

國立交通大學

資訊學院

資訊科學與工程研究所

博士論文

高容量 JPEG 資訊隱藏及其在影像分享、驗證與
修復之應用

High-Capacity JPEG Data Hiding and its Applications to
Image Sharing, Authentication, and Recovery

研究生：陳李書滕

指導教授：林志青 博士

中華民國九十九年六月

高容量 JPEG 資訊隱藏及其在影像分享、驗證與
修復之應用

High-Capacity JPEG Data Hiding and its Applications to
Image Sharing, Authentication, and Recovery

研究生：陳李書滕

Student: Lee Shu-Teng Chen

指導教授：林志青 博士

Advisor: Dr. Ja-Chen Lin



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 Science and Information Engineering
June 2010
Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

高容量 JPEG 資訊隱藏及其在影像分享、驗證與修復之應用

研究生：陳李書滕

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

本論文基於 Joint Photographic Experts Group (JPEG) 技術提出了資訊隱藏、影像分享、影像驗證與修復之方法。資訊隱藏是藉由將機密資料藏於掩護影像，使得攻擊者無法從偽裝影像中察覺出機密資料的存在，以達到保護機密資料的一種方式；然而，大部份資訊隱藏方法所產生的偽裝影像大小並不經濟。因此，將偽裝影像壓縮以節省儲存空間是必要的，但壓縮經常會破壞藏於偽裝影像的機密資料。為了解決這個問題，我們提出了一個高容量可逆式 JPEG 資訊隱藏方法。這個方法將機密資料藏於 JPEG 壓縮碼。在機密資料被無失真取出後，原來用於藏機密資料的 JPEG 壓縮碼，也可以完整重建回來。

機密影像分享藉由將機密影像產生 n 張有雜訊的分存來保護機密影像。任意 k 份分存 ($2 \leq k \leq n$)，可以還原機密影像，但少於 k 份分存，則無法還原機密影像。這種性質(不是機密影像被還原就是只有雜訊)，只有當被分享的影像為最高機密才有用處；然而，在真實世

界中，並不是所有的影像都為最高機密，分享的影像有可能是每天需要處理的重要影像，而不是最高機密影像。因此，還原重要影像可能會涉及到某些品質層次。為了可以根據收集分存的數量多寡，來還原重要影像的多種品質，我們提出了一個漸進式影像分享方法。這個方法所產生的每個分存都非常的小，所以這些小分存可以成功地藏於 JPEG 壓縮碼，使得這些隱藏於 JPEG 壓縮碼下的小分存在一個不友善的環境傳送時，可以減低被攻擊的機率。

影像驗證是用來檢驗影像的完整性。有些具有自我修復能力的影像驗證方法除了可以偵測影像是否遭到惡意竄改，而且在影像被偵測遭到竄改時，也可進而修復；然而，當影像遭到更嚴重破壞時，所遺留的修復資料並不足以用來修復。因此，我們提出了一個影像驗證與交叉修復方法來保護一組 n 張 JPEG 影像。這個方法可以檢驗該組 n 張 JPEG 影像有哪幾張被竄改，而且還可以藉由其它未被竄改的 JPEG 影像之間的相互合作來修復被竄改的 JPEG 影像。

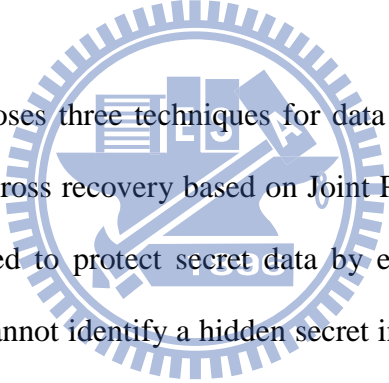
High-Capacity JPEG Data Hiding and its Applications to Image Sharing, Authentication, and Recovery

Student : Lee Shu-Teng Chen

Advisor : Dr. Ja-Chen Lin

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

Abstract



This dissertation proposes three techniques for data hiding, image sharing, and image authentication with cross recovery based on Joint Photographic Experts Group (JPEG). Data hiding is used to protect secret data by embedding them in a cover image such that attackers cannot identify a hidden secret in the stego-image. However, the stego-images generated by many existing data hiding techniques are not economic in size, and therefore the compression to the stego-images is needed to reduce storage space. Unfortunately, the compression usually destroys the hidden secret in the stego-images. To solve this dilemma, a reversible JPEG data hiding method with high hiding capacity is proposed. The secret data are embedded in a JPEG code. In the decoding process, following lossless extraction of the hidden secret data, the JPEG code used to embed the secret data can also be reconstructed without any error.

Secret image sharing is used to protect a secret image by splitting the secret image into n noise-like shadows. Any k of the n shadows ($2 \leq k \leq n$) can reconstruct the secret image, but fewer than k shadows cannot. This all-or-nothing property is

useful when the image being shared is top secret. However, in the real world, not all images are top secret. The shared image might be a daily-used important image but not a top-secret one, and therefore the reconstruction of the important image can involve certain quality levels. To reconstruct the important image with various quality levels based on the number of collected shadows, a progressive image sharing method with compact shadows is proposed. All n shadows are very compact, and so can be hidden successfully in the JPEG codes of cover images to reduce the probability of being attacked when transmitted in an unfriendly environment.

Image authentication is used to verify the integrity of an image. Some Image authentication schemes with self-recovery ability not only identify unauthorized manipulations of the image but also recover the tampered areas with approximation of the original ones. However, when the image is seriously damaged, the recovery data left are insufficient to recover the the seriously damaged image. Therefore, an image authentication and cross-recovery method is proposed to protect a group of n JPEG image. The proposed method can verify which JPEG images in the group are tampered with, and recover the tampered JPEG images approximately through the cooperation of the survived members.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Ja-Chen Lin, for his invaluable assistance and helpful guidance. I also wish to thank to Dr. Shang-Kuan Chen, Dr. Wen-Pinn Fang, Dr. Yu-Jie Chang, Dr. Kun-Yuan Chao, Mr. Sian-Jheng Lin, and Mr. Suiang-Shyan Lee for their discussion and suggestions. I would like to thank all members of Computer Vision Laboratory for their help. Last, I would like to express my deepest appreciation to my family and my friends for their encouragement and support.

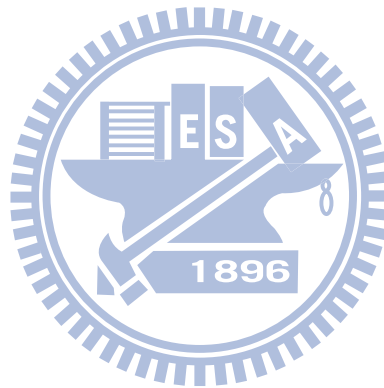


Table of Contents

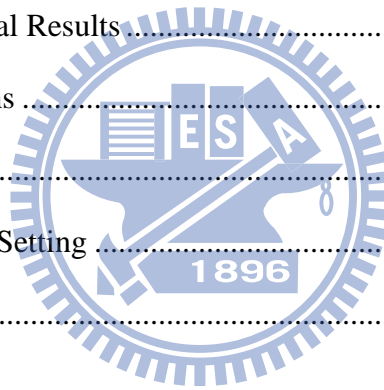
Abstract in Chinese.....	I
Abstract in English	III
Table of Contents	V
List of Figures.....	VIII
List of Tables	XI

Chapter 1 Introduction.....	1
1.1 Motivation.....	1
1.2 Related Studies.....	5
1.2.1 JPEG Data Hiding.....	5
1.2.2 Image Sharing	6
1.2.3 Image Recovery	7
1.3 Dissertation Overview	8
1.3.1 Reversible JPEG Data Hiding with High Hiding Capacity	9
1.3.2 Multi-Threshold Progressive Image Sharing with Compact Shadows	9
1.3.3 Authentication and Cross Recovery of Multiple JPEG images	9
1.4 Dissertation Organization	10

Chapter 2 Reversible JPEG Data Hiding with High Hiding Capacity.....	11
--	----

2.1 Review of JPEG.....	11
-------------------------	----

2.2 Proposed Method	12
2.2.1 Mapping Quantized Coefficients to Stego Quantized Coefficients	12
2.2.2 Secret Embedding	14
2.2.2.1 Construction of Hiding Capacity Table and Modified Quantization Table	14
2.2.2.2 Hiding Secret Data in Quantized Blocks	16
2.2.3 Extraction of Hidden Data and Reconstruction of Original JPEG Code	17
2.2.4 Example of Secret Embedding and Extraction Processes.....	18
2.3 Experiments and Comparisons	19
2.3.1 Experimental Results	19
2.3.2 Comparisons	20
2.4 Discussions	22
2.4.1 Parameters Setting	22
2.4.2 Security	23
2.5 Summary	25



Chapter 3 Multi-Threshold Progressive Image Sharing With Compact Shadows.....33

3.1 Related Works	34
3.1.1 Secret Image Sharing Method of Thien and Lin.....	34
3.1.2 Galois Field.....	35
3.2 Proposed method.....	36
3.2.1 Encoding	36
3.2.2 Decoding	38
3.2.3 Example of Sharing and Inverse-Sharing Processes based on GF(256)	

.....	39
3.3 Experiments and Comparisons	39
3.3.1 Experimental Results	39
3.3.2 Comparisons	41
3.4 Security Discussion.....	43
3.5 Summary	47
Chapter 4 Authentication and Cross Recovery of Multiple JPEG	
images	58
4.1 Reed-Solomon Codes.....	58
4.2 Proposed method.....	59
4.2.1 Encoding	59
4.2.2 Decoding	62
4.3 Experiments and Comparisons	64
3.3.1 Experimental Results	64
4.3.2 Comparisons	65
4.4 Summary	66
Chapter 5 Conclusions and Future Works.....	72
5.1 Conclusions.....	72
5.2 Future Works.....	73
References.....	75

Vita

Publication List of Lee Shu-Teng Chen

List of Figures

Figure 1.1. Nine sets $\{R_i: 1 \leq i \leq 9\}$ of quantized coefficients for secret embedding of Chang et al. [18].....6

Figure 1.2. Framework of dissertation.....8

Figure 2.1. Relation between QF values and the corresponding quality of a 512×512 JPEG decompressed grayscale image Lena12

Figure 2.2. The quantizer $Q(i, j)$ is modified to a smaller quantizer $Q'(i, j)$ that satisfies $\lfloor Q(i, j)/Q'(i, j) \rfloor \geq 2$, enabling $m=F(i, j)-0.5$ and $n=F(i, j)+0.5$ to be mapped to $M(i, j)$ and $N(i, j)$, respectively13

Figure 2.3. Images decompressed from JPEG codes. (a) The 41.75-dB image Lena decompressed from the JPEG-Q85 code without any hidden secret. (b) The 39.12-dB stego-image Lena decompressed from our JPEG stego code. (c) The 41.75-dB decrypted-decompressed image Lena derived from our JPEG stego code. (d)-(f) Same as (a)-(c) except that the image Lena is replaced by Jet, and the PSNRs of (d) to (f) are 41.15dB, 38.75 dB, and 41.15dB, respectively.....26

Figure 2.4. Comparisons between Chang et al.'s method [18] and ours when the two parameters are $N_{bit}=2$ and $N_{QC}=16$. The comparisons are (a) hiding capacity, (b) hiding ratio, (c) stego-image quality, and (d) average length of the JPEG stego codes.....27

Figure 2.5. Same as Figure 2.4 except that the two parameters are $N_{bit}=1$ and $N_{QC}=10$ 28

Figure 2.6. Results of Chi-square analysis: (a) the image Lena without any hidden secret; (b) the stego-image Lena generated using the LSB substitution

method; (c) the stego-image Lena decompressed from our JPEG stego code after a secret of size 253,952 bits is hidden; (d)-(f) the results of Chi-square analysis of the pixels in (a), (b), and (c), respectively.....29

Figure 2.7. Results of StegDetect analysis: (a) the cover image Lena decompressed from the JPEG-Q85 code without any hidden secret; (b)-(d) the stego-images decompressed from the JPEG stego codes of JPHide [10], JSteg [12], OutGuess [11], respectively; (e)-(l) the stego-images decompressed from our eight JPEG stego codes, respectively; (m) the results of StegDetect analysis of the twelve JPEG codes, the decompressed images of which are in (a)-(l)30

Figure 3.1. Flowchart for encoding.....49

Figure 3.2. Original important image Lena used in the first experiment.....50

Figure 3.3. Six images Peppers, Jet, Boat, Lake, Baboon, and Zelda, which are utilized to cover the important image Lena50

Figure 3.4. Six images decompressed from our six JPEG stego codes51

Figure 3.5. Four versions of the important image Lena reconstructed from various numbers of received JPEG stego codes: (a) from any three JPEG stego codes (PSNR=27.32 dB); (b) from any four JPEG stego codes (PSNR=33.67 dB); (c) from any five JPEG stego codes (PSNR=39.35 dB); (d) from the six JPEG stego codes (and identical to the original important image Lena).....52

Figure 3.6. Same as Figure 3.4 except that the to-be-hidden shadows are generated using the proposed $[(k_1=2, k_2=3, k_3=4, k_4=5), n=6]$ threshold scheme53

Figure 3.7. Results of Chi-Square analysis: (a) for the original JPEG cover image Peppers; (b) for our JPEG stego-image Peppers in Figure 3.6(a); (c) for the original JPEG cover image Jet; (d) for our JPEG stego-image Jet in

Figure 3.6(b).....	54
Figure 3.8. Results of StegSpy analysis: (a) for the original JPEG code of Peppers; (b)-(c) for the JPEG stego codes created using the Hiderman and JPegX hiding tools, respectively; (d)-(i) for our six JPEG stego codes, the decompressed images of which are in Figure 3.6.....	55
Figure 4.1. A group of four images Baboon, Scene, Lena, and Jet.....	67
Figure 4.2. Four images decompressed from the four JPEG stego codes in the first experiment without any extraction of the hidden authentication and recovery data.....	67
Figure 4.3. Two JPEG stego codes in the first experiment are lost. (a)-(b) The recovered JPEG images 'Baboon' (PSNR=24.87 dB) and Lena' (PSNR=34.89 dB) by using two survived JPEG stego codes, the decompressed images of which are in Figures 4.2(b) and 4.2(d). (c)-(d) The recovered JPEG images 'Scene' (PSNR=30.24 dB) and Jet' (PSNR=31.82 dB) by using two survived JPEG stego codes, the decompressed images of which are in Figures 4.2(a) and 4.2(c).....	68
Figure 4.4. Another group of four images Peppers, Boat, Tiffany, and House.....	69
Figure 4.5. Four images decompressed from the four JPEG stego codes in the second experiment without any extraction of the hidden authentication and recovery data.....	69
Figure 4.6. Two JPEG stego codes in the second experiment are lost. Any two of the four JPEG stego codes, the decompressed images of which are in Figure 4.5, are lost, and the recovered JPEG images are identical to two of the four images, which are (a) Peppers' (PSNR=34.64 dB), (b) Boat' (PSNR=30.79 dB), (c) Tiffany' (PSNR=31.91 dB), and (d) House' (PSNR=32.51 dB).....	70

List of Tables

Table 2.1. Comparisons of hiding capacity, hiding ratio, and stego-image quality among the related JPEG hiding methods [12,15,17] and ours when the grayscale image Lena is compressed using JPEG with $QF=85$	31
Table 2.2. Same as Table 2.1 except that the image Lena is replaced by Jet.....	31
Table 2.3. Our hiding capacity, hiding ratio, and stego-image quality when the grayscale image Lena is compressed using JPEG with $QF=75$	32
Table 3.1. Bit rates (bpp) of the JPEG codes of cover images.....	56
Table 3.2. Comparisons among image sharing methods [21,25-30] and ours.....	56
Table 3.3. Comparison of shadow sizes in non-expanded methods [21,26,28-30] and ours. The (largest) threshold is set to six for all works except that Hung et al.'s method [30] uses five as the largest threshold value.....	57
Table 3.4. Same as Table 3.3 except that (largest) threshold value is set to five.....	57
Table 3.5. Same as Table 3.1 except that the JPEG codes are created using $QF=65$..	57
Table 4.1. Comparisons among image recovery schemes [35-39] and ours.....	71
Table 4.2. Comparison of shadow sizes between Chang et al.'s method [39] and ours when the group of four grayscale images are Baboon, Scene, Lena, and Jet	71
Table 4.3. Same as Table 4.2 except that the four grayscale images are replaced by Peppers, Boat, Tiffany, and House	71

Chapter 1

Introduction

In this chapter, motivation for the dissertation is introduced in Section 1.1. A brief review of the related studies is presented in Section 1.2. An overview of the proposed methods is reported in Section 1.3. Last, the organization of the dissertation is described in Section 1.4.

1.1 Motivation

The transmission of digital media including text, image, audio, and video, via computer networks is becoming increasingly popular. However, because networks are open environments, transmitted digital signals may be intercepted or distributed by illegal users. To solve this problem, data hiding is often applied. The concept of data hiding is to embed secret data in an ordinary-look cover image such that the generated stego-image is a diversion to the attackers when the stego-image is transmitted to the networks along with other images. Data hiding schemes can be further classified into two categories: spatial domain and frequency domain. Spatial domain hiding schemes [1-9] generated a stego-image by embedding secret data in the pixel values of a cover image. Because the stego-images generated by the spatial domain hiding schemes are usually very large, some well-known image compression techniques such as Joint Photographic Experts Group (JPEG) have been developed to reduce storage space.

However, using higher compression rate usually destroys or damages the hidden content in the stego-images. Therefore, various frequency domain hiding methods have been investigated to hide secret data in a JPEG code [10-20]. Among these

frequency domain hiding methods, some are reversible [14,18]. A reversible JPEG hiding technique can loss-freely reconstruct not only the hidden secret data but also the JPEG code. Therefore, we propose a reversible JPEG hiding method with high hiding capacity in Chapter 2. The method intends to embed a large-size secret in a JPEG code. After lossless extraction of the secret data, the JPEG code used to embed the secret can be reconstructed.

Secret image sharing is an approach for protecting secret images [21-26]. Polynomials are used to share a secret image and generate n shadows. Any k of the n shadows ($2 \leq k \leq n$) can reconstruct the secret image, but fewer than k shadows cannot. The missing-allowable feature makes secret image sharing methods useful to the distributed storage of a secret image. Specifically, the n shadows of an image can be stored in n places. Later, to reconstruct the image, the n shadows are grabbed over n distinct channels. Some of the n communication channels or the n storage disks may be out of service temporarily or deliberately if the owner of a shadow refuses to cooperate, but neither case will affect the reconstruction, as long as the number of missing shadows is not more than $n-k$. The potential problem of losing an image forever is therefore erased using image sharing. Additionally, collecting fewer than k shadows yields nothing but noise, and this feature increases security.

Conventional secret image sharing methods reveal either the entire secret image (when any k of the n shadows are collected) or nothing (when fewer than k shadows are collected). This all-or-nothing property is useful when the image being shared is top secret. However, not all images in daily life are top secret. In many circumstances, the shared image might be sensitive in some way and yet not top secret. Restated, although an image must not be viewed by only a minority of the participants, the reconstruction of the image can still involve certain quality levels, such as low quality, middle quality, and high quality, based on whether the number of collected shadows

reaches the corresponding thresholds. Such sharing is called progressive sharing: the sensitive image is reconstructed with improving quality, as determined by the number of the collected shadows in the decoding meeting [27-30].

Progressive sharing has a range of applications. Consider, for example, image searching in an antiterrorism intelligence office or a witness-protection program, when an authorized officer searches for a sensitive image from a missing-allowable database system with n distributed storage places. If the shadows of each image have been formed earlier by a traditional all-or-nothing sharing scheme, a user must wait for the entire image to be downloaded (by collecting k out of the n shadows) and then check whether the reconstructed image is useful. In contrast, using shadows in a progressive manner can reduce searching time: in the earlier stage of the reconstruction, people can obtain a rough version of the image by collecting a smaller number (k_1) of shadows ($2 \leq k_1 < k \leq n$); they can then abort further transmission as soon as the rough version indicates that the candidate image is absolutely not the sought image. Meanwhile, fewer than k_1 shadows reveal nothing but noise, providing a certain degree of protection of a sensitive image.

Another example involves the increasing use of mobile devices or computers to browse the web. Providing progressive versions of an image allow each authorized customer or team member to have more choices. Moreover, the number of downloadable shadows, which control the quality of the reconstructed versions, can be determined by the level of the paid membership (or authorized rank) of the downloader. In particular, if the image is too offensive or violent or only allowed to be inspected by a particular police team or intelligence squad, then controlling the number of shadows in a progressive manner can yield flexible design benefits or facilitate management of the system (as members of a single team but with different ranks are authorized to download different numbers of the n created shadows).

Therefore, we propose a multi-threshold progressive reconstruction method with compact shadows in Chapter 3. The method intends to reconstruct the sensitive image with progressively improved quality, according to the number of collected shadows. Also, each shadow is very compact and so can reduce storage space and transmission time, and facilitate the hiding of shadows.

Protecting the integrity of digital images has recently become an important research topic because digital images are easily copied and illegally distributed through networks. Image authentication is used to verify the integrity of an image. Some image authentication schemes [31-34] embedded authentication data in an image, and the embedded authentication data could then be extracted to verify the validity of the image. Several image authentication methods with extra tamper recovery ability have also been investigated [35-38]. These methods could identify unauthorized manipulations of an image and recovered tampered areas with approximation of the original ones.

However, these self-recovery methods [35-38] focused on a single image. Once the protected image is seriously tampered with, the recovery data left are insufficient to recover the seriously tampered image. Accordingly, Chang et al. [39] proposed a (k, n) threshold scheme with authentication and cross-recovery ability to protect a group of n images. When some of the n images are tampered with or even lost, the tampered or lost ones can be approximately recovered by the support of k survived members. However, if the n protected images in Chang et al.'s work [39] are lossy compressed to reduce storage space, then their cross-recovery ability is gone.

To make the verification and recovery mechanisms feasible for the images stored in JPEG format, we propose an image authentication and cross-recovery method for protecting a group of n JPEG images in Chapter 4. The proposed method generates n JPEG stego codes by embedding authentication and recovery data in the n JPEG

images. When some (up to $n-k$) of the n JPEG stego codes are destroyed, the destroyed JPEG images can be approximately recovered using any k survived JPEG stego codes.

1.2 Related Studies

1.2.1 JPEG Data Hiding

JSteg [12] was a simple hiding tool that embedded secret data in a JPEG code. First, a cover image was divided into non-overlapping blocks of 8×8 pixels each. The 8×8 pixels of each block were transformed using discrete cosine transform (DCT) and then quantized according to the default JPEG quantization table. The secret data were embedded in the least significant bit (LSB) of the quantized coefficients, the values of which were not 1, 0, or -1. Last, each embedded quantized block was compressed further using the JPEG entropy coding. After all blocks have been processed, the JPEG stego code is generated.

Because the quantized coefficients are almost all zero after the DCT transformation and quantization steps of JPEG, the hiding capacity in JSteg is quite limited. Therefore, Chang et al. [15] modified the default JPEG quantization table and embedded secret data in the 26 quantized coefficients located in the middle-frequency part. Because the quantizers in the middle-frequency part were modified to one, the quantized coefficients in the middle-frequency part were usually non-zero. To find a balance between the hiding capacity and stego-image quality, Tseng and Chang [17] proposed further the adaptive JPEG data hiding method, which used the human visual system and their capacity table to determine the embedded length of secret data. Iwata et al. [16] also modified the boundaries between zero and non-zero quantized coefficients to hide a secret.

Some JPEG hiding techniques with additional reversibility have also been investigated [14,18]. Fridrich et al. [14] proposed a reversible watermarking scheme for authenticating JPEG codes. A secret watermark was generated by hashing quantized coefficients. The generated watermark and further compressed result of the quantized coefficients were then embedded in the LSB of the quantized coefficients. Chang et al. [18] utilized the successive zero sequence of each set R_i ($1 \leq i \leq 9$) in Figure 1.1 to embed secret data because many quantized coefficients in the lower right portion of Figure 1.1 were zeros. They also modified the JPEG quantization table to improve the quality of their stego-images.

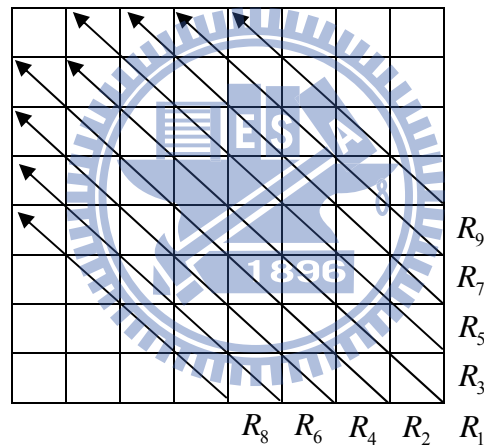


Figure 1.1. Nine sets $\{R_i: 1 \leq i \leq 9\}$ of quantized coefficients for secret embedding of Chang et al. [18].

1.2.2 Image Sharing

Blakley [40] and Shamir [41] were the first to propose the idea of a (k, n) threshold sharing scheme. Thien and Lin [21] extended the work of Shamir [41] by sharing a secret image and generated n shadows. The size of each shadow in their work is only $1/k$ of that of the original secret image, and so the storage space and transmission time are kept low. To reduce the size of each shadow further, Tso [26] quantized the secret image and then shared it. Lin and Tsai [24] also transformed the

secret image into the frequency domain and then shared the first DCT coefficient, which was used as a seed in a random number generator to yield a sequence of numbers that were then used to rearrange the values of the second to tenth DCT coefficients in each transformed block. In almost all sharing methods, since each generated shadow looks like noise, data hiding may be employed to hide each noise-like shadow in a cover image.

Various progressive image sharing schemes have recently been investigated [27-30]. However, in Fang's scheme [27], the size of each shadow was expanded to four times larger than the input image. To avoid expansion, people may use approaches [28-30] that were based on the sharing scheme of Thien and Lin [21]. Chen and Lin [28] adopted a bit-plane scanning procedure to rearrange the pixels of the shared image, and the rearranged data were then shared. Wang and Shyu [29] decomposed the shared image using spatial and depth information simultaneously and then shared the decomposed image. Hung et al. [30] shared the quantized DCT coefficients of the input image. Although the shadows in their work were much smaller than those in the preceding three works [27-29], the reconstruction of their image was not lossless when all n shadows were collected.

1.2.3 Image Recovery

Chang et al. [39] proposed an image authentication and cross-recovery method to protect a group of n images. First, the v LSBs ($1 \leq v \leq 4$) of each of the n images were set to zero to create n rough images using a modulus-based zeroing approach that was based on Thien and Lin's hiding scheme [3]. Next, the n rough images were compressed using JPEG2000, and each of the n generated JPEG2000 codes was shared using a two-layer sharing that is based on Thien and Lin's sharing scheme [21]. Then, the n generated shadows were embedded in the n rough images, respectively, to

generate n stego-images. For each of the n stego-images, the authentication data were computed using the MD5 hash function [42]. Last, the n authentication data were embedded in the n stego-images, respectively, using the v -LSB substitution method. Their n watermarked images were therefore generated. When some of the n watermarked images were tampered with, the tampered images could be approximately recovered using the survived ones.

1.3 Dissertation Overview

The dissertation proposes three techniques for data hiding, image sharing, and image authentication with cross recovery based on JPEG. The proposed methods contain a reversible JPEG data hiding method with high hiding capacity, a progressive image sharing method with compact shadows, and an image authentication method with cross-recovery ability. The framework of the dissertation is shown in Figure 1.2, and a brief overview of each proposed method is described in the following subsections.

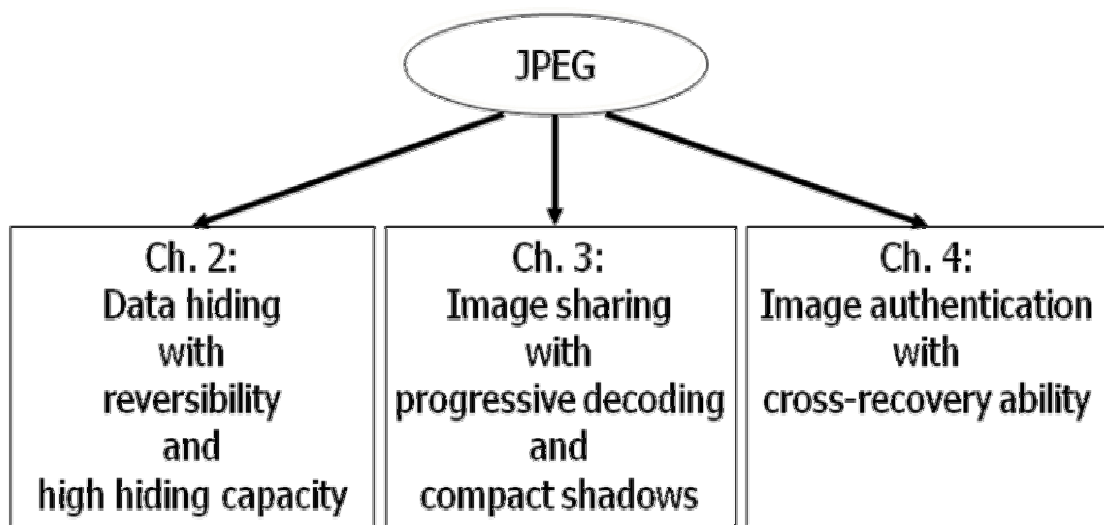


Figure 1.2. Framework of dissertation.

1.3.1 Reversible JPEG Data Hiding with High Hiding Capacity

Chapter 2 proposes a reversible JPEG data hiding method for hiding a large-size secret in a JPEG code. The JPEG quantization table attached to the JPEG code is modified, and the two JPEG quantization tables (original and modified) together can map the quantized DCT coefficients to the larger quantized DCT coefficients with hidden secret data. In the decoding process, the secret data and the original JPEG code can be reconstructed without any error.

1.3.2 Multi-Threshold Progressive Image Sharing with Compact Shadows

Chapter 3 proposes a multi-threshold progressive image sharing method for reconstructing an image with various quality levels, based on whether the number of collected shadows reaches the corresponding thresholds. All n products (or n shadows) of the image are compact and so can be hidden in stego media easily without excessively affecting the image quality of the cover media. Any of the shadows could be missing, so the sender or the receiver need not worry about which shadows are sent or collected first. The probability of success of the decoding meeting is therefore increased.

1.3.3 Authentication and Cross Recovery of Multiple JPEG Images

Chapter 4 proposes an image authentication method with cross-recovery ability for protecting a group of n JPEG images. The proposed method generates n JPEG stego codes with hidden authentication and recovery data. The hidden authentication data can be used to verify which JPEG stego codes are attacked, and when some of the n JPEG stego codes are corrupted, the corrupted JPEG images can be approximately recovered by the cooperation of the survived JPEG stego codes.

1.4 Dissertation Organization

In the rest of this dissertation, the proposed reversible JPEG data hiding method with high hiding capacity is presented in Chapter 2. The proposed multi-threshold progressive image sharing method with compact shadows is reported in Chapter 3. The proposed image authentication method with cross-recovery ability is described in Chapter 4. Last, conclusions and future work are drawn in Chapter 5, along with recommendation for future research.



Chapter 2

Reversible JPEG Data Hiding with High Hiding Capacity

This chapter proposes a reversible JPEG data hiding method for hiding a large-size secret in a JPEG code. The quantization table of the JPEG code is modified, and the proposed mapping function is utilized to map quantized coefficients to larger quantized coefficients with hidden secret data. In the decoding process, both the secret data and the original JPEG code can be reconstructed without any error. Experimental results demonstrate that the proposed method provides a high hiding capacity and acceptable stego-image quality, to an extent comparable with other JPEG data hiding methods. The rest of this chapter is organized as follows. Section 2.1 reviews the JPEG technique. Section 2.2 presents the proposed method. Section 2.3 reports the experimental results and makes comparisons with other methods. Section 2.4 discusses parameters setting and security. Last, Section 2.5 summarizes this chapter.

2.1 Review of JPEG

JPEG is an international image compression standard that is used commonly on the Internet. A given image is divided into several blocks of 8×8 pixels each. The 8×8 pixels of each block are transformed using DCT, and the 8×8 transformed coefficients are quantized using a quantization table. The 8×8 quantized coefficients are then scanned in zigzag order for entropy coding. After all of the blocks have been sequentially processed, the JPEG code is generated. The quality factor QF between 0 and 100 controls the quality of the JPEG decompressed image. A higher QF

corresponds to higher quality of the JPEG decompressed image. This phenomenon is shown in Figure 2.1, and the peak signal-to-noise ratio (PSNR), defined by

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE}, \quad (2.1)$$

is used to measure the similarity between the original image and its JPEG decompressed image where MSE represents the mean square error between the pixel values of the original image and those of the JPEG decompressed image.

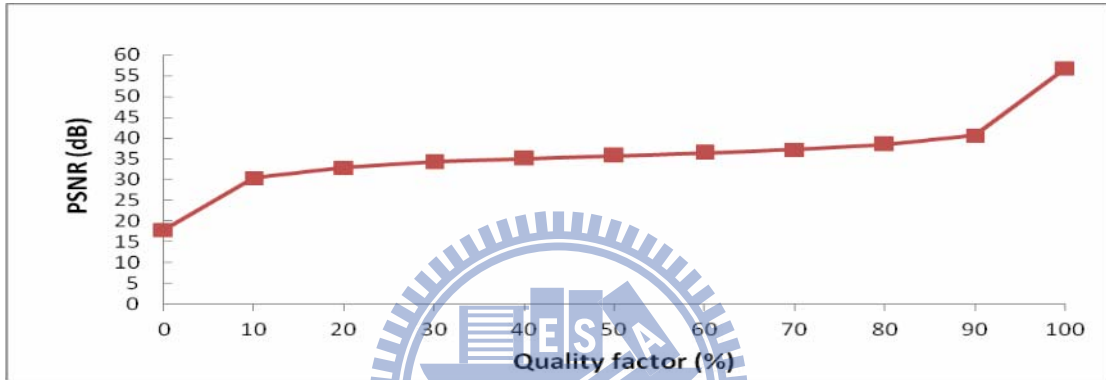


Figure 2.1. Relation between QF values and the corresponding quality of a 512×512 JPEG decompressed grayscale image Lena.

2.2 Proposed Method

2.2.1 Mapping Quantized Coefficients to Stego Quantized Coefficients

In the quantization process of JPEG, each DCT coefficient $D(i, j)$ is divided by its quantizer $Q(i, j)$ grabbed from a quantization table, and then rounded to the nearest integer $F(i, j)$, as

$$F(i, j) = \text{Round}[D(i, j) / Q(i, j)], \quad (2.2)$$

where $0 \leq i, j < 8$. To generate the stego quantized coefficient $F'(i, j)$ by embedding secret data in the quantized coefficient $F(i, j)$, a mapping that maps $F(i, j)$ to $F'(i, j)$ is introduced in the following. As shown in Figure 2.2, consider a line segment \overline{mn} in the real axis containing the integer point $F(i, j)$ and its two half-unit-away non-integer

points $m=F(i, j)-0.5$ and $n=F(i, j)+0.5$. The quantizer $Q(i, j)$ is modified to a smaller quantizer $Q'(i, j)$ that satisfies $\lfloor Q(i, j)/Q'(i, j) \rfloor \geq 2$, enabling each real number z to be mapped to an integer, as

$$h(z) = \lceil z \times Q(i, j) / Q'(i, j) \rceil. \quad (2.3)$$

According to Eq. (2.3), the non-integer points $m=F(i, j)-0.5$ and $n=F(i, j)+0.5$ are mapped to the integer points $M(i, j)$ and $N(i, j)$, respectively, using

$$M(i, j) = \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil, \quad (2.4)$$

$$N(i, j) = \lceil [F(i, j) + 0.5] \times Q(i, j) / Q'(i, j) \rceil. \quad (2.5)$$

After the values of $M(i, j)$ and $N(i, j)$ have been determined, based on the to-be-hidden secret digit, an integer point in the half-open interval $[M(i, j), N(i, j))$ is indentified as the stego quantized coefficient $F'(i, j)$. For example, let $F'(i, j)$ be $M(i, j)$ if the to-be-hidden secret is 0; let $F'(i, j)$ be $M(i, j)+1$ if the to-be-hidden secret is 1; let $F'(i, j)$ be $M(i, j)+2$ if the to-be-hidden secret is 2; and so on. Notably, if the final value of the stego quantization coefficient $F'(i, j)$ is used to indicate one of the 2^n possible readings of an n -bit secret, the half-open interval $[M(i, j), N(i, j))$ in Figure 2.2 must contain at least 2^n integer points.

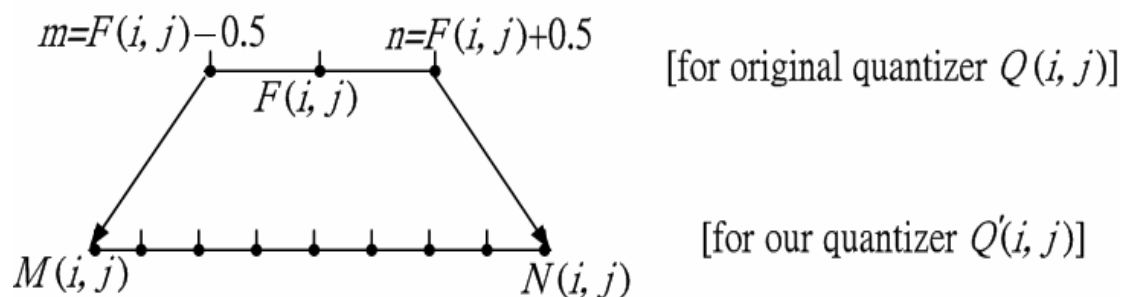


Figure 2.2. The quantizer $Q(i, j)$ is modified to a smaller quantizer $Q'(i, j)$ that satisfies $\lfloor Q(i, j)/Q'(i, j) \rfloor \geq 2$, enabling $m=F(i, j)-0.5$ and $n=F(i, j)+0.5$ to be mapped to $M(i, j)$ and $N(i, j)$, respectively.

To reconstruct the original quantized coefficient $F(i, j)$ from the stego quantized coefficient $F'(i, j)$, all integers in the range $[M(i, j), N(i, j))$ must be traced back to the unique integer $F(i, j)$. This can be done by applying the $Q'(i, j)$ -to- $Q(i, j)$ inverse mapping to reverse $F'(i, j)$ back to a real number $F'(i, j) \times Q'(i, j) / Q(i, j)$, which is near $F(i, j)$. In fact, according to the design shown in Figure 2.2, this real number will stay in the interval $[F(i, j) - 0.5, F(i, j) + 0.5)$, and it can be rounded to the nearest integer as the original $F(i, j)$. In summary, $F(i, j)$ is reconstructed using

$$F(i, j) = \text{Round}[F'(i, j) \times Q'(i, j) / Q(i, j)]. \quad (2.6)$$

2.2.2 Secret Embedding

2.2.2.1 Construction of Hiding Capacity Table and Modified Quantization Table

For $0 \leq i, j < 8$, let $HC(i, j)$, $Q(i, j)$, and $Q'(i, j)$ denote the elements in the i 'th row and j 'th column of the 8×8 hiding capacity table HC , the original JPEG quantization table Q , and the modified quantization table Q' , respectively. The three tables are used for all blocks of the cover image. The integer $HC(i, j)$, which is the number of secret bits to be hidden in a quantized coefficient $F(i, j)$, is randomly chosen from the range

$$\min\{N_{bit}, \lfloor \log_2 Q(i, j) \rfloor\} \leq HC(i, j) \leq \lfloor \log_2 Q(i, j) \rfloor. \quad (2.7)$$

The reason why $HC(i, j)$ depends on the original quantization table Q and integer parameter N_{bit} is explained in the following. Based on Range Property below, $\lfloor Q(i, j) / Q'(i, j) \rfloor \leq N(i, j) - M(i, j) \leq \lceil Q(i, j) / Q'(i, j) \rceil$. Therefore, in the half-open interval $[M(i, j), N(i, j))$, there are at most $\lfloor Q(i, j) / Q'(i, j) \rfloor$ integer points. To embed secret data of $HC(i, j)$ bits in the quantized coefficient $F(i, j)$, there are $2^{HC(i, j)}$ possible combinations of the secret bits. The embedding of each combination corresponds to one of the integer points located in the half-open interval $[M(i, j), N(i, j))$. Therefore, the inequality

$$2^{HC(i,j)} \leq \lfloor Q(i,j)/Q'(i,j) \rfloor \quad (2.8)$$

must be satisfied for each $0 \leq i, j < 8$. Because the values of $Q'(i,j)$ and $Q(i,j)$ are integers between 1 and 255, $\lfloor Q(i,j)/Q'(i,j) \rfloor \leq \lfloor Q(i,j) \rfloor = Q(i,j)$. Therefore,

$$2^{HC(i,j)} \leq Q(i,j). \quad (2.9)$$

The integer parameter N_{bit} also appears in Eq. (2.7) because if directly let $HC(i,j)$ be $\lfloor \log_2 Q(i,j) \rfloor$, according to Eq. (2.10), the corresponding $Q'(i,j)$ in the modified quantization table Q' will be one ($\lfloor Q(i,j)/2^{HC(i,j)} \rfloor = \lfloor Q(i,j)/2^{\lfloor \log_2 Q(i,j) \rfloor} \rfloor = 1$), and therefore the first N_{QC} elements in the modified quantization table Q' are all one, making attackers feel suspicious. Last, to determine the 8×8 modified quantization table Q' , Eq. (2.8) implies that $2^{HC(i,j)} \leq \lfloor Q(i,j)/Q'(i,j) \rfloor \leq Q(i,j)/Q'(i,j)$. Because both $Q(i,j)$ and $Q'(i,j)$ are positive integers, $Q'(i,j) \leq Q(i,j)/2^{HC(i,j)}$. Therefore, the element $Q'(i,j)$ of the modified quantization table Q' is simply defined as

$$Q'(i,j) = \lfloor Q(i,j)/2^{HC(i,j)} \rfloor, \quad (2.10)$$

where $0 \leq i, j < 8$. Eq. (2.10) also implies that

$$HC(i,j) = \lfloor \log_2 [Q(i,j)/Q'(i,j)] \rfloor. \quad (2.11)$$

Range Property: $\lfloor Q(i,j)/Q'(i,j) \rfloor \leq N(i,j) - M(i,j) \leq \lceil Q(i,j)/Q'(i,j) \rceil$.

Proof:

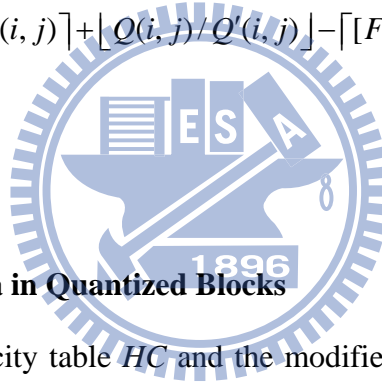
(i) Proof of the upper bound:

$$\begin{aligned} & N(i,j) - M(i,j) \\ &= \lceil [F(i,j) + 0.5] \times Q(i,j)/Q'(i,j) \rceil - \lceil [F(i,j) - 0.5] \times Q(i,j)/Q'(i,j) \rceil \\ &= \lceil [F(i,j) + 0.5] \times Q(i,j)/Q'(i,j) \rceil - \lceil [F(i,j) + 0.5 - 1] \times Q(i,j)/Q'(i,j) \rceil \end{aligned}$$

$$\begin{aligned}
&= \lceil [F(i, j) + 0.5] \times Q(i, j) / Q'(i, j) \rceil - \lceil [F(i, j) + 0.5] \times Q(i, j) / Q'(i, j) - Q(i, j) / Q'(i, j) \rceil \\
&\leq \lceil [F(i, j) + 0.5] \times Q(i, j) / Q'(i, j) \rceil - \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil + \lceil Q(i, j) / Q'(i, j) \rceil \\
&= \lceil Q(i, j) / Q'(i, j) \rceil.
\end{aligned}$$

(ii) Proof of the lower bound:

$$\begin{aligned}
&N(i, j) - M(i, j) \\
&= \lceil [F(i, j) + 0.5] \times Q(i, j) / Q'(i, j) \rceil - \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil \\
&= \lceil [F(i, j) - 0.5 + 1] \times Q(i, j) / Q'(i, j) \rceil - \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil \\
&= \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) + Q(i, j) / Q'(i, j) \rceil - \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil \\
&\geq \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil + \lceil Q(i, j) / Q'(i, j) \rceil - \lceil [F(i, j) - 0.5] \times Q(i, j) / Q'(i, j) \rceil \\
&= \lceil Q(i, j) / Q'(i, j) \rceil.
\end{aligned}$$



2.2.2.2 Hiding Secret Data in Quantized Blocks

After the hiding capacity table HC and the modified quantization table Q' have been generated, secret data are embedded in quantized blocks. The secret embedding algorithm is summarized here:

The secret embedding algorithm

Input: A JPEG code; secret data; two integer parameters $N_{bit} \in \{1, 2, 3\}$ and $N_{QC} \in \{1, 2, \dots, 32\}$.

Output: The JPEG stego code.

Step 1: Decode the JPEG code using the JPEG entropy decoding to obtain an 8×8 quantization table Q and all quantized blocks $\{F\}$ formed of 64 quantized coefficients each.

Step 2: Use Eqs. (2.7) and (2.10) to compute the first N_{QC} elements of the 8×8 hiding capacity table HC and the 8×8 modified quantization table Q' , respectively. The other $(64-N_{QC})$ elements of the table HC are set to zeros. The other $(64-N_{QC})$ elements of the table Q' are copied from the corresponding $(64-N_{QC})$ elements of the quantization table Q .

Step 3: Repeat Steps 3a-3b in a block-by-block manner until all quantized blocks $\{F\}$ have been processed.

Step 3a: For the first N_{QC} quantized coefficients $\{F(i, j)\}$ of the 8×8 quantized block F currently being processed, use Eq. (2.4) to determine $M(i, j)$ from $F(i, j)$. Then, sequentially grab the next $HC(i, j)$ not-yet-embedded bits from the secret data, and let the single decimal value $Z(i, j)$ be the decimal equivalent of the just-grabbed binary number with $HC(i, j)$ bits. Compute the stego quantized coefficient $F'(i, j)$ using

$$F'(i, j) = M(i, j) + Z(i, j). \quad (2.12)$$

Replace $F(i, j)$ by $F'(i, j)$ because $F(i, j)$ is no longer needed.

Step 3b: Apply the JPEG entropy coding to the embedded quantized block F' .

Step 4: Put the modified quantization table Q' in the JPEG file header. The JPEG stego code is generated. (The JPEG stego code, the original quantization table Q , and the integer parameter N_{QC} are transmitted to the receiver.)

2.2.3 Extraction of Hidden Data and Reconstruction of Original JPEG Code

When people receive the JPEG stego code, if they also know the original JPEG quantization table Q and the integer parameter N_{QC} , they can extract the secret data, followed by reconstructing the original JPEG code. The secret data and JPEG code reconstruction algorithm is summarized here:

The secret data and JPEG code reconstruction algorithm

Input: The JPEG stego code; the original 8×8 JPEG quantization table Q ; the integer parameter $N_{QC} \in \{1, 2, \dots, 32\}$.

Output: The secret data and the original JPEG code.

Step 1: Apply the JPEG entropy decoding to the JPEG stego code and obtain the 8×8 modified quantization table Q' and all stego quantized blocks $\{F'\}$.

Step 2: Reconstruct the first N_{QC} elements of the 8×8 hiding capacity table HC using Eq. (2.11).

Step 3: Repeat Steps 3a-3b in a block-by-block manner until all stego quantized blocks $\{F'\}$ have been processed.

Step 3a: For the first N_{QC} stego quantized coefficients $\{F'(i, j)\}$ of the 8×8 stego quantized blocks F' currently being processed, reverse each $F'(i, j)$ back to $F(i, j)$ using Eq. (2.6). Map $F(i, j) - 0.5$ to $M(i, j)$ using Eq. (2.4). Extract the decimal equivalent $Z(i, j)$ of the secret bits using

$$Z(i, j) = F'(i, j) - M(i, j). \quad (2.13)$$

Overwrite $F'(i, j)$ by $F(i, j)$. Therefore, not only the secret data $Z(i, j)$ hidden in the stego quantized coefficient $F'(i, j)$ but also the original JPEG quantized coefficient $F(i, j)$ at the current block position are reconstructed.

Step 3b: Encode the current (secret-already-extracted) quantized block F using the JPEG entropy coding for recycling use.

Step 4: Put the original quantization table Q in the JPEG file header. The original JPEG code is reconstructed without any error.

2.2.4 Example of Secret Embedding and Extraction Processes

Assume the quantized coefficient $F(0, 0)$ is 12 and the original quantizer $Q(0, 0)$ used by JPEG is 17. Let the modified quantizer $Q'(0, 0)$ be four. According to Eqs.

(2.4) and (2.5), the values $M(0, 0)$ and $N(0, 0)$ are computed using

$$M(0, 0) = \lceil [F(0, 0) - 0.5] \times Q(0, 0) / Q'(0, 0) \rceil = \lceil (12 - 0.5) \times 17 / 4 \rceil = 49,$$

$$N(0, 0) = \lceil [F(0, 0) + 0.5] \times Q(0, 0) / Q'(0, 0) \rceil = \lceil (12 + 0.5) \times 17 / 4 \rceil = 54.$$

Therefore, in the half-open interval $[M(0, 0)=49, N(0, 0)=54)$, the value $F'(0, 0)=49$ is the stego quantized coefficient after 0 is embedded. [$F'(0, 0)=50$ after 1 is embedded; $F'(0, 0)=51$ after 2 is embedded; $F'(0, 0)=52$ after 3 is embedded.] Later, assume that the stego quantized coefficient $F'(0, 0)=50$ is obtained. To reconstruct the quantized coefficient $F(0, 0)$ from $F'(0, 0)=50$, use Eq. (2.6) to evaluate

$$F(0, 0) = \text{Round}[F'(0, 0) \times Q'(0, 0) / Q(0, 0)] = \text{Round}(50 \times 4 / 17) = 12.$$

Then, use Eq. (2.4) to compute

$$M(0, 0) = \lceil [F(0, 0) - 0.5] \times Q(0, 0) / Q'(0, 0) \rceil = \lceil (12 - 0.5) \times 17 / 4 \rceil = 49.$$

Last, from Eq. (2.13), the decimal equivalent of the hidden data is extracted as

$$Z(0, 0) = F'(0, 0) - M(0, 0) = 50 - 49 = 1.$$

2.3 Experiments and Comparisons

2.3.1 Experimental Results

Six 512×512 grayscale images, Lena, Jet, Boat, Peppers, Baboon, and Zelda, are tested in the experiments. They are all compressed using the JPEG source code taken from the fourth public release of the Independent JPEG Group's free JPEG software [43]. The quality of an image is measured by the PSNR. Figures 2.3(a) and 2.3(d) show the images Lena and Jet decompressed from the JPEG-Q85 codes using JPEG with $QF=85$ without any hidden secret. Figures 2.3(b) and 2.3(e) display the stego-images decompressed from our JPEG stego codes after two secret data of size 253,952 bits each are embedded in the JPEG-Q85 codes of Lena and Jet, respectively.

Figures 2.3(c) and 2.3(f) depict the decrypted-decompressed images derived from our JPEG stego codes, the decompressed images of which are in Figures 2.3(b) and 2.3(e).

2.3.2 Comparisons

Table 2.1 (for image Lena) and Table 2.2 (for image Jet) compare some non-reversible JPEG hiding methods [12,15,17] with the proposed reversible JPEG hiding method in terms of hiding capacity (size of the hidden secret), hiding ratio (size of the hidden secret over size of the JPEG stego code), and the quality of the stego-image when the cover image is compressed using JPEG with $QF=85$. For simplicity, the value of each element $HC(i, j)$ in the 8×8 hiding capacity table HC is set to the left-hand-side value of Eq. (2.7).

As presented in Table 2.1, the proposed method provides a higher hiding capacity, a higher hiding ratio, and better stego-image quality than those in the related works [12,15,17]. For example, the proposed method has the PSNR value 39.12 dB when the hiding capacity is 253,952 bits with the hiding ratio being 35.23% while the PSNR value in Chang et al.'s method [15] is 29.64 dB (< 39.12 dB) when the hiding capacity is 212,992 bits ($< 253,952$ bits) with the hiding ratio being 30.0% ($< 35.23\%$).

Similarly, the proposed method has the PSNR value 41.11 dB when the hiding capacity is 122,880 bits with the hiding ratio being 23.16% while the PSNR value in JSteg [12] is 40.70 dB (< 41.11 dB) when the hiding capacity is 31,933 bits ($< 122,880$ bits) with the hiding ratio being 8.85% ($< 23.16\%$), and the PSNR value in Tseng and Chang's method [17] is 38.10 dB (< 41.11 dB) when the hiding capacity is 54,652 bits ($< 122,880$ bits) with the hiding ratio being 14.0% ($< 23.16\%$). Most of all, after the secret data have been extracted, the proposed method can reconstruct the original JPEG code, but the aforementioned methods [12,15,17] cannot.

The comparisons between Chang et al.'s reversible method [18] and ours are given in the following. Notably, both Fridrich et al.'s [14] and Chang et al.'s [18] methods are reversible. However, because the original purpose in Fridrich et al.'s method [14] is to authenticate JPEG codes (the hidden secret is a much smaller authentication watermark of size about 4,000 bits rather than our larger-size secret, which can be as large as 253,952 bits), their method is not in the same application group as ours or the associated works [12,15,17,18]. Therefore, the comparisons between Fridrich et al.'s method [14] and ours are omitted.

Figures 2.4(a)-(c) compare Chang et al.'s method [18] with ours in terms of hiding capacity, hiding ratio, and stego-image quality when the image is Lena. Figure 2.4(d) compare the average length of the JPEG stego codes between Chang et al.'s method [18] and ours when the six images are Lena, Jet, Boat, Peppers, Baboon, and Zelda. The two parameters used in the proposed method are $N_{bit}=2$ and $N_{QC}=16$. As displayed in Figures 2.4(a)-(c), the hiding capacity, hiding ratio, and stego-image quality in the proposed method are all better than those in Chang et al.'s method [18].

On the other hand, as shown in Figure 2.4(d), the length of our JPEG stego code would be less attractive than that of theirs because a higher hiding capacity causes a longer output JPEG stego code. But this trade off is still worthy and more efficient because the hiding capacity is high. If the readers want to reduce the length of the JPEG stego code, they may reduce the not-small gains in hiding capacity and hiding ratio by using lower N_{bit} and N_{QC} values. One such example is shown in Figure 2.5 where the two parameters are $N_{bit}=1$ and $N_{QC}=10$. From Figure 2.5, the hiding capacity, hiding ratio, and stego-image quality in the proposed method are still a little bit more competitive than those in Chang et al.'s method [18] while our code-length curve gets a tie with theirs.

2.4 Discussions

2.4.1 Parameters Setting

The rules in the following can be used to determine the two parameters N_{bit} and N_{QC} . As a key rule, using $N_{bit}=1$ is for the secret of moderate size, and using $N_{bit}=2$ or 3 is for the secret of large size. After the N_{bit} value has been determined, adjust the N_{QC} value. When the N_{QC} value is adjusted, start from a moderate value such as $N_{QC}=16$. If the secret is too large to be hidden, increase the N_{QC} value; otherwise, decrease the N_{QC} value slightly to reduce the length of the JPEG stego code, as long as the secret can still be hidden after the cut of the N_{QC} value.

Additionally, under the same N_{QC} value, using a higher N_{bit} , which forces the use of a higher $HC(i, j)$, causes a higher hiding capacity than using a lower N_{bit} because $HC(i, j)$ is the number of secret bits to be embedded in a quantized coefficient. Also, a little improvement of the stego-image quality might be obtained by using a higher N_{bit} . Another experiment is conducted to observe this phenomenon. Let the image Lena be compressed using JPEG with $QF=75$. The generated JPEG code is utilized to embed the secret data using various setting of N_{bit} and N_{QC} values. For simplicity, the value of each element $HC(i, j)$ in the 8×8 hiding capacity table HC is set to be the left-hand-side value of Eq. (2.7). According to this assignment rule, using a higher N_{bit} really causes a higher hiding capacity than using a lower N_{bit} , as shown in Table 2.3.

However, the improvement of the hiding capacity and the quality of the stego-images are less obvious when the N_{bit} value jumps from three to four. Also, using a higher N_{bit} causes a more increasing of the length of the JPEG stego code than using a lower N_{bit} . Therefore, using $N_{bit}=4$ is not suggested for security concern. In summary, it is recommended to use $N_{bit}=2$ and $N_{bit}=3$. Skip the use of $N_{bit}=1$ if very large hiding capacity is required.

2.4.2 Security

Both the Chi-square [44] and StegDetect [11] analysis tools are used by attackers to detect whether a stego-image or a JPEG code contains secret data or not. The proposed method can resist these attacks.

For the Chi-square analysis, Figure 2.6(a) shows the 512×512 grayscale image, Lena, without any hidden secret. Figure 2.6(b) displays the stego-image Lena generated by embedding secret data in Figure 2.6(a) using the LSB substitution scheme. Figure 2.6(c) depicts the stego-image Lena decompressed from our JPEG stego codes after a secret of size 253,952 bits is hidden. Figures 2.6(d)-(f) plot the results obtained when Guillermi's Chi-square analysis tool is employed to examine the pixels in Figures 2.6(a)-(c), respectively. The cross curve represents the probability that the pairs of values are randomly distributed, and the circular curve represents the average value of all LSBs in one block of pixels. The circular curves in Figures 2.6(d)-(f) suggest to the attackers nothing because these circular curves are all around $(0+1)/2=0.5$. However, the cross curve in Figure 2.6(e) is close to one, indicating that some secret data are very probably embedded in Figure 2.6(b). In contrast, the cross curves in Figures 2.6(d) and 2.6(f) are both close to zero. Therefore, the attackers may ignore the hidden data in Figure 2.6 (c).

Provos's StegDetect analysis tool [11], introduced in Provos and Honeyman's method [45] and used in some methods [46-48], is adopted for the StegDetect analysis. Let OriginalLena.jpg be the JPEG-Q85 code generated by compressing a 24-bit color image Lena of size 512×512 using JPEG with $QF=85$. Figure 2.7(a) shows the image Lena decompressed from OriginalLena.jpg. When people use the JPEG hiding methods JPHide [10], JSteg [12], and OutGuess [11] to hide secret data in OriginalLena.jpg, let the generated JPEG stego codes be Stego-of-JPHide.jpg, Stego-of-JSteg.jpg, and Stego-of-OutGuessOld.jpg, respectively. Figures 2.7(b)-(d)

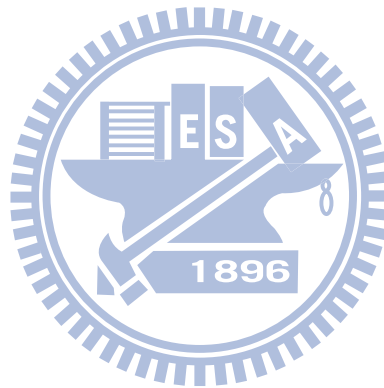
shows the stego-images decompressed from Stego-of-JPHide.jpg, Stego-of-JSteg.jpg, and Stego-of-OutGuessOld.jpg, without any extraction of the hidden secret data. Figures 2.7(e)-(l) plot the eight stego-images decompressed from our eight JPEG stego codes when the two parameters (N_{bit}, N_{QC}) are (1, 16), (1, 24), (1, 32), (2, 16), (2, 24), (2, 32), (3, 16), (3, 24), respectively. (The hidden secret data are left untouched when the eight JPEG decompressions are performed.) Notably, the proposed method only uses the quantized coefficients of the brightness component Y of the cover image to hide secret data. The quantized coefficients of the color components Cb and Cr of the cover image are not used in hiding to minimize color distortion.

As shown in Figure 2.7(m), three JPEG stego codes Stego-of-JPHide.jpg, Stego-of-JSteg.jpg, and Stego-of-OutGuessOld.jpg yield the strongest response (the highest score *** in terms of the measure is provided by Provos's StegDetect analysis tool [11]), indicating that the three JPEG stego codes must contain some hidden data. However, when our JPEG stego codes are examined by the StegDetect analysis tool, all the responses are negative. Therefore, our JPEG stego codes can pass StegDetect analysis. Similar observations are made when the color image Lena is replaced by Jet or other images.

Last, the length of our JPEG stego code is discussed. When the 512×512 grayscale image Lena is compressed using JPEG with a quality factor in the range of 10 to 95, the length of the JPEG code (without any hidden secret) is between 8,119 and 94,581 bytes. The lengths of the JPEG stego codes of Lena are 55,432, 62,236, 70,237, 66,307, 78,484, 90,114, 67,257, and 82,207 bytes when (N_{bit}, N_{QC}) are (1, 16), (1, 24), (1, 32), (2, 16), (2, 24), (2, 32), (3, 16), and (3, 24), respectively. Therefore, the attackers will not be suspicious about the lengths of our JPEG stego codes since their lengths all fall in the reasonable range of 8,119 to 94,581 bytes. Similar phenomena also hold when the grayscale image Lena is replaced by Jet or others.

2.5 Summary

This chapter proposes a reversible JPEG hiding method with high hiding capacity and high hiding ratio. The secret data are embedded in a JPEG code. In decoding, after lossless extraction of the secret data, the original JPEG code is reconstructed. From Tables 2.1-2.2 and Figures 2.4-2.5, the proposed method outperforms that of the related JPEG hiding methods [12,15,17,18] in terms of hiding capacity, hiding ratio, and stego-image quality. The reversibility makes the reconstructed JPEG code to be used many times without any degrading.



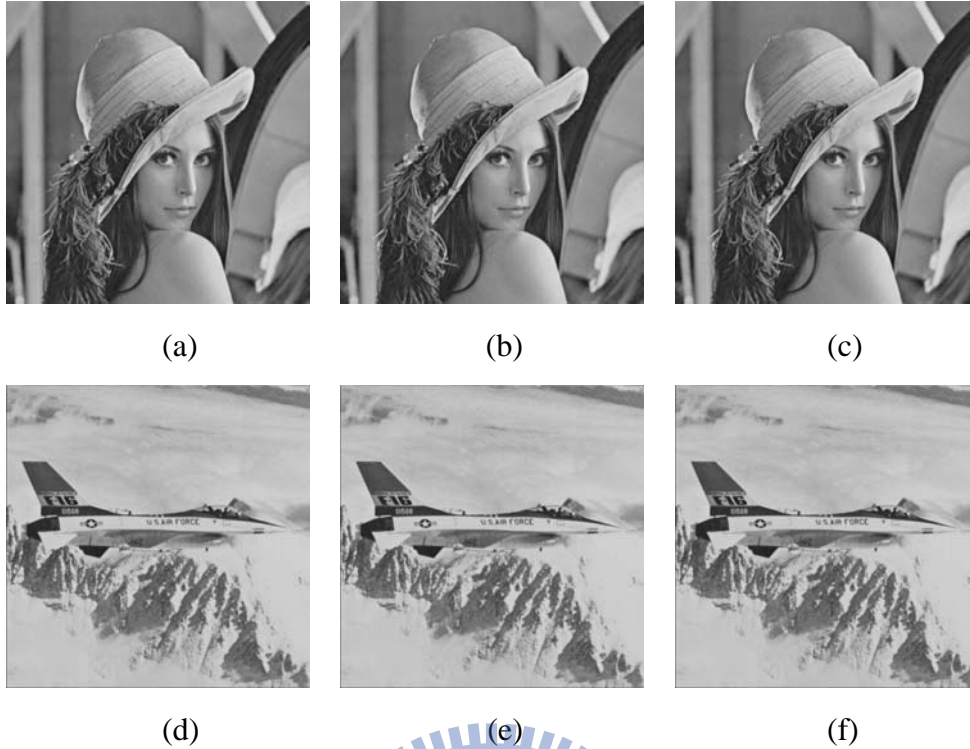
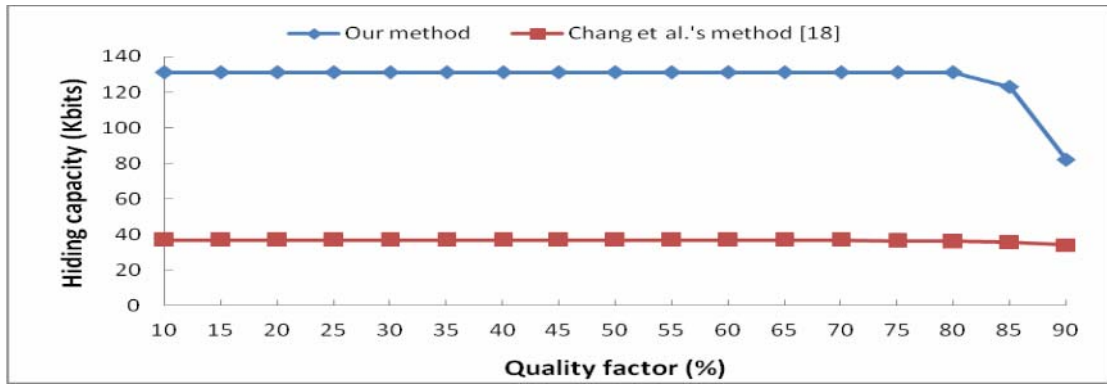
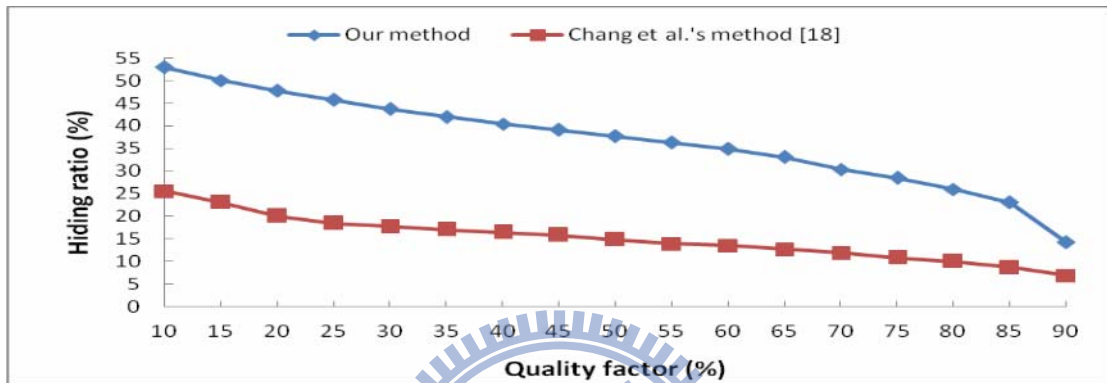


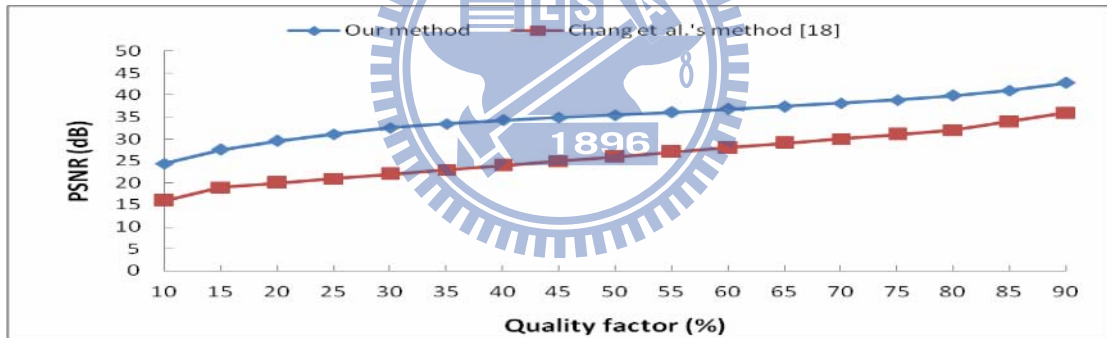
Figure 2.3. Images decompressed from JPEG codes. (a) The 41.75-dB image Lena decompressed from the JPEG-Q85 code without any hidden secret. (b) The 39.12-dB stego-image Lena decompressed from our JPEG stego code. (c) The 41.75-dB decrypted-decompressed image Lena derived from our JPEG stego code. (d)-(f) Same as (a)-(c) except that the image Lena is replaced by Jet, and the PSNRs of (d) to (f) are 41.15dB, 38.75 dB, and 41.15dB, respectively.



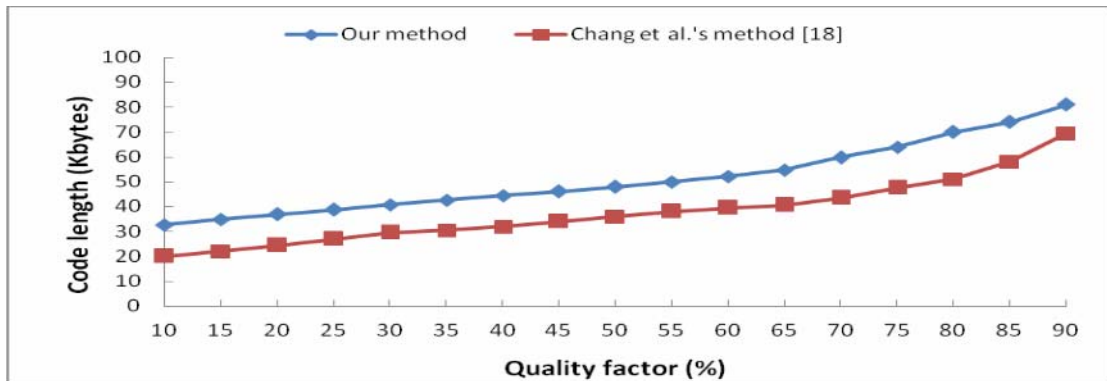
(a)



(b)

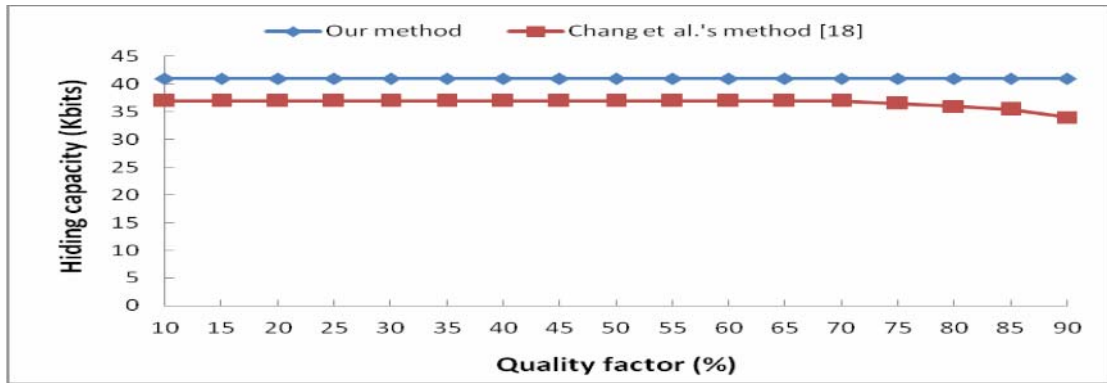


(c)

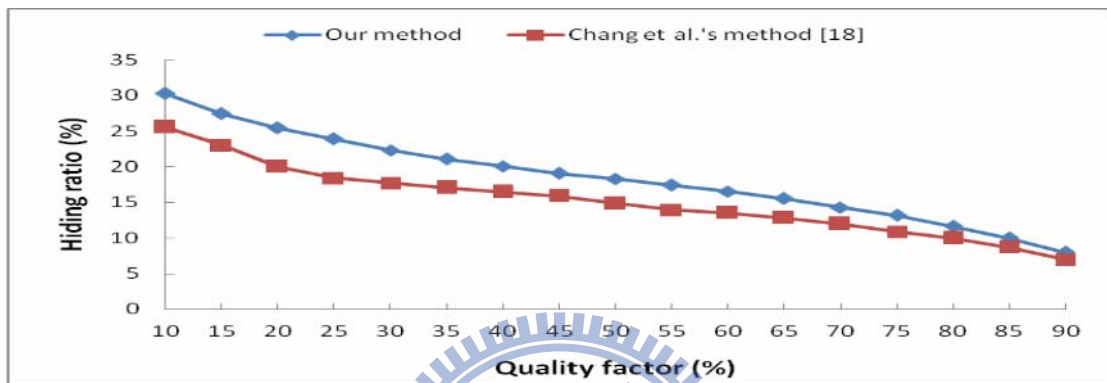


(d)

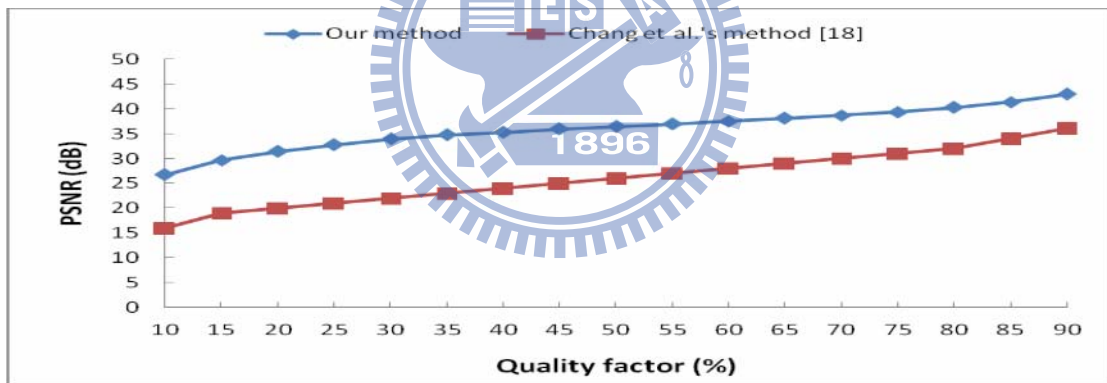
Figure 2.4. Comparisons between Chang et al.'s method [18] and ours when the two parameters are $N_{bit}=2$ and $N_{QC}=16$. The comparisons are (a) hiding capacity, (b) hiding ratio, (c) stego-image quality, and (d) average length of the JPEG stego codes.



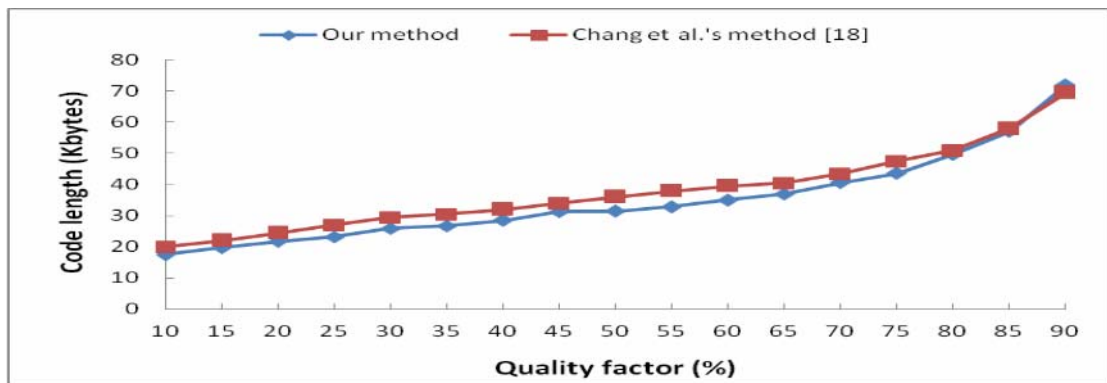
(a)



(b)



(c)



(d)

Figure 2.5. Same as Figure 2.4 except that the two parameters are $N_{bi}=1$ and $N_{QC}=10$.

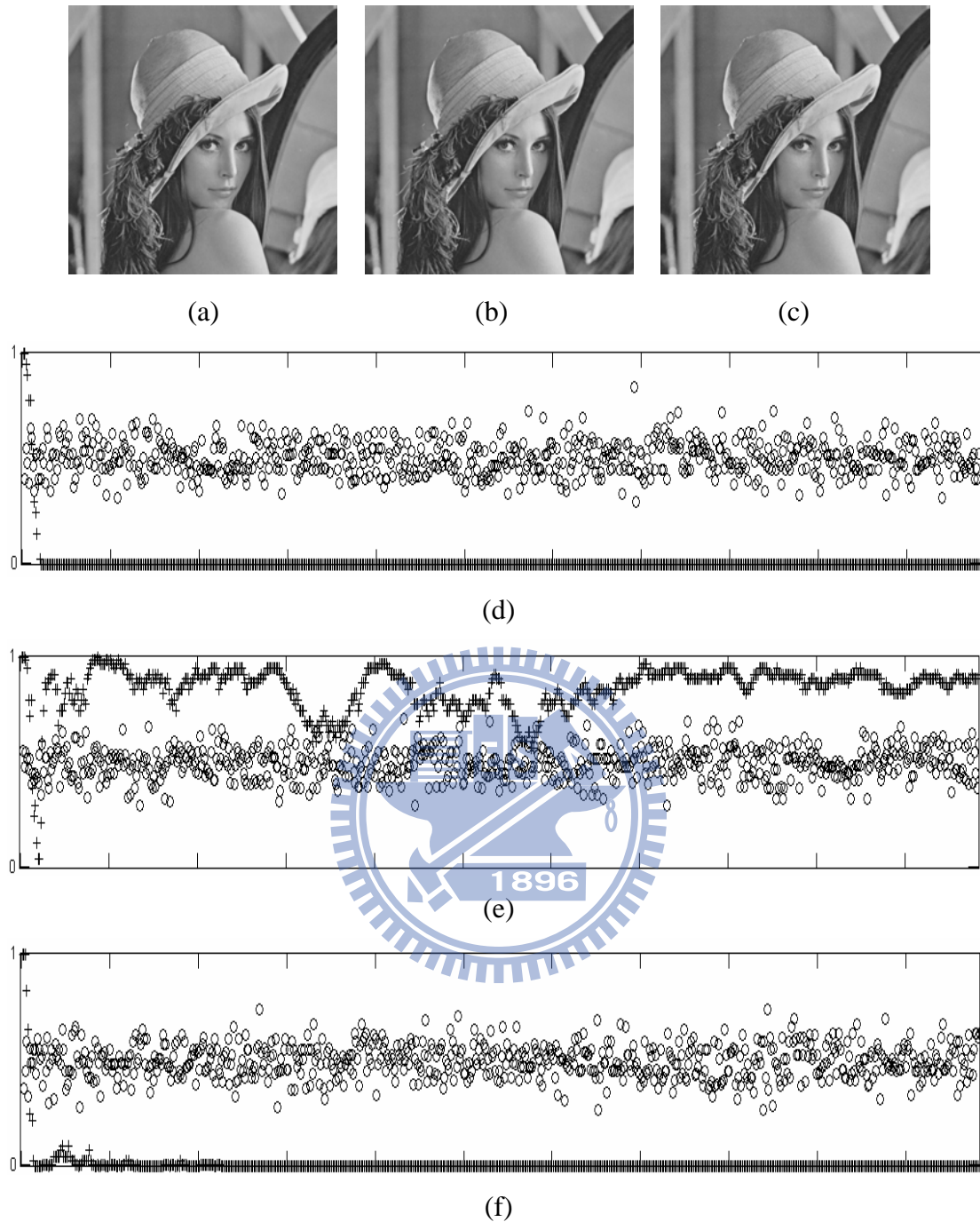


Figure 2.6. Results of Chi-square analysis: (a) the image Lena without any hidden secret; (b) the stego-image Lena generated using the LSB substitution method; (c) the stego-image Lena decompressed from our JPEG stego code after a secret of size 253,952 bits is hidden; (d)-(f) the results of Chi-square analysis of the pixels in (a), (b), and (c), respectively.



Filename	Detection
F:\StegDetect\OriginalLena.jpg	negative
F:\StegDetect\Stego-of-JPHide.jpg	jphide(***)
F:\StegDetect\Stego-of-JSteg.jpg	jsteg(***)
F:\StegDetect\Stego-of-OutGuessOld.jpg	outguess(old)(***)
F:\StegDetect\ourStego-(1,16).jpg	negative
F:\StegDetect\ourStego-(1,24).jpg	negative
F:\StegDetect\ourStego-(1,32).jpg	negative
F:\StegDetect\ourStego-(2,16).jpg	negative
F:\StegDetect\ourStego-(2,24).jpg	negative
F:\StegDetect\ourStego-(2,32).jpg	negative
F:\StegDetect\ourStego-(3,16).jpg	negative
F:\StegDetect\ourStego-(3,24).jpg	negative

(m)

Figure 2.7. Results of StegDetect analysis: (a) the cover image Lena decompressed from the JPEG-Q85 code without any hidden secret; (b)-(d) the stego-images decompressed from the JPEG stego codes of JPHide [10], JSteg [12], OutGuess [11], respectively; (e)-(l) the stego-images decompressed from our eight JPEG stego codes, respectively; (m) the results of StegDetect analysis of the twelve JPEG codes, the decompressed images of which are in (a)-(l).

Table 2.1. Comparisons of hiding capacity, hiding ratio, and stego-image quality among the related JPEG hiding methods [12,15,17] and ours when the grayscale image Lena is compressed using JPEG with $QF=85$.

Scheme	(N_{bit}, N_{QC})	Capacity factor	Hiding capacity (bits)	Hiding ratio (%)	Stego-image quality (dB)
[12]			31,933	8.85%	40.70
[15]			212,992	30.0%	29.64
[17]		0.2	33,790	8.68%	40.21
		0.6	54,652	14.0%	38.10
		1.0	63,083	16.2%	36.56
Our scheme	(1, 16)		65,536	14.78%	40.99
	(1, 24)		98,304	19.74%	40.10
	(1, 32)		131,072	23.33%	38.22
	(2, 16)		122,880	23.16%	41.11
	(2, 24)		188,416	30.01%	40.53
	(2, 32)		253,952	35.23%	39.12
	(3, 16)		126,976	23.60%	41.13
	(3, 24)		208,896	31.76%	40.49

Table 2.2. Same as Table 2.1 except that the image Lena is replaced by Jet.

Scheme	(N_{bit}, N_{QC})	Capacity factor	Hiding capacity (bits)	Hiding ratio (%)	Stego-image quality (dB)
[12]			32,868	9.11%	40.07
[15]			212,992	29.9%	28.16
[17]		0.2	35,678	9.26%	39.38
		0.6	58,254	15.1%	37.32
		1.0	66,363	17.2%	35.77
Our scheme	(1, 16)		65,536	14.70%	40.47
	(1, 24)		98,304	19.58%	39.67
	(1, 32)		131,072	23.24%	37.91
	(2, 16)		122,880	23.03%	40.58
	(2, 24)		188,416	29.74%	40.06
	(2, 32)		253,952	35.01%	38.75
	(3, 16)		126,976	23.44%	40.60
	(3, 24)		208,896	31.43%	40.03

Table 2.3. Our hiding capacity, hiding ratio, and stego-image quality when the grayscale image Lena is compressed using JPEG with $QF=75$.

(N_{bit}, N_{QC})	Hiding capacity (bits)	Hiding ratio (%)	Stego-image quality (dB)
(1, 16)	65,536	18.91%	38.56
(2, 16)	131,072	28.56%	38.87
(3, 16)	163,840	32.38%	38.94
(4, 16)	167,936	32.81%	38.96
(1, 24)	98,304	24.15%	37.09
(2, 24)	196,608	35.76%	37.78
(3, 24)	262,144	41.07%	37.91
(4, 24)	278,528	41.96%	37.96
(1, 32)	131,072	27.60%	34.48
(2, 32)	262,144	40.59%	35.59
(3, 32)	360,448	46.74%	35.95
(4, 32)	405,504	47.40%	35.81



Chapter 3

Multi-Threshold Progressive Image Sharing with Compact Shadows

This chapter proposes a multi-threshold progressive reconstruction method. An important image is encoded three times using JPEG: first with a low-quality factor, then with a medium-quality factor, and last with a high-quality factor. Huffman coding is employed to encode the difference between the important image and the high-quality JPEG decompressed image. The three JPEG codes and the Huffman code are shared, respectively, according to four pre-specified thresholds. The n generated equally important shadows can be stored or transmitted using n channels in parallel. Cooperation among these shadows can progressively reconstruct the important image. The reconstructed image is loss-free when the number of collected shadows reaches the largest threshold. Each shadow is very compact and so can be hidden successfully in the JPEG codes of cover images to reduce the probability of being attacked when transmitted in an unfriendly environment. The proposed scheme is easier than the progressive image sharing schemes [27-30] to apply to scalable Moving Picture Experts Group (MPEG) video transmission [49,50], and the shadows herein can resist differential attack [51-53]. Comparisons with other image sharing methods are also made. The rest of this chapter is organized as follows. Section 3.1 reviews related works. Section 3.2 presents the proposed scheme. Section 3.3 reports the experimental results and makes comparisons with other methods. Section 3.4 discusses security. Last, Section 3.5 summarizes this chapter.

3.1 Related Works

3.1.1 Secret Image Sharing Method of Thien and Lin

Thien and Lin [21] propose a (k, n) threshold method for splitting a grayscale secret image into n shadows. First, all of the gray values between 251 and 255 in the secret image must be truncated to 250 because the arithmetic operations in Eq. (3.1) are modulo 251. They then use a key to permute the pixels of the secret image, and the permuted image is partitioned into several sectors of k pixels each. For each not-yet processed sector, define a polynomial

$$f(z) = r_0 + r_1z + r_2z^2 + \dots + r_{k-1}z^{k-1} \pmod{251}, \quad (3.1)$$

where r_0 to r_{k-1} are the k pixel values. The n values $f(1)$, $f(2)$, ..., and $f(n)$ are calculated and then attached to the n shadows.

After all sectors have been processed, the n shadows are generated. Since every k pixels in the secret image contribute a single pixel to each of the n generated shadows, each shadow size is $1/k$ of the secret image size. In collecting at least k shadows, Thien and Lin take the first not-yet-used pixel from each of the k shadows and use these k pixel values $f(z_1)$, $f(z_2)$, ..., and $f(z_k)$ to evaluate the k coefficients in Eq. (3.1) for the first sector by reconstructing the $(k-1)$ -degree polynomial $f(z)$ as

$$\begin{aligned} f(z) = & f(z_1) \frac{(z - z_2)(z - z_3) \dots (z - z_k)}{(z_1 - z_2)(z_1 - z_3) \dots (z_1 - z_k)} \\ & + f(z_2) \frac{(z - z_1)(z - z_3) \dots (z - z_k)}{(z_2 - z_1)(z_2 - z_3) \dots (z_2 - z_k)} \\ & + \dots + f(z_r) \frac{(z - z_1)(z - z_2) \dots (z - z_{k-1})}{(z_k - z_1)(z_k - z_2) \dots (z_k - z_{k-1})} \pmod{251}. \end{aligned} \quad (3.2)$$

By processing all pixels of the k shadows in order, they obtain the permuted image, which is then de-permuted to reveal the secret image.

An example is presented in the following. To divide $k=2$ pixel values 100 and 200 into $n=3$ shadows, Eq. (3.1) is used to compute the three shadows: $f(1) =$

$(100+200 \times 1) \bmod 251 = 49$; $f(2) = (100+200 \times 2) \bmod 251 = 249$; and $f(3) = (100+200 \times 3) \bmod 251 = 198$. Later, if two shadows $f(1)=49$ and $f(3)=198$ are received, Eq. (3.2) is used to reconstruct the polynomial as $f(z) \equiv f(1) \times (z-3)/(1-3) + f(3) \times (z-1)/(3-1) \equiv 49 \times (z+248)/249 + 198 \times (z+250)/2 \equiv 49 \times (z+248) \times 125 + 198 \times (z+250) \times 126 \equiv 100 + 200z \pmod{251}$. The original pixel values, 100 and 200, are therefore obtained.

3.1.2 Galois Field

A Galois field (GF) is a finite field of p^t elements with addition (+) and multiplication (\times) operations that satisfy commutative, associative, and distributive laws where p is a prime number and t is a positive integer. The arithmetic over $\text{GF}(p)$ is generally the same as modulo p , and therefore Thien and Lin [21] use a Galois field with $p^t = p^1 = p = 251$ elements. The proposed method employs a Galois field with 2^t elements, and arithmetic over $\text{GF}(2^t)$ is based on the representation of each element in $\text{GF}(2^t)$. An element in $\text{GF}(2^t)$ is generally represented using a polynomial-basis representation, as a binary polynomial of degree less than t . The t -tuple of coefficients of the binary polynomial corresponds to the binary representation of an integer between 0 and $2^t - 1$.

Let $A = (a_{t-1} \dots a_1 a_0)_2$ and $B = (b_{t-1} \dots b_1 b_0)_2$ be two t -bit binary elements in $\text{GF}(2^t)$. In the polynomial-basis representation, A and B are $A(X) = a_{t-1}X^{t-1} + \dots + a_1X + a_0$ and $B(X) = b_{t-1}X^{t-1} + \dots + b_1X + b_0$, respectively. Define the addition of A and B as

$$A + B = \sum_{i=0}^{t-1} a_i \oplus b_i, \quad (3.3)$$

where \oplus is the exclusive-or (XOR) operator. For the subtraction, because each element in $\text{GF}(2^t)$ is its own additive inverse, the subtraction of B from A is defined as

$$A - B = A + (-B) = A + B = \sum_{i=0}^{t-1} a_i \oplus b_i. \quad (3.4)$$

The multiplication and division involve a primitive polynomial $P(X)$ where $P(X)$ is a t -degree irreducible polynomial (i.e., it has no nontrivial factors). To multiply A by B , the remainder $C(X) = c_{t-1}X^{t-1} + \dots + c_1X + c_0$ is computed in a long division, defined by

$$C(X) = A(X)B(X) \bmod P(X), \quad (3.5)$$

where the operations of the binary coefficients in the polynomial multiplication and in the mod $P(X)$ operations are all modulo two such that all the resulting coefficients are still in $\{0, 1\}$ and therefore binary. After the binary polynomial $C(X) = c_{t-1}X^{t-1} + \dots + c_1X + c_0$ has been determined using Eq. (3.5), the multiplication of the two t -bit binary elements A and B in $\text{GF}(2^t)$ is defined as

$$A \times B = C = (c_{t-1} \dots c_1 c_0)_2. \quad (3.6)$$

Last, to divide A by B , Eqs. (3.5) and (3.6) are used to multiply A by B^{-1} , where B^{-1} is the unique element U in $\text{GF}(2^t)$ such that $[B(X)U(X)] \bmod P(X) = 1$.

3.2 Proposed method

3.2.1 Encoding

The quality factor $QF \in \{0, 1, \dots, 100\}$ in JPEG is used to control image quality. To reconstruct an important image with various quality levels based on the number of received JPEG stego codes, a low-quality factor $QF_L \in \{0, 1, \dots, 5\}$, a medium-quality factor $QF_M \in \{10, 11, \dots, 25\}$, and a high-quality factor $QF_H \in \{55, 56, \dots, 85\}$ are used to generate, respectively, a low-quality JPEG code s_1 , a medium-quality JPEG code s_2 , and a high-quality JPEG code s_3 . The quality levels of the JPEG images q_1 , q_2 , and q_3 decompressed from the codes s_1 , s_2 , and s_3 , respectively, are around 18-28 dB, 30-34 dB, and 36-40 dB. To reconstruct the important image error-freely, a difference image q_4 is created by subtracting from the important image the high-quality JPEG image q_3 decompressed from the high-quality JPEG code s_3 . The difference image q_4 is

compressed using Huffman coding to generate the Huffman code c_4 . Last, based on the five user-defined integer parameters $2 \leq k_1 < k_2 < k_3 < k_4 \leq n$, the generated codes s_1 , s_2 , s_3 , and s_4 are shared according to Eq. (3.7).

As shown in Figure 3.1, the proposed $[(k_1, k_2, k_3, k_4), n]$ threshold scheme comprises four phases: (1) codes generation, (2) sharing, (3) shares combining, and (4) data hiding. The $[(k_1, k_2, k_3, k_4), n]$ threshold sharing algorithm is summarized here:

The $[(k_1, k_2, k_3, k_4), n]$ threshold sharing algorithm

Input: An important image; five positive integer parameters k_1, k_2, k_3, k_4 , and n , where

$$2 \leq k_1 < k_2 < k_3 < k_4 \leq n; n \text{ cover images.}$$

Output: The n JPEG stego codes.

Step 1a: Compress the important image using JPEG three times (with quality factors of QF_L , QF_M , and QF_H , respectively), yielding a low-quality JPEG code s_1 , a medium-quality JPEG code s_2 , and a high-quality JPEG code s_3 .

Step 1b: Compute the difference image q_4 by subtracting from the important image its high-quality JPEG image q_3 decompressed from the JPEG code s_3 . Compress the difference image q_4 using Huffman coding to create the Huffman code s_4 .

Step 2: For each code s_i ($i=1,2,3,4$), use Eq. (3.7) to split the code s_i into n shares.

Step 3: For each $x=1,2,\dots,n$, the x 'th shadow is formed by binding together the x 'th share of s_1 , the x 'th share of s_2 , the x 'th share of s_3 , and the x 'th share of s_4 .

Step 4: Use the JPEG data hiding method in Chapter 2 to hide the n shadows in the n JPEG codes of the n cover images, respectively. (This generates n desired JPEG stego codes, and the cooperation of several JPEG stego codes can view the important image at certain quality levels.)

Note: In step 2 above, the code s_i is divided into sectors of k_i bytes each. Each byte is treated as a number between 0 and 255. Our share-generating polynomial is

$$g(x) = b_0 + b_1x + b_2x^2 + \dots + b_{k_i-1}x^{k_i-1} \text{ [over GF(256)]}, \quad (3.7)$$

where b_0 to b_{k_i-1} are the k_i numbers of each sector, and the computations in Eq. (3.7) are over GF(256). Then, $g(1)$ to $g(n)$ are sequentially assigned to n shares. Since each sector of k_i bytes contributes only a single byte [a value in the range 0 to 255 and determined by Eq. (3.7)] to each share of the code s_i , each share size is k_i times smaller than that of the code s_i . In step 3, each shadow has size $\sum_{i=1}^4 |s_i| / k_i$, where $|s_i|$ denotes the length of the code s_i . In step 4, to avoid attracting the attention of attackers, the n shadows are hidden in n JPEG codes.

3.2.2 Decoding

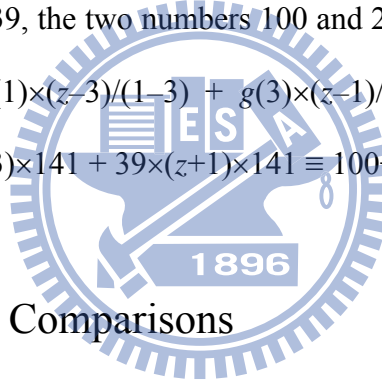
When any k_1 of the n JPEG stego codes are received, the k_1 shadows can be extracted from the k_1 JPEG stego codes by inverse hiding. For each $x=1,2,\dots,k_1$, the x 'th shadow is partitioned to yield the x 'th share of each code s_i ($1 \leq i \leq 4$). The k_1 shares of the code s_1 are used to reconstruct the low-quality JPEG code s_1 using the Lagrange interpolation method. The reconstructed JPEG code s_1 is then decompressed to yield the low-quality JPEG image q_1 , which is an approximate version of the original important image.

When k_2 (or k_3) JPEG stego codes are available, the reconstruction process is similar to that described earlier, and the reconstructed image q_2 (or q_3) will be of medium (or high) quality. Last, if at least k_4 JPEG stego codes are received, the k_4 shadows can also be extracted from the k_4 JPEG stego codes by inverse hiding. Then, for each $x=1,2,\dots,k_4$, the x 'th shadow is divided to generate the x 'th share of each code s_i ($1 \leq i \leq 4$). The k_4 shares of the code s_4 are used in inverse sharing to reconstruct the

Huffman code s_4 by Lagrange interpolation, and the reconstructed code s_4 is then decompressed to generate the difference image q_4 . Adding the difference image q_4 to the image q_3 yields the error-free important image.

3.2.3 Example of Sharing and Inverse-Sharing Processes based on GF(256)

An example of the sharing and inverse-sharing processes based on GF(256) and $P(X)=X^8+X^4+X^3+X+1$ is presented. To partition $k_i=2$ numbers 100 and 200 of 8 bits each into $n=3$ shares, Eq. (3.7) is used to compute the three shares: $g(1) \equiv 100+200 \times 1 \equiv 100+200 \equiv 172$ [over GF(256)]; $g(2) \equiv 100+200 \times 2 \equiv 100+139 \equiv 239$ [over GF(256)]; and $g(3) \equiv 100+200 \times 3 \equiv 100+67 \equiv 39$ [over GF(256)]. In obtaining the two shares $g(1)=172$ and $g(3)=39$, the two numbers 100 and 200 are revealed by Lagrange interpolation, as $g(z) \equiv g(1) \times (z-3)/(1-3) + g(3) \times (z-1)/(3-1) \equiv 172 \times (z+3)/(1+3) + 39 \times (z+1)/(3+1) \equiv 172 \times (z+3) \times 141 + 39 \times (z+1) \times 141 \equiv 100+200z$ [over GF(256)].



3.3 Experiments and Comparisons

3.3.1 Experimental Results

The inequalities $(k_1=3) < (k_2=4) < (k_3=5) < (k_4=6)$ and the irreducible polynomial $P(X)=X^8+X^4+X^3+X+1$ are used to generate $n=6$ shadows of the important image. The JPEG source code used in the experiments is taken from the fourth public release of the Independent JPEG Group's free JPEG software [43]. The quality of an image is measured by the PSNR.

In the first experiment, the 512×512 grayscale important image Lena, displayed in Figure 3.2, is encoded using JPEG with three quality factors $QF_L=5$, $QF_M=25$, and $QF_H=85$. The four codes s_1 , s_2 , s_3 , and s_4 have lengths 5,750, 13,787, 45,972, and 115,458 bytes, respectively. The six cover images Peppers, Jet, Boat, Lake, Baboon,

and Zelda, shown in Figure 3.3, are all encoded using JPEG with $QF=75$ to hide the six shadows, and therefore generate the six JPEG stego codes. Figure 3.4 displays the $n=6$ images decompressed from our six JPEG stego codes without any extraction of the hidden shadows, and the PSNRs of the six decompressed images are 37.42, 37.41, 36.40, 34.39, 32.73, and 38.67 dB, respectively. When different numbers of JPEG stego codes are received, the reconstructed versions q_1, q_2, q_3 , and q_4 of Lena are as plotted in Figure 3.5, and the respective PSNRs are 27.32, 33.67, 39.35, and ∞ dB.

In the second experiment, the important image is the 512×512 grayscale image Tiffany. The six shadows are generated and then remain hidden in the six JPEG codes generated in the first experiment. The PSNRs of the decompressed images from the six JPEG stego codes are 37.75, 37.70, 36.65, 34.61, 32.97, and 38.96 dB, respectively. (These values are a little better than the 37.42, 37.41, 36.40, 34.39, 32.73, and 38.67 dB values obtained in the first experiment.) For Tiffany, the PSNRs of the versions reconstructed using any three, four, and five JPEG stego codes are 28.37, 34.12, and 39.79 dB, respectively (these values are a little better than those, 27.32, 33.67, and 39.35 dB, for Lena). When six JPEG stego codes are collected, the reconstructed Tiffany is identical to the original Tiffany.

Last, Table 3.1 shows the bit rates [bits per pixel (bpp)] of the JPEG-Q75 codes created using JPEG with $QF=75$ before and after hiding our shadows. The bit rate will increase significantly after hiding a large-size secret. However, the bit rate of our JPEG stego code still falls in the reasonable range of JPEG, i.e., the bit rate of our JPEG stego code is smaller than that of the JPEG-Q95 code generated using JPEG with $QF=95$, as shown in Table 3.1. This alleviates the problem of code length. [The reason using $QF=95$ as the upper bound to examine the bit rate of our JPEG stego codes is that, as stated in Kim et al.'s work [54], the general quality factors (QF s) used in digital cameras are between 90 and 95.]

3.3.2 Comparisons

Table 3.2 compares other sharing schemes [21,25-30] with ours in terms of shadow-size expansion, progressive ability, and lossless reconstruction ability. Each shadow in two of the related works [25,27] is four times larger than the original important image, indicating that size expansions occur. In contrast, each shadow in all of the associated works [21,26,28-30] and ours is smaller than the original important image. Although Thien and Lin [21], Tso [26], and Hung et al. [30] all shared the image without size expansion, Thien and Lin [21] and Tso [26] could not reconstruct the image progressively, whereas Hung et al. [30] could not reconstruct the image in an error-free manner. Only Chen and Lin [28], Wang and Shyu [29], and ourselves have achieved reconstruction with all three desired characteristics.

Among these three methods, as presented in Table 3.3, each shadow size herein (12.89% of 512×512 bytes) is smaller than those in Chen and Lin's method [28] (22.22%) and Wang and Shyu's [29] (50%). Therefore, the transmission time in the proposed method is less, and the survival rate in an unfriendly environment, in which the network connection time is unstable among the n channels used to store the n shadows, is increased. Equivalently, in the proposed method, the storage space in a distributed storage system is most reduced. The smaller size of the shadows also facilitates the hiding of shadows in stego media.

The construction of Table 3.3, which compares the shadow sizes among non-expanded schemes, is explained in the following. For fairness of comparison, the shadow sizes in Table 3.3 are all measured before hiding: all are shadow sizes, and none is a stego media size. This action eliminates the size-altering effects of particular post-processing (hiding) approaches. Assume that the important image is the 512×512 grayscale image Lena, and the (largest) threshold value is set to six for all schemes, except that Hung et al.'s scheme [30] uses five as the largest threshold value because

their scheme did not provide a version with a threshold value being six. For each $x=1,2,\dots,n$, the four x 'th shares are combined to form the x 'th shadow, and therefore each shadow in the proposed method has size $\sum_{i=1}^4 |s_i|/k_i = (5,750/3) + (13,787/4) + (45,972/5) + (115,458/6) = 33,802$ bytes (which is 12.89% of the size of the 512×512 grayscale image Lena).

According to Table 3.3, the shadow sizes in the proposed method and Hung et al.'s [30] are more economic than those in the related works [21,26,28,29]. However, Hung et al.'s method [30] is not lossless when all shadows are collected. In fact, if the original important image can be satisfactorily reconstructed with some loss, then our step 1b can be omitted, such that no Huffman code s_4 is generated. Then, each of our shadows can be reduced to $\sum_{i=1}^3 |s_i|/k_i = (5,750/3) + (13,787/4) + (45,972/5) = 14,559$ bytes (which is 5.55% of the size of the 512×512 grayscale image Lena). Restated, the size of each shadow in this lossy version is about half of that in Hung et al.'s scheme. Moreover, in this lossy version, the total shadow size is $14,559 \times 6 = 87,354$ bytes, which is still smaller than $30,723 \times 5 = 153,615$ bytes obtained by Hung et al. [30]. When the five shadows are collected, the 39.35-dB Lena [identical to that in Figure 3.5(c)] is reconstructed, better than the 37.04-dB Lena revealed by Hung et al. [30].

Last, the size of each shadow in the proposed $[(k_1, k_2, k_3, k_4), n]$ threshold scheme is $\sum_{i=1}^4 |s_i|/k_i$. Therefore, it is suggested that the readers set the largest threshold k_4 to n to save storage space. However, if the readers want to have more freedom, they may use their own choice of a threshold k_4 being less than n , at the price of wasting space for the shadows. When k_4 is less than n , a simulation is done in the following. Assume that n , the number of cover images, is at least six. In general, the smallest threshold k_1 cannot be one because the purpose of sharing is that no participant alone can be trusted. Therefore, $\{1 < k_1 = 2 < k_2 = 3 < k_3 = 4 < k_4 = 5 < n = 6\}$ is used to generate $n=6$ shadows,

the size of which is then compared to those in the image sharing schemes [21,26,28-30] when the (largest) threshold value is set to five for all these schemes. The comparison results are shown in Table 3.4. It is observed that each shadow in the proposed method is still smaller than those in the related works [21,26,28,29], and each shadow in our lossy approach is also smaller than that in Hung et al.'s lossy approach [30].

3.4 Security Discussion

The code s_i ($1 \leq i \leq 4$) cannot easily be revealed if fewer than k_i shadows are intercepted. To determine the coefficients b_0 to b_{k_i-1} in Eq. (3.7), k_i equations are required. If only k_i-1 equations are available [and without loss of generality, suppose that $g(1), g(2), \dots,$ and $g(k_i-1)$ are intercepted], then the following k_i-1 equations can be reconstructed:

$$\begin{cases} g(1) = (b_0 + b_1 + \dots + b_{k_i-1}) \quad [\text{over GF}(256)], \\ g(2) = (b_0 + 2b_1 + \dots + 2^{k_i-1}b_{k_i-1}) \quad [\text{over GF}(256)], \\ \vdots \\ g(k_i-1) = [b_0 + (k_i-1)b_1 + \dots + (k_i-1)^{k_i-1}b_{k_i-1}] \quad [\text{over GF}(256)] \end{cases}$$

The preceding k_i-1 equations are solved for the k_i unknown coefficients. The set of possible solutions has 256 members, so the probability of guessing the right solution is $1/256$. Since $|s_i|/k_i$ sectors exist for the code s_i , the probability of obtaining the right code s_i is $(1/256)^{|s_i|/k_i}$. For example, for a low-quality JPEG code s_1 of size 5,000 bytes, the number of sectors is 2,500 if k_1 is 2. The probability of obtaining the correct JPEG code s_1 is $(1/256)^{2500} = 10^{-6020}$.

If the security of the shadows is to be increased, a seed may be used in a random number generator to generate a random sequence for each shadow, and then XOR

operations can be applied between the random sequence and the shadow. Each row of the XOR-encrypted shadows of the code s_i can then be circularly shifted by a certain number of bytes. Similar operations are applied to each column. This process will transform the shadows to their safer versions. Notably, each seed used to generate a random sequence is based on the creation time and shadow index. Each seed can then be kept by all n participants or held by the company leader if the leader insists on attending the decoding meeting.

As stated in three works [51-53], attackers may slightly change the plaintext and then observe the change in the ciphertext. This is so-called differential attack, which the related progressive image sharing methods [27-29] cannot handle. The attackers may try to find a relationship between the plaintext and its ciphertext. Therefore, to ensure high security, the change in the ciphertext should cover a very large area if change occurs over a small area in the plaintext.

To check this phenomenon, the number of pixels change rate (NPCR) is used to measure the number of pixels that are changed in the ciphertext when only one pixel is changed in the plaintext. To define NPCR, let X and Y be two ciphertexts of size $W \times H$, where the plaintexts of X and Y differ by only one pixel. Let $X(i, j)$ and $Y(i, j)$ be the pixel values at the position (i, j) in X and Y , respectively. Define

$$NPCR = \frac{\sum_{i,j} E(i, j)}{W \times H} \times 100\%, \quad (3.10)$$

where $E(i, j)$ is defined as

$$E(i, j) = \begin{cases} 0, & X(i, j) = Y(i, j), \\ 1, & X(i, j) \neq Y(i, j) \end{cases} \quad (3.11)$$

The unified average changing intensity (UACI) is used to measure the average intensity of the differences between two ciphertexts X and Y . It is defined as

$$UACI = \frac{1}{W \times H} \left(\sum_{i,j} \frac{|X(i,j) - Y(i,j)|}{255} \right) \times 100\%. \quad (3.12)$$

For random images, the expected values of NPCR and UACI are 99.609375% and 33.46354%, respectively, according to Kwok and Tang's work [53], which is an image encryption method rather than an image sharing method. Our NPCR values are between 99.54% and 99.60% (very close to 99.609375%), indicating that each XOR-enhanced shadow is very sensitive to a change in a single byte in the code s_1 ; our UACI values are between 33.36% and 33.45% (very close to 33.46354%), suggesting that the change of each XOR-enhanced shadow that is associated with a single-byte change in the code s_1 is very large. Similar observations are made when the code s_1 is replaced by the code $s_2, s_3,$ or s_4 . Accordingly, the enhanced version can resist differential attack. Notably, to achieve this ability to resist differential attack, the design is based on simple XOR operations, unlike other designs [51-53]. Also, this XOR-enhanced version does not increase the shadow size.

Last, since the shadows are hidden using the JPEG data hiding method in Chapter 2, the security after hiding is discussed in the following. Possible attack due to visual inspection is avoided. As presented in Section 3.3.2, in the $\{1 < k_1 = 2 < k_2 = 3 < k_3 = 4 < k_4 = 5\}$ experiment, each shadow has size $\sum_{i=1}^4 |s_i| / k_i = (5,750/2) + (13,787/3) + (45,972/4) + (115,458/5) = 42,056$ bytes. If other sets of $\{1 < k_1 < k_2 < k_3 < k_4\}$ are used to generate n shadows, then each shadow still has size smaller than 42,056 bytes. Therefore, 42,056 bytes is the largest possible shadow size for all possible combinations of $\{1 < k_1 < k_2 < k_3 < k_4\}$. Each shadow of size 42,056 bytes can be hidden in a JPEG-Q65 code of a cover image after a JPEG compression with $QF=65$. Figure 3.6 shows the six images decompressed from our six created JPEG stego codes without any extraction of the hidden shadows, and the PSNRs of the six decompressed images

are 35.73, 35.57, 34.65, 33.14, 30.94, and 35.94 dB, respectively. Visual quality of these decompressed images is acceptable, reducing the probability that the JPEG stego codes get attacked when the attackers use visual inspection to find suspicious images.

Moreover, suspicious JPEG code length is avoided. Besides the evidence shown in Table 3.1, Table 3.5 lists the bit rates of the JPEG-Q65 codes before and after hiding the largest shadow of size 42,056 bytes. The bit rates of the JPEG codes corresponding to $QF=95$ are also listed to observe whether the bit rates of our JPEG stego codes are reasonable. From Table 3.5, it is observed that even after hiding the largest shadow of size 42,056 bytes, the bit rate of our JPEG stego code is still below that of the plain JPEG-Q95 code. Therefore, the attackers will not be suspicious about the length of our JPEG stego code.

Furthermore, the proposed method can pass the Chi-square [44] and StegSpy [55] analyses, which are tools to determine whether secret data are hidden in an image or a JPEG code. For the images Peppers and Jet, Figures 3.7(a)-(d) display the results after Guillermito's Chi-square analysis tool is utilized to examine each pixel of the JPEG images mentioned there. The cross curve represents the probability that pairs of values follow a random distribution, and the circular one represents the average value of all LSBs in one block of pixels. The circular curves in Figures 3.7(a)-(d) suggest to the attackers that there is nothing strange in these JPEG images because all four circular curves are around $(0+1)/2=0.5$. Also, after a sort of latency, all cross curves are flat at zero, indicating that the possibility of the existence of the hidden secret is very low. Similar phenomena also hold when the image Peppers or Jet is replaced by other images Boat, Lake, etc.

As for the StegSpy analysis, let OriginalPeppers.jpg be the JPEG-Q65 code of the grayscale image Peppers. A random secret of size 3,000 bytes is then hidden in

OriginalPeppers.jpg using the Hiderman [56] and JPegX [57] hiding tools to generate two JPEG stego codes: Stego-of-Hiderman.jpg and Stego-of-JPegX.jpg. Let our six JPEG stego codes, the decompressed images of which are in Figure 3.6, be ourStegoPeppers.jpg, ourStegoJet.jpg, etc. Figure 3.8 displays the results after the StegSpy analysis tool inspects these nine JPEG codes. The tool finds that some data are hidden in Stego-of-Hiderman.jpg and Stego-of-JPegX.jpg, but not in our JPEG stego codes, and therefore the hidden data will be ignored by the attackers.

3.5 Summary

This chapter proposes a $[(k_1, k_2, k_3, k_4), n]$ threshold progressive reconstruction method. The contributions of the proposed method are as follows.

(1) The proposed method has progressive ability. (Those in other works [21,26] do not have progressive ability, whereas the progressive methods [28-30], which also use the sharing algorithm of Thien and Lin [21] to generate shadows, either have shadows that are larger than ours or cannot achieve lossless reconstruction.) Traditional image sharing schemes [21-26] are suitable for sharing top-secret images because of their all-or-nothing property. Progressive schemes provide a flexible means of viewing sensitive images progressively at certain quality levels. As indicated in Tables 3.2 and 3.3, each of the shadows in the proposed method is smaller than those in the related works [21,26,28,29] (and about 40% smaller than that in Hung et al.'s [30], as determined by comparing their lossy approach with our lossy approach). This improvement reduces storage space and transmission time, and facilitates the hiding of shadows. Therefore, the proposed method is better suited to transmit an image through limited communication channels.

(2) Easiness to apply to scalable MPEG video transmission. The proposed

scheme can also be adopted in the transmission of scalable MPEG video. Scalable MPEG video transmission depends on the adapting of video compression bit streams with a range of quality levels to meet various network environments or different end-user requirements. Since an MPEG video encoder also uses quality factors to control the quality of decoded video, MPEG video codes of various quality levels can be generated with various quality factors, and then these MPEG codes can be shared. Therefore, the proposed method conveniently provides a scalable MPEG video transmission system.

(3) Each of the shadows in the proposed scheme can resist differential attack, whereas the image sharing schemes [21,26,28,29] in Table 3.3 cannot. Simple XOR and circular-shift operations are adopted to enhance the security of noise-like shadows, to enable them to resist differential attack.

(4) Unlike in several works [21,25-29], the to-be-shared data in the proposed method are the compression code rather than the raw file. The use of the compression code has at least the following two advantages: (4a) since compression disturbs the correlation between adjacent pixels of an image, the permutation process that is employed elsewhere [21,26,28,29] before the image is shared can be omitted; (4b) after the inverse-sharing reconstruction, the compression code requires less storage space than the raw file used in several works [21,25-29]; yet the benefit of lossless reconstruction when most of the shadows are collected is retained.

(5) Arithmetic operations are performed over $GF(2^8)$ which can be replaced by $GF(2^t)$ for any positive integer t , rather than $GF(p)$, which is used in the image sharing schemes [21,22,24,28,30] for a prime number p . This increases the convenience of sharing digital data, which are often in binary form, regardless of whether they are pixel values or bit streams.

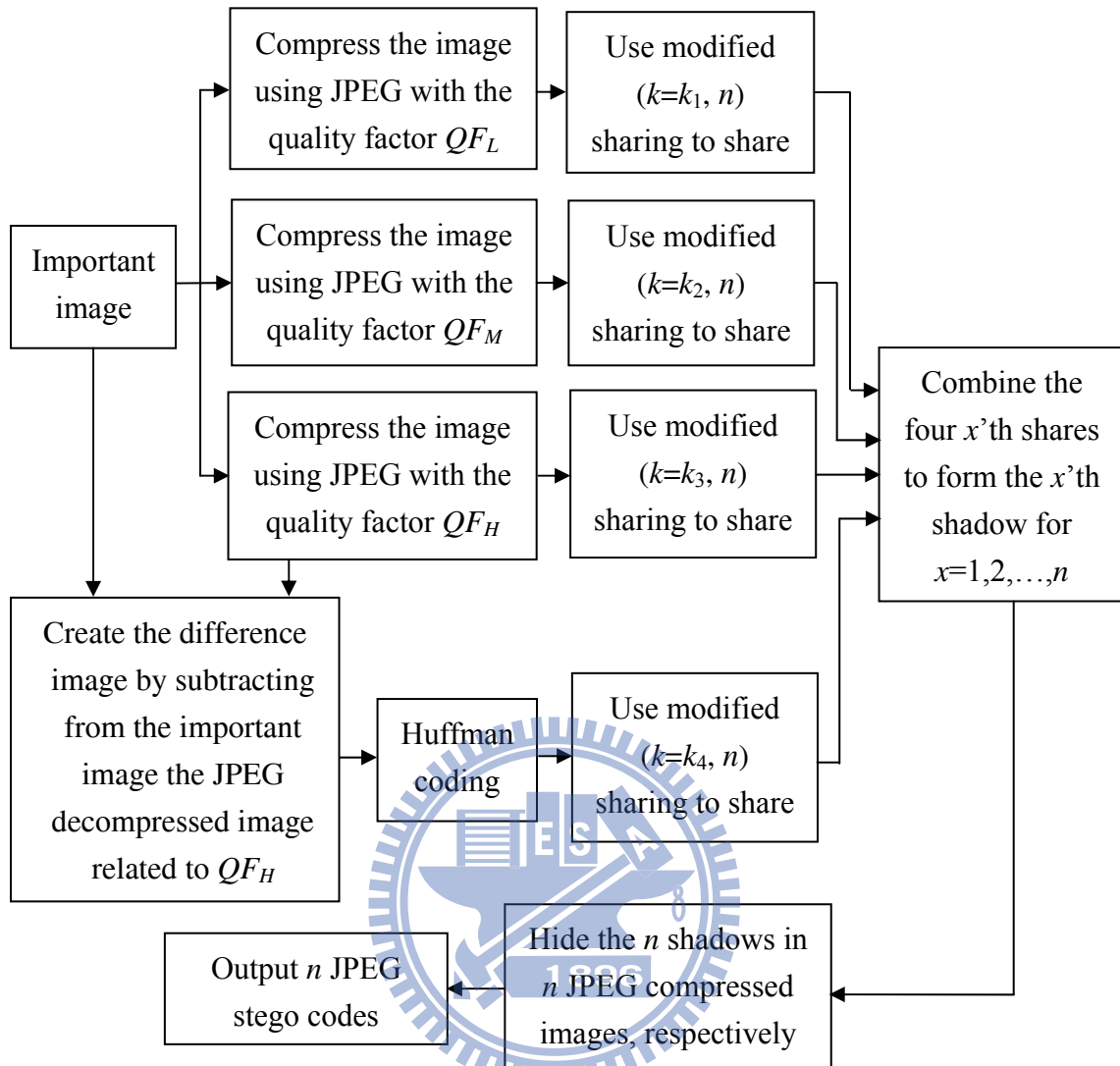


Figure 3.1. Flowchart for encoding.

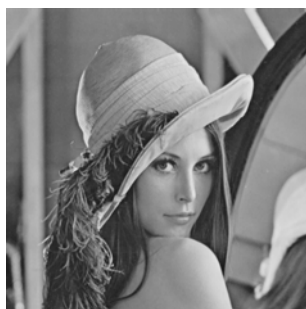


Figure 3.2. Original important image Lena used in the first experiment.



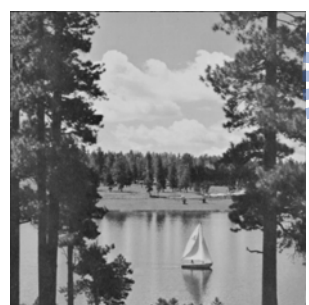
(a)



(b)



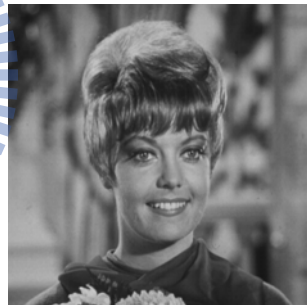
(c)



(d)



(e)



(f)

Figure 3.3. Six images Peppers, Jet, Boat, Lake, Baboon, and Zelda, which are utilized to cover the important image Lena.



(a)



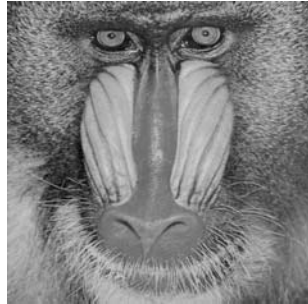
(b)



(c)



(d)



(e)



(f)

Figure 3.4. Six images decompressed from our six JPEG stego codes.

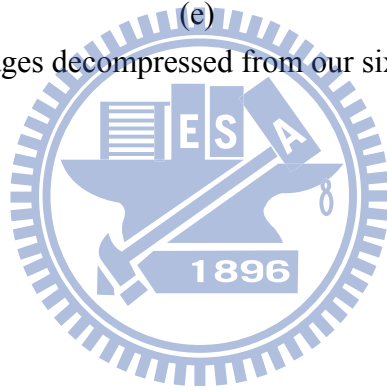




Figure 3.5. Four versions of the important image Lena reconstructed from various numbers of received JPEG stego codes: (a) from any three JPEG stego codes (PSNR=27.32 dB); (b) from any four JPEG stego codes (PSNR=33.67 dB); (c) from any five JPEG stego codes (PSNR=39.35 dB); (d) from the six JPEG stego codes (and identical to the original important image Lena).



(a)



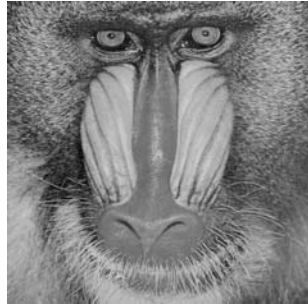
(b)



(c)



(d)

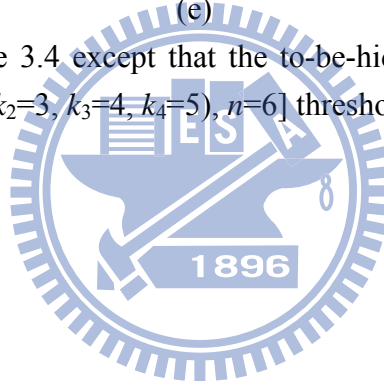


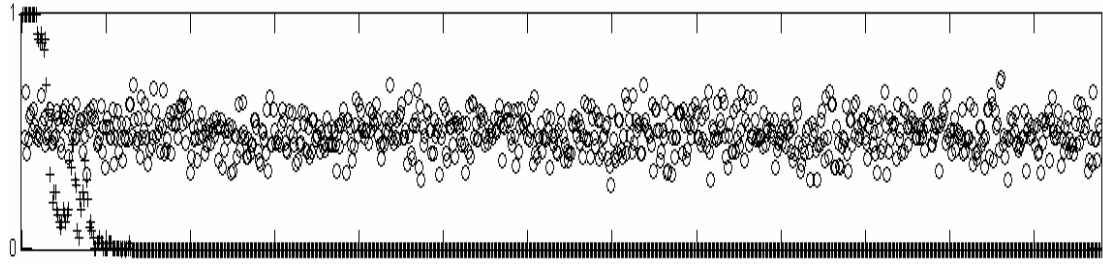
(e)



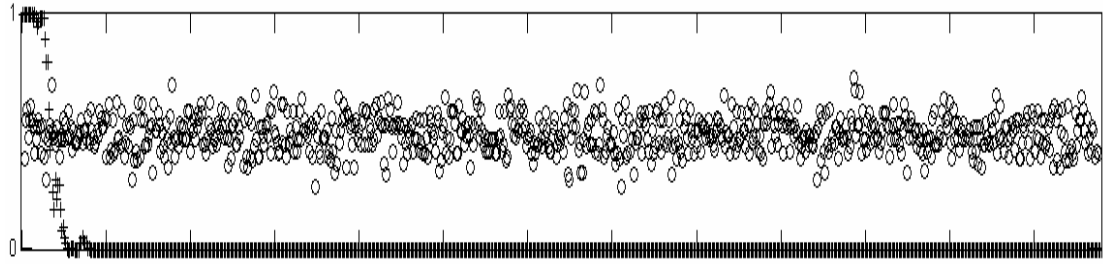
(f)

Figure 3.6. Same as Figure 3.4 except that the to-be-hidden shadows are generated using the proposed $[(k_1=2, k_2=3, k_3=4, k_4=5), n=6]$ threshold scheme.

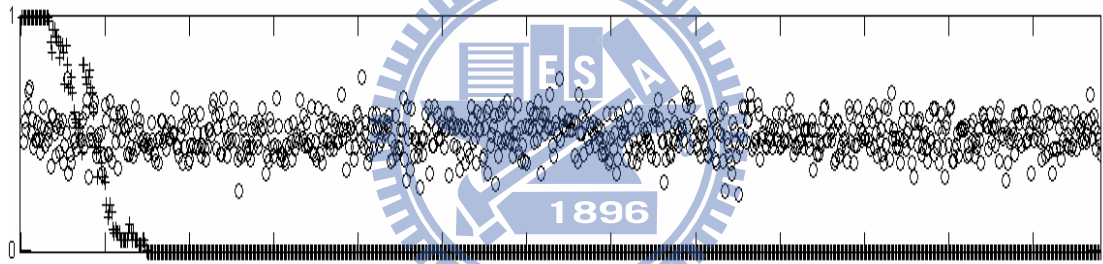




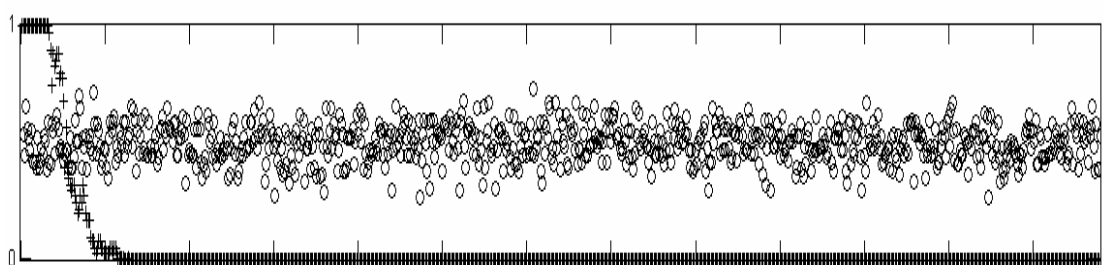
(a)



(b)



(c)



(d)

Figure 3.7. Results of Chi-Square analysis: (a) for the original JPEG cover image Peppers; (b) for our JPEG stego-image Peppers in Figure 3.6(a); (c) for the original JPEG cover image Jet; (d) for our JPEG stego-image Jet in Figure 3.6(b).

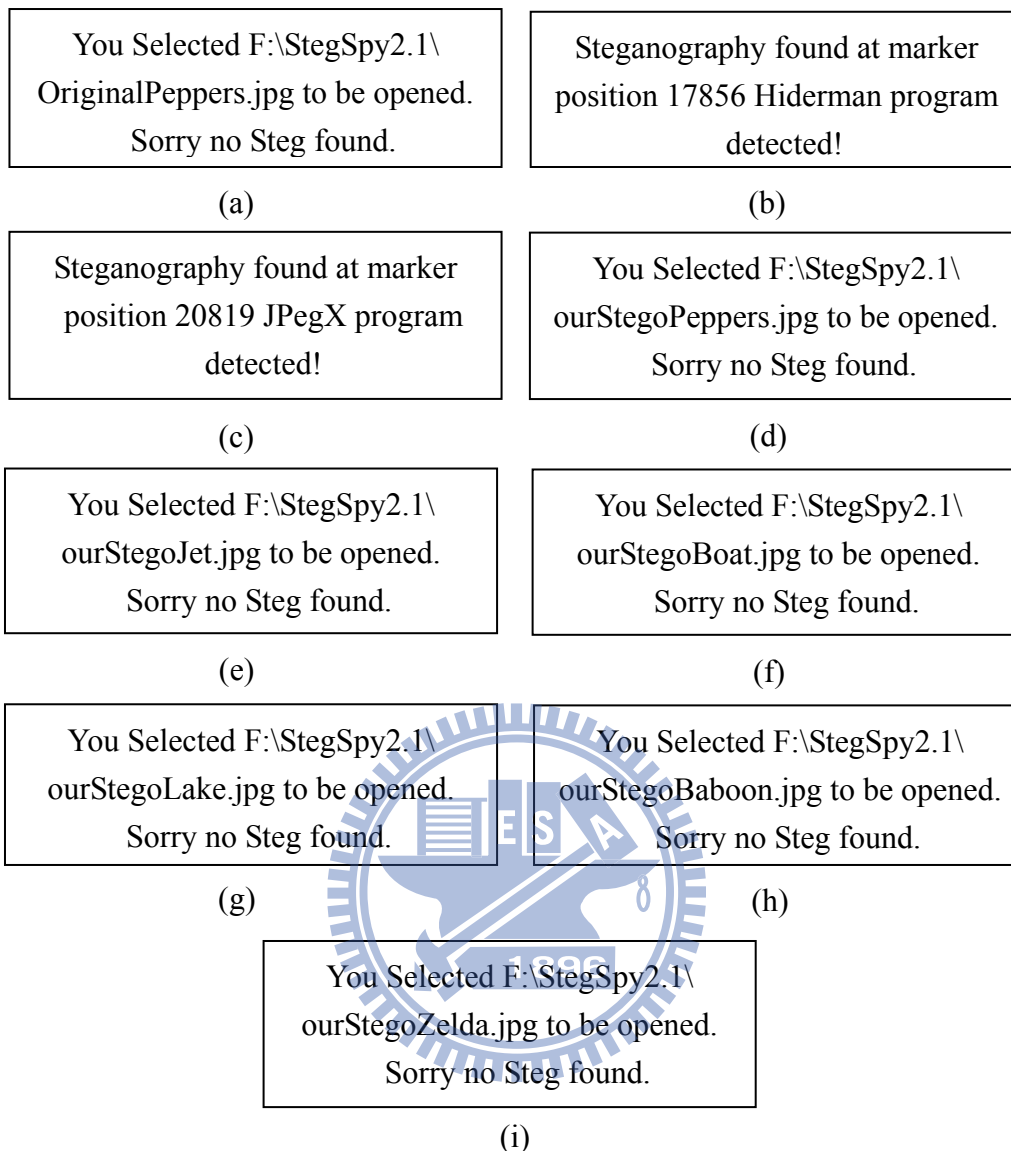


Figure 3.8. Results of StegSpy analysis: (a) for the original JPEG code of Peppers; (b)-(c) for the JPEG stego codes created using the Hiderman and JPegX hiding tools, respectively; (d)-(i) for our six JPEG stego codes, the decompressed images of which are in Figure 3.6.

Table 3.1. Bit rates (bpp) of the JPEG codes of cover images.

Cover image	bpp of the (no-hiding) JPEG-Q75 code	bpp after hiding a shadow of size 14,559 bytes in the JPEG-Q75 code	bpp after hiding a shadow of size 33,802 bytes in the JPEG-Q75 code	<i>bpp of the (no-hiding) JPEG-Q95 code</i>
Peppers	0.94	1.62	2.33	2.52
Jet	0.94	1.64	2.37	2.47
Boat	1.02	1.73	2.46	2.74
Lake	1.25	1.95	2.64	3.44
Baboon	1.89	2.63	3.25	4.09
Zelda	0.81	1.48	2.21	2.54

Table 3.2. Comparisons among image sharing methods [21,25-30] and ours.

Scheme	Non-expanded size of each shadow	Progressive ability	Lossless reconstruction using all shadows
[25]			✓
[27]			✓
[21]	✓		✓
[26]	✓		✓
[28]	✓	✓	✓
[29]	✓	✓	✓
[30]	✓	✓	
Our scheme	✓	✓	✓

Table 3.3. Comparison of shadow sizes in non-expanded methods [21,26,28-30] and ours. The (largest) threshold is set to six for all works except that Hung et al.'s method [30] uses five as the largest threshold value.

Scheme	Each shadow size (bytes)	Each shadow size over 512×512 (given image size)	Lossless reconstruction ability
[21]	43,691	16.67%	Yes
[26]	43,691	16.67%	Yes
[28]	58,254	22.22%	Yes
[29]	131,072	50%	Yes
[30]	30,720	11.72%	No (37.04 dB)
Our scheme	33,802	12.89%	Yes
	14,559	5.55%	No (39.35 dB)

Table 3.4. Same as Table 3.3 except that (largest) threshold value is set to five.

Scheme	Each shadow size (bytes)	Each shadow size over 512×512 (given image size)	Lossless reconstruction ability
[21]	52,429	20%	Yes
[26]	52,429	20%	Yes
[28]	74,898	28.57%	Yes
[29]	131,072	50%	Yes
[30]	30,720	11.72%	No (37.04 dB)
Our scheme	42,056	16.04	Yes
	18,964	7.23%	No (39.35 dB)

Table 3.5. Same as Table 3.1 except that the JPEG codes are created using $QF=65$.

Cover image	bpp of the (no-hiding) JPEG-Q65 code	bpp after hiding a shadow of size 42,056 bytes in the JPEG-Q65 code	<i>bpp of the (no-hiding) JPEG-Q95 code</i>
Peppers	0.76	2.40	2.52
Jet	0.77	2.45	2.47
Boat	0.83	2.52	2.74
Lake	1.02	2.65	3.44
Baboon	1.57	3.16	4.09
Zelda	0.65	2.30	2.54

Chapter 4

Authentication and Cross Recovery of Multiple JPEG Images

This chapter proposes a system with authentication and cross-recovery ability to protect a group of n JPEG images. Given n images, they are scaled down and then encoded using JPEG. The n generated JPEG codes, which are the n recovery data, are shared, according to a pre-determined threshold k ($2 \leq k < n$), using Reed-Solomon $(n-1, k)$ coding to generate n shadows. To identify malicious attacks on the system, the n authentication data are generated and then embedded together with the n shadows in the n JPEG codes of the n given images. The n JPEG stego codes are therefore generated, and they can be stored in a distributed storage system. In the daily maintenance of the storage system, the hidden authentication data in the n JPEG stego codes are used to verify which ones are attacked, and if some (up to $n-k$) of the n JPEG stego codes are corrupted, the corrupted JPEG images are approximately recovered using any k survived stego codes. Experimental results demonstrate the effectiveness of the proposed scheme. Comparisons with other image recovery methods are also made. The rest of this chapter is organized as follows. Section 4.1 reviews Reed-Solomon codes. Section 4.2 presents the proposed scheme. Section 4.3 reports the experimental results and makes comparisons with other methods. Last, Section 4.4 summarizes this chapter.

4.1 Reed-Solomon Codes

Reed-Solomon (RS) codes [58] can be used as error correction and erasure

correction codes. An RS(n, k) erasure code encodes k data symbols to n code symbols, and any k of the n code symbols can reconstruct the original k data symbols. Specifically, given the k data symbols b_0 to b_{k-1} of t bits each, where b_0 to b_{k-1} are all in GF(2^t), the n code symbols c_0 to c_{n-1} are computed as

$$[c_0 \ c_1 \ \dots \ c_{n-1}] = [b_0 \ b_1 \ \dots \ b_{k-1}] \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & n \\ 1^2 & 2^2 & 3^2 & \dots & n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1^{k-1} & 2^{k-1} & 3^{k-1} & \dots & n^{k-1} \end{bmatrix} \quad [\text{over GF}(2^t)]. \quad (4.1)$$

When any k of the n code symbols are collected, and assume without loss of generality that the first k code symbols c_0 to c_{k-1} are collected, the original k data symbols b_0 to b_{k-1} are reconstructed using

$$[b_0 \ b_1 \ \dots \ b_{k-1}] = [c_0 \ c_1 \ \dots \ c_{k-1}] \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & k \\ 1^2 & 2^2 & 3^2 & \dots & k^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1^{k-1} & 2^{k-1} & 3^{k-1} & \dots & k^{k-1} \end{bmatrix}^{-1} \quad [\text{over GF}(2^t)]. \quad (4.2)$$

4.2 Proposed method

4.2.1 Encoding

The goal of the proposed method is to protect a group of n JPEG images: when some (up to $n-k$) of the n JPEG images are destroyed or lost, the destroyed or lost JPEG images can be approximately recovered through the cooperation of any k survived members. Given n images, they will be compressed using JPEG to hide authentication and recovery data. Therefore, the hiding space is quite limited. To reduce the amount of n recovery data, the n images are scaled down by n factors, respectively. The n down-scaled images are encoded using JPEG, and the n generated JPEG codes are combined with their respective factors to form the n recovery data.

To offer the missing-allowable ability that allows the $n-k$ JPEG images to be completely lost in the recovery phase, each of the n recovery data is shared using RS($n-1, k$) erasure coding. In the sharing process, every t bits of each recovery data are sequentially grouped into a sharing coefficient, and every k generated sharing coefficients form a sector. Then, the k sharing coefficients a_1 to a_{k-1} of the first sector are used to compute the $n-1$ values w_0 to w_{n-2} using

$$[w_0 \ w_1 \ \dots \ w_{n-2}] = [a_0 \ a_1 \ \dots \ a_{k-1}] \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & n-1 \\ 1^2 & 2^2 & 3^2 & \dots & (n-1)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1^{k-1} & 2^{k-1} & 3^{k-1} & \dots & (n-1)^{k-1} \end{bmatrix} \text{ [over GF}(2^t)\text{]}. \quad (4.3)$$

The $n-1$ values w_0 to w_{n-2} are assigned to $n-1$ shares. By sequentially processing all sectors of each recovery data, the $n-1$ shares of which are generated. Let the $n-1$ generated shares of the first recovery data be $S_{12}, S_{13}, S_{14}, \dots$, and S_{1n} ; let the $n-1$ generated shares of the second recovery data be $S_{21}, S_{23}, S_{24}, \dots$, and S_{2n} ; let the $n-1$ generated shares of the third recovery data be $S_{31}, S_{32}, S_{34}, \dots$, and S_{3n} ; and so on. Then, for each $x=1,2,\dots,n$, the $n-1$ shares S_{wx} ($w=1,2,\dots,n$ and $w \neq x$) are combined to form the x 'th shadow.

After the n shadows have been generated, they are embedded in the n JPEG codes of the n original images, respectively, using the JPEG data hiding method in Chapter 2. To later check whether the n to-be-generated JPEG stego codes are corrupted, the authentication data of each of the n modified JPEG codes (with hidden shadows) are computed using the SHA-256 hash function [59]. Then, the JPEG data hiding method in Chapter 2 is used again to hide the n authentication data in the n modified JPEG codes, respectively, to generate the n JPEG stego codes. Notably, the purposes of generating the JPEG codes of the down-scaled images and generating the JPEG codes of the original images are quite different. The JPEG codes of the

down-scaled images are used to recover the tampered JPEG images while the JPEG codes of the original images are utilized to embed the recovery and authentication data. The encoding algorithm is summarized here:

The encoding algorithm

Input: A group of n images; n factors; the positive integer parameter k ($2 \leq k < n$).

Output: The n JPEG stego codes.

Step 1: Scale down the n images by the n factors, respectively, to create n down-scaled images. Encode the n down-scaled images using JPEG to generate n JPEG codes. Combine the n JPEG codes with their respective factors to form n recovery data.

Step 2: Use Eq. (4.3) to split each of the n recovery data into $n-1$ shares. Let the $n-1$ created shares of the first recovery data be $S_{12}, S_{13}, S_{14}, \dots$, and S_{1n} ; let the $n-1$ created shares of the second recovery data be $S_{21}, S_{23}, S_{24}, \dots$, and S_{2n} ; let the $n-1$ created shares of the third recovery data be $S_{31}, S_{32}, S_{34}, \dots$, and S_{3n} ; and so on.

Step 3: For each $x=1,2,\dots,n$, combine the $n-1$ shares S_{wx} ($w=1,2,\dots,n$ and $w \neq x$) to form the x 'th shadow.

Step 4: Encode the n original images using JPEG to generate n JPEG codes. Modify the n JPEG codes to hide the n shadows using the JPEG data hiding method in Chapter 2.

Step 5: For each of the n modified JPEG codes (with hidden shadows), apply the SHA-256 hash function to generate a 256-bit authentication data. Hide the n authentication data in the n modified JPEG codes, respectively, using the JPEG data hiding method in Chapter 2, and generate the n JPEG stego codes.

4.2.2 Decoding

The decoding process consists of the verification and recovery phases. In the verification phase, to determine whether each of the n generated JPEG stego codes is authentic, the authentication data are extracted from the JPEG stego code (the hidden shadow is left untouched), and a by-product JPEG code is also obtained since the JPEG data hiding method used in the proposed method is reversible. The authentication data of the by-product JPEG code are then computed using the SHA-256 hash function. If the extracted authentication data is the same as the computed one, the JPEG stego code is authentic; otherwise, it is non-authentic.

After all n JPEG stego codes have been verified, if at least one JPEG stego code is non-authentic and there exist at least k authentic JPEG stego codes, then the recovery phase starts. The recovery of $n-k$ non-authentic JPEG images through the remaining k authentic JPEG stego codes is described in the following. Without loss of generality, assume that the first k JPEG stego codes are authentic. The k shadows are extracted from the first k JPEG stego codes, respectively, by inverse hiding. For each $y=1,2,\dots,k$, the y 'th of the k shadows is partitioned to yield the $n-1$ shares S_{wy} ($w=1,2,\dots,n$ and $w \neq y$). For each $z=k+1,k+2,\dots,n$, the k shares $S_{z1}, S_{z2}, \dots,$ and S_{zk} are employed to reconstruct the z 'th recovery data by inverse sharing.

The inverse-sharing process takes the first k not-yet-processed pixels of t bits each from each of the obtained k shares, and uses these k pixel values w_0 to w_{k-1} to solve for the k coefficients a_0 to a_{k-1} in Eq. (4.3) for the first sector, as

$$[a_0 \ a_1 \ \dots \ a_{k-1}] = [w_0 \ w_1 \ \dots \ w_{k-1}] \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & k \\ 1^2 & 2^2 & 3^2 & \dots & k^2 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1^{k-1} & 2^{k-1} & 3^{k-1} & \dots & k^{k-1} \end{bmatrix}^{-1} \quad [\text{over GF}(2^t)]. \quad (4.4)$$

By processing all pixels of the k shares in order, the z 'th recovery data are

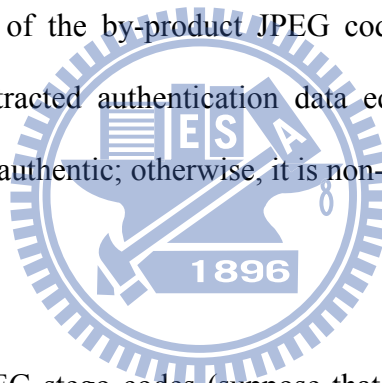
reconstructed. Last, since the z 'th recovery data consist of a JPEG code and a factor, the JPEG code is decompressed and then scaled up by the factor to create the up-scaled image, which is the approximate version of the z 'th non-authentic JPEG image. The verification and recovery algorithms are summarized here:

The verification algorithm

Input: The n JPEG stego codes.

Output: The verification results of the n JPEG stego codes.

Step 1: For each JPEG stego code, extract the authentication data from the JPEG stego code (a by-product JPEG code is also obtained). Compute the authentication data of the by-product JPEG code using the SHA-256 hash function. If the extracted authentication data equal the computed one, the JPEG stego code is authentic; otherwise, it is non-authentic.



The recovery algorithm

Input: The k authentic JPEG stego codes (suppose that there are $n-k$ non-authentic JPEG stego codes, and assume without loss of generality that the first k JPEG stego codes are authentic).

Output: The $n-k$ recovered JPEG images.

Step 1: Extract the k shadows from the first k JPEG stego codes, respectively.

Step 2: For each $y=1,2,\dots,k$, partition the y 'th of the k shadows to yield the $n-1$ shares S_{wy} ($w=1,2,\dots,n$ and $w \neq y$).

Step 3: For each $z=k+1,k+2,\dots,n$, use Eq. (4.4) to reconstruct the z 'th recovery data from the k shares S_{z1}, S_{z2}, \dots , and S_{zk} . Then, use the z 'th recovery data including a JPEG code and a factor to recover the z 'th non-authentic JPEG image approximately.

4.3 Experiments and Comparisons

4.3.1 Experimental Results

The fourth public release of the Independent JPEG Group's free software [43] is used in the experiments. The PSNR is used to measure the quality of an image. In the first experiment, a group of four 512×512 grayscale images Baboon, Scene, Lena, and Jet, shown in Figure 4.1, are all scaled down by a factor of two in each dimension, and the four down-scaled images are encoded using JPEG with $QF=85$. After the four recovery data have been generated, they are shared using RS(3, 2) erasure coding [over $GF(2^8=256)$] to generate four shadows. The four shadows and four authentication data are embedded in the four JPEG codes, created by encoding the four images in Figure 4.1 using JPEG with $QF=85$, to generate four JPEG stego codes. Figure 4.2 displays the four images decompressed from the four JPEG stego codes without any extraction of the hidden authentication and recovery data, and the PSNRs of the four decompressed images are 35.70, 36.27, 40.52, and 40.04 dB, respectively. Visual quality of these decompressed images is acceptable.

In fact, each of these $n=4$ JPEG stego codes can work alone to obtain its own decompressed image, but recovering the lost $n-k=2$ JPEG images will require the cooperation of any $k=2$ survived JPEG stego codes. Figures 4.3(a)-(b) show the recovered JPEG images Baboon' (PSNR=24.87 dB) and Lena' (PSNR=34.89 dB) by using the two survived JPEG stego codes, the decompressed images of which are in Figures 4.2(b) and 4.2(d). Similarly, Figures 4.3(c)-(d) display the recovered JPEG images Scene' (PSNR=30.24 dB) and Jet' (PSNR=31.82 dB) by using the two survived stego codes, the decompressed images of which are in Figures 4.2(a) and 4.2(c).

In the second experiment, another group of four 512×512 grayscale images

Peppers, Boat, Tiffany, and House, displayed in Figure 4.4, are tested. The four JPEG stego codes are generated likewise. Figure 4.5 depicts the four images decompressed from these four JPEG stego codes, and the quality of the decompressed images is visually acceptable. When any two of the four JPEG stego codes are lost, the lost JPEG images can be approximately recovered using the remaining two members, and the recovered JPEG images are always identical to two of the four images plotted in Figure 4.6, the PSNRs of which are 34.64, 30.79, 31.91, and 32.51 dB, respectively.

4.3.2 Comparisons

Table 4.1 compares several image recovery methods [35-39] with ours in terms of authentication ability and recovery ability. All of the associated works [35-39] and ours have authentication and recovery ability. However, four of the related works [35-38] focus on the recovery of a single image. Once the protected image in these self-recovery methods is seriously damaged, it is difficult for their methods to recover the seriously damaged image. In contrast, Chang et al.'s cross-recovery method [39] and ours can handle this situation: the damaged image can be approximately recovered as long as k members are authentic.

The comparisons between Chang et al.'s cross-recovery method [39] and ours are given in the following. Both Chang et al.'s method and the proposed method have the missing-allowable ability that allows $n-k$ members to be lost. However, Chang et al. use a two-layer sharing that is based on Thien and Lin's method [21] to share their n recovery data. In the first-layer sharing, their n recovery data are shared to create $n-k$ temporary shadows; in the second-layer sharing, the created $n-k$ temporary shadows are shared again and then combined to form their n shadows. To the contrary, our sharing process based on RS erasure coding is one layer, and therefore simple.

Moreover, Table 4.2 compares the shadow sizes between Chang et al.'s method

and ours when the group of four 512×512 grayscale images are Baboon, Scene, Lena, and Jet. Each of the four shadows in our method is smaller than that in Chang et al.’s method no matter the threshold k is set to two or three. Similarly, Table 4.3 compares the shadow sizes between Chang et al.’s method and ours when the group of four 512×512 grayscale images are Peppers, Boat, Tiffany, and House. Each shadow in our method is still smaller than those in Chang et al.’s method no matter the threshold k is set to two or three. Therefore, the n shadows in our method can be hidden successfully in the JPEG codes of the n images to reduce storage space. Notably, if the n protected images in Chang et al.’s method are compressed using JPEG to save storage space, then their cross-recovery ability is gone.

4.4 Summary

This chapter proposes an image authentication method with cross-recovery ability for protecting a group of n JPEG images. The proposed method has the missing-allowable ability that allows $n-k$ JPEG stego codes to be completely lost, and the lost JPEG images can be approximately recovered using the remaining k JPEG stego codes. The proposed method owns the following benefits. (1) The n JPEG stego codes in the proposed method require less storage space than the n stego-images in Chang et al.’s work [39]; yet the quality of the recovered images is maintained. (2) The one-layer sharing (or inverse-sharing) in the proposed method is simpler than the two-layer sharing (or inverse-sharing) in Chang et al.’s work [39]. (3) The proposed method is useful for the images stored and transmitted in JPEG format.

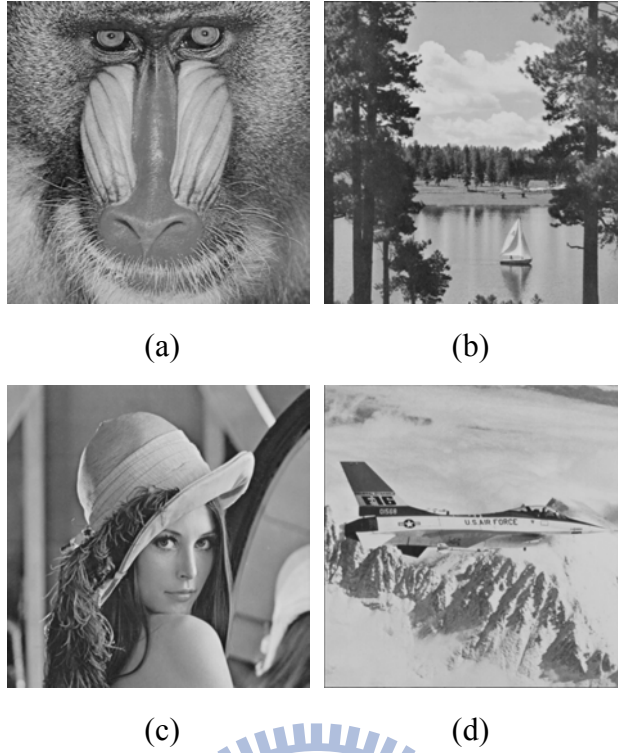


Figure 4.1. A group of four images Baboon, Scene, Lena, and Jet.



Figure 4.2. Four images decompressed from the four JPEG stego codes in the first experiment without any extraction of the hidden authentication and recovery data.

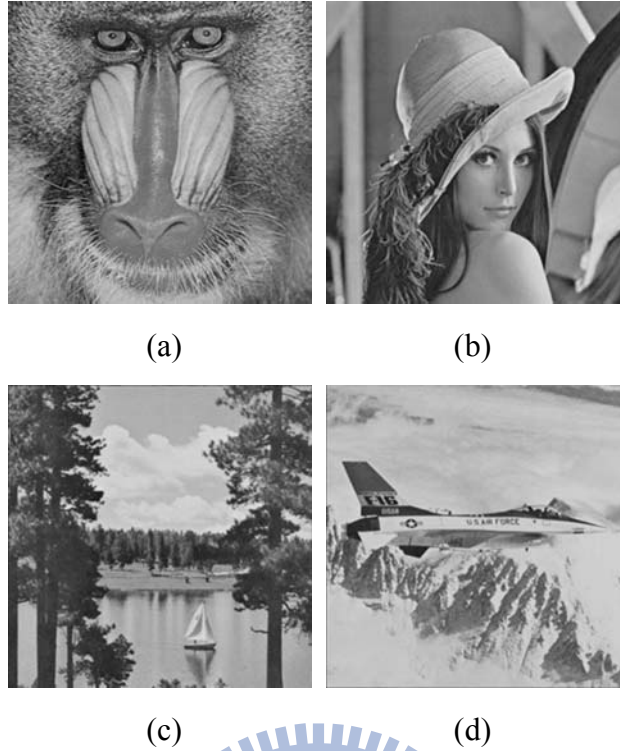


Figure 4.3. Two JPEG stego codes in the first experiment are lost. (a)-(b) The recovered JPEG images Baboon' (PSNR=24.87 dB) and Lena' (PSNR=34.89 dB) by using two survived JPEG stego codes, the decompressed images of which are in Figures 4.2(b) and 4.2(d). (c)-(d) The recovered JPEG images Scene' (PSNR=30.24 dB) and Jet' (PSNR=31.82 dB) by using two survived JPEG stego codes, the decompressed images of which are in Figures 4.2(a) and 4.2(c).



(a)



(b)



(c)



(d)

Figure 4.4. Another group of four images Peppers, Boat, Tiffany, and House.



(a)



(b)



(c)



(d)

Figure 4.5. Four images decompressed from the four JPEG stego codes in the second experiment without any extraction of the hidden authentication and recovery data.

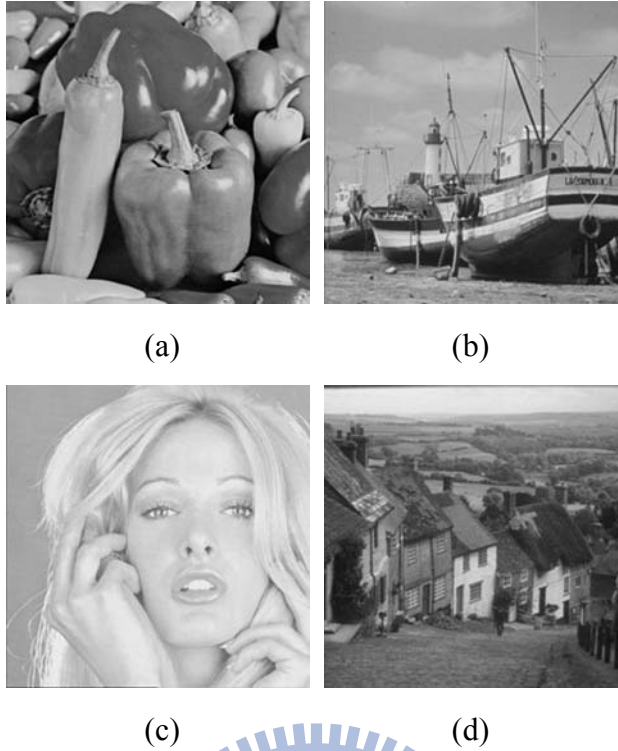


Figure 4.6. Two JPEG stego codes in the second experiment are lost. Any two of the four JPEG stego codes, the decompressed images of which are in Figure 4.5, are lost, and the recovered JPEG images are identical to two of the four images, which are (a) Peppers' (PSNR=34.64 dB), (b) Boat' (PSNR=30.79 dB), (c) Tiffany' (PSNR=31.91 dB), and (d) House' (PSNR=32.51 dB).

Table 4.1. Comparisons among image recovery schemes [35-39] and ours.

Scheme	Authentication ability	Recovery ability
[35]	Yes	Yes (single image)
[36]	Yes	Yes (single image)
[37]	Yes	Yes (single image)
[38]	Yes	Yes (single image)
[39]	Yes	Yes (multiple images)
Our scheme	Yes	Yes (multiple JPEG images)

Table 4.2. Comparison of shadow sizes between Chang et al.'s method [39] and ours when the group of four grayscale images are Baboon, Scene, Lena, and Jet.

Scheme	Threshold k	The size (bytes) of each of the four shadows			
		Shadow 1	Shadow 2	Shadow 3	Shadow 4
[39]	2	65,457	65,457	65,457	65,457
Our scheme	2	22,374	24,741	26,445	26,442
[39]	3	21,819	21,819	21,819	21,819
Our scheme	3	14,917	16,495	17,631	17,629

Table 4.3. Same as Table 4.2 except that the four grayscale images are replaced by Peppers, Boat, Tiffany, and House.

Scheme	Threshold k	The size (bytes) of each of the four shadows			
		Shadow 1	Shadow 2	Shadow 3	Shadow 4
[39]	2	65,003	65,003	65,003	65,003
Our scheme	2	20,660	20,425	21,919	19,781
[39]	3	21,668	21,668	21,668	21,668
Our scheme	3	13,774	13,617	14,613	13,187

Chapter 5

Conclusions and Future Works

5.1 Conclusions

We propose three methods for data hiding, image sharing, and image authentication with cross recovery based on JPEG. The proposed methods include a reversible JPEG data hiding method with high hiding capacity in Chapter 2, a multi-threshold progressive image sharing method with compact shadows in Chapter 3, and an image authentication method with cross-recovery ability in Chapter 4.

In Chapter 2, when a large-size secret is transmitted to the Internet, the secret is embedded in a JPEG code to reduce the probability of being attacked. In collecting the generated JPEG stego code, the secret is completely extracted and the JPEG code is reconstructed. The proposed method owns the following advantages: (1) the hiding capacity, hiding ratio, and stego-image quality in the proposed method are better than those in the related works [12,15,17,18]; (2) the proposed method can resist Chi-square and StegDetect attacks, reducing the probability that the attackers would notice the existence of the hidden data in our JPEG stego codes; (3) the lengths of our JPEG stego codes are smaller than that of the JPEG code generated using $QF=95$ without any hidden secret, and therefore the attackers will not be suspicious about the lengths of our JPEG stego codes; (4) the reversibility makes the reconstructed JPEG code to be reused many times to hide a new secret.

In Chapter 3, the transmission or distributed storage of an important image uses n shadows, and the important image with various quality levels are reconstructed based on the number of the collected shadows. The proposed method owns the following advantages: (1) each shadow in the proposed method is more economic than those in

the related works [21,25-29], and each shadow in our lossy approach is much smaller than that in Hung et al.'s lossy approach [30]. This improvement keeps storage space and transmission time low, and facilitates the hiding of shadows in stego media; (2) the proposed method is easier than the progressive image sharing schemes [27-30] to be used in developing a scalable MPEG video transmission system; (3) the security of the shadows is enhanced using XOR and circular shift operations, and the XOR-enhanced version can resist differential attack. Also, each shadow in the proposed method is very compact and so can be hidden in the JPEG codes using the JPEG data hiding method in Chapter 2 to resist Chi-square and StegSpy attacks.

In Chapter 4, given n images, both of their n authentication and recovery data are embedded in the n JPEG codes of the n images to generate n JPEG stego codes. When some (up to $n-k$) of the n JPEG stego codes are lost, the lost JPEG images can be approximately recovered by the support of the k survived JPEG stego codes. The proposed method owns the following advantages: (1) the n JPEG stego codes require less storage space than the n stego-images in Chang et al.'s method [39]; (2) the one-layer sharing in the proposed method is simpler than the two-layer sharing in Chang et al.'s method [39].

5.2 Future Works

Some suggestions for extending the proposed methods in the future are listed in the following. (1) JPEG and Graphics Interchange Format (GIF) are two most widely used image formats on the web. Therefore, we can try to design a high-capacity GIF data hiding method together with its applications such as progressive image sharing or image authentication or image recovery. (2) In Chapter 3 and Chapter 4, the sharing of JPEG codes is based on polynomial or matrix computations. The polynomial-based or

matrix-based sharing method is generally good in obtaining small-size shadows, but not fast. Therefore, we can try to design a Boolean-based sharing method to accelerate the sharing speed.



References

- [1] D. C. Wu and W. H. Tsai, "Spatial-domain image hiding using image differencing," *IEE Proceedings-Vision, Image and Signal Processing*, Vol. 147, No. 1, pp. 29-37, 2000.
- [2] 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, 2001.
- [3] 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. 13, pp. 2875-2881, 2003.
- [4] D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recognition Letters*, Vol. 24, pp. 1613-1626, 2003.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] C. Y. Yang and J. C. Lin, "Image hiding by base-oriented algorithm," *Optical Engineering*, Vol. 45, No. 11, pp. 117001.1-117001.10, 2006.

- [9] 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.
- [10] A. Latham, JPHide hiding tool, available at <http://linux01.gwdg.de/~alatham/stego.html>, 1999.
- [11] N. Provos, OutGuess hiding tool and StegDetect steganography test program, available at <http://www.outguess.org/download.php>.
- [12] D. Upham, JSteg hiding tool, available at <http://packetstormsecurity.org/crypt/stego/DOS/jsteg.zip>.
- [13] H. Kobayashi, Y. Noguchi, and H. Kiya, "A method of embedding binary data into JPEG bitstreams," *IEICE Transaction on Information and Systems*, Vol. J83-D-2, No. 6, pp. 1469-1476, 2000.
- [14] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication watermark for JPEG images," *Proceedings of the IEEE International Conference on Information Technology*, Las Vegas, Nevada, USA, pp. 223-227, 2001.
- [15] 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.
- [16] M. Iwata, K. Miyake, and A. Shiozaki, "Digital steganography utilizing features of JPEG images," *IEICE Transaction on Fundamentals*, Vol. E87-A, No. 4, pp. 929-936, 2004.
- [17] H. W. Tseng and C. C. Chang, "High capacity data hiding in JPEG-compressed images," *Informatica*, Vol. 15, No. 1, pp. 127-142, 2004.
- [18] C. C. Chang, C. C. Lin, C. S. Tseng, and W. L. Tai, "Reversible hiding in DCT-based compressed images," *Information Sciences*, Vol. 177, No. 14, pp. 2768-2786, 2007.

- [19] X. Li and J. Wang “A steganographic method based upon JPEG and particle swarm optimization algorithm,” *Information Sciences*, Vol. 177, No. 15, pp. 3099-3109, 2007.
- [20] C. L. Liu and S. R. Liao “High-performance JPEG steganography using complementary embedding,” *Pattern Recognition*, Vol. 41, No. 9, pp. 2945-2955, 2008.
- [21] C. C. Thien and J. C. Lin, “Secret image sharing,” *Computers and Graphics*, Vol. 26, No. 5, pp. 765-770, 2002.
- [22] 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. 12, No. 13, pp. 1161-1169, 2003.
- [23] C. C. Lin and W. H. Tsai, “Visual cryptography for gray-level images by dithering techniques,” *Pattern Recognition Letter*, Vol. 24, No. 1-3, pp. 349-358, 2003.
- [24] 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, 2005.
- [25] 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.
- [26] H. K. Tso, “Sharing secret images using Blakley’s concept,” *Optical Engineering*, Vol. 47, No. 7, pp. 077001.1-077001.3, 2008.
- [27] W. P. Fang, “Friendly progressive visual secret sharing,” *Pattern Recognition*, Vol. 41, No. 4, pp. 1410-1414, 2008.
- [28] S. K. Chen and J. C. Lin, “Fault-tolerant and progressive transmission of images,” *Pattern Recognition*, Vol. 38, No. 12, pp. 2466-2471, 2005.

- [29] R. Z. Wang and S. J. Shyu, "Scalable secret image sharing," *Signal Processing: Image Communication*, Vol. 22, No. 4, pp. 363-373, 2007.
- [30] K. H. Hung, Y. J. Chang, and J. C. Lin, "Progressive sharing of an image," *Optical Engineering*, Vol. 47, No. 4, p. 047006.1-047006.14, 2008.
- [31] C. S. Lu and H. Y. M. Liao, "Multipurpose watermarking for image authentication and protection," *IEEE Transactions on Image Processing*, Vol. 10, No. 10, pp. 1579-1592, 2001.
- [32] 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.
- [33] C. H. Tzeng and W. H. Tsai, "A new approach to authentication of binary images for multimedia communication with distortion reduction and security enhancement," *IEEE Communication Letter*, Vol. 7, No. 9, pp. 443-445, 2003.
- [34] C. C. Chang and C. S. Lin, "A GA-based nearly optimal image authentication approach," *International Journal of Innovative Computing, Information and Control*, Vol. 3, No. 3, pp. 631-640, 2007.
- [35] A. Piva, F. Bartolini, and R. Caldelli, "Self recovery authentication of images in the DWT domain," *International Journal of Image and Graphic*, Vol. 5, No. 1, pp. 149-165, 2005.
- [36] R. Chamlawi, A. Khan, and A. Idris, "Wavelet based image authentication and recovery," *Journal of Computer Science and Technology*, Vol. 22, No. 6, pp. 795-804, 2007.
- [37] Y. J. Chang, R. Z. Wang, and J. C. Lin, "A sharing-based fragile watermarking method for authentication and self-recovery of image tampering," *EURASIP Journal on Advances in Signal Processing*, Vol. 2008, No. 200, pp. 1-17, 2008.
- [38] C. C. Chang and H. A. Ke, "Image Authentication under DCT domain with

- attack recovery,” accepted by *International Journal of Innovative Computing, Information and Control*, Vol. 6, No. 3(A), pp. 885-896, 2010.
- [39] Y. J. Chang, S. J. Lin, and J. C. Lin, “Authentication and cross-recovery for multiple images,” *Journal of Electronic Imaging*, Vol. 17, No. 4, pp. 043007.1-043007.12, 2008.
- [40] G. R. Blakley, “Safeguarding cryptographic keys,” *Proceedings of the National Computer Conference*, New Jersey, New York, USA, pp. 313-317, 1979.
- [41] A. Shamir, “How to share a secret,” *Communication of the ACM*, Vol. 22, No. 11, pp. 612-613, 1979.
- [42] R. L. Rivest, “The MD5 message digest algorithm,” *Technique Report of MIT Laboratory for Computer Science and RSA Data Security*, 1992.
- [43] Fourth public release of the Independent JPEG Group's free JPEG software, available at <http://packetstormsecurity.nl/crypt/stego/jpeg-steg/jpeg-v4.tar.gz>.
- [44] S. Guillermito, Chi-square steganography test program, available at <http://www.guillermito2.net/stegano/tools/index.html>, 2004.
- [45] N. Provos and P. Honeyman, “Hide and seek: an introduction to steganography,” *IEEE Security and Privacy*, Vol. 1, No. 3, pp. 32-44, 2003.
- [46] F. Goel, M. Garuba, C. Liu, and T. Nguyen, “The security threat posed by steganographic content on the Internet,” *Proceedings of the IEEE International Conference on Information Technology*, Las Vegas, Nevada, USA, pp. 794-798, 2007.
- [47] G. C. Kessler, “An overview of steganography for the computer forensics examiner,” *Forensic Science Communications*, Vol. 6, No. 13, pp. 45-48, 2004.
- [48] H. Malik, R. Chandramouli, and K. P. Subbalakshmi, “Steganalysis: Trends and Challenges”, in *Multimedia Forensics and Security*, C. T. Li, Ed., *Idea Group Publishing*, 2008.

- [49] U. Horn and B. Girod, "Scalable video transmission for the Internet," *Computer Networks and ISDN Systems*, Vol. 29, No. 15, pp. 1518-1529, 1997.
- [50] T. Schierl, T. Stockhammer, T. Wiegand, "MPEG video transmission using scalable video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 9, pp. 1204-1217, 2007.
- [51] N. K. Pareek, V. Patidar, and K. K. Sud, "Image encryption using chaotic logistic map," *Image and Vision Computing*, Vol. 24, No. 9, pp. 926-934, 2006.
- [52] H. E. H. Ahmed, H. M. Kalash, and O. S. F. Allah, "An efficient chaos-based feedback stream cipher (ECBFSC) for image encryption and decryption," *Informatica*, Vol. 4, No. 31, pp. 121-129, 2007.
- [53] H. S. Kwok and W. K. S. Tang, "A fast image encryption system based on chaotic maps with finite precision representation," *Chaos, Solitons and Fractals*, Vol. 32, No. 4, pp. 1518-1529, 2007.
- [54] J. Kim, Y. Byun, and J. Choi, "Image forensics technology for digital camera," in *Pacific-Rim Conference on Multimedia*, K. Alzawa, Y. Nakamura, and S. Satoh, Eds., *Lecture Notes in Computer Science* 3333, pp. 331-339, 2004.
- [55] M. T. Raggio, StegSpy steganography test program, available at <http://www.spy-hunter.com/stegspydownload.htm>.
- [56] Hiderman hiding tool, available at <http://www.softsea.com/review/Hiderman.html>.
- [57] JPegX hiding tool, available at <http://www.globalfreeware.com/Jpegx-211.html>.
- [58] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 8, No.2, pp. 300-304, 1960.
- [59] National Institute of Standards and Technology, *Federal Information Processing Standard (FIPS) Publication 180-2, Secure Hash Standard (SHS)*, 2004.

Vita

LEE SHU-TENG CHEN was born in Taipei, Taiwan, in 1976. He received his BS degree in computer science from National Chiao Tung University, Taiwan, in 1999, and MS degree in computer science and information engineering from National Taiwan University, Taiwan, in 2001. He is currently a PhD candidate in the Department of Computer Science and Information Engineering at National Chiao Tung University. His current research interests include data hiding and image sharing.



Publication List of Lee Shu-Teng Chen

A. Journal papers

Published:

1. L. S. T. Chen, S. J. Lin, and J. C. Lin, "Reversible JPEG-based hiding method with high hiding-ratio", *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 24, No. 3, pp. 1-23, 2010. (SCI)
2. L. S. T. Chen and J. C. Lin, "Steganography scheme based on side match vector quantization", *Optical Engineering*, Vol. 49, No. 3, pp. 037008.1-037008.7, 2010. (SCI)
3. L. S. T. Chen and J. C. Lin, "Multithreshold progressive image sharing with compact shadows", *Journal of Electronic Imaging*, Vol. 19, No. 1, pp. 013003.1-013003.12, 2010. (SCI)
4. S. J. Lin, L. S. T. Chen, and J. C. Lin, "Fast-weighted secret image sharing", *Optical Engineering*, Vol. 48, No. 7, pp. 077008.1-077008.7, 2009. (SCI)
5. L. S. T. Chen, W. K. Su, and J. C. Lin, "Secret image sharing based on vector quantization", *International Journal of Circuits, Systems and Signal Processing*, Vol. 3, No. 3, pp. 137-144, 2009.
6. W. K. Su, L. S. T. Chen, S. K. Chen, and J. C. Lin, "Secret image recovery based on search order coding", *International Journal of Computers*, Vol. 3, No. 3, pp. 321-328, 2009.

Revised:

7. L. S. T. Chen, S. K. Chen, and J. C. Lin, "Authentication and cross-recovery of multiple JPEG images", *International Journal of Innovative Computing, Information and Control*, revised. (SCI)

B. Conference papers

Accepted or Published:

1. L. S. T. Chen, S. S. Lee, W. T. Chang, and J. C. Lin, "Data hiding based on side match vector quantization and modulus function", *Proceeding of the IEEE International Conference on Intelligence Information and Multimedia Signal Processing*, accepted.
2. W. K. Su, L. S. T. Chen, and J. C. Lin, "Fault-tolerant VQ-style secret image sharing", *Proceeding of the WSEAS International Conference on Applied Computer and Applied Computational Science*, pp. 40-43, Hangzhou, China, May 2009.
3. W. K. Su, L. S. T. Chen, and J. C. Lin, "Error correction of secret images by search order coding", *Proceeding of the WSEAS International Conference on Multimedia Systems and Signal Processing*, pp. 49-52, Hangzhou, China, May 2009.

