*(44.03*) |
| | *ours without MSOC* | 256×256 Tiffany | *Error-free* | *46.33 (46.32)* |
| | *ours without MSOC* | 256×256 Jet | *Error-free* | *46.33 (46.31)* |
| | *ours* | 256×256 Tiffany | *Error-free* | *47.52 (47.58)* |
| | *ours* | 256×256 Jet | *Error-free* | *47.76 (47.78)* |
| | [18] | 256×512 Tiffany | Error-free | 32.90* (32.95*) |
| | [18] | 256×512 Jet | Error-free | 32.71* (32.79*) |
| | [20] | 256×512 Tiffany | Error-free | 34.84* (34.79*) |
| | [31] | 256×512 Tiffany | Error-free | 34.80* |
| | [31] | 256×512 Jet | Error-free | 34.76* |
| | [32] | 256×512 Jet | Error-free | 31.05* (31.09*) |
| | *ours without MSOC* | 256×512 Tiffany | *Error-free* | *34.74 (34.73)* |
| | *ours without MSOC* | 256×512 Jet | *Error-free* | *34.76 (34.73)* |
| | *ours* | 256×512 Tiffany | *Error-free* | *37.54 (37.81)* |
| | *ours* | 256×512 Jet | *Error-free* | *39.80 (39.79)* |
| | [22] | 512×512 Jet | 30.01* | 32.50* |
| | [23] | 512×512 Tiffany | 32.02* | 44.42* |
| | *ours*$^{\alpha}$ *without MSOC* | 512×512 Tiffany | *32.02*$^{\alpha}$ | *51.68*$^{\alpha}$ *(51.69*$^{\alpha}$*)* |
| | *ours*$^{\alpha}$ *without MSOC* | 512×512 Jet | *31.43*$^{\alpha}$ | *51.69*$^{\alpha}$ *(51.69*$^{\alpha}$*)* |
| | *ours*$^{\alpha}$ | 512×512 Tiffany | *32.02*$^{\alpha}$ | *53.37*$^{\alpha}$ *(53.37*$^{\alpha}$*)* |
| | *ours* | 512×512 Jet | *40.55* | *37.69 (37.81)* |

Table 3.4. The results of testing 30 important images.

| Size of each important image | Versions | 30 extracted images | | 30 stego-images | |
|---|---|---|---|---|---|
| | | PSNR (Range) | PSNR (Average) | PSNR (Range) | PSNR (Average) |
| 256×256 | No MSOC | Error-free | Error-free | 46.31—46.35 | 46.335 |
| | $T$=1 in MSOC | Error-free | Error-free | 46.08—51.40 | 47.480 |
| 256×512 | $T$=3 in MSOC | 47.75—52.29 | 48.886 | 38.16—47.84 | 43.968 |
| 512×512 | $T$=8 in MSOC | 38.64—43.66 | 39.881 | 30.60—40.56 | 37.242 |

Table 3.5. The results of testing 30 cover images. "α" means "the input of the MSOC is the VQ index file of the important image, rather than the important image itself".

| Cover images | Scheme | Size of the important image | PSNR (extracted) | PSNR (stego) |
|---|---|---|---|---|
| 30 tests using 30 cover images | *ours without MSOC* | 256×256 Tiffany | *Error-free* | *46.28 —46.34* |
| | *ours without MSOC* | 256×256 Jet | *Error-free* | *46.27 —46.35* |
| | *ours* | 256×256 Tiffany | *Error-free* | *47.51 —47.61* |
| | *ours* | 256×256 Jet | *Error-free* | *47.62 —47.81* |
| | *ours without MSOC* | 256×512 Tiffany | *Error-free* | *34.71 —34.78* |
| | *ours without MSOC* | 256×512 Jet | *Error-free* | *34.70 —34.79* |
| | *ours* | 256×512 Tiffany | *Error-free* | *37.50 —37.86* |
| | *ours* | 256×512 Jet | *Error-free* | *39.76 —39.83* |
| | *ours*$^{\alpha}$ *without MSOC* | 512×512 Tiffany | $32.02^{\alpha}$ | $51.65^{\alpha}$ —$51.72^{\alpha}$ |
| | *ours*$^{\alpha}$ *without MSOC* | 512×512 Jet | $31.43^{\alpha}$ | $51.66^{\alpha}$ —$51.72^{\alpha}$ |
| | *ours*$^{\alpha}$ | 512×512 Tiffany | $32.02^{\alpha}$ | $53.32^{\alpha}$ —$53.44^{\alpha}$ |
| | *ours*$^{\alpha}$ | 512×512 Jet | $31.43^{\alpha}$ | $53.45^{\alpha}$ —$53.58^{\alpha}$ |
| | *ours* | 512×512 Tiffany | *40.15* | *38.74 —39.21* |
| | *ours* | 512×512 Jet | *40.55* | *37.36 —37.83* |

Table 3.6. The encoding time of each procedure in our method. (unit: second)

| Size of important image | MSOC | Permutation | Embedding | Total |
|---|---|---|---|---|
| 256×256 | 0.203 | 0.406 | 0.078 | 0.687 |
| 256×512 | 0.375 | 0.483 | 0.085 | 0.943 |
| 512×512 | 0.723 | 0.667 | 0.093 | 1.483 |

Table 3.7. The decoding time of our method. (unit: second)

| Size of important image | Decoding |
|---|---|
| 256×256 | 0.599 |
| 256×512 | 0.734 |
| 512×512 | 1.197 |

# Chapter 4
# A Sharing-Based Authentication and Self-recovery Method against Image Tampering

In this chapter, we present an authentication method with self-recovery ability. The hidden watermark in the proposed method not only detects and locates the tampered portion of an image, but also self-recovers the tampered portion. To increase the ability of recovery from tampering, the recovery data are shared by using an ($r$, $n$) threshold scheme, and then scattered all over the image. Experimental results indicate that the visual quality of the watermarked image is acceptable, and that the positions of the tampered portions can be identified. Most importantly, the *self*-recovery result is very competitive.

The rest of this chapter is organized as follows. An introduction of image authentication technique with tampering recovery ability is given in Section 4.1. Section 4.2 briefly reviews the secret image sharing method. The encoding (watermarking) and decoding (detection, locating, and recovery) of the proposed method are described in Sections 4.3 and 4.4, respectively. Experimental results are presented in Section 4.5. The discussion and comparison are in Sec. 4.6, and the conclusions are in Sec. 4.7.

## 4.1 Introduction

Transmission of digital images by network has become very popular due to the rapid development of Internet and computer technologies. However, tampering with digital images is also easier, for modern pervasive and powerful image manipulation tools have made imperceptible modification of images very simple. Therefore, protecting the integrity of images is an important issue. Image authentication is a technique which can verify the integrity of a digital image by detecting whether there is any malicious manipulation to the protected image. Besides tampering detection, the recovery of tampered area has become a very popular research issue in recent years. Several image authentication schemes with tampering recovery ability have been proposed [70-78]. In their methods, the hidden watermark can not only detect and locate the tampered portion of an image, but also recover the tampered portions. Most of them backup the recovery data of a block by another block whose address is determined by a permutation function. If a block is marked as a tampered block after the detection phase, its recovery data is retrieved from the backup block to reconstruct the appearance of the tampered block. However, if the watermarked image is tampered extensively in a large area and randomly, the block and its backup block could be tampered simultaneously. In this case, the recovery of the tampered block fails. To solve this problem, in this chapter, we design an image authentication with higher self-recovery ability to handle the case that a large area is tampered even if 50% of the watermarked image is cropped. The details of the proposed method are presented in the following sections.

## 4.2 Review of Secret Image Sharing

Because the proposed method utilizes the sharing polynomial of Thien and Lin's method [90], their work is reviewed below to provide some necessary background knowledge. In Ref. [90], a secr◦et image $O$ containing $m$ pixels is shared by $n$ participants based on Shamir's polynomial $(r, n)$-threshold scheme [3] with a module base *p=251*. The image $O$ is first transformed into a noisy image $Q$ by permuting pixels according to a secret key. Then, $Q$ is further divided into $m/r$ non-overlapping sections so that each section contains $r$ pixels. Let $q(x)$ be the $x^{\text{th}}$ shadow image, and $q_j(x)$ be the $j^{\text{th}}$ pixel in $q(x)$, where $1 \le x \le n$ and $1 \le j \le m/r$. For each section $j$, the $r$ coefficients $a_0, a_1, \ldots, a_{r-1}$ of the corresponding polynomial

$$q_j(x) = a_0 + a_1 \times x + \cdots + a_{r-1} \times x^{r-1} \pmod{p} \qquad (4.1)$$

are used as the gray values of the $r$ pixels of the corresponding section $j$ in $Q$. The $x^{\text{th}}$ shadow image $q(x)$ is the collection $\{q_j(x) \mid j = 1,2,\ldots,m/r\}$. Since each section $j$ has $r$ pixels and contributes only one pixel $q_j(x)$ to the $x^{\text{th}}$ shadow image, the size of each generated shadow image is only $1/r$ of that of the secret image $O$. This property holds for every shadow image, i.e. for every $x \in \{1,2,3,\ldots,n\}$. Any $r$ of the $n$ shadow images can be utilized to reconstruct $Q$, because the inverse-finding of the $r$ coefficients $a_0, a_1, \ldots, a_{r-1}$ in Equation (4.1) only needs $r$ of the $n$ values $\{q_j(1), q_j(2), \ldots, q_j(n)\}$. The detail is omitted.

To enhance the recovery ability of tampered image, the proposed method employs the polynomial sharing equation (4.1) to share the information needed for recovery. This procedure is described in detail in Section 4.3.1. Notably, Thien and Lin's method [90] uses $p$ = 251, but we use $p = 2^{\lceil (log_2 L)/r \rceil}$, where $L$ is the VQ codebook size. Additionally, all arithmetic calculations in the proposed method are performed in the power-of-two Galois Field GF($2^{\lceil (log_2 L)/r \rceil}$), and thus reducing the number of bits needed to express a share value (the left hand side of Eq. (4.1)) from $\lceil log_2 251 \rceil = 8$ bits to $\lceil (log_2 L)/r \rceil$ bits.

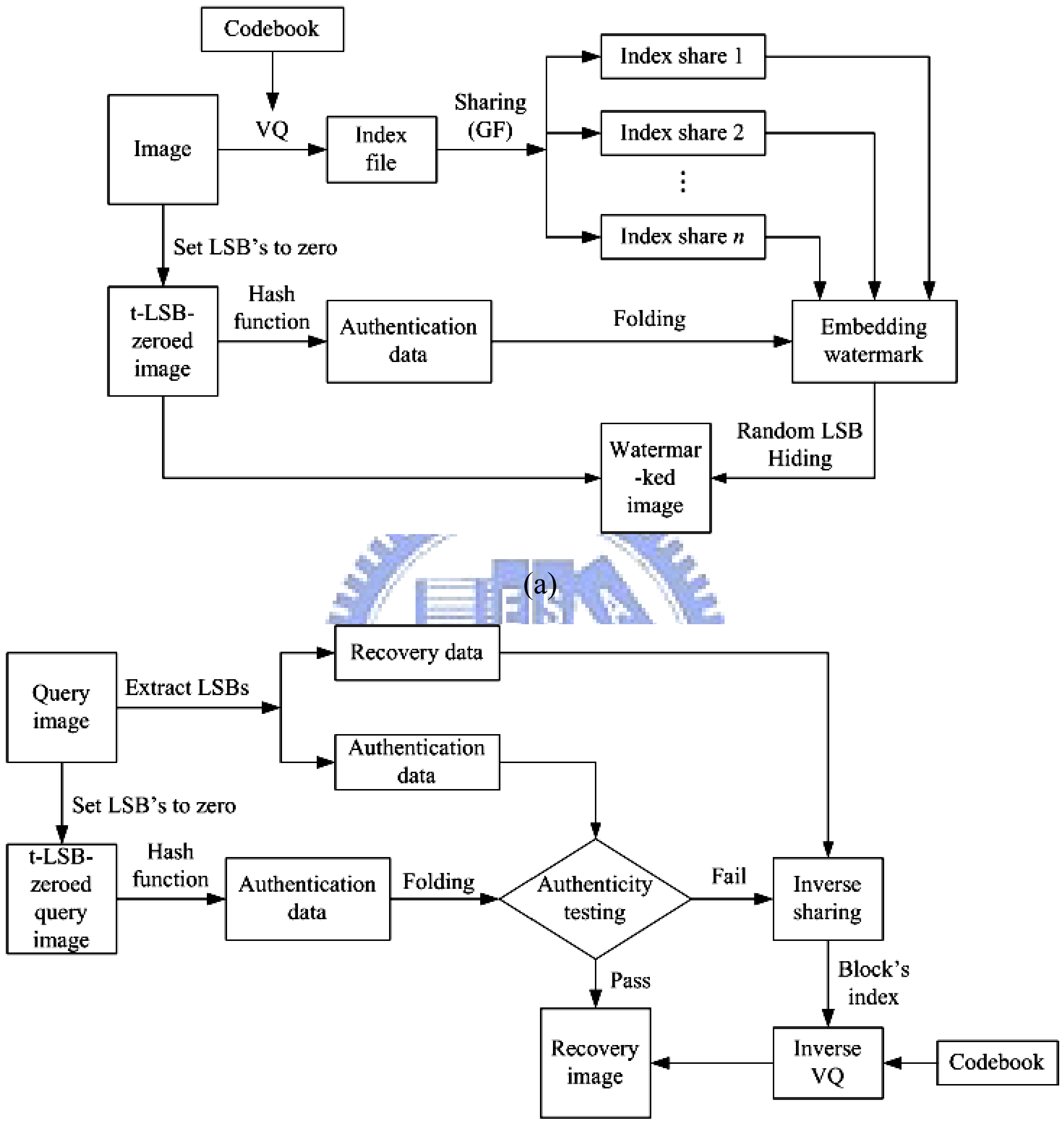


Fig. 4.1. Two flowcharts of the proposed method: (a) the encoding (watermarking) procedure; (b) the decoding (verification-and-recovery) procedure.

## 4.3 The Proposed Method (Encoding)

Figure 4.1(a) shows the flowchart of the encoding (watermarking) procedure; while Fig. 4.1(b) illustrates the decoding (verification and recovery) procedure. Section 4.3 explains Fig. 4.1(a) only.

The encoding procedure consists of three parts: i) The input 8-bit grayscale image is first encoded by VQ compression technique [15] to generate a VQ-index file of the image. The VQ-index file of the image is then shared by Thien and Lin's method [90] (see Eq. (4.1) above) to generate *n* shares. These index shares are treated as the recovery data. ii) Next, the authentication data for each image block are generated using a cryptographic hash function, in which the input include the local information of the image block, and the interrelated information (about the image) to resist the collage attack. iii) Finally, the recovery data of other block are combined with the authentication data to form the watermark for the block. The watermark is embedded in the LSBs of the block in the image. The position of the so-called "other block" may be determined by a permutation method based on the Mersenne Twister (MT) pseudo-random number generator [9].

The details of the encoding are described in the following three subsections accordingly.

### 4.3.1 Generation of the Recovery Data

The image recovery data are generated first. To do this, the image is partitioned into non-overlapping blocks of $h \times w$ pixels each, where $h$ is the block height, and $w$ is the block width. Each block of the image is then represented by an index value according to a given codebook of size $L$ (i.e. the number of codewords in the codebook is $L$). Therefore, the compression result of the image is an index file. The index value of each image block can be used as the recovery information when the block is tampered.

To increase the survival rate in the recovery process, Thien and Lin's sharing method [90] is utilized to share the index file of the VQ-compressed image, as described below. The index file in a sequence of *index value by index value* (or equivalently, *block by block* is shared according to the raster-scan order of the image, which is from left to right and top to bottom). Each index value is transformed into a binary stream of $log_2 L$ bits, and this binary stream is then divided into $r$ sections. Therefore, each section has $\left\lceil \frac{log_2 L}{r} \right\rceil$ bits, where $L$ is the codebook size, and $\lceil \cdot \rceil$ denotes the *ceiling* operation. Each section of the binary stream is then transformed into a decimal number, and these $r$ decimal numbers of an index value are assigned as the $r$ coefficients in Eq. (4.1). By substituting $n$ distinct values for the variable $x$ in Eq. (4.1), $n$ shares can be generated. In other words, the index value of an image block is shared among $n$ index shares. Notably, while Thien and Lin [90] used $p$ = 251, as stated at the end of Sec. 4.2, the proposed method uses $p = 2^{\lceil (log_2 L)/r \rceil}$, and performs all arithmetic calculations in the power-of-two Galois Field GF($2^{\lceil (log_2 L)/r \rceil}$). This change reduces the number of bits needed to express a share value (the left hand side of Eq. (4.1)) from $\lceil log_2 251 \rceil = 8$ bits to $\lceil (log_2 L)/r \rceil$ bits. Additionally, the bit-length of each share value (which has $\lceil (log_2 L)/r \rceil$ bits) is only about $1/r$ of that of an index value (which has $log_2 L$ bits).

Since the ($r$, $n$) threshold scheme is used to share an index value among $n$ shares, the index value can still be recovered even if (up to) $n-r$ shares are lost. This increases the survival rate (the recovery ability) of a tampered block if the $n$ shares of the tampered block are stored in $n$ distinct places of the image in advance. Moreover, because the bit-length of each share value is only $1/r$ of that of the index value, it is easy to embed such a compact share value in the watermark.

Here we explain more clearly how ($r$, $n$) sharing and uniform scattering increase the survival rate (the recovery ability) of a tampered block. Notably, the proposed method uses VQ-index as the recovery data. For an image block, the matched codeword is more informative to show the texture of the block than the mean value or the halftone result used in some other publications. Although the size of a VQ-index might be larger than that of the mean value or the half-tone representation of the block, we will use ($r$, $n$) sharing to shorten it. (Each share is $1/r$ times the length of the VQ-index being shared). Hence, unlike published recovery methods, the proposed method allows many backup shares ($n$ shares) for each block without significantly increasing the total size of recovery data. Since the proposed method has more backup pieces ($n$ shares rather than 1 or 2 copies), we can use a hybrid two-layer strategy to scatter the backup (the $n$ created index shares) all over the image in a uniformly and not-too-sparse manner. (This two-layer strategy is described in detail in Sec. 4.3.3) With this strategy, the recovery data can be uniformly scattered throughout the whole image to resist a cropping attack of an extensive area. Finally, since the ($r$, $n$) sharing being used is a threshold system, if the image is tampered in certain area, recovering a tampered block $A$ only requires that the remaining (non-tampered) areas contain at least $r$ of the $n$ scattered shares for $A$. This missing-allowable property ($n - r$ shares can be missed), together with the uniformly-scattered distribution of the $n$ shares, increase the survival rate of the recovery data. As for previous publications, they use a backup approach to store the recovery data directly. In such approaches, if a block and its single or double backup blocks are tampered at the same time (e.g. in a cropping attack of an extensive area), then the recovery data of this block is lost completely.

### 4.3.2 Generation of the Authentication Data

After generating the recovery data, the next work is to generate the authentication data to verify the image's integrity. The authentication data of each block consists of some important information about the block, including the MSBs of the pixels within the block and the block's position, as well as some private information, such as the image-owner's secret key, and the image identification number, width, and length.

The watermark in the proposed scheme is hidden in the $t$ LSB planes of the image, and the remaining $(8-t)$ MSBs of each pixel within the block are used in the generation procedure of authentication data. The parameter $t < 8$ must satisfy

$$\left\lceil \frac{\lceil (log_2 L)/r \rceil \times n}{h \times w} \right\rceil < t < 8. \tag{4.2}$$

This is because if each block has $h{\times}w$ pixels, then $t$ LSB planes together provide $h{\times}w{\times}t$ bits to hide the recovery data of $n$ shares, of which each share has $\lceil (log_2 L)/r \rceil$ bits. Of course, the authentication data naturally occupies additional bits. The number of additional bits should be added to the numerator of Eq. (4.2) if the "< $t$" in Eq. (4.2) is to be replaced by "=$t$". According to the decided parameter $t$, the image is transformed into the $t$-LSB-zeroed image by setting all $t$ LSBs of each pixel in the image to zero. Here, we design a modulus-based zeroing scheme to obtain the $t$-LSB-zeroed image rather than directly setting all $t$ LSBs of each pixel of the original image to zero, and the details are described in Sec. 5.2.1. Notably, if the parameter value of $t$ is increased, then the hiding capacity of each block (to embed recovery and authentication data) rises, leading to stronger recovery and authentication ability. However, the quality of the watermarked image worsens if the value of $t$ increases.

The $t$-LSB-zeroed image is used to generate the authentication data for verification, as shown in Fig. 4.1(a). Consider a cryptographic hash function

$$H(S_I) = (d_1, d_2, \ldots, d_u), \tag{4.3}$$

where $S_I$ is an input bit string of arbitrary length; $d_i$ is the output binary bits of the hash function for $1 \le i \le u$, and $u$ is the length of the output bit string. Wong and Memon [69] concluded that a cryptographic hash function should have the property that once an input bit string $S_I$ and its corresponding output $(d_1, d_2, \ldots, d_u)$ are given, then finding another input bit string of any length that will be hashed to the same output $(d_1, d_2, \ldots, d_u)$ should be computationally infeasible. For example, MD5 [8] is a famous collision-resistant one-way hash function, in which any input bit string is hashed into a bit stream of 128 bits, i.e., $u = 128$. Here, the proposed method employs MD5 as the hash function, and produces a single output 128-bit message (the output hash value) from the input message. Of course, other cryptographic hash functions can also be utilized to replace MD5.

Let *ID* be the identification number of an image, and *SK* denote a secret key of the image-owner. The *SK* can be either constant or dynamic, where dynamic means changing randomly or according to the image *ID*. Consider the $k^{\text{th}}$ block of an image ($k = 1, 2, \cdots, \dfrac{H \times W}{h \times w}$ if the image size is given as $H \times W$). The gray value of the $h \times w$ pixels of the block (computed using the $(8 - t)$ MSBs only) are denoted by $\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_{h\times w}^k$. The authentication data of the $k^{\text{th}}$ block can be computed as

$$H(ID \,\|\, SK \,\|\, H \,\|\, W \,\|\, k \,\|\, \hat{y}_1^k \,\|\, \hat{y}_2^k \cdots \|\, \hat{y}_{h\times w}^k) = (d_1^k, d_2^k, \ldots, d_u^k), \tag{4.4}$$

where the symbol || is the concatenation of all input streams. Notably, according to the discussion in Ref. [69], the image identification number *ID* and the block sequence number *k* are two important parameters for resisting the collage attack [88].

Finally, because each pixel within a block only has *t* LSBs to be embedded, each block might have insufficient embedding space for embedding the recovery data and the 128-bit authentication data. In this case, the bit stream of the authentication data need to be folded, that is, converted into another shorter bit stream. In the folding procedure, the length of the embedding authentication data $\delta$ is determined by

$$\delta = h \times w \times t - \left\lceil \frac{\log_2 L}{r} \right\rceil \times n. \tag{4.5}$$

The original bit stream generated by Eq. (4.4) is then divided equally into several sections of $\delta$ bits each, and folded into $\delta$-bit embedding authentication data with an exclusive-or operation. For example, a bit stream "10100111" can be folded into "1101", since 1101 = 1010$\oplus$0111. In this case, $\delta = 4$; "1010" is the first section of the original bit stream, and "0111" is the last section of the original bit stream. If the last section does not have enough bits, then it is padded with "0" to make it convenient to fold.

### 4.3.3 The Embedding Procedure

Finally, as shown in Fig. 4.1(a), the recovery and the authentication data of a given image are combined to form the embedding watermark. This section describes the procedure that embeds the watermark. First, the image is divided equally into ($n$+1) non-overlapping big regions, where $n$ is a number not less than the threshold value $r$. The block addresses {1, 2, 3,…} are assigned to the blocks of each region according to the raster scan order, i.e. from left to right and top to bottom. The $n$ index shares of each block are then embedded respectively into the blocks, which are in the corresponding positions of the other $n$ regions. For example, assuming that $r$ = 2 and $n$ = 3, then, as shown in Fig. 4.2, the image is divided equally into $n$+1 = 3+1 non-overlapping regions, and numbers are assigned sequentially to the blocks of each region by raster scan order. According to Section 4.3.1, the VQ index value of the first block in Region I is shared among $n$ = 3 index shares, which are embedded into the first block in the remaining $n$ = 3 regions (Regions II, III, and IV), respectively.

Likewise, the three index shares of the first block in the Region II are embedded into the first block in the remaining $n$ = 3 regions (Regions III, IV, and I), respectively. Similar action is performed for each block in each region.

On the other hand, each block has its own authentication data for verification purpose, according to Section 4.3.2. The authentication data of a block are embedded into the block itself. Therefore, the data to be embedded into each block include its own authentication data and the $n$ index shares obtained from the corresponding block of the remaining $n$ regions (one share per region). To simplify the embedding, the concatenation operation is used to combine these $n$ index shares and the authentication data to form a watermark for the block. The watermark is then transformed into a binary stream, and is embedded into the $t$ LSBs of the pixels in the block. To increase security level, the MT pseudo-random number generator [9] may also be integrated into the embedding procedure to confuse the embedding positions of the binary stream, as illustrated below using the aforementioned example ($r$ = 2 and $n$ = 3). Assume that the embedding watermark has 27 bits, which consist of $n$ = 3 index shares ($b_0 - b_4$, $b_5 - b_9$, and $b_{10} - b_{14}$) and a set of local authentication data ($b_{15} - b_{26}$). This watermark is embedded into the $t$=3 LSBs of the block with a size of 3×3. Figure 4.3(a) shows the positions of the 3×3 = 9 pixels in the block. The 3 LSBs of the nine pixels in the block provide a hiding space of 27 bits, which are assigned position numbers 0–26, as shown in Fig. 4.3(b). Inputting a randomly chosen 32-bit natural number, called the “seed”, into the MT pseudo-random number generator [9], generates a series of pseudo-random real numbers $R_j$ ($j = 0,1,2,\ldots,2^{19937} - 1$), which are uniformly distributed on the interval [0, 1]. After a simple mapping (the detail is omitted), we get a pseudo-random sequence whose elements are integers from 0 to 26. The embedding binary stream is then embedded into the corresponding locations in the block according to the pseudo-random integer sequence. For example, if the generated pseudo-random embedding sequence for 0–26 is {21, 4, 26, 13, 6, 9, 0, 25, 13, 14, 22, 3, 16, 2, 17, 18, 5, 19, 7, 20, 8, 11, 1, 23, 15, 24, 10}, then the bit $b_0$ of the binary stream watermark is embedded in the bit-position marked as No. 21 in Fig. 4.3(b), and the bit $b_1$ is embedded in the bit-position marked as No. 4 in Fig. 4.3(b), and similarly along the entire sequence. Figure 4.3(c) shows the result of embedding the 27-bit watermark *b* in this example.

The watermarked image is obtained by performing the embedding procedure block by block until the entire image is processed. The seed used in the MT pseudo-random number generator [9] can be the secret key (*SK*) in Eq. (4.4), and only the legal users who own the same secret key can obtain the same pseudo-random embedding sequence to verify correctly the authenticity and integrity of the image.

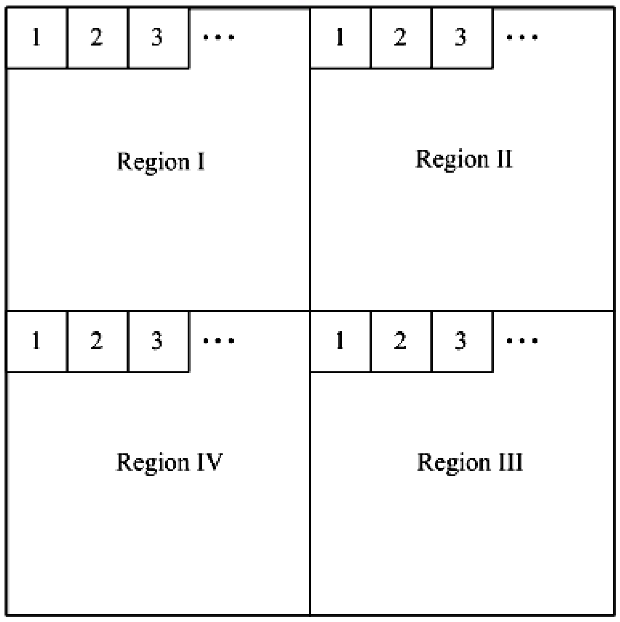


Fig. 4.2. An example of region division and number assignment.

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

(a)

| pixel | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| bit 7 | | | | | | | | | |
| 6 | | | | | | | | | |
| 5 | | | | | | | | | |
| 4 | | | | | | | | | |
| 3 | | | | | | | | | |
| 2 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 16 |
| 1 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

(b)

| pixel | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| bit 7 | | | | | | | | | |
| 6 | | | | | | | | | |
| 5 | | | | | | | | | |
| 4 | | | | | | | | | |
| 3 | | | | | | | | | |
| 2 | $b_{15}$ | $b_{17}$ | $b_{19}$ | $b_0$ | $b_{10}$ | $b_{23}$ | $b_{25}$ | $b_7$ | $b_2$ |
| 1 | $b_5$ | $b_{26}$ | $b_{21}$ | $b_8$ | $b_3$ | $b_9$ | $b_{24}$ | $b_{12}$ | $b_{14}$ |
| 0 | $b_6$ | $b_{22}$ | $b_{13}$ | $b_{11}$ | $b_1$ | $b_{16}$ | $b_4$ | $b_{18}$ | $b_{20}$ |

(c)

Fig. 4.3. An example of embedding a 27-bit watermark *b* in a 3×3 block: (a) the arrangement of the 9 pixels; (b) the arrangement for the 3-LSBs of the 9 pixels; (c) the result of embedding.

## 4.4 The Proposed Method (Decoding)

### 4.4.1 Verification

In the verification phase, the query image is first divided into non-overlapping blocks of $h \times w$ pixels, and the verification is performed block by block until all blocks are processed. The watermark of each query block is extracted from the $t$ LSBs of each of its $h \times w$ pixels. Notably, only legal users know (the codebook and) the parameters used earlier in the watermark embedding procedure. Legal users can obtain the previously used pseudo-random embedding sequence from the secret key $SK$, and thus obtain the authentication data and recovery data from the extracted watermark, according to the locations corresponding to the obtained sequence. The authentication data "extracted" this way using the $t$ LSBs of the query block's pixels should be the same as the authentication data "re-computed" directly from the $(8-t)$ MSBs of the block's pixels, with the re-computation as described in the two paragraphs containing Eqs. (4.4) and (4.5). A block in which the extracted authentication data coincide with the re-computed authentication data is called an authentic block; otherwise, it is regarded as a tampered block. An authentic block is one that almost certainly has no tampering occurs, and the recovery data embedded in this block are trustworthy for reconstructing other blocks in the recovery phase, if other blocks are tampered. In contrast, a block that fails the authentication test is regarded as a tampered block, and the back-up content stored in it should never be trusted; i.e., should not be used to reconstruct other blocks in the later recovery phase.

Because the hiding capacity is not enough to hide the whole hashed sequence, a folding procedure is performed to shorten the hashed sequence. This procedure is a many-to-one mapping ($2^{128}$ to $2^{\delta}$), leading to the possibility that two hashed sequences have the same folded sequence. (A smaller $\delta$ value leads to a higher collision probability.) However, experiments results show that the folded sequence can locate almost all tampered places when $\delta$=11 bits (see Figs. 4.6(d)-(f) and 4.7(d)-(f)).

Moreover, to reduce the probability of misdetection resulting from collision, a hierarchical check system can be used to locate the tampered area. In other words, each authentic block can be checked again after performing the verification procedure described in Sec. 4.4.1. If the eight neighboring blocks of an authentic block are marked as tampered blocks, then the block status can be changed from authentic to tampered, because a collision might have occurred in the block. This block is then recovered by recovery data, which can still yield an acceptable look for this block, even if no collision occurred.

### 4.4.2 Recovery of Tampered Area

The recovery phase starts if any tampered block was found by the block authentication step. For a tampered block, we collect its back-up data, which are the $n$ VQ-index-value shares embedded in the $n$ corresponding blocks of other $n$ regions (one share per region, see Sec. 4.3.3), to reconstruct the VQ index value of the block. Of course, some of these $n$ back-up blocks might also be marked as "tampered" after the authentication test in Sec. 4.4.1. These tampered, and hence useless, back-up blocks are simply skipped. The VQ index value of the tampered block can be traced back by inverse sharing if at least $r$ (out of the $n$) VQ-index-value shares survive. The tampered block can then be rebuilt by this VQ index value and the codebook.

## 4.5 Experimental Results

This section presents the experimental results and comparison to demonstrate the performance and feasibility of the proposed method. All images were 8-bit gray-valued, and the VQ codebook was generated by the LBG algorithm [14]. Each block had 4×4 pixels, and a set of 512×512 standard gray-value images was used as the training set to generate the VQ codebook. The ($r$=2, $n$=3) threshold sharing scheme was used to share the VQ-index values. The image was divided equally into $n$+1 = 4 non-overlapping regions according to Sec. 4.3.3, and the parameter $t$ became $t$=2 by Eq. (4.2) when $L$ = 16,384 and $h$×$w$ = 4×4, which means that the watermark was embedded in the $t$=2 Least-Significant-Bits of each pixel of the image.

Figure 4.4 shows the four original images "Lena", "Jet", "Baboon", and "Barbara". Each image was 512×512. Figure 4.5 shows their watermarked version, after executing the proposed watermarking method described in Sec. 4.3. The qualities of all watermarked images and recovery images were measured by the Peak-Signal-to-Noise Ratio (PSNR), defined as

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{\text{MSE}}, \tag{4.6}$$

in which MSE denotes the Mean Square Error between the pixel values of the original and of the watermarked images. From Fig. 4.5, we can see that the qualities of the watermarked images are acceptable. Their PSNR values are all greater than 44.13 dB. The images in Figs. 4.4(a–d) and Figs. 4.5(a–d) are visually indistinguishable using naked eyes. In other words, the proposed watermarking method has transparency property (the watermark is perceptually invisible). As for the processing time, the average encoding and decoding time of our method are shown in Tables 4.3. The decoding time includes both detection and recovery phases.

Experiments were performed to tamper with the watermarked images shown in Fig. 4.5 to measure the performance of the proposed method in tampering detection, location, and recovery.

## A. Cropping attacks

A part of each watermarked image was cropped. Figure 4.6(a) shows the watermarked image Lena$^{(W)}$ with 25% of the image cropped. Figures 4.6(b) and 4.6(c) show the same image with 50% of the image cropped. The cropped parts were detected by the verification procedure described in Section 4.4.1, as shown in Fig. 4.6(d–f), where the black blocks are the detected tampered blocks, and the white blocks are the detected authentic blocks. Finally, Figs. 4.6(g–i) show the recovery results obtained by applying the recovery techniques in Section 4.4.2 on the images Figs. 4.6(a–c), respectively (using the white blocks in Figs. 4.6 (d–f) to recover the black blocks). The PSNR values between the recovery images in Figs. 4.6(g–i) and the original images in Figs. 4.6(a–c) were 40.04 dB, 34.77 dB, and 36.54 dB, respectively. In summary, the cropped portions were correctly detected and located, and the cropped portions were recovered although the cropped portions occupied a big area of the watermarked image in both Fig. 4.6(b) and Fig. 4.6(c).

## B. Collage attacks

As reviewed in Sec. 1.2.2, a block-based watermarking scheme may be vulnerable to a collage attack [88]. Therefore, an experiment was performed to verify the effectiveness of the proposed scheme against a collage attack. To perform collage attack, several authentic blocks were collected from a set of watermarked images to form a “forgery” codebook as used in collage attack. Figure 4.7(a) shows a tampered watermarked image “Lena$^{(W)}$” (Fig. 4.5(a)), in which a flower (formed of blocks chosen from forgery codebook, i.e. blocks chosen from another watermarked image) is inserted into her hat. Similarly, Fig. 4.7(b) shows a tampered watermarked image “Jet$^{(W)}$” (Fig. 4.5(b)), in which the country name “U.S.” and the emblem printed on the plane are replaced respectively by the country name “R.O.C.” and another emblem. Finally, Fig. 4.7(c) shows a tampered watermarked image “Barbara$^{(W)}$” (Fig. 4.5(d)), in which a part of the book-shelf in the top-left corner is removed by copying some blocks of the same watermarked image. Figures 4.7(d), (e), and (f) respectively show the detection results of the tampered images, and Figs. 4.7(g), (h), and (i) respectively show the recovery results, of which the PSNR values were 41.58 dB, 42.45 dB, and 43.01 dB, respectively. The experiment results indicate that the proposed method accurately detected and located the replaced parts, and recovered back the replaced parts with acceptable quality. Additionally, inserting the block index $k$ and the image identification $ID$ in the authentication data helped resist collage attack [88], as suggested by Wong and Memon [69].

## 4.6 Discussions

### 4.6.1 Modified Version

A security-enhanced approach can be used to increase the security level. In this new model, the image is divided equally into at least $n$+1 non-overlapping regions. Then, for each block, the block’s $n$ VQ-index-value shares are embedded into $n$ blocks chosen from $n$ of the remaining regions (one corresponding block per mentioned region). However, the positions of the corresponding blocks in each region can be randomized and made distinct among regions. Therefore, the MT pseudo-random number generator [9] could be applied to perform this rearrangement.

### 4.6.2 Comparison

Because the proposed scheme is a watermarking-based approach with detecting-locating and recovery abilities of tampered areas, Table 4.1 compares it with other watermarking-based methods that have similar functions.

The data in Table 4.1 indicate that the proposed scheme is competitive. The qualities of our watermarked images are competitive to the counterparts obtained in those elegant methods reported in literature. Because the PSNR values of the watermarked images in Ref. [72] and the recovered images in Refs. [72, 70] are not shown in the published papers [72, 70], they are denoted by the symbol "N/A (not available)". Although the PSNR value of the watermarked images of Ref. [72] is not shown in their published paper, the value must be very high, since the halftone information is embedded by 1-LSB (in the worst case, even if the LSB bit of each pixel is altered, the PSNR was still higher than 48 dB). Even so, their recovered images were still slightly worse than those obtained by the proposed method, as shown in Fig. 4.8 (where Figs. 4.8(b) and 4.8(e) are from page 168 of Ref. [72] and page 157 of Ref. [70], respectively). Additionally, Figs. 4.8(g–h) show the experimental result of Ref. [73]. Clearly, the lost upper 25% recovered by Ref. [73] (Fig. 4.8(h)) contains more distortion than that of the proposed method (Fig. 4.8(i)).

To measure the performance of tampering recovery, the quality of the recovered images were compared with those published methods when the same area was cropped from the watermarked images of both sides. From Table 4.1 and Fig. 4.8, we can see that the quality of the recovered image in the proposed method is better than those of the published methods. This is because their schemes can recover the tampered parts of the protected image by using the recovery data, which is often embedded in blocks of other areas of the same image. Therefore, a block and its backup block may both be tampered at the same time if a watermarked image is tampered extensively in a large area and randomly. In this case, the recovery ability of the tampered block in many approaches is gone. In contrast to the aforementioned approach (which uses a backup block to store the recover data), the recovery data of a block in the proposed method are the *n* "shares" of the block's VQ-index value, which are embedded into blocks located in $n$ other non-overlapping regions of the image. (Notably, instead of directly embedding the recovery data, the proposed method shares the data and spreads the $n$ shares throughout the whole image. The scattering of the $n$ shares in the whole image reduces the chance that all VQ-index-value shares are lost when a connected portion of the image is damaged.) With sharing (and the scattering in the whole image), the tampered query-block can still be recovered even if only $r$ out of the $n$ back-up blocks of a tampered query-block survive in an attack. This increases the recovery rate of the proposed method, as compared with those methods in which each query-block only has one back-up block. Thus, the proposed scheme can handle the case in which a large area is tampered even if 50% of the watermarked image is cropped, as long as $r$ out of the $n$ back-up blocks are authentic.

To compare with published methods, Table 4.2 shows the compression ratio, amount of the recovery data, and preservation system of each method. The compression ratio is the ratio of the recovery data size to the original data size. For a 256×256 host image, because Ref. [71] stores the 6-bit mean value of each 2×2 block, its compression ratio is $\frac{2\times2\times8}{6}$ = 5.33 : 1 = 5.33. Notably, Ref. [71] uses only one backup to preserve the recovery data, and thus has $\frac{256\times256}{2\times2}\times6$ = 98,304 bits of recovery data, which are embedded in the 2-LSB of the watermarked image. (The recovery data of a block is preserved in another block whose address is determined by a permutation function.) Consequently, the block and its backup block could be tampered simultaneously if the watermarked image is tampered extensively in a large area and randomly. In this case, the recovery of the block fails. Lin et al. [71] found that the recovery rate was not higher than 94% when the cropping area occupies 50% of the watermarked image. In contrast, our method had a recovery rate of 100%. In Ref. [73], Wang and Tsai preserved the recovery data of ROI (regions of importance), and embedded them in the host image. In their experiments, because the size of the so-called range block was 4×4, and the codes of each range block were 31 bits, the compression ratio is $\frac{4\times4\times8}{31}$=4.13. Their scheme also uses a backup approach to preserve the recovery data. Hence, if the range of ROI is about 60% of the host image, then $\frac{256\times256}{4\times4}\times31\times0.6$ = 76,186 bits are embedded in the 256×256 watermarked image. The recovery ability is good in the ROI, but slightly worse in the non-ROI area, because the tampered non-ROI area is recovered using image-inpainting technique. Luo et al. [72] used a binary halftone image with the same size (512×512) as the gray-valued host image as the recovery data, thus yielding a compression ratio of 8. Therefore, their recovery data have 512×512×1= 262,144 bits. Because their method has a small amount of embedding data, its watermarked image quality is good, but the recovery ability is not good enough to resist cropping attack of an extensive area (see Fig. 4.8(b)). Wu and Chang's method [70] backs up three copies of the recovery data, which are the results of edge detection (one bit per group of 4×4 pixels). The compression ratio is $\frac{4\times4\times8}{1}$=128, and recovery data size is $\frac{512\times512}{4\times4}\times3$= 49,152 bits. The advantage of Ref. [70] is that the resulting image is a JPEG-compressed image, and can be used directly in transmission. However, the recovered image is slightly worse than the proposed method because the recovery data are the result of edge detection (see Fig. 4.8(e)). Yeh and Lee [74] proposed content-based fragile watermarking to recover tampered areas of an image with JPEG-compressed quality. They categorized each block as smooth, mid-textured, or textured type, and the recovery data of the smooth and mid-textured blocks were replicated to raise the recovery ability. The relationship between a block and its backup block was by an automorphism. The compression ratio is $\frac{8\times8\times8}{56}$=9.14, and recovery data size is

$\frac{512\times512}{8\times8}\times(56\times2)=458{,}752$ bits.

Our experiments used a codebook with 16,384 codewords to encode each 4×4 block, and hence the index value of each block had $(log_2 16384)=14$ bits. The compression rate was $\frac{4\times4\times8}{\log_2 16384}=9.14$. After ($r$=2, $n$=3) threshold sharing, each block's VQ-index had three index shares, so each share had $\lceil(log_2 L)/r\rceil=\lceil(log_2 16384)/2\rceil=7$ bits. The total amount of recovery data was $\frac{512\times512}{4\times4}\times7\times3=$ 344,064 bits for a 512×512 image (or $\frac{256\times256}{4\times4}\times7\times3=$ 86,016 bits for a 256×256 image).

In general, the recovery ability is related to the amount of the recovery data embedded in the protected image. In case of 256×256 host images, the total amount of recovery data in our method was 86,016 bits after (2, 3) threshold sharing (57,344 bits before sharing); and was 98,304 and 76,186 bits in Ref. [71] and Ref. [53], respectively. Although our method did not have the largest data size, its recovery ability was most competitive. This is because the ($r$, $n$) threshold sharing (Sec 4.3.1) and the widely scatter manner (Sec 4.3.3) were used to scatter the recovery data in a distributed and missing-allowable manner.

In case of 512×512 host images, our total amount of recovery data was 344,064 bits after sharing (229,376 bits before sharing), while Ref. [72] and Ref. [70] were 262,144 and 49,152 bits, respectively. The proposed method had a larger data size than Refs. [72] and [70], so it is not surprising that our recovery quality (see Fig. 4.8(c) and (f)) is better than Refs. [72] and [70] (see Figs. 4.8(b) and (f)). Nonetheless, because the ($r$, $n$) threshold sharing was used to distribute the more-detailed recovery data, the total size of all $n$ shares for a VQ-index was only $n\times\lceil(log_2 L)/r\rceil$ bits (3×7=21 bits), rather than $n$×$L$ (=3×14= 42) bits as used in the traditional approach, in which $n$=3 copies of the recovery data are made directly. This is why the watermarked image of our method still has a good PSNR (44.1 dB) quality, after embedding the detailed recovery data.

Some limitations of the proposed scheme are: i) The codebook used in the VQ compression procedure must be available during tamper detection and recovery. But this codebook can be public, so it is not a big problem. ii) Due to the collision of the hashed sequence, a non-secure pass (in which a tampered block is treated as authentic block) might occur. A hierarchical check system (see the ending paragraph of Sec. 4.4.1) can avoid this problem, but it creates the chance of false alarm (in which a non-tampered block is considered as a tampered block). Nonetheless, the mis-alarmed block is replaced by its recovery data, which can still yield an acceptable look for this block, since our recovery data cover enough information about the block.

### 4.6.3 Security Analysis of the Recovery Data

If an image block is marked as tampered, then the recovery data embedded in it can no longer be used for recovery. Let us discuss the situation when an attack damages an extensive area of the watermarked image. It suffices to use Figs. 4.8(a) or 4.8(g), where the damaged area is one horizontal quarter of the image, as an example to show how to analyze the recovery rate of the proposed method. The $(r, n)$ threshold is (2, 3) in Fig 4.8; hence, to recover a block $B$ in the damaged area needs any $r$=2 of its $n$=3 index shares distributed in other three regions (i.e. Regions II, III, and IV, if block $B$ is in Region I).

The recovery rate of the proposed method is 100% for Figs. 4.8(a) or 4.8(g), because two of the four regions {I-IV}are never touched (see the four regions shown Fig. 4.2), which means that two authentic shares required to rebuild block $B$ can always be obtained from these two untouched regions. Therefore, some attackers might try to scratch an area across all Regions I-IV, but they can only achieve this when they know how the image was partitioned into ($n$+1) regions. (An image can be partitioned into four regions in many ways besides the one shown in Fig. 4.2; for instance, Region I could be a long vertical bar in the leftmost quarter, with Regions II, III, and IV obtained by dividing the remaining area into three horizontal bars. Notably, the number of possible partitions increases significantly if each region itself is allowed to be an unconnected set.) To continue the analysis, still assume the partition of regions is as in Fig. 4.2.

Then, even if the scratched horizontal quarter belt (the dark area in Fig. 4.9(a)) happens to touch all four regions; a tampered block $B$ can still be recovered as long as $r$=2 of the $n$=3 support (recovery) shares are in the remaining 75% of the image (i.e. the white part in Fig. 4.9(a)). The recovery rates are distinct between the lower pixels (intersection of dark area with Regions III and IV) and the upper pixels (intersection of dark area with Regions I and II) of the quarter belt. More specifically, the recovery rate is

$$P_L = \frac{1}{(256)^3}\left[(256)^3 - (256)^2 x + 512x^2 - 2x^3\right] \tag{4.7}$$

for each lower pixel; and

$$P_U = \frac{1}{256^3}\left[\frac{3}{4}(256)^3 + \frac{1}{2}(256)^2 x - 256x^2 + 2x^3\right] \tag{4.8}$$

for each upper pixel. For the dark quarter-belt shown in Fig. 4.9(a), there are 512/4=128 rows of pixels. Among them, $x$ rows of the dark pixels are in the upper plane, and (128−$x$) rows are in the lower plane. Therefore, in Fig. 4.9(a), the recovery rate for a dark-area pixel is

$$\frac{128-x}{128}P_L + \frac{x}{128}P_U = \frac{1}{(256)^3}\left[(256)^3 - \frac{3}{2}(256)^2 x + 1280x^2 - 8x^3 + \frac{1}{32}x^4\right] \tag{4.9}$$

after certain evaluations. Now, as the value of $x$ varies from 0 through 128, the average probability to recover a dark area pixel is the integration of the above equation on the range from $x$=0 to $x$=128, followed by a division over the range length

128－0=128, which is

$$\frac{1}{128}\times\frac{1}{(256)^3}\left\{(256)^3 x-49152x^2+\frac{1280}{3}x^3-2x^4+\frac{1}{160}x^5\right\}\Bigg|_{x=0}^{x=128}=89.\overline{16}\%. \quad (4.10)$$

Finally, consider that there are 1/3= 33.3% (or 2/3=66.6%) chances that the quarter belt will (will not) cross all four regions {I-IV}, the recovery rate of a tampered pixel for the quarter-belt tamper is

$$(\frac{2}{3}\times 100\%)+(\frac{1}{3}\times 89.\overline{16}\%)=96.39\%. \quad (4.11)$$

To see why there exists 1/3= 33.3% chance that the quarter belt will cross all four regions {I-IV}, we can use Fig. 4.9(b) to explain. It is obvious that if the dark horizontal belt gradually goes through the whole 512×512 image from bottom to top, with the constraint that the whole belt must stay in the image, then the upper boundary of the 128-rows dark belt goes from the y-coordinate value 127 through 511 (assuming that the lowest line of the image has coordinate y=0). Obviously, the quarter belt will touch all four regions {I-IV} when the dark belt's top row hits the 512×512 image's y=256 line (or y=257, or y=258, …., or y=382 or y=256+128－1=383). Therefore, the probability that all four regions {I-IV} are touched by the belt is $\frac{383-256+1}{511-127+1}\approx\frac{1}{3}$.

Below we discuss how we got the formulas for $P_L$ and $P_U$ above. Without the loss of generality, we only prove the formula for $P_L$. Because of the symmetry (left vs. right), we may assume that the damaged block *B* is in Region IV, as shown in Fig. 4.9(a). (The recovery rate for the case that block *B* is in Region III is identical to the recovery rate for the case that *B* is in Region IV.) Now, as stated earlier, block *B* can be recovered as long as any *r*=2 of its *n*=3 supporting shares (stored in Regions {I, II, III}, respectively) are in the white part of Fig. 4.9(a). Now, according to the sequence I, II, III, there are four sub-cases this requirement is satisfied. They are: (Dark, White, White), (White, Dark, White), (White, White, Dark), and (White, White, White). For example, (Dark, White, White) means that the supporting share in Region I is in dark area, but the supporting shares in Region II and III are both in white area. The probabilities for these four cases are ($\frac{x}{256}\times\frac{256-x}{256}\times\frac{128+x}{256}$), ($\frac{256-x}{256}\times\frac{x}{256}\times\frac{128+x}{256}$), ($\frac{256-x}{256}\times\frac{256-x}{256}\times\frac{128-x}{256}$), and ($\frac{256-x}{256}\times\frac{256-x}{256}\times\frac{128+x}{256}$), respectively. Summing up these four terms, we get

$$
\begin{aligned}
P_L &= 2\left(\frac{x}{256}\times\frac{256-x}{256}\times\frac{128+x}{256}\right)+\left[\left(\frac{256-x}{256}\times\frac{256-x}{256}\right)\times\left(\frac{128-x}{256}+\frac{128+x}{256}\right)\right] \\
&= \left(\frac{256x+2x^2}{256^2}+\frac{256^2-256x}{256^2}\right)\times\frac{256-x}{256} \\
&= \left(\frac{2x^2+256^2}{256^2}\right)\times\frac{256-x}{256} \\
&= \frac{1}{(256)^3}\left[(256)^3-(256)^2x+512x^2-2x^3\right]
\end{aligned}
$$

The analysis for $P_U$ is similar. Notably, increasing the value of $n$ increases the recovery rate for a fixed $r$, but reduces the PSNR quality of the watermarked image.

### 4.6.4 Recovery in Case of Multiple Users

The method described in Sections 4.3 and 4.4 can be regarded as a recovery work for a single user who owns the image. The recovery data of the image is uniformly embedded into the protected image itself. A single user can perform the recovery work alone.

On the other hand, the case of multiple users can also be considered. For example, consider a case where a team of four members have to produce a new product according to its blueprint. First, divide the blueprint into ($n$+1=3+1=4) non-overlapping regions as shown in Fig. 4.2, and each member holds one region. Then, for each 4×4 block of Region I, use ($r$=2, $n$=3) sharing to generate $n$=3 index shares (recovery data) and embed, respectively, the three index shares in other regions (i.e. Regions II, III, and IV). Perform the analogous process for each block of each region. By this, each member knows nothing about the complete look of the new product, and thus cannot leak the information to a competitor. On the other hand, if one of the four members is absent in the team meeting, then the region held by the absent member can still be reconstructed through the mutual support of other members. The recovery procedure is similar to Sec. 4.4.2. In fact, since *r*=2, any two of the four members (for example, members I and II) can meet together to handle the team work by roughly reconstructing the other two quarter-images owned by the other two team members (III and IV) in order to review the approximated look of the whole product.

The compression ratio of the recovery data in this multiple-users case is $\frac{4\times4\times8}{\log_2 16384}$ = 9.14. If one member transmits the recovery data of the absent region to another member, for instance, in the previous example (*r*=2, *n*=3), if member I wants to transmit data to member II in order to help member II to construct Region III), then the amount of transmitted data is $(\frac{512\times512}{4\times4}\times7)\times\frac{1}{4}$ = 28,672 bits for a 512×512 blueprint. Of course, this amount is doubled if member I wants to help member II to build up both Regions III and IV. On the other hand, each member holds the recovery data of other three regions, which are embedded in the held region, and the total data size is $[(\frac{512\times512}{4\times4}\times7)\times\frac{1}{4}]\times3$ = 86,016 bits. Anyway, we can increase the application of the proposed method by the mutual cooperation of multiple users. In the above (*r*=2, *n*=3) case, the recovery work is performed by any two team members instead of a single user.

## 4.7 Conclusions

The chapter proposes a watermarking method for image authentication, and it is with good self-recovery ability. The proposed method has the following functions: 1) detecting whether the watermarked image is tampered; 2) indicating the locations of the tampered area; 3) self-recovering the tampered portion using the non-tampered portion of the same watermarked image; and 4) enhancing the recovery ability by utilizing (*r*, *n*) threshold sharing [90], followed by scattering the shares all over the image.

Feature (4) above gives the proposed method a good recovery rate. The sharing polynomial of Thien and Lin's method [90], which was devised to share a secret image among several participants, is used to reduce the amount of recovery data without significantly degrading the visual quality of the watermarked image. Experimental results (Figs. 4.5–4.8) and the comparison tables 4.1 and 4.2 show that the proposed method is competitive.

In conclusion, the proposed method has the following novelty compared with previously published schemes (particularly, image authentication or recovery methods [66-73] and image sharing methods [3-4, 89-92, 94, 96]):

i) The recovery data are generated by using VQ compression technique (an index file). A VQ index file has at least three advantages. i-a) The matched codeword of an image block is more suitable for showing the texture of the block than the mean value or the halftone result, as used in some other publications. i-b) VQ compression technique is block-based, and a block-based approach is sufficiently easy to apply for tampering recovery. i-c) VQ decompression is simple and has a very short decoding time, thus reducing the reconstruction cost.

ii) To increase the survival rate of the recovery data (VQ-index file), a modified version of the ($r$, $n$) threshold sharing [90] is used to generate $n$ index shares. Some remarks about this are given below.

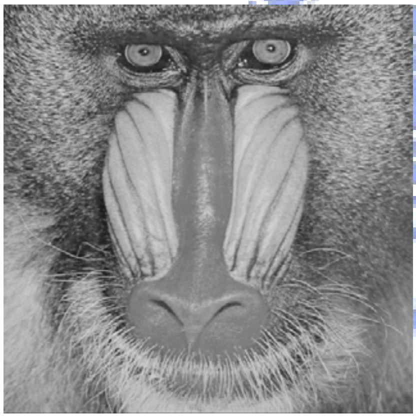
ii-a) When applying secret sharing techniques to images, people often generated $n$ shares for each pixel (or for each block of the pixels) of the secret image. Hence, the share value has a wide range, and each share is represented by $m$ bits, where $m$ is the number of bits per pixel. For example, if each pixel is represented by 8 bits, then each share needs 8 bits. However, the proposed method assumes that the codebook has $L$ codewords, and divides each VQ-index value into $r$ sections before generating $n$ index shares for each index value. Thus, the size of each index share is reduced to $\lceil (log_2 L)/r \rceil$, which is usually smaller than 8 (for example, $\lceil (log_2 L)/r \rceil$=2 bits if ($L$=1024, $r$=5), or 3 bits if ($L$=4096, $r$=4)). The smaller size of each share helps maintain the quality of the watermarked image in the embedding process.

ii-b) All arithmetic calculations in the proposed method are performed in the power-of-two Galois Field GF($2^{\lceil (log_2 L)/r \rceil}$), rather than the Mod$_{251}$ used in Thien and Lin's image sharing [90]. This modification not only reduces the number of bits of each index share from $\lceil log_2 251 \rceil = 8$ bits to $\lceil (log_2 L)/r \rceil$ bits, but also makes the proposed method more suitable for the various codebook sizes $L$, thus increasing flexibility.

iii) Many published image recovery techniques embed the recovery data of a unit (i.e. a block or a pixel) for backuping into another block or pixel according to a permutation function. (Notably, having many copies of the recovery data might increase survival rate, while decreasing the quality of the watermarked image.) The proposed method is sharing-based, and each share is small in size. Hence, unlike published recovery methods, the proposed method allows many backup shares (*n* shares) without significantly increasing the total size of recovery data. Since the proposed method has more backup pieces (*n* shares rather than 1 or 2 copies), it can use a hybrid two-layer strategy to scatter the backup (the *n* index shares created to backup a VQ index value). More specifically, the host image is divided into at least (*n*+1) non-overlapping regions, and the *n* index shares of each block are respectively embedded into the blocks of *n* other regions. After this layer of *n*-to-*n* random mapping, the position of the corresponding block in each region can be randomized again in the second layer and be distinct among regions by a MT pseudo-random number generator [9]. With this strategy, the recovery data can be uniformly scattered (*in a distributed and missing-allowable manner*) in the whole host image to resist a cropping attack of an extensive area.

(a)

(b)



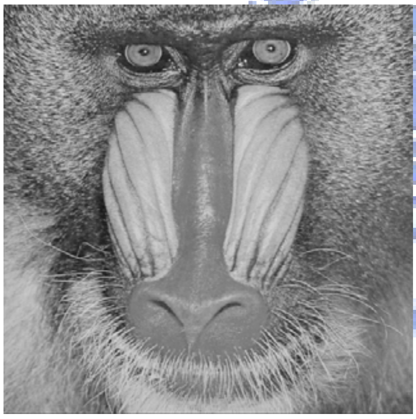
(c)

(d)

Fig. 4.4. Four original images: (a) Lena, (b) Jet, (c) Baboon, (d) Barbara.

(a)

(b)



(c)

(d)

Fig. 4.5. The watermarked images of Fig. 4.4: (a) Lena$^{(W)}$ (PSNR = 44.14 dB), (b) Jet$^{(W)}$ (PSNR = 44.13 dB), (c) Baboon$^{(W)}$ (PSNR = 44.15 dB), and (d) Barbara$^{(W)}$ (PSNR = 44.15 dB).

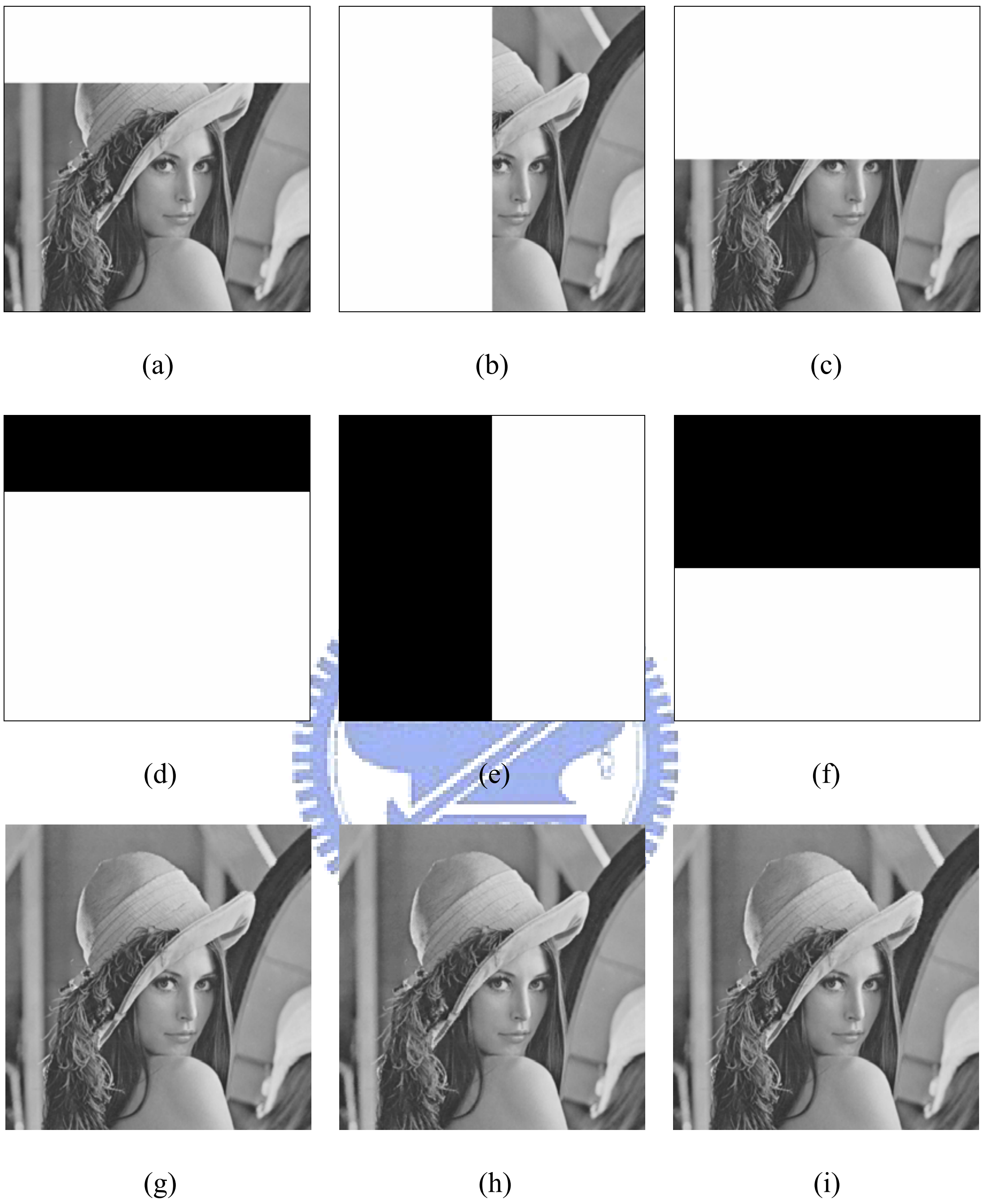


Fig. 4.6. The cropping attack experiments: (a) 25% of the watermarked image Lena$^{(W)}$ is cropped; (b) a vertical-cropping of 50%; (c) a horizontal-cropping of 50%; (d–f) the detected tampered-blocks of (a–c), respectively; (g–i) the images recovered from (a–c), respectively. The PSNR values are 40.04 dB for (g); 34.77 dB for (h); and 36.54 dB for (i).

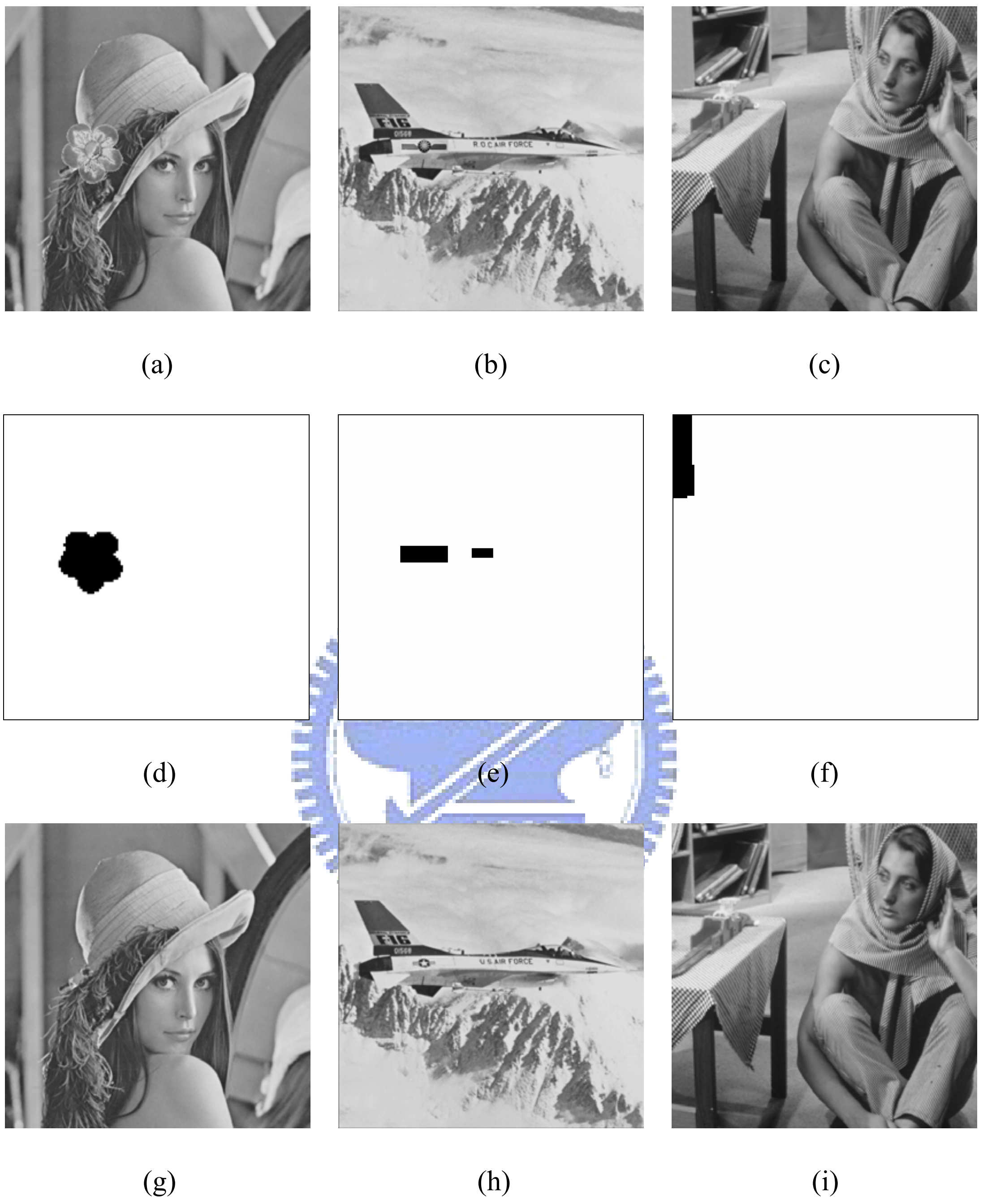


Fig. 4.7. The collage attack experiments: (a) a tampered version of Lena$^{(W)}$; (b) a tampered version of Jet$^{(W)}$; (c) a tampered version of Barbara$^{(W)}$, respectively; (g–i) the images recovered from (a–c), respectively. The PSNR values are 41.58 dB for (g); 42.45 dB for (h); and 43.01 dB for (i).

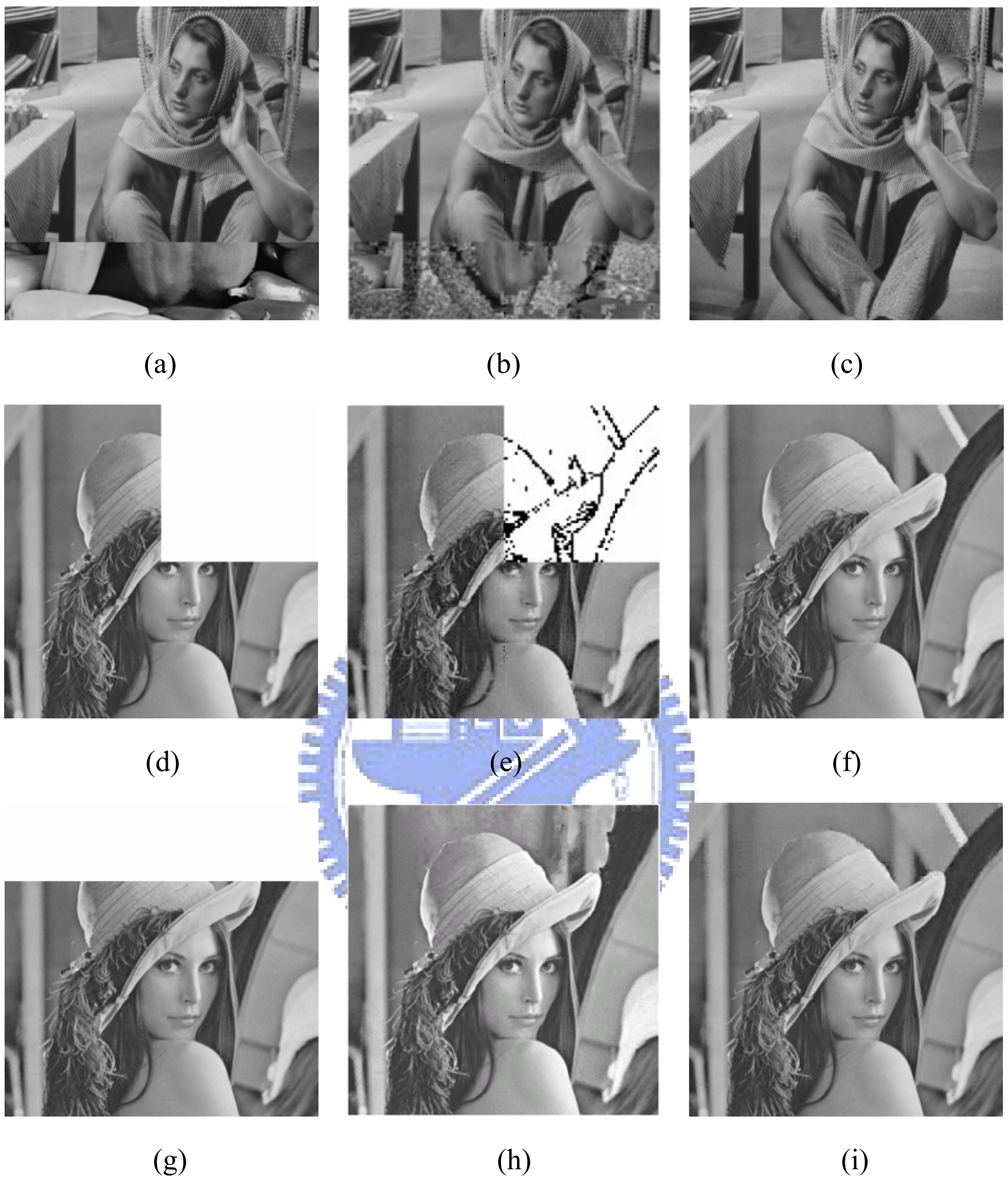
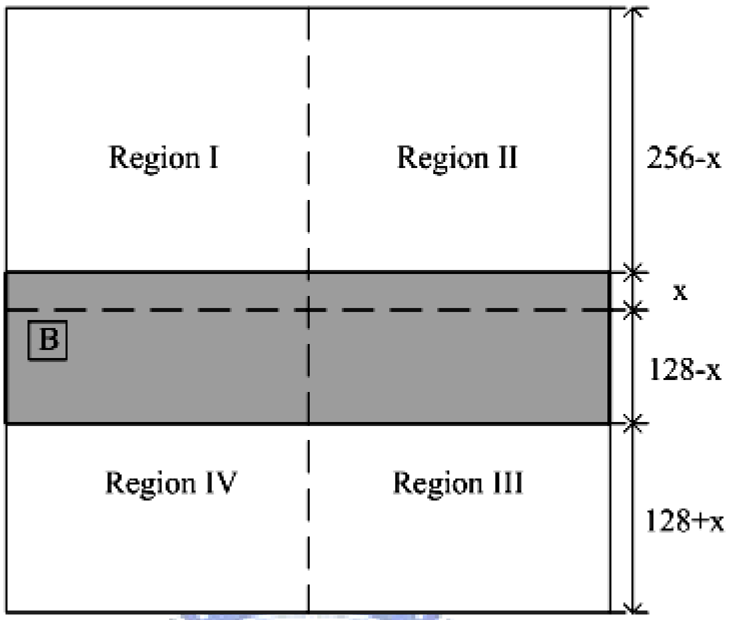
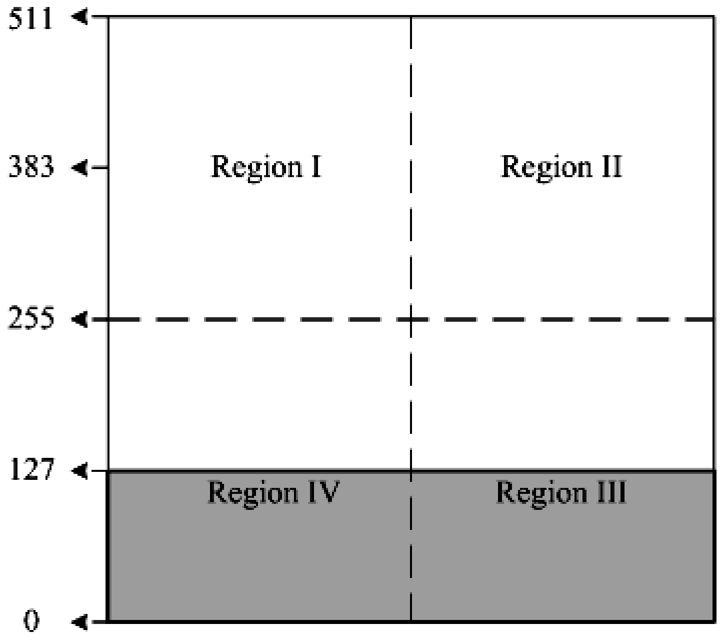


Fig. 4.8. A comparison of the recovered images of Refs. [72], [70], [73] and the proposed method, with tampering or cropping in the same region: (a) 25% of a 512×512 watermarked image Barbara$^{(W)}$ is replaced; (b) the recovered image of Ref. [72]; (c) the recovered image of the proposed method. (d) 25% of a 512×512 watermarked image Lena$^{(W)}$ is cropped; (e) the recovered image of Ref. [70]; (f) the recovered image of the proposed method; (g) 25% of a 256×256 watermarked image Lena$^{(W)}$ is cropped; (h) the recovered image of Ref. [73]; (i) the recovered image of the proposed method.



(a)



(b)

Fig. 4.9. Analysis of a 25% horizontal-bar cropping for a 512×12 image. Dark area is tampered.

Table 4.1. A comparison between the reported recovery methods and the proposed method. ("*" means "quoted directly from the reported paper", and "N/A" means not mentioned in the reported paper). The unit of PSNR is the dB.

| Method | Abilities to detect and locate tampering | PSNR of the watermarked image | PSNR of the recovered image |
|---|---|---|---|
| [71] | Yes | 44.37* (Beach 256×256) | 30.85* (cropping 7.1%) |
| [73] | Yes | 42.11* (Lena 256×256) | 30.14* (cropping 12.5%) 25.39* (Fig. 4.8(h) where cropping is 25%) |
| Ours | Yes | 44.15 (Beach 256×256) | 42.44 (cropping 7.1%) |
| | | 44.15 (Lena 256×256) | 41.28 (cropping 12.5%) 38.81 (Fig. 4.8(i) where cropping is 25%) |
| | | 44.14 (Lena 512×512) | 43.20 (cropping 6%) 34.77 or 36.54 (Fig. 4.6(h) or 4.6(i), respectively; each cropping is 50%) |
| | | 44.13 (Jet 512×512) | 43.27 (cropping 6%) 34.34 or 36.07 (if crop 50% like Fig. 4.6(b) or 4.6(c) does) |
| [72] | Yes | N/A | N/A (but see Fig. 4.8(b)) |
| [70] | Yes | 34.34* (Lena 512×512) | N/A (but see Fig. 4.8(e)) |
| [74] | Yes | 44 (Jet 512×512) | 42 (cropping 6%) 32 (cropping 49%) |

Table 4.2. Comparison of the size of recovery data.

| Method | Compression ratio (C.R.) of the recovery data | Total amount of the recovery data (counting the copies) | Manner of preserving the recovery data |
|---|---|---|---|
| [70] | Represent each 4×4 block by 1 bit. C.R.= $\frac{4\times4\times8}{1}$ =128 | $\frac{512\times512}{4\times4}\times3 = 49{,}152$ bits | Three backup copies of the recovery data are embedded in another 3 blocks |
| [71] | Represent each 2×2 block by 6 bits. C.R.= $\frac{2\times2\times8}{6}$ =5.33 | $\frac{256\times256}{2\times2}\times6 = 98{,}304$ bits | Backup in another block |
| [72] | Represent each pixel by one bit. C.R.= 8 | 512×512 = 262,144 bits | Backup in another pixel |
| [73] | Represent each 4×4 block by 31 bits. C.R.= $\frac{4\times4\times8}{31} = 4.13$ | $\frac{256\times256}{4\times4}\times31\times0.6 = 76{,}186$ bits | Backup in another block |
| [74] | Represent each 8×8 block by 56 bits. C.R.= $\frac{8\times8\times8}{56} = 9.14$ | $\frac{512\times512}{8\times8}\times(56\times2) = 458{,}752$ bits | Two backup copies of smooth and mid-textured blocks are embedded in another 2 blocks |
| Ours | Represent each 4×4 block by 14 bits. C.R.= $\frac{4\times4\times8}{14} = 9.14$ | For 256×256 image: Before sharing: $\frac{256\times256}{4\times4}\times14 = 57{,}344$ bits After (2, 3) sharing: $\frac{256\times256}{4\times4}\times\frac{14}{2}\times3 = 86{,}016$ bits | *n* shares are embedded in blocks of *n* distinct regions. |
| | | For 512×512 image: Before sharing: $\frac{512\times512}{4\times4}\times14 = 229{,}376$ bits After (2, 3) sharing: $\frac{512\times512}{4\times4}\times\frac{14}{2}\times3 = 344{,}064$ bits | |

Table 4.3. The processing time of our method. (unit: second)

| Size of image | Encoding | Decoding |
|---|---|---|
| 256×256 | 5.21 | 3.42 |
| 512×512 | 9.36 | 5.67 |

# Chapter 5
# Authentication and Cross-Recovery for Multiple Images

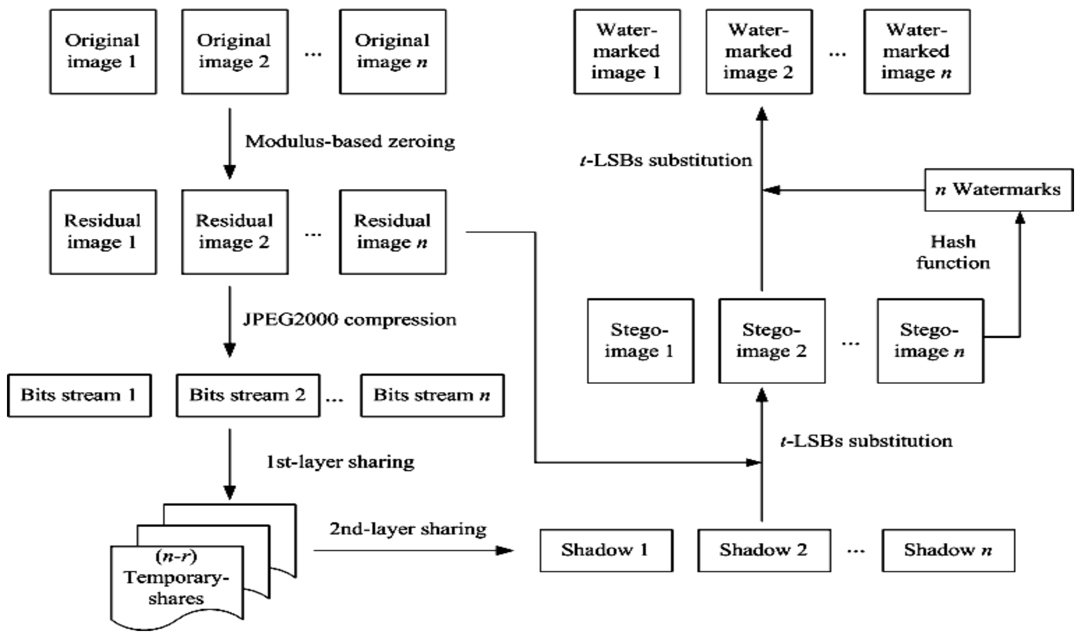
In this chapter, we propose a system with both image authentication and cross-recovery ability to protect a group of $n$ given digital images. The system is an ($r$, $n$) threshold scheme ($r$ is a pre-specified threshold satisfying $2 \le r < n$ ). Any $r$ of these images can reconstruct the whole group of $n$ images, but less than $r$ images cannot. Therefore, the system has cross-recovery ability because if some [up to ($n-r$)] images in the group are destroyed or lost in a distributed storage scheme or transmission mission, the destroyed or lost can be rebuilt vividly by the mutual support of $r$ survived members. As for the authentication ability, it is used in the daily maintenance of the storage system. The authentication data hidden inside the images enable them to check themselves which images are attacked or replaced, and hence call for a cross-recovery automatically. The whole design consists of compression, two-layer sharing, cryptographic hash function, and information hiding.

The rest of the chapter is organized as follows. An introduction to the applications of cross-recovery for multiple images is given in Sec. 5.1. The encoding and decoding (the latter includes verification and recovery) of the proposed method are described in Secs. 5.2 and 5.3, respectively. Experimental results are shown in Sec. 5.4. The comparison is in Sec. 5.5; and the summary is in Sec. 5.6.

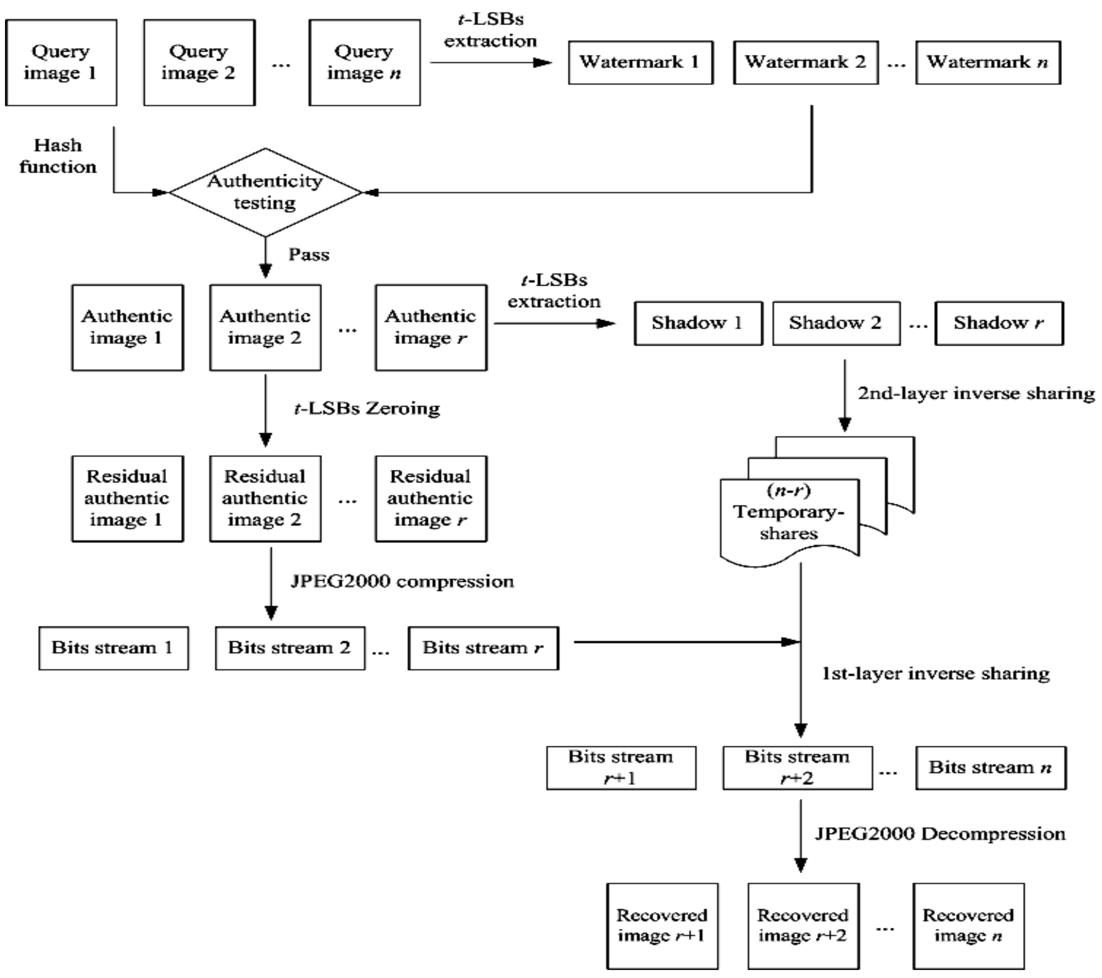
## 5.1 Introduction

In the transmission of multiple images via networks, it is not unusual for the network connection to be unstable or under hacker attacks. A general remedy is to request the sender end to resend again, but then transmission time has been wasted. Unless the sender immediately handles the request and the network is stable for awhile, the waiting time may seriously influence an important business decision when a cooperative company is eager to know what the new product images look like. Another situation is that in a distributed storage system of multiple images, if some images are lost by the hardware failure or the manager’s carelessness, a trivial way to recover them is using back-up copies identical to the lost images but stored elsewhere. Although the back-up copies can increase the reliability of a storage system, it also increases the cost of the system.

As a result, an interesting and important issue becomes how to protect the integrity of multiple images during a transmission or in a storage system. Several image authentication methods [61-78] have been proposed. Most of them emphasize the detection of whether malicious manipulations have occurred, and some have the position-locating ability for the tampered parts of the test image. Some approaches [70-78] additionally possess the capability of automatic recovery for the tampered parts to a certain extent, after detection and locating work. However, these recovery approaches usually only target a single image rather than a group of images. Moreover, the recovery ability of these approaches is limited, for example, if the tampered portions of the image exceed 50%, or even worse, if the whole image is lost, then the image is unable to be recovered. To solve this problem, we develop here an image authentication method with cross-recovery ability for multiple images. The word “cross-recovery” means that if some of the images in the group are lost during transmission or tampered by malicious manipulations, then the lost/tampered images can be reconstructed by the mutual support of the surviving members at the receiver end. The details of the proposed method are presented in the following sections.



(a)



(b)

Fig. 5.1. Two flowcharts of the proposed method: (a) the encoding procedure; (b) the authentication and recovery procedure.

## 5.2 The Proposed Method (Encoding)

Fig. 5.1(a) shows the flowchart of the encoding procedure, while Fig. 5.1(b) is for the decoding (verification and recovery) procedure. Sec. 5.2 only discusses Fig. 5.1(a). Assume that there are $n$ grayscale (8-bit) images in the group to be protected. Our goal is that after watermarking these $n$ images, if at least $r$ (out of the $n$) watermarked images survive (must pass the authentication test) in a transmission or disk crash, where $r$ is a pre-specified threshold and $2 \le r < n$, all non-survived images can be recovered to a certain extent. In short, our method can tolerate the loss or modification of up to ($n-r$) watermarked images.

The encoding procedure consists of three parts. First, create the rough image by setting all $t$ least significant bits (LSBs) of each pixel of the original image to zero. To reduce the size of recovery data, encode each of the $n$ rough images respectively by the JPEG2000 compression technique to generate $n$ bit streams. These $n$ bit streams are treated as the recovery data. Then, share these $n$ bit streams by means of a two-layer sharing method to generate $n$ shadows. Use a module method, modified from the simple LSB substitution method, to embed these $n$ shadows in the $n$ original images of the group, respectively, to form $n$ stego-images. Finally, generate the authentication data (a 128-bits watermark) for each stego-image by using a cryptographic hash function, in which the input includes the local and interrelated information of the stego-image. The authentication data are embedded into a certain space of the stego-image to form the watermarked image. The watermarked images are the final results, which not only look like the original images but also contain the authentication and cross-recovery data needed to protect the transmission or storage of the group. Details of the encoding are described in the following three sections accordingly.

### 5.2.1 Generation of the Recovery Data

In the proposed method, the recovery data of each image is generated first. As a remark, all the bit values of the *t* LSBs at each pixel of the original image will be completely destroyed (erased) to hide the recovery and authentication data. Therefore, after the hiding procedure, only the remaining (8 − *t*) most significant bits (MSBs) of the original image are directly related to the local gray values. Hence, the rough image, which is a $2^{(8-t)}$-level image formed of the (8 − *t*) MSB bit-planes, of each original image is the input data for generating the corresponding recovery data.

Furthermore, a small skill used in our scheme is that the rough image is obtained by using the modulus operation rather than directly setting all *t* LSBs of each pixel of the original image to zero. (The modulus-based zeroing scheme here is inspired by Thien and Lin's modulus-based hiding method [31]. Reference [31] has proven that when doing hiding, the gray value distortion of the simple LSB substitution method is never smaller than that caused by modulus-based hiding.)

Our modulus-based zeroing scheme is as follows. To set *t* LSBs of a pixel value $y$ $(0 \le y \le 255)$ of an 8-bit grayscale image to zero, the new value $\hat{y}$ of our rough image can be expressed as

$$\hat{y} = 2^t \times rounding(\frac{y}{2^t}), \tag{5.1}$$

where the $rounding(\cdot)$ operator means rounding its content to the nearest integer. This rounding operator is better than the truncation operator, which directly truncates the non-integer part; and the latter is exactly the one used in a simple LSB method. Of course, we need to check whether or not the $\hat{y}$ falls in the valid gray-value range $0 \le \hat{y} \le 255$. If it is out of the range, then $\hat{y}$ should be corrected further as

$$\hat{y} = \begin{cases} \hat{y} + 2^t & \text{if } \hat{y} < 0 \\ \hat{y} - 2^t & \text{if } \hat{y} > 255 \end{cases}. \tag{5.2}$$

Now, the rough image is an image whose gray values are $\hat{y}$, of which $2^t$ is a factor, rather than the old pixel values $y$ of the original image.

For example, assume that a rough image of zeroing 2-LSB (i.e., $t$=2) is desired, and the gray value $y$ of a pixel in the original image is 107 = $(01101011)_2$. Instead of using 104 = $(01101\underline{0}\boldsymbol{00})_2$, which is the result of directly erasing the 2 LSBs by zeros, we use the modulus-based zeroing formula in Eq. (5.1), i.e., the value of the corresponding pixel in our rough image is $2^2 \times rounding(\frac{107}{2^2}) = 108 =$ $(01101\underline{1}\boldsymbol{00})_2$. Obviously, the distortion (108 − 107=1) of using Eq. (5.1) is smaller than the distortion (107 − 104=3) of using direct truncation. Before going further, we remark here on the influence of the value of $t$. The bigger the parameter value of $t$, the larger the hiding capacity of each image (to embed the recovery and authentication data into the space provided by $t$ bit planes), and hence, the better the quality of the recovered image. However, the quality of the watermarked image will become worse as the value of $t$ increases, for the final $t$ bits have been changed at each pixel. So, there is a tradeoff.

Next we discuss the design of the recovery data. In general, the data amount of a digital image is large, so it is difficult to embed all information of a protected image in quite a limited hiding space. For this reason, we first use a compression technique to reduce the data amount, so that it is easier to perform the embedding procedure later. JPEG is a very common image compression standard on the Internet. In contrast to JPEG, JPEG2000 is a new international standard for image compression that is advantageous over JPEG in terms of compression ratio, freedom of compression mode selection, and adaptation to different kinds of images. Therefore, JPEG2000 is employed as the image compression technique in our method to generate the recovery data. As shown in Fig. 5.1(a), after modulus-base zeroing, $n$ rough images of the group are compressed by the JPEG2000 compression encoder, and the compression result of each rough image is a bit stream. The *n* bit streams are the input of the first-layer sharing introduced in Sec. 5.2.2. As a rule, the bit stream of each rough image is treated as the recovery data for that image, i.e., the decompressed image obtained from the corresponding bit stream will be used as the recovered image to replace a watermarked image, should the watermarked image be tampered or lost.

### 5.2.2 Two-Layer Sharing

After generating the recovery data, the next thing to do is to share it. We design here a two-layer sharing scheme to share the information needed in recovery.

#### 5.2.2.1 First-Layer Sharing

As shown in Fig. 5.1(a), the recovery data (the *n* compressed bit streams of the rough images) are shared in the first-layer sharing process to generate the $(n-r)$ temporary shares. The process is explained here. Assume that each bit stream has *m* bits. Divide each stream into $m/8$ non-overlapping cells so that each cell contains 8 bits. Then, each cell is converted from base 2 to base 10 (a decimal number ranges between 0 and 255); in other words, each cell value is a decimal value in the range of commonly-seen gray values. For each cell *j*, where $1 \leq j \leq m/8$, let $\{A_1, A_2, \ldots, A_n\}$ be, respectively, the *n* decimal values of the *j*th cell in the *n* bit streams. Then, let the first-layer sharing polynomial (for the *j*th cell) be

$$p_j(x) = [A_1 \times (c_1)^x + A_2 \times (c_2)^x + \cdots + A_n \times (c_n)^x] \,(\mathrm{mod}\, 256), \tag{5.3}$$

where the arbitrarily chosen positive integer constants $\{c_1, c_2, \ldots, c_n\}$ satisfy $c_i \neq c_k$, for all $i \neq k$. For example, let $c_1=1$, $c_2=2$, …, $c_n=n$. Now, for each integer $x \in \{1, 2, \cdots, (n-r)\}$, let

$$p(x) = [p_1(x), p_2(x), \ldots, p_{m/8}(x)] \tag{5.4}$$

denote the $x$th temporary share (so each $p_j(x)$ is just the $j$th cell value of $p(x)$). Since each cell value $p_j(x)$ is still an 8-bit value by Eq. (5.3), each temporary share $p(x)$ can be treated as an image; and it has 8×($m$/8)=$m$ bits, just like the length of each bit stream.

After first-layer sharing, there are $(n - r)$ temporary shares. Because the tampered or lost parts of some bit streams can be reconstructed by the cooperation between the $(n-r)$ temporary shares and the $r$ survived bit streams (the details are illustrated in Sec. 5.3.2), the $(n-r)$ temporary shares are the key data for the recovery purpose. To ensure the survival of these $(n-r)$ temporary-shares, they are shared again among the $n$ images.

Notably, the original recovery data ($n$ bit streams), which keeps the main information about the $n$ original images, are condensed to the $(n-r)$ temporary shares by the first-layer sharing. In other words, the first-layer sharing not only contributes to the recovery work, but also decreases further the total amount of the recovery data, for there are $(n-r)$ temporary shares of $m$ bits each, which are more economic (in total size) than the $n$ bit strings of $m$ bits each.

### 5.2.2.2 Second-Layer Sharing

In second-layer sharing, we use Thien and Lin's ($r$, $n$) threshold sharing method [90] to create $n$ shadows, which share the $(n-r)$ temporary shares just generated from the first layer. The detail is as follows. For each temporary share $p(x) = [p_1(x), p_2(x), \ldots, p_{m/8}(x)]$, divide it into sectors of $r$ pixels each. [Recall from Eq. (5.3) that each element $p_j(x)$ has 8 bits, and hence can be treated as a pixel.] Then, for each sector, use its $r$ pixels as $r$ coefficients in Eq. (4.1) (see Sec. 4.2), and then plug in $n$ pre-specified integer values for the variable $x$. This creates $n$ transformed values for a sector. After doing this for all sectors of a temporary share $i$, and collecting the transformed values for each $x$, the temporary share $i$ is transformed to $n$ shadows. As $i$ runs from 1 through $(n-r)$, each temporary share gets its own $n$ shadows. Then, the first shadows of all $(n-r)$ temporary shares are concatenated, and the result is still called shadow 1. Similarly, second shadows of all $(n-r)$ temporary shares are concatenated, and the result is still called shadow 2. As the process goes on, the concatenation sequentially yields shadows 1, 2, …, $n$.

As shown in Fig. 5.1(a), after the aforementioned two-layer sharing phase, the recovery data of the $n$ original images in the group are condensed to $n$ shadows. To have the ability to recover later any polluted image by using the remaining images of the group, we may "hide" (see Refs. [18-21] for examples of hiding techniques) the $n$ obtained shadows, respectively, into the $n$ rough images of the protected group. Recall that the $t$ least significant bits of each rough image have been zeros, so we may use these $t$ least significant bits to embed shadow (embed one shadow in one rough image). This hiding technique is the famous $t$-LSB substitution method. Now, $n$ stego-images are thus generated, which still look like their original images. The reason we use $t$-LSB substitution for hiding is because it is simple, fast, and with great hiding capacity without downgrading too much the quality of the stego-images. Notably, in the hiding step here, we use $t$-LSB substitution rather than the modulus-based substitution [31], because we do not want to change the $(8-t)$ MSBs of the rough images when creating stego-images. This ensures the $(8-t)$ MSBs are the same between each rough image and its stego-image. Therefore, in later days, when a lost or tampered stego-image is to be recovered, we can grab the rough images obtained from the $(8-t)$ MSBs of non-tampered stego-images, then do JPEG2000 compression, and thus obtain the recovery data [see Figs. 5.1(a) and 5.1(b)].

### 5.2.3 Generation of the Authentication Data

After the hiding procedure, the *n* original images of the protected group are transformed to the *n* stego-images. Later, to determine whether a stego-image of the protected group is tampered or not, people can use image authentication to verify the integrity of each stego-image. Below we describe the details of the generation of our authentication code (the watermark).

As the final work for the encoding procedure, the watermark to verify the stego-image's integrity is generated, and it is then embedded in stego-images to obtain the watermarked image. In our method, we use a cryptographic hash function to generate the watermark of each stego-image. Similar to Sec. 4.3.2, we use MD5 [8] as the hash function, and produces a single output 128-bit message (the output hash value) from the input message (see Eq. (4.3)). The input message consists of some important information of the stego-image, including width, length, the image's identification number, and the pixel values of the stego-image. Let *ID* be the image's identification number of the original image in the protected group. Now, according to Eq. (4.3), for each stego-image whose size is *H*×*W*, the authentication data can be computed as

$$H(ID \,\|\, H \,\|\, W \,\|\, z_1 \,\|\, z_2 \,\|\cdots\|\, z_{H\times W-128} \,\|\, \hat{z}_{H\times W-127} \,\|\cdots\|\, \hat{z}_{H\times W}) = (d_1, d_2, \ldots, d_u), \qquad (5.5)$$

where the symbol || is the concatenation of all input streams, and $z_1, z_2, \ldots, z_{H\times W-128}$ are the stego-image pixels' gray values according to the raster-scan order of the stego-image, which is from left to right and top to bottom. Because the least-significant bit of the final 128 pixels $\hat{z}_{H\times W-127}, \hat{z}_{H\times W-126}, \cdots, \hat{z}_{H\times W}$ of each stego-image are particularly reserved for embedding its 128-bits authentication data, their pixel values $\hat{z}_{H\times W-127}, \hat{z}_{H\times W-126}, \cdots, \hat{z}_{H\times W}$ are computed just using the 8－1=7 most significant bits (MSBs) only. Finally, the generated 128-bits watermark is embedded in the LSB of the final 128 pixels of the corresponding stego-image to form the watermarked image. These watermarked images are used in transmission or storage.

The version in the previous paragraph, which sequentially hides the 128-bits authentication data (watermark) in the LSB of the last 128 pixels of a stego-image, is just the simplest version. For security concerns, we can use a pseudo-random number generator to randomize the embedding locations of the watermark in the stego-image. More specifically, assume that the pixels of the stego-image are numbered sequentially from 0 to $(H\times W)-1$ according to the raster scan order, which is from left to right and top to bottom. In other words, $(H\times W)$ is the total number of the pixels in the stego-image (i.e., $(H\times W)$ is 262,144 for a 512×512 image). By a pseudo-random number generator, we can generate a sequence of random integer numbers. Then, we can embed the watermark into the stego-image according to the generated sequence one by one. For example, if the generated sequence is {214,257, 133,785, 34,480, 9284,…}, the first bit of the watermark is embedded into the LSB of the pixel number 214,257 in the stego-image, and the second bits is embedded into the LSB of the pixel number 214,257, and so on. Therefore, the pseudo-random sequence is used to increase the difficulty in getting the watermark for an unauthorized user. In our method, the Mersenne Twister (MT) pseudo-random number generator [9] is employed as the permutation function of the watermark, and its seed can be regarded as a secret key. In the image-disclosure site, only the authorized person who owns the same secret key can obtain the same sequence of pseudo-random integer numbers to extract the watermark in the verification phase.

## 5.3 The Proposed Method (Decoding)

This section introduces the decoding procedure. Fig. 5.1(b) shows the flowchart for decoding. The detail sub-procedures for verifying and recovering *n* query images are as follows.

### 5.3.1 Verification

When a user receives some of the *n* query images from the protected group, the first thing to do is to verify the integrity of each query image. This is because the pervasive and powerful image manipulation tools now have made the imperceptible modification of images become very easy, and a user usually cannot determine whether the query image is tampered or not by using the naked eye. According to the final paragraph of Sec. 5.2.3, a legal user can have the secret key, which is the seed of the MT pseudo-random number generator [9], to obtain the embedding locations of the hidden watermark.

For each query image, its extracted watermark should be the same as the authentication data recomputed directly according to the hash function described in the paragraph containing Eq. (5.5). If the extracted watermark coincides with the recomputed watermark, then the query image is called an authentic image; otherwise, it is regarded as a tampered image. An authentic image means that it is extremely likely that no tampering occurs in this image, and the recovery data embedded in this image are therefore considered trustworthy in joining the reconstruction team to recover some tampered images. Note that if an image fails to pass the authentication test, then the recovery information stored in it should never be used.

## 5.3.2 Cross-Recovery of Tampered Images through the Cooperation of Authentic Images

After checking the authentication status of all query images, the recovery phase starts if there is at least one tampered or lost image; along with the condition that there exist at least $r$ authentic images. To recover a tampered or lost image, we collect its recovery data from any $r$ authentic images. The recovery data were embedded earlier in the encoding phase in the $t$ LSBs of all $n$ images (one shadow per image, as stated in Sec. 5.2.2). When at least $r$ (out of the $n$) images are authentic, we can extract $r$ shadows from the $t$ LSBs of the $r$ authentic images. Then, by doing the inverse of the second-layer sharing, we can use these $r$ shadows to extract back the $(n-r)$ temporary shares, which are the result of the first-layer sharing. (The detail steps of inverse sharing are similar to those in Sec. 3.2 of Ref. [90].)

Meanwhile, the $r$ authentic stego-images can replace all their $t$ LSBs by zeros to obtain $r$ rough images. Then use JPEG2000 compression (as described in Sec. 5.2.1 and Fig. 5.1) to obtain $r$ bit streams. Because the $r$ authentic stego-images are verified to have not been tampered, the $r$ rough images obtained now in the decoding phase, and hence the $r$ bit streams obtained now, are identical to those obtained in the encoding phase.

Then, the $r$ bit streams are combined with the $(n-r)$ temporary shares obtained from the aforementioned second-layer inverse sharing, to rebuild other $(n-r)$ bit streams by solving $n$ coefficients ($A_1, A_2, \ldots, A_n$) of Eq. (5.3).

Notably, these $(n-r)$ bit streams are the compression result corresponding to those $(n-r)$ tampered or lost images, so we may decompress these $(n-r)$ bit streams to recover the $(n-r)$ tampered or lost images in lower resolution.

In Lemma 6.1, we prove that the $(n-r)$ bit streams could be recovered successfully through the cooperation of the $r$ (out of the $n$) bit streams and the $(n-r)$

temporary shares.

***Lemma 5.1***

If any $r$ (out of the $n$) bit streams and ($n-r$) temporary shares are obtained, then the absent ($n-r$) bit streams can be recovered successfully.

***Proof:***

In first-layer sharing, the ($n-r$) temporary shares $p(1), p(2), \ldots, p(n-r)$ are generated by Eq. (5.3). For each cell $j$, the following shows the relation between the coefficients $\{A_1, \ldots, A_n\}$ of the $n$ bit streams and temporary share values $\{p_j(x): x=1,2,\ldots,(n-r)\}$. For convenience, we dropped the subscript $j$ in this proof.

$$\begin{bmatrix} c_1^{\,1} & c_2^{\,1} & c_3^{\,1} & \cdots & c_n^{\,1} \\ c_1^{\,2} & c_2^{\,2} & c_3^{\,2} & \cdots & c_n^{\,2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ c_1^{\,n-r} & c_2^{\,n-r} & c_3^{\,n-r} & \cdots & c_n^{\,n-r} \end{bmatrix} \bullet \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(2) \\ \vdots \\ p(n-r) \end{bmatrix}. \tag{5.6}$$

Without the loss of generality, assume that the obtained $r$ (out of the $n$) bit streams are $A_1, A_2, \ldots, A_r$, and the tampered or lost ($n-r$) bit streams that need to be reconstructed are $A_{r+1}, A_{r+2}, \ldots, A_n$. Then, Eq. (5.6) can be rewritten as

$$\begin{bmatrix} c_1^{\,1} & c_2^{\,1} & \cdots & c_r^{\,1} \\ c_1^{\,2} & c_2^{\,2} & \cdots & c_r^{\,2} \\ \vdots & \vdots & \cdots & \vdots \\ c_1^{\,n-r} & c_2^{\,n-r} & \cdots & c_r^{\,n-r} \end{bmatrix} \bullet \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_r \end{bmatrix} + \begin{bmatrix} c_{r+1}^{\,1} & c_{r+2}^{\,1} & \cdots & c_n^{\,1} \\ c_{r+1}^{\,2} & c_{r+2}^{\,2} & \cdots & c_n^{\,2} \\ \vdots & \vdots & \cdots & \vdots \\ c_{r+1}^{\,n-r} & c_{r+2}^{\,n-r} & \cdots & c_n^{\,n-r} \end{bmatrix} \bullet \begin{bmatrix} A_{r+1} \\ A_{r+2} \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(2) \\ \vdots \\ p(n-r) \end{bmatrix}. \tag{5.7}$$

Equation (5.7) can be expressed as the abbreviation

$$C_k \cdot \vec{A}_k + C_u \cdot \vec{A}_u = \vec{P}. \tag{5.8}$$

Equation (5.8) therefore becomes

$$C_u \cdot \vec{A}_u = \vec{P} - C_k \cdot \vec{A}_k. \tag{5.9}$$

From Eq. (5.9), because $c_i \neq c_k \neq 0, \text{if } i \neq k$, we can see that

$$C_u^T = \begin{bmatrix} c_{r+1}^{1} & c_{r+1}^{2} & \cdots & c_{r+1}^{n-r} \\ c_{r+2}^{1} & c_{r+2}^{2} & \cdots & c_{r+2}^{n-r} \\ \vdots & \vdots & \cdots & \vdots \\ c_n^{1} & c_n^{2} & \cdots & c_n^{n-r} \end{bmatrix}$$, and obviously, it is a Vander Monde matrix.

Note that for any Vander Monde matrix

$$V = \begin{bmatrix} 1 & e_1 & e_1^2 & \cdots & e_1^{h\text{-}1} \\ 1 & e_2 & e_2^2 & \cdots & e_2^{h\text{-}1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & e_h & e_h^2 & \cdots & e_h^{h\text{-}1} \end{bmatrix}, \tag{5.10}$$

the determinant will be

$$\det(V) = \prod_{l<m} (e_m - e_l), \tag{5.11}$$

and hence nonzero as long as all $e_i$ are distinct. Because matrix $C_u^T$ is the form of the Vander Monde matrix, det ($C_u^T$) is nonzero, that is, det($C_u$) is nonzero. It means that $C_u$ has the inverse matrix $C_u^{-1}$. Therefore, Eq. (5.9) can be rewritten as

$$\vec{A}_u = C_u^{-1}\left(\vec{P} - C_k \cdot \vec{A}_k\right). \tag{5.12}$$

Because the $r$ (out of the $n$) bit streams $\vec{A}_k$= ($A_1$,…, $A_r$) are known, and the ($n-r$) temporary shares $\vec{P}$ ={ $p(1)$ , …., $p(n-r)$}are also known, ( $\vec{P} - C_k \cdot \vec{A}_k$ ) can be calculated. By the inverse operation of matrix $C_u$, the absent ($n-r$) bit streams $\vec{A}_u$ = ($A_{r+1}$, …, $A_n$) can be recovered easily by the multiplication operation between $C_u^{-1}$ and ( $\vec{P} - C_k \cdot \vec{A}_k$ ). *-End of proof-*

## 5.4 Experimental Results and Robustness-Related Issues

### 5.4.1 Experimental Results

Experimental results are presented to demonstrate the performance and feasibility of the proposed method. A group of 512×512 standard 8-bit gray-scaled images (Baboon, Lena, Jet, Scene) shown in Fig. 5.2 are used as the test images in the experiments.

In the experiments, we used a ($r$=2, $n$=4) threshold scheme in the two-layer sharing procedure to share the recovery data, i.e., the group of four images could survive even if ($n - r$ = 4 − 2) of the watermarked images was lost or tampered. The parameter setting for the $t$-LSBs is $t$=2, i.e., the watermark is embedded into the 2 LSBs of the images' pixels. Fig. 5.3 shows the watermarked images of Fig. 5.2 using the proposed method described in Sec. 5.2. The qualities of all watermarked images and recovered images are measured by the peak signal-to-noise ratio (PSNR) defined in Eq. (4.6). From Fig. 5.3, we can see that the qualities of the watermarked images are acceptable (their PSNR values range between 44.14 and 44.17 dB). It is visually indistinguishable between Figs. 5.2 and 5.3 using the naked eye. In other words, our watermarked images have the transparency property for the hidden data, i.e., the recovery data and watermark are perceptually invisible. To inspect our scheme's recovery ability, some of the watermarked images shown in Fig. 5.3 are lost or tampered in the following experiments. As for the processing time, the average encoding and decoding time of our method are shown in Tables 5.3. The decoding time includes both detection and recovery phases.

#### *5.4.1.1 Case 1: one watermarked image is lost*

When the watermarked image "Lena" [Fig. 5.3(b)] is lost due to transmission error or hardware storage failure, the remaining images are Baboon, Jet, and Scene, as shown in Figs. 5.4(a)-5.4(c). After the verification procedure described in Sec. 5.3.1, these three images are authentic, and thus participate in the recovery team to save the lost member Lena. Using the support from the three authentic images Baboon, Jet, and Scene, we recover the Lena′ shown in Fig. 5.4(d), whose PSNR value is 41.57 dB.

### *5.4.1.2 Case 2: two watermarked images are tampered*

In the experiment, we consider the situation when (any) two of the four watermarked images shown in Fig. 5.3 are tampered by manipulations or processing such as rotation, filtering, cropping, noise addition, resizing, and replacement (see Fig. 5.5). Because the authentication code was generated by a hash function using the pixel values of the stego-image, the image's width and length, and the image's identification number as the input information (see Sec. 5.2.3), any manipulation of pixel values, image size, or image *ID* would cause the input image to fail our verification test. So, these images shown in Figs. 5.5(a)-5.5(f) are judged as tampered images by the authentication test. Therefore, the recovery phase begins, and the two tampered members of the protected group are successfully reconstructed using the remaining two authentic images (all images in Fig. 5.3 can pass the authentic test, so any two from Fig. 5.3 can achieve this).

One such example is shown in Fig. 5.6, in which Figs. 5.3(c) and 5.3(d) are tampered and become Figs. 5.6(c) and 5.6(d). Their recovery versions become Jet′ and Scene′ in Figs. 5.6(e) and 5.6(f). As for the other examples of ($r$=2, $n$=4), each time two of the four images in Fig. 5.3 are replaced by two of the six images shown in Fig. 5.5. The two images identical to the images shown in Fig. 5.3 can pass the authentication test and thus need no recovery. Moreover, after passing the test, they will back-up the recovery of the two images missing from Fig. 5.3. No matter which two are tampered or missing from Fig. 5.3, the recovered versions are always identical to two of the four images shown in Fig. 5.7.

### 5.4.2 Robustness-Related Issues

Now we discuss the robustness of the watermarked images. There are two parts of information carried in each watermarked image: (1) watermark (i.e., the authentication code), and (2) recovery data. Their robustness are discussed respectively below.

(1). The authentication code is powerful enough so that if a received image $R$ is not exactly (100%) a product of our method (a watermarked image of ours), then the received image $R$ is rejected immediately, no matter if the difference between $R$ and our product is small or big, and also no matter if the difference is due to noise addition, filtering, resizing, rotation, cropping, replacement, hacker attack, etc. This is shown in the experiment (related to Figs. 5.6 and 5.7). The reason we require the authentication code to be so sensitive is that we want to make sure the recovery data that we grab from the received image $R$ is 100% exact, avoiding confusion in the later step of helping the recovery of other broken images $\{R_2, R_3, \ldots\}$. (For example, if the recovery of $R_2$ from $r$=3 images $\{I_A, I_B, I_C\}$ are distinct from $\{I_A, I_B, R\}$, or $\{I_B, I_C, R\}$, or $\{I_A, I_C, R\}$, then there are some other troubles to judge which recovery of $R_2$ is to be believed.

(2). As a result of this, after the authentication test, if a received image $R$ cannot pass the authentication test, then we discard all information hidden in or carried by image $R$, because we do not think this information is trustworthy. Therefore, in some sense, we might say that the recovery data are not robust against any change of the watermarked image. But the authentic watermark is quite robust.

(3). The above discussion in (1) and (2) is from logic sense. As for the technique sense, notably, the recovery data are usually much larger in size than the authentication code. Hiding much larger sized data and making it robust is more difficult than hiding much smaller sized data and making it robust. This is because the robustness process against every kind of manipulation or attack often enlarges the hidden data. If the original data to be hidden are small, such as the authentication code, then robust hiding is less difficult. But if the original data to be hidden is already large, such as the cross-recovery data of multiple images, then this is difficult.

## 5.5 Discussion

Table 5.1 compares our method with Refs. [66-67, 69-71, 74-75] and [98]. Among them, Refs. [66-67, 69] are for authentication only. References [70-71, 74-75] can have both authentication and self-recovery ability of a “single” image. Reference [98] is the only one (other than ours) dealing with multiple images. So we introduce and compare with Ref. [98] in more detail in the following paragraphs. Before that, let us take a look at single-image recovery methods. Usually, for single-image methods, their schemes can recover the tampered parts of the protected image by using the recovery data, which is often embedded in blocks of other areas of the same image. However, when a watermarked image is tampered extensively in a large area and randomly, it is not rare that a block and its back-up block are tampered at the same time. In this case, the recovery ability of the tampered block in their approaches is gone, but our scheme can handle this case, even if the whole image is lost, as long as the remaining $r$ members are authentic.

Next we compare our method with Ref. [98] in which Tsai et al. proposed a method whose goal is somehow related to ours. Their goal is to share multiple secret images among a group of cover images, so that each pair of stego-images can recover a unique secret image for that pair [see Fig. 5.8(a)]. In Fig. 5.8(a), the six secret images {Jet, Goldhill, Girl, Toys, Boat, and Scene} with size of $200\times200$ pixels are shared in the four cover images {Lena, Baboon, Tiffany, and Zelda} with size of $600\times600$ pixels. The secret image Jet is shared between the pair of stego-images Lena and Baboon; the secret image Boat is shared between the pair of stego-images Lena and Tiffany, and so on. The PSNR values of their stego-images range between 42.41 and 42.61 dB.

Reference [98] has two main advantages. First, it is effective. By an ingenious design of the sharing order and an adequate utilization of bit planes of the cover image, one can share multiple secret images with multiple cover images, and the quality of all stego-images is visually acceptable to cover communication. Second, it is efficient, because the stego-images are generated using simple operators such as addition and exclusive-or.

However, according to the experiment done in Ref. [98] [shown in our Fig. 5.8(a)], when one of the stego-images is tampered or lost, three of the multiple secret images cannot be reconstructed and then lost forever. For example, if the stego-images Lena is tampered or lost, the secret images Jet, Boat, and Toys cannot be recovered by the remaining three stego-images. Therefore, the method of Ref. [98] is not fault tolerant when some stego-images are polluted. As a contrast, our recovery still works even if $(n-r)$ of the $n$ watermarked images are tampered or lost. All member images of the protected group can be recovered by our method.

In summary, each method has its own advantage. Reference [98] is faster, and ours is fault tolerant. Reference [98] is for sharing multiple secrets, and ours is for easy recovery of distributed storage systems in unstable environments.

To compare with Ref. [98] further, we use a structure shown in Fig. 5.8(b) so that Ref. [98] can also be used for cross-recovery of multiple images. Here, the hidden secret images are in fact the JPEG2000-compressed images of the four cover images. Then we generate the authentication code of each stego-image using the skill aforementioned in Sec. 5.2.3, and finally embed the code in stego-images to form four watermarked images in Fig. 5.8(b).

In Fig. 5.8(b), if one of the four watermarked images is lost or tampered, then the remaining three members can reconstruct the lost image. For example, if Baboon is lost, we can reconstruct it by the cooperation between Lena and Jet or Jet and Scene. Of course, the recovered version of the lost image is its JPEG2000-compressed version. However, when two of the four watermarked images are lost, only one of the lost images can be reconstructed. For example, if Jet and Scene are lost, then the JPEG2000-compressed Scene can be reconstructed by the cooperation between Baboon and Lena, but the image Jet is gone. To the contrary, our (2, 4) scheme can tolerate two tampered or lost images, for they can be recovered by the cooperation of the remaining two authentic images (see Case 2 of Sec. 5.4.1).

Table 5.2 lists the comparison between our ($r$=2, $n$=4) scheme and Ref. [98]. From Table 5.2, we can see that the qualities of the watermarked and recovered images in our scheme are better than that of Ref. [98]. Moreover, in the recovery procedure, the fault-tolerant ability of our scheme is better than the method of Ref. [98]. As for processing speed, Ref. [98] is faster than ours because Ref. [98] uses logic operations rather than arithmetical computations for their sharing process.

## 5.6 Summary

We propose an authentication and cross-recovery method for a group of $n$ images. The goal of this work is that if some images in the group are tampered or lost, these images can be identified and reconstructed by the mutual support of the $r$ survived members. In our method, two-layer sharing is designed to create $n$ shadows that share the recovery data. The sharing design reduces the recovery data amount; and it also makes the recovery fault tolerant, for up to $n-r$ shadows can be lost.

The features and the experimental results show that: 1) the quality of our watermarked images is acceptable, i.e., the proposed method keeps the transparency of the hidden data, including the recovery data and watermark; 2) authentication and cross-recovery can both be done when some watermarked images are altered, tampered, or lost; 3) the visual quality of the recovered damaged images is maintained; and 4) the method can be applied to distributed image storage system or multiple image transmission.

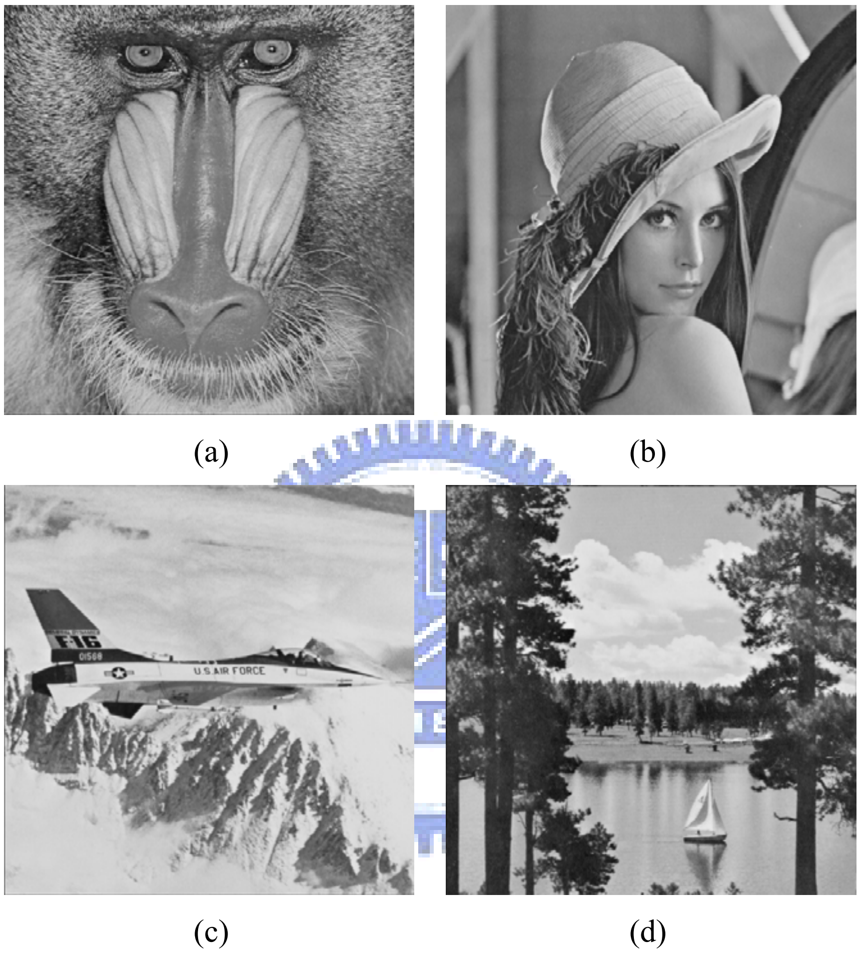
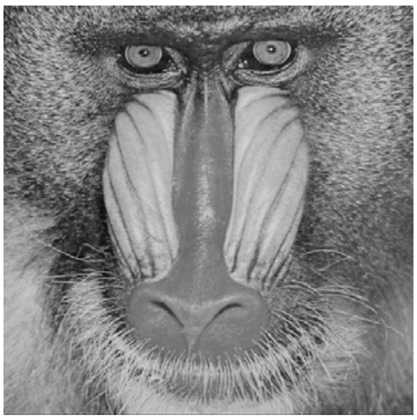
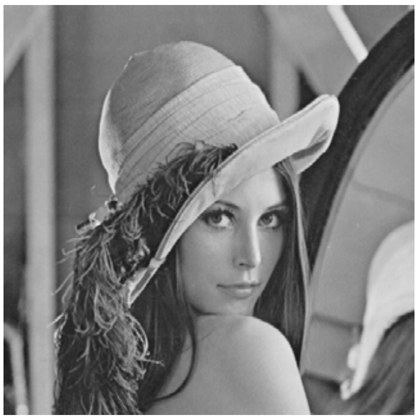
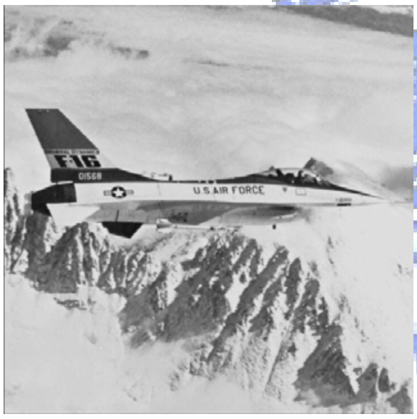


Fig. 5.2. A group of test images: (a) Baboon, (b) Lena, (c) Jet, and (d) Scene.


(a)


(b)



(c)

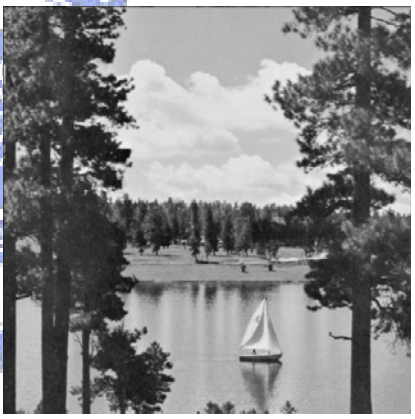

(d)

Fig. 5.3. The watermarked image of Fig. 5.2: (a) Baboon (PSNR = 44.17 dB), (b) Lena (PSNR = 44.14 dB), (c) Jet (PSNR = 44.16 dB), and (d) Scene (PSNR = 44.15 dB).

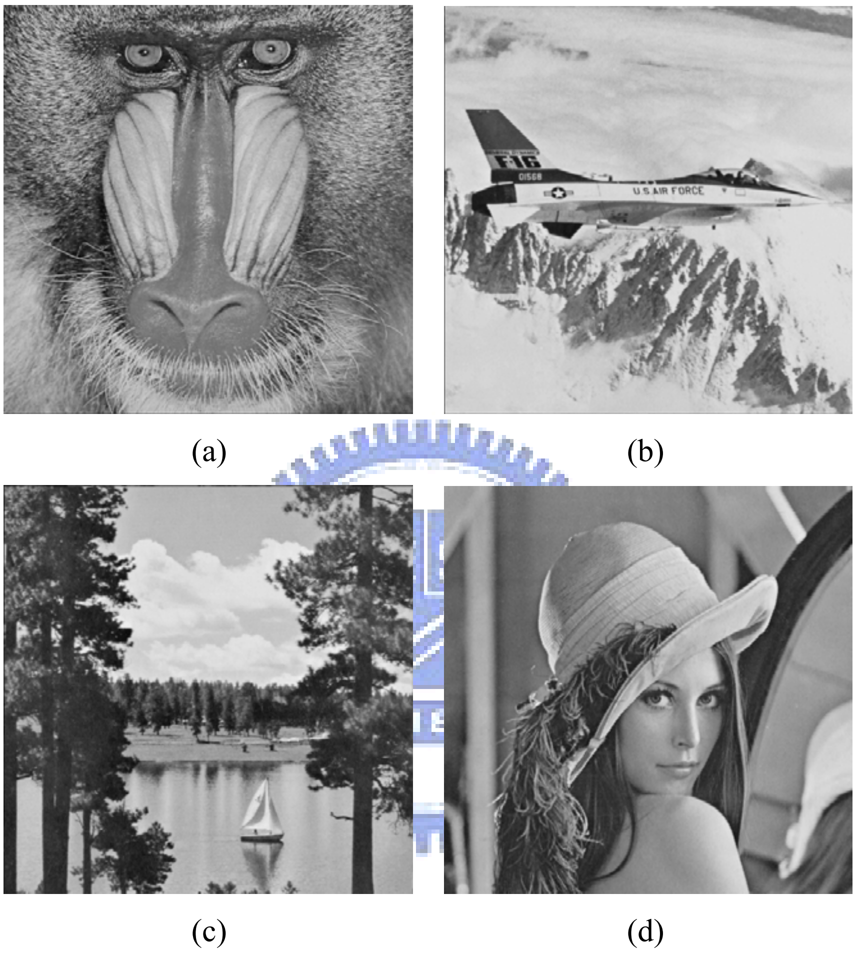
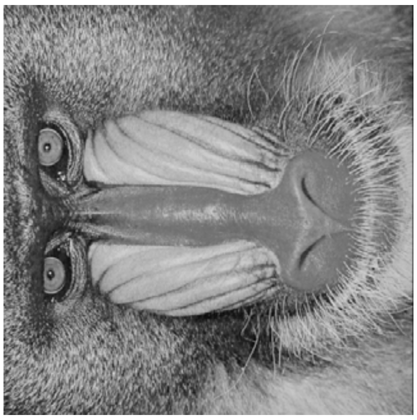
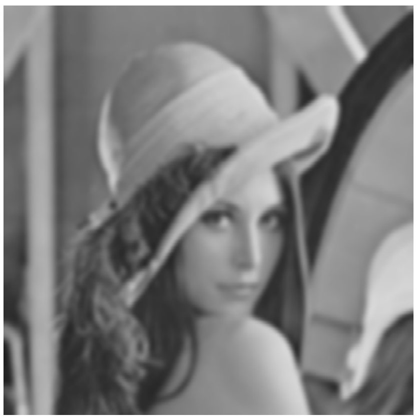
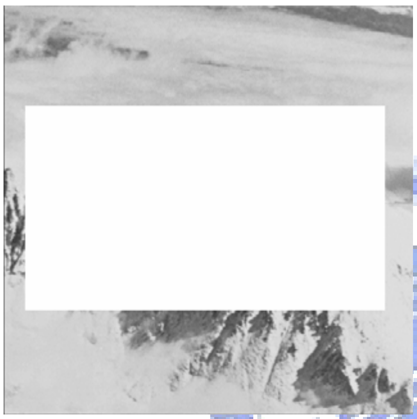
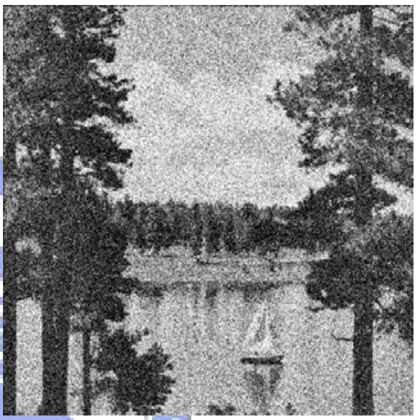


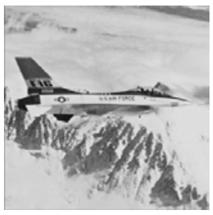
Fig. 5.4. The experiment result when the watermarked image Lena is lost. (a-c) The survived 44.1 dB images that passed the authentication test; (d) the recovered image Lena′ (PSNR = 41.57 dB) saved from using (a), (b), and (c).


(a)


(b)


(c)


(d)


(e)

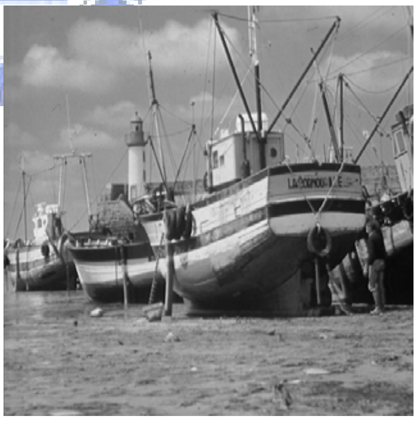

(f)

Fig. 5.5. Some alternations of Fig. 5.3: (a) the watermarked image Baboon is rotated through 90 degree; (b) the watermarked image Lena filtered by a low-pass filter; (c) the watermarked image Jet is cropped; (d) noise is uniformly added to the watermarked image Scene; (e) the watermarked image Jet is shrunk to a quarter of its size; and (f) the watermarked image Scene is completely replaced by another image Boat.

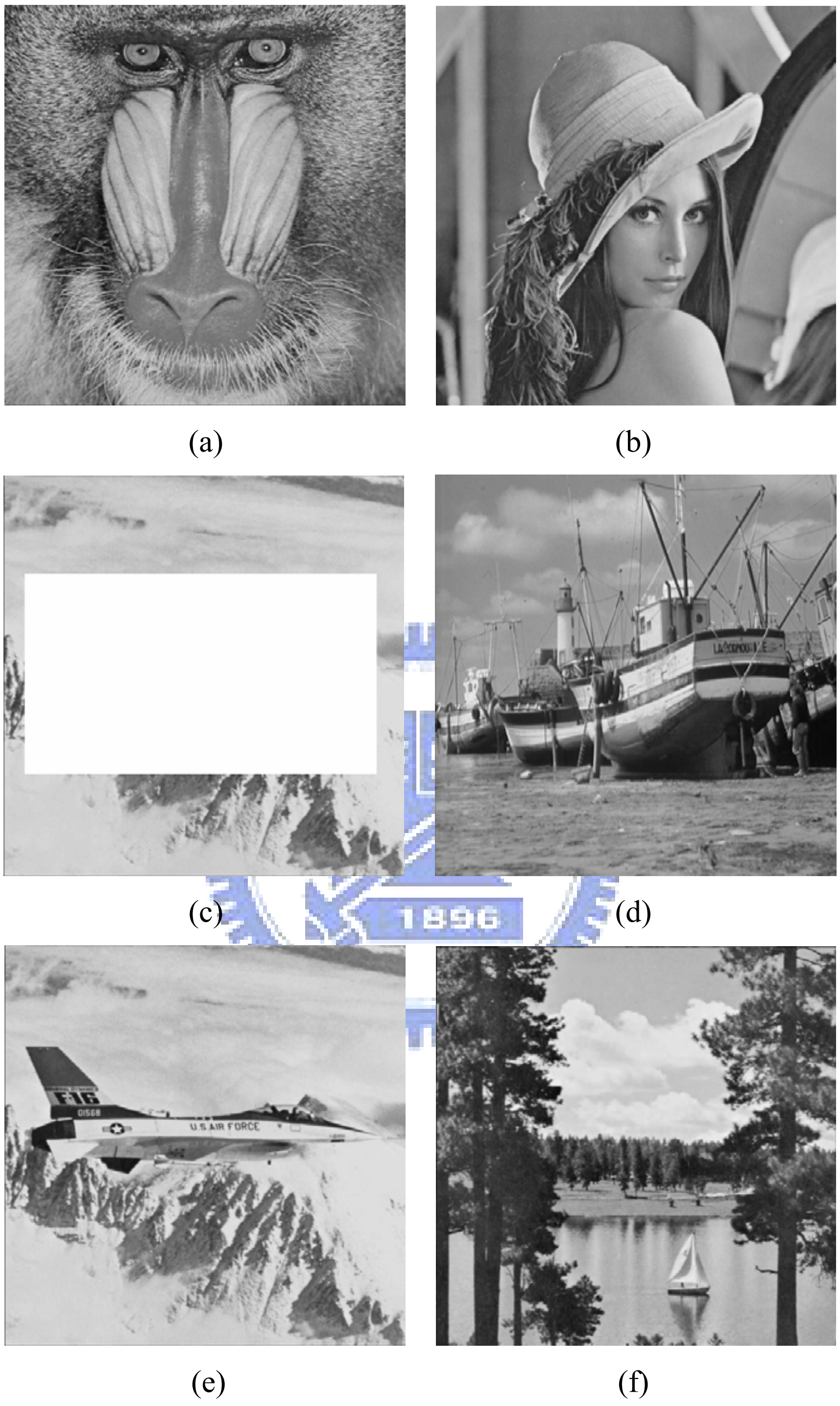


Fig. 5.6. An experiment when two images are tampered. (a-b) The watermarked images [Figs. 5.3(a) and 5.3(b)] that pass the authentication test; (c-d) the tampered images that fail the test, where (c) is the cropped watermarked image Jet, and (d) is when the watermarked image Scene is completely replaced by another image Boat. Finally, using (a) and (b), the two recovered images are the image Jet′ (PSNR = 42.54 dB) shown in (e), and the image Scene′ (PSNR = 38.32 dB) shown in (f).

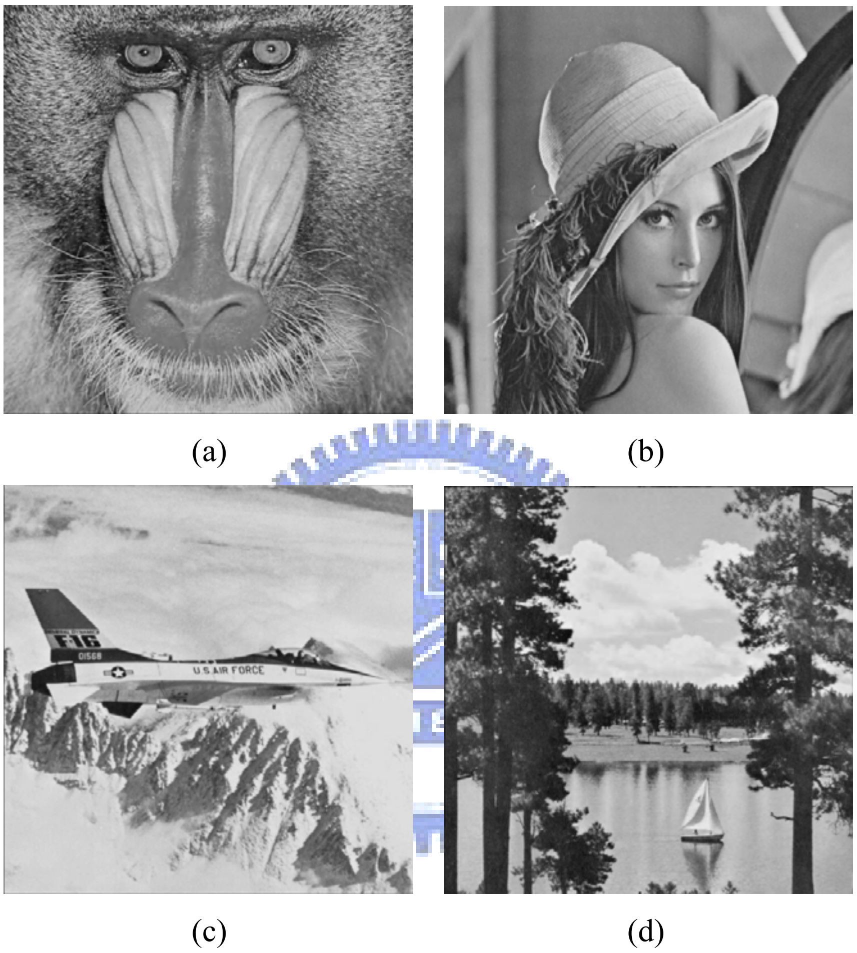
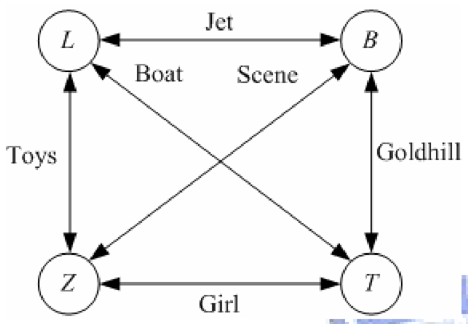
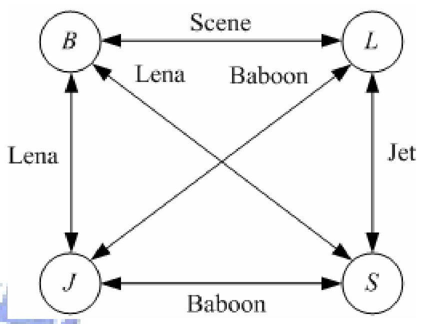


Fig. 5.7. The recovered versions (33.82-dB Baboon′, and/or 41.57-dB Lena′, and/or 42.54-dB Jet′, and/or 38.32-dB Scene′) of the two lost or tampered images in a sequence of ($r$=2, $n$=4) experiments. In each experiment, only two (=$r$) of the received images are exactly the watermarked images shown in Fig. 5.3 (and hence can pass the authentic test). The remaining $n$–$r$ = 4–2 = 2 images are either lost or tampered (for example, they can be any two images in Fig. 5.5).

Table 5.1. A comparison between some published methods and our scheme. (* means "quoted directly from the reported paper," and N/A means not mentioned in the reported paper.) The unit of PSNR is decibels.

| Schemes | PSNR of the watermarked image (images, if Ref. [98] or ours) | Authentication check ability | Recovery ability | PSNR of the recovered image (images, if Ref. [98] or ours) |
|---|---|---|---|---|
| Ref. [66] | 49.19* (Jet 512×512) | Yes | No | No |
| Ref. [67] | N/A | Yes | No | No |
| Ref. [69] | N/A | Yes | No | No |
| | | | | |
| Ref. [70] | 34.34* (Lena 512×512) | Yes | Self (single image) | N/A |
| Ref. [71] | 44.37* (Beach 256×256) | Yes | Self (single image) | 30.85*–48.48* |
| Ref. [74] | 44* (Jet 512×512) | Yes | Self (single image) | 32.82*–44* |
| Ref. [75] | N/A | Yes | Self (single image) | N/A |
| | | | | |
| Ref. [98] | 42.41–42.61 (four 600×600 images) | No (but it can be modified and become "yes") | ***Cross (Multiple images)*** | 27.92–39.63 for Fig. 5.8(b) |
| Ours | 44.14-44.17 (four 512×512 images) | Yes | ***Cross (Multiple images)*** | 33.82–42.54 |





(a) (b)

Fig. 5.8. (a) The relationships among four cover images {L, Z, B, T} in the original experiment of Ref. [98]. (b) The relationships among four cover images {B, L, J, S} in a new version slightly modified from Ref. [98] (it is an application version of Ref. [98], the hidden images being the JPEG2000 version of the cover images. *L* means Lena, *B* means Baboon, etc.).

Table 5.2. Comparison between two recovery methods for multi-images: 1) Ref. [98] using the structure shown in Fig. 5.8(b); and 2) our scheme with ($r$=2, $n$=4) threshold. (The unit of PSNR is the decibels.)

| Scheme | PSNR of the watermarked images | PSNR of the recovered images | Number of images allowed to be lost in the recovery procedure | Number of cover or water-marked images | Purpose of the design in the original paper | Features |
|---|---|---|---|---|---|---|
| Ref. [98] | 42.41－42.61 | 27.92－39.63 for Fig. 5.8(b) | None, if cover and hidden images are irrelevant [Fig. 5.8(a)]. One image if cover and hidden images are related [Fig. 5.8(b)] | 4 | To cover multiple secret images [So, its application to cross-recovery will use some structures such as Fig. 5.8(b)] | Faster processing speed, but less tolerable about missing images |
| Ours | 44.14－44.17 | 33.82－42.54 | $n-r$ = 4－2 = 2 images | $n$=4 in this case | To mutually support images of the same group | Slower processing speed, but more tolerable about missing images |

Table 5.3. The processing time of our method. (unit: second)

| Size of image | Encoding | Decoding |
|---|---|---|
| 256×256 | 7.25 | 5.52 |
| 512×512 | 12.36 | 8.47 |

# Chapter 6
# Conclusions and Future works

## 6.1 Conclusions

In this dissertation, we have proposed several techniques to protect important data and private images in a transmission or storage system. The techniques include two data hiding methods (the data hiding using VQ index file in Chapter 2, and the data hiding using MSOC and modulus function in Chapter 3) for protection of confidential images and two authentication-and-recovery methods (the sharing-based image authentication with self-recovery ability in Chapter 4, and the image authentication with cross-recovery ability in Chapter 5) for maintaining the integrity of important images.

In Chapter 2, we proposed a method for hiding secret data in the vector quantization (VQ) index file of an image. Because the cover media is the compression result of an ordinary image, the waiting time of the receiver end can be reduced. As the transmission of a compression file is a common activity, the transmission will not cause eavesdropper's perception. Although both our method and Chang et al's method [17] use the search-order coding (SOC) [13] to reduce the size of the resulted file, our approach has better bpp compression rate than Ref. [17] if hiding capacity is the same. Also, our method has better hiding capacity when the bpp compression rates are similar. Although some secret data are hidden in indices, the decompressed cover image is identical to the one decompressed by traditional VQ; moreover, the compression rate is often better than VQ's if we use $e$ = 1 mode (see Tables 2.2-2.3 and Figs. 2.4-2.6).

In Chapter 3, we develop an image hiding method in which the cover images are not necessarily bigger than the important images. The method is suitable for secret data whose amount is larger than that of Chapters 2. The scheme contains three parts: 1) MSOC encoding; 2) pseudo-random permutation; and 3) an embedding process using variance-based modulus function. The MSOC method makes the important image more compact and hence more suitable for the embedding procedure later. A user-specified non-negative integer threshold $T$ is introduced to control the length of the MSOC code, which in turn affects later the quality of the extracted important image. If the size of important image is not small, then we might need to use a larger value of $T$ to handle the case. However, if the size of important image is small, then the readers may just use $T$=1 so that the extracted important image is error-free. Therefore, the use of $T$ provides more flexibility for practical applications. To increase the security level, a pseudo-random permutation algorithm has been utilized by applying the MT pseudo-random number generator [9]. In the embedding part, we use a variance-based criterion to estimate the hiding capacity of a pixel in the cover image. The criterion is based on human visual system (HVS): more bits can be hidden in a pixel of busy area. From the experimental results, it can be seen that the variance-based estimation is more suitable than its counterpart in Ref. [32] which uses the pixel value to estimate the hiding capacity of a pixel. Experimental results show that the quality of both the stego-images and extracted important images are competitive to those obtained in many existing steganographic methods reported recently. From Table 3.3, it can be seen that the proposed method can create low-profile stego-images to protect the important image (because stego-images are with competitive qualities), and yet preserve the fidelity of the important image.

In Chapter 4, we propose a sharing-based watermarking method for image authentication, and it is with good self-recovery ability. The proposed method is suitable for the environment of protecting an important image's integrity. The method has the following functions: 1) detecting whether the watermarked image is tampered; 2) indicating the locations of the tampered area; 3) self-recovering the tampered portion using the non-tampered portion of the same watermarked image; and 4) enhancing the recovery ability by utilizing ($r$, $n$) threshold sharing [90], followed by scattering the shares all over the image. Feature (4) above gives the proposed method a good recovery rate. The sharing polynomial of Thien and Lin's method [90], which was devised to share a secret image among several participants, is used to reduce the amount of recovery data without significantly degrading the visual quality of the watermarked image. Experimental results (Figs. 4.5–4.8) and the comparison tables 4.1 and 4.2 show that the proposed method is very competitive. Our method has the following novelty compared with previously published schemes (particularly, image authentication or recovery methods [66-73]) and image sharing methods [3-4, 89-92, 94, 96]): i) The recovery data are generated by using VQ compression technique (an index file). A VQ index file has at least three advantages. i-a) The matched codeword of an image block is more suitable for showing the texture of the block than the mean value or the halftone result, as used in some other publications. i-b) VQ compression technique is block-based, and a block-based approach is sufficiently easy to apply for tampering recovery. i-c) VQ decompression is simple and has a very short decoding time, thus reducing the reconstruction cost. ii) To increase the survival rate of the recovery data (VQ-index file), a modified version of the ($r$, $n$) threshold sharing [90] is used to generate $n$ index shares. When applying secret sharing techniques to images, people often generated $n$ shares for each pixel (or for each block of the pixels) of the secret image. Hence, the share value has a wide range, and each share is represented by $8$ bits, where $8$ is the number of bits per pixel. However, the proposed method assumes that the codebook has $L$ codewords, and divides each VQ-index value into $r$

sections before generating *n* index shares for each index value. Thus, the size of each index share is reduced to $\lceil (log_2 L)/r \rceil$, which is usually smaller than 8 (for example, $\lceil (log_2 L)/r \rceil$=2 bits if ($L$=1024, $r$=5), or 3 bits if ($L$=4096, $r$=4)). The smaller size of each share helps maintain the quality of the watermarked image after embedding. iii) Many published image recovery techniques embed the recovery data of a unit (i.e. a block or a pixel) for backuping into another block or pixel according to a permutation function. (Notably, having many copies of the recovery data might increase survival rate, while decreasing the quality of the watermarked image.) Ours is sharing-based, and each share is small in size. Hence, unlike published recovery methods, the proposed method allows many backup shares (*n* shares) without significantly increasing the total size of recovery data. Since ours has more backup pieces (*n* shares rather than 1 or 2 copies), we can uniformly scatter the backup the recovery data (*in a distributed and missing-allowable manner*) in the whole host image to resist a cropping attack of an extensive area.

In Chapter 5, we propose an authentication and cross-recovery method to protect a *group* of *n* images. The goal is that if some images in the group are tampered or lost, these images can be identified and reconstructed by the mutual support of the *r* survived members ($r<n$ is a specified threshold). In the proposed method, a two-layer sharing algorithm is designed to create *n* shadows that share the recovery data. The two-layer sharing not only reduces the recovery data amount, but also makes the recovery fault tolerant, for up to $n-r$ shadows can be lost. The features and the experimental results show that: i) the quality of our watermarked images is acceptable, i.e., the proposed method keeps the transparency of the hidden data, including the recovery data and watermark; ii) authentication and cross-recovery can both be done when some watermarked images are altered, tampered, or lost; iii) the visual quality of the recovered damaged images is maintained; and iv) the method can be applied to

distributed image storage system or multiple image transmission. Finally, we show the comparison of Chapter 4 and Chapter 5 in Table 6.1.

Table 6.1. The comparison of Chapter 4 and Chapter 5.

| | Chapter 4<br>Self-recovery | Chapter 5<br>Cross-recovery |
|---|---|---|
| Processing time | Faster<br>(One-layer sharing) | Slower<br>(Two-layer sharing) |
| Image's quality (PSNR) | Watermarked: 44.1 dB<br>Recovery: (Quality is varied according to the area of the tampered portions in an image.)<br>34.77 (cropping 50%) ~ 43.2 (cropping 6%) | Watermarked: 44.1 dB<br>Recovery: (Fixed quality for an image, no matter how big the damaged area is.)<br>33.82 ~ 42.54 |
| When to use? | When some parts (e.g. 25%) of an image are tampered. (If the tampered portions are not too large, then the quality of the recovered image is better than that of Chapter 5.) | If the tampered portions of an image exceed 50%, or even worse, if the whole image is lost. |

## 6.2 Future works

Below are some extensions of the proposed method in the future.

1. Reversible data hiding becomes a popular research issue in recent years. In general, the hiding capacity of a reversible data hiding method is not high; this is because some hiding space must be preserved for embedding the information of the original image. We may try to extend our method or design a new one so that the extension or new design is reversible, and with acceptable hiding capacity as a worthy, but not easy, target.

2. In the future, we plan to extend the proposed method in Chapter 5 to equip with self-recovery ability for each member image, as long as the damage to each image is small in area. This can reduce the time waiting for the cross-recovery data from other computers storing other images. The main difficulty is that the amount of the recovery data becomes very enormous, so it is a challenge to embed those recovery data in the images without causing too more visual distortions to the watermarked images.
3. In the topic of image authentication, because palette-based images (such as those in GIF format) are also popular in Internet applications such as WWW homepage, we can try to design an image authentication method to verify the integrity of this kind of images, and equipped with tampering-recovery function.
4. The statistical style steganalysis methods are powerful tools to reveal tiny alternation in an image’s statistical behavior caused by hiding activities. Therefore, designing data hiding methods that can pass steganalysis is always a job worthy of trying.

# References

[1] National Bureau of Standards, “Data Encryption Standard,” U.S. Department of Commerce, Federal Information Processing Standards Publication 46, January 1977.

[2] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public key cryptosystems,” *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126, 1978.

[3] A. Shamir, “How to share a secret,” *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613, 1979.

[4] G. R. Blakey, “Safeguarding cryptography keys,” *Proceedings of AFIPS 1979 National Computing Conference*, New York, Vol. 48, pp. 313–317, 1979.

[5] M. Naor and A. Shamir, “Visual cryptography,” *Advances in Cryptography - Eurocrypt’94, Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Vol. 950, pp. 1–12, 1995.

[6] G. R. Chen, Y. B. Mao, and C. K. Chui, “A symmetric image encryption scheme based on 3D chaotic cat maps,” *Chaos, Solitons & Fractals*, Vol. 21, No. 3, pp. 749–761, 2004.

[7] Hossan El-din H. Ahmed, Hamdy M. Kalash, and Osama S. Farag Allah, “Encryption quality analysis of the RC5 block cipher algorithm for digital images,” *Optical Engineering*, Vol. 45, No. 10, pp. 107003–(1–7), 2006.

[8] R. L. Rivest, “The MD5 message digest algorithm,” Request for Comments (RFC) 1321, Internet Activities Board, Internet Privacy Task Force, April 1992.

[9] M. Matsumoto and T. Nishimura, “Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator,” *ACM Transactions on*

*Modeling and Computer Simulation*, Vol. 8, No. 1, pp. 3–30, 1998.

[10] M. Y. Rhee, "Cryptography and Secure Communication," McGraw-Hill Book Co, Singpore, 1994.

[11] V. Klima, "Finding MD5 collisions - A toy for a notebook," Cryptology ePrint Archive, Report 2005/075, 2005.

[12] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," *Advances in Cryptology – CRYPTO 2005, Lecture Notes in Computer Science*, Springer-Verlag, Vol. 3621, pp. 17–36, 2005.

[13] C. H. Hsieh and J. C. Tsai, "Lossless compression of VQ index with search-order coding," *IEEE Transactions on Image Processing*, Vol. 5, No. 11, pp. 1579–1582, 1996.

[14] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, Vol. 28, No. 1, pp. 84–95, 1980.

[15] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Information Theory*, Vol. 44, No. 6, pp. 2325–2383, 1998.

[16] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding – A survey," *Proceedings of the IEEE*, Vol. 87, No. 7, pp. 1062–1078, 1999.

[17] C. C. Chang, G. M. Chen, and M. H. Lin, "Information hiding based on search-order coding for VQ indices," *Pattern Recognition Letters*, Vol. 25, No. 11, pp. 1253–1261, 2004.

[18] R. Z. Wang, 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.

[19] C. C. Chang, M. H. Lin, and Y. C. Hu, "A fast and secure image hiding scheme based on LSB substitution," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 16, No. 4, pp. 399–416, 2002.

[20] C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB

substitution," *Pattern Recognition*, Vol. 37, No. 3, pp. 469–474, 2004.

[21] C. C. Chang, 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.

[22] K. L. Chung, C. H. Shen, and L. C. Chang, "A novel SVD- and VQ-based image hiding scheme," *Pattern Recognition Letters*, Vol. 22, No. 9, pp. 1051–1058, 2001.

[23] Y. C. Hu and M. H. Lin, "Secure image hiding scheme based upon vector quantization," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 18, No. 6, pp. 1111–1130, 2004.

[24] Y. C. Hu, "High-capacity image hiding scheme based on vector quantization," *Pattern Recognition*, Vol. 39, No. 9, pp. 1715–1724, 2006.

[25] C. C. Chang and H. W. Tseng, "A steganographic method for digital images using side match," *Pattern Recognition Letters*, Vol. 25, No. 12, pp. 1431–1437, 2004.

[26] C. C. Chang and T. C. Lu, "Reversible index-domain information hiding scheme based on side-match vector quantization," *The Journal of Systems and Software*, Vol. 79, No. 8, pp. 1120–1129, 2006.

[27] R. Z. Wang and Y. D. Tsai, "An image-hiding method with high hiding capacity based on best-block matching and *k*-means clustering," *Pattern Recognition*, Vol. 40, No. 2, pp. 398–409, 2007.

[28] D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recognition Letters*, Vol. 24, No. 10, pp. 1613–1626, 2003.

[29] H. C. Wu, N. I. Wu, C. S. Tsai, and M. S. Hwang, "Image steganographic scheme based on pixel-value differencing and LSB replacement methods," *IEE*

*Proceedings – Vision, Image and Signal Processing*, Vol. 152, No. 5, pp. 611–615, 2005.

[30] S. L. Li, K. C. Leung, L. M. Cheng, and C. K. Chan, "A novel image-hiding scheme based on block difference," *Pattern Recognition*, Vol. 39, No. 6, pp. 1168–1176, 2006.

[31] C. C. Thien and J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function," *Pattern Recognition*, Vol. 36, No. 12, pp. 2875–2881, 2003.

[32] S. J. Wang, "Steganography of capacity required using modulo operator for embedding secret image," *Applied Mathematics and Computation*, Vol. 164, No. 1, pp. 99–116, 2005.

[33] C. C. Chang, C. S. Chan, and Y. H. Fan, "Image hiding scheme with modulus function and dynamic programming strategy on partitioned pixels," *Pattern Recognition*, Vol. 39, No. 6, pp. 1155–1167, 2006.

[34] C. C. Chang, T. S. Chen, and L. Z. Chung, "A steganographic method based upon JPEG and quantization table modification," *Information Sciences*, Vol. 141, No. 1, pp. 123-138, 2002.

[35] Y. C. Tseng, Y. Y. Chen, and H. K. Pan, "A secure data hiding scheme for binary images," *IEEE Trans. Communications*, Vol. 50, No. 8, pp. 1227–1231, 2002.

[36] S. S. Maniccam and N. Bourbakis, "Lossless compression and information hiding in images," *Pattern Recognition*, Vol. 37, No. 3, pp. 475–486, 2004.

[37] M. Y. Wu, Y. K. Ho, and J. H. Lee, "An iterative method of palette-based image steganography," *Pattern Recognition Letters*, Vol. 25, No. 3, pp.301–309, 2004.

[38] Xinpeng Zhang and Shuozhong Wang, "Steganography using multiple-base notational system and human vision sensitivity," *IEEE Signal Processing Letters*, Vol. 12, No. 1, pp. 67–70, 2005.

[39] Y. H. Yu, C. C. Chang, and Y. C. Hu, "Hiding secret data in images via predictive coding," *Pattern Recognition*, Vol. 38, No. 5, pp. 691–705, 2005.

[40] E. Besdok, "Hiding information in multispectral spatial images," *AEU – International Journal of Electronics and Communications*, Vol. 59, No. 1, pp. 15–24, 2005.

[41] R. Z. Wang and Y. S. Chen, "High-payload image steganography using two-way block matching," *IEEE Signal Processing Letters*, Vol. 13, No. 3, pp. 161–164, 2006.

[48] C. Y. Yang and J. C. Lin, "Image hiding by base-oriented algorithm," *Optical Engineering*, Vol. 45, No. 11, pp. 117001–(1–10), 2006.

[49] T. Y. Liu and W. H. Tsai, "A new steganographic method for data hiding in Microsoft Word documents by a change tracking technique," *IEEE Trans. Information Forensics and Security*, Vol. 2, No. 1, pp. 24–30, 2007.

[50] Y. H. Yu, C. C. Chang, and I. C. Lin, "A new steganographic method for color and grayscale image hiding," *Computer Vision and Image Understanding*, Vol. 107, No. 3, pp.183–194, 2007.

[51] N. F. Johnson and S. Jajodia, "Steganalysis of images created using current steganography software," *Lecture Notes in Computer Science*, Vol. 1525, Springer, Berlin, pp. 273–289, 1998.

[52] A. Westfeld and A. Pfitzmann, "Attack on steganographic systems," *Lectures Notes in Computer Science*, Vol. 1768, pp. 61–75, 2000.

[53] P. A. Watters, F. Martin, and S. H. Stripf, "Visual steganalysis of LSB-encoded natural images," *International Conference on Information Technology and Applications*, Vol. 1, pp. 746–751, 2005.

[54] B. B. Zhu, M. D. Swanson, and A. H. Tewfik, "When seeing isn't believing," *IEEE Signal Processing Magazine*, Vol. 21, No. 2, pp. 40–49, 2004.

[55] C. Y. Lin and S. F. Chang, "Robust digital signature for multimedia authentication: a summary," *IEEE Circuits and Systems Magazine*, October 2003.

[56] C. S. Lu, S. K. Huang, C. J. Sze, and H. Y. M. Liao, "Cocktail watermarking for digital image protection," *IEEE Trans. Multimedia*, Vol. 2, No. 4, pp. 209–224, 2000.

[57] Y. Wang, J. F. Doherty, and R. E. Van Dyck, "A wavelt-based watermarking algorithm for ownership verification of digital images," *IEEE Trans. Image Process.*, Vol. 11, No. 2, pp. 77–88, 2002.

[58] D. C. Lou, J. M. Shieh, and H. K. Tso, "A robust buyer-seller watermarking scheme based on DWT," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 20, No. 1, pp. 79–90, 2006.

[59] C. S. Lu and H. Y. M. Liao, "Multipurpose watermarking for image authentication and protection," *IEEE Trans. Image Processing*, Vol. 10, No. 10, pp. 1579–1592, 2001.

[60] Z. M. Lu, D. G. Xu, and S. H. Sun, "Multipurpose image watermarking algorithm based on multistage vector quantization," *IEEE Trans. Image Processing*, Vol. 14, No. 6, pp. 822–831, 2005.

[61]C. S. Lu and H. Y. M. Liao, "Structural digital signature for image authentication: An incidental distortion resistant scheme," *IEEE Trans. Multimedia*, Vol. 5, No. 2, pp. 161–173, 2003.

[62] C. W. Wu, "On the design of content-based multimedia authentication systems," *IEEE Trans. Multimedia*, Vol. 4, No. 3, pp. 385–393, 2002.

[63] C. Y. Lin and S. F. Chang, "A robust image authentication method distinguishing JPEG compression from malicious manipulation," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 11, No. 2, pp. 153–168, 2001.

[64] P. Tsai, Y. C. Hu, and C. C. Chang, “Using set partitioning in hierarchical tree to authenticate digital images,” *Signal Process.: Image Comm.*, Vol. 18, No. 9, pp. 813–822, 2003.

[65] A. H. Paquet, R. K. Ward, and I. Pitas, “Wavelet packets-based digital watermarking for imge verification and authentication,” *Signal Processing*, Vol. 83, No. 10, pp. 2117–2132, 2003.

[66] C. C. Chang, Y. S. Hu, and T. C. Lu, “A watermarking-based image ownership and tampering authentication scheme,” *Pattern Recognition Letters*, Vol. 27, No. 5, pp. 439–446, 2006.

[67] M. U. Ceilk, G. Sharma, E. Saber, and A. M. Tekalp, “Hierarchical watermarking for secure image authentication with localization,” *IEEE Trans. Image Process.*, Vol. 11, No. 6, pp. 585–594, 2002.

[68] P. W. Wong, “A public key watermark for image verification and authentication,” in *Proc. IEEE Int. Conf. Image Processing, Chicago, IL, USA*, pp. 425–429, October 1998.

[69] P. W. Wong and N. Memon, “Secret and public key image watermarking schemes for image authentication and ownership verification,” *IEEE Trans. Image Process.*, Vol. 10, No. 10, pp. 1593–1601, 2001.

[70] H. C. Wu and C. C. Chang, “Detection and Restoration of Tampered JPEG Compressed Images,” *The Journal of Systems and Software*, Vol. 64, No. 2, pp. 151–161, 2002.

[71] P. L. Lin, C. K. Hsieh, and P. W. Huang, “A hierarchical digital watermarking method for image tamper detection and recovery,” *Pattern Recognition*, Vol. 38, No. 12, pp. 2519–2529, 2005.

[72] H. Luo, S. C. Chu, and Z. M. Lu, “Self embedding watermarking using halftoning technique,” *Circuits Systems and Signal Processing*, Vol. 27, No. 2, pp.

155–170, 2008.

[73] S. S. Wang and S. L. Tsai, "Automatic image authentication and recovery using fractal code embedding and image inpainting," *Pattern Recognition*, Vol. 41, No. 2, pp. 701–712, 2008.

[74] F. H. Yeh and G. C. Lee, "Content-based watermarking in image authentication allowing remedying of tampered images," *Optical Engineering*, Vol. 45, No. 7, pp. 077004-1–10, 2006.

[75] C. S. Chan and C. C. Chang, "An efficient image authentication method based on Hamming code," *Pattern Recognition*, Vol. 40, No. 2, pp. 681–690, 2007.

[76] S. J. Lin (Supervised by J. C. Lin), "Image Recovery by using Sharing," Mater thesis, National Chiao Tung University, 2006.

[77] M. S. Wang and W. C. Chen, "A majority-voting based watermarking scheme for color image tamper detection and recovery," *Computer Standards and Interfaces*, Vol. 29, No. 5, pp. 561–570, 2007.

[78] K. H. Chiang, K. C. Chang-Chien, R. F. Chang, and H. Y. Yen, "Tamper detection and restoring system for medical images using wavelet-based reversible data embedding," *Journal of Digital Imaging*, Vol. 21, No. 1, pp. 77–90, 2008.

[79] Z. F. Yang and W. H. Tsai, "Watermark approach to embedded signature-based authentication by channel statistics," *Optical Engineering*, Vol. 42, No. 4, pp. 1157–1165, 2003.

[80] H. T. Lu, R. M. Shen, and F. L. Chung, "Fragile watermarking scheme for image authentication," *Electronics Letters*, Vol. 39, No. 12, pp. 898–900, 2003.

[81] Chang-Tsun Li, "Digital fragile watermarking scheme for authentication of JPEG images," *IEE Proceedings – Vision, Image and Signal Processing*, Vol. 151, No. 6, pp. 460–466, 2004.

[82] S. Suthaharan, “Fragile image watermarking using a gradient image for improved localization and security,” *Pattern Recognition Letters*, Vol. 25, No. 16, pp. 1893–1903, 2004.

[83] M. U. Celik, G. Sharma, and A. M. Tekalp, “Lossless watermarking for image authentication: a new framework and an implementation,” *IEEE Trans. Image Processing*, Vol. 15, No. 4, pp. 1042–1049, 2006.

[84] C. H. Tzeng, Z. F. Yang, and W. H. Tsai, “Adaptive data hiding in palette images by color ordering and mapping with security protection,” *IEEE Trans. Communications*, Vol. 52, No. 5, pp. 791–800, 2004.

[85] C. H. Tzeng and W. H. Tsai, “A combined approach to integrity protection and verification of palette images using fragile watermarks and digital signatures,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E87-A, No. 6, pp. 1612–1619, 2004.

[86] C. C. Chang and P. Y. Lin, “A color image authentication method using partitioned palette and morphological operations,” *IEICE Transactions on Information and Systems*, Vol. E91-D, No. 1, pp. 54–61, 2008.

[87] M. Holliman and N. Memon, “Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes,” *IEEE Trans. Image Processing*, Vol. 9, No. 3, pp. 432–441, 2000.

[88] J. Fridrich, M. Goljan, and N. Memon, “Further attacks on Yeung-Mintzer fragile watermarking scheme,” in *Proceedings of SPIE Conference on Security and Watermarking of Multimedia Contents*, Vol. 3971, pp. 428–437, 2000.

[89] C. C. Chang and R. J. Huang, “Sharing secret images using shadow codebooks,” *Information Sciences*, Vol. 111, No. 1－4, pp. 335–345, 1998.

[90] C. C. Thien and J. C. Lin, “Secret image sharing,” *Computers and Graphics*, Vol. 26, No. 5, pp. 765–770, 2002.

[91] C. C. Thien and J. C. Lin, "An image-sharing method with user-friendly shadow images," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 12, pp. 1161–1169, 2003.

[92] C. C. Lin and W. H. Tsai, "Secret image sharing with steganography and authentication," *The Journal of Systems and Software*, Vol. 73, No. 3, pp. 405–414, 2004.

[93] S. K. Chen and J. C. Lin, "Fault-tolerant and progressive transmission of images," *Pattern Recognition*, Vol. 38, No. 12, pp. 2466–2471, 2005.

[94] R. Z. Wang and S. J. Shyu, "Scalable secret image sharing," *Signal Processing: Image Communication*, Vol. 22, No. 4, pp. 363–373, 2007.

[95] S. J. Lin and J. C. Lin, "VCPSS: A two-in-one two-decoding-options image sharing method combining visual cryptography (VC) and polynomial-style sharing (PSS) approaches," *Pattern Recognition*, Vol. 40, No. 12, pp. 3652－3666, 2007.

[96] C. C. Lin and W. H. Tsai, "Secret image sharing with capability of share data reduction," *Optical Engineering*, Vol. 42, No. 8, pp. 2340–2345, 2003.

[97] J. B. Feng, H. C. Wu, C. S. Tsai, and Y. P. Chu, "A new multi-secret images sharing scheme using Largrange's interpolation," *The Journal of Systems and Software*, Vol. 76, No. 3, pp. 327–339, 2005.

[98] C. S. Tsai, C. C. Chang, and T. S. Chen, "Sharing multiple secrets in digital images," *The Journal of Systems and Software*, Vol. 64, No. 2, pp. 163–170, 2002.

[99] C. C. Lin and W. H. Tsai, "Visual cryptography for gray-level images by dithering techniques," *Pattern Recognition Letters*, Vol. 24, No. 1–3, pp. 349–358, 2003.

[100] H. C. Wu and C. C. Chang, "Sharing visual multi-secrets using circle shares,"

*Computer Standards and Interfaces*, Vol. 28, No. 1, pp. 123–135, 2005.

[101] R. Z. Wang and C. H. Su, “Secret image sharing with smaller shadow images,” *Pattern Recognition Letters*, Vol. 27, No. 6, pp. 551–555, 2006.

[102] C. N. Yang, T. S. Chen, K. H. Yu, and C. C. Wang, “Improvements of image sharing with steganography and authentication,” *The Journal of Systems and Software*, Vol. 80, No. 7, pp. 1070–1076, 2007.

[103] C. Y. Lin and C. C. Chang, “Hiding data in VQ-compressed images using dissimilar pairs,” *Journal of Computers*, Vol. 17, No. 2, pp. 3–10, 2006.

[104] C. C. Chang, T. D. Kieu, and Y. C. Chou, “Reversible information hiding for VQ indices based on locally adaptive coding,” *Journal of Visual Communication and Image Representation*, Vol. 20, No. 1, pp. 57–64, 2009.

# Vita

YU-JIE CHANG was born in Kaohsiung, Taiwan, in 1977. He received the BS in computer science and information engineering in 1999 from National Central University, Taiwan. In 2001, he received his MS in computer and information science from National Chiao Tung University. He is now a PhD candidate in the computer science department of National Chiao Tung University. His research interests include digital watermarking, image processing, and pattern recognition.

# Publication List of Yu-Jie Chang

## A. Journal papers

**Accepted or Published:**

1. Yu-Jie Chang, Ran-Zan Wang, and Ja-Chen Lin, "Hiding Images Using Modified Search-Order Coding and Modulus Function," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 22, No. 6, pp. 1215–1240, 2008. (SCI)
2. Kuo-Hsien Hung, Yu-Jie Chang, and Ja-Chen Lin, "Progressive sharing of an image," *Optical Engineering*, Vol. 47, No. 4, 047006, 2008. (SCI)
3. Yu-Jie Chang, Sian-Jheng Lin, and Ja-Chen Lin, "Authentication and Cross-Recovery for Multiple Images," *Journal of Electronic Imaging*, Vol. 17, No. 4, 043007, 2008. (SCI)
4. Yu-Jie Chang, Ran-Zan Wang, and Ja-Chen Lin, "A sharing-based fragile watermarking method for authentication and self-recovery of image tampering," *EURASIP Journal on Advances in Signal Processing*, accepted. (SCI)

## B. Conference papers

**Published:**

1. Yu-Jie Chang and Ran-Zan Wang, "An Image Steganography Based on Modified Search-Order Coding," *Proceedings of the 2006 IPPR Conference on Computer Vision, Graphics and Image Processing*, Tao-yuan, Taiwan, R.O.C., Aug. 2006.
2. Yu-Jie Chang and Ja-Chen Lin, "Data Hiding Using VQ Index File," *Proceedings*

*of IEEE International Conference on Intelligence and Security Informatics (ISI 2008)*, Taipei, Taiwan, June 2008. (EI)

3. Yu-Jie Chang, Shang-Kuan Chen, and Ran-Zan Wang, “Image Steganography Using Maximum Cross-Number Plane,” *2008 International Conference on Digital Contents (ICDC 2008)*, Taoyuan, Taiwan, 2008.