

國立交通大學

資訊學院

資訊工程研究所

博士論文

新型態的影像分享技術

New Types of Image Sharing Techniques



研究生：方文聘

指導教授：林志青 教授

中華民國 九十五年 七月

新型態的影像分享技術

New Types of Image Sharing Techniques

研究生：方文聘

Student：Wen-Pinn Fang

指導教授：林志青 教授

Advisor：Ja-Chen Lin

國立交通大學

資訊工程學系

博士論文



Submitted to Department of Computer and Information Science

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

新型態的影像分享技術

學生：方文聘

指導教授：林志青博士

國立交通大學資訊工程學系(研究所)博士班

摘 要

本論文提出四種新型的影像分享方法。在翻轉型影像分享，我們產生出兩張投影片。直接將這兩張投影片相疊會看到一張機密影像；而如果將其中一張翻轉後再相疊，則可以看到另一機密影像。在兩層型影像分享，我們也產生出兩張投影片。如果直接對齊相疊，則可以看到機密影像，如果將某一張偏移 to 特定位置後再將兩張投影片相疊，則可以看到驗證用的更進一層資訊。在通用型影像分享，我們設計出一張由公司召集人持有的特殊分存。此特殊分存可以通用於任意多張機密影像之分享，召集人可持此特殊分存參加任一機密影像之解碼會議。這可以解決公司有太多機密影像要分享時，召集人手上的分存數量日益增加的問題。最後，在快速解碼型影像分享，我們用位元平面的分解法去分享灰階影像。在影像的解碼會議上，解碼可快速達成。

New Types of Image Sharing Techniques

Student: Wen-Pinn Fang

Advisor: Dr. Ja-Chen Lin

Institute of Computer Science

College of Computer Science

National Chiao Tung University

ABSTRACT



This dissertation proposes four new types of image sharing: turnover, two-level, universalizing, and fast decoding. In the turnover type, for any two given secret images, two corresponding transparencies are produced. Both transparencies look noisy. If we stack the front view of both transparencies, then we can see the first secret image. On the other hand, if we stack the front view of Transparency 1 with the back view (the turnover) of Transparency 2, then the second secret image is unveiled.

In the two-level type, we present a two-in-one visual cryptography scheme, which not only shares an image of moderate confidentiality between two noisy transparencies, but also hides in these two transparencies a more confidential text file, which is either the information for authentication purpose or the information describing the image. More specifically, if we stack the two transparencies without any shift, then we can see the secret image. On the other hand, if we shift Transparency 1 to a predefined amount before stacking with Transparency 2, then we can see some other information of more confidential level in the shifted stacking.

In the third type, we design a so-called universal share. A company's organizer can hold this special share to attend any unveiling meeting of any secret image shared in his company. No matter how many secret images are shared in his company, the organizer only have to hold this special share, rather than thousands or millions of shares.

Finally, in the forth type, we use bit-plane decomposition to design a scheme to share gray-value secret images. The decoding speed is fast. The method is particularly useful if the threshold r in the (r,n) sharing scheme equals n . In this case, not only the decoding speed, but also the encoding speed, is very fast.



Acknowledgements

I would like to acknowledge and express gratitude to all those who gave me the possibility to complete this thesis.

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

Appreciations are also given to Dr. Chih-Ching Thein and Dr. Yu-Yang Liu for their helpful discussion and suggestions. Thanks are also given to all colleagues in the Computer Vision Laboratory at National Chiao Tung University for their assistance and helps: Dr. Shang-Kuan Chen, Yu-Jay Chang and Sen-Jen Lin.

Furthermore, the support and encouragement from my family were appreciated and greatly needed. I am very grateful for the love of my parents and my brothers. I would like to thank them whose ardent support has been indispensable to me.

Finally, I sincerely offer my thanks to my lovely wife Anne Lin for her love, encouragement and support. This thesis is dedicated her.

Table of Contents

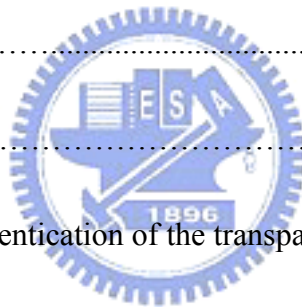
ABSTRACT IN CHINESE.....	i
ABSTRACT IN ENGLISH.....	ii
AACKNOWLEDGES.....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES.....	viii
Chapter 1 Introduction.....	1
1.1 Motivation.....	1
1.2 Notations and Background for the Related Topic	2
1.3 Overview of the proposed method.....	3
1.3.1 Visual Cryptography in Turnover Style	3
1.3.2 Visual Cryptography in Two-level Style: The VC with Extra Ability of Hiding Confidential Data.....	4
1.3.3 Universal share for the sharing of multiple images.....	4
1.3.4 Fast-decoding sharing of a gray-value image: by using bit-level sharing and economic size shares.....	5
1.4 Dissertation Organization.....	5
Chapter 2 Visual Cryptography in Turnover Style	6

2.1 The goal	7
2.2 Analysis	10
2.3 The design of shares (transparencies) S_1 and S_2	14
2.4 Experiment results.....	17
2.5 Perturbed version of the design.....	18
2.6 Summary.....	23

Chapter 3 Visual Cryptography in Two-level style: the VC with Extra Ability of

Hiding Confidential Data 28

3.1 The method.....	29
3.2 Experimental results	36
3.3 Application to the authentication of the transparencies	38
3.4 Techniques to improve the security.....	38
3.5 When a decoding-computer is not available	40
3.6 Summary.....	46



Chapter 4 Universal share for the sharing of multiple images..... 50

4.1 The method.....	51
4.2 Experimental results	54
4.3 Comparison.....	57
4.4 Summary.....	58

Chapter 5 Fast-decoding type sharing of a gray-value image: by using bit-level sharing and economic size shares	60
5.1 The method.....	61
5.2 Experimental results	76
5.3 Summary.....	80
Chapter 6 Conclusions and future works	82
6.1 Conclusions.....	82
6.2 Future works.....	84
REFERENCE	85
VITA	92
Publication List of Wen-Pinn Fang	93



List of Figures

Fig. 1.1 Some sharing blocks seen in Ref. [NS1995] (not used here in the current thesis).....3

Fig. 2.1 Visual Cryptography in turn-over style.....8

Fig. 2.2 The "turn-over" operation of a transparency. (a) is a transparency, and (b) is its turn-over version (the back view of the transparency in (a)).....10

Fig. 2.3 Two types of stacking.....13

Fig. 2.4 The pattern blocks $\{\alpha_1, \beta_1, \alpha_2, \beta_2\}$ used in our method to paint the two transparencies. Columns (α_1) and (β_1) are, respectively, the pattern blocks in position $\alpha=(x,y)$ and position $\beta=(256-x,y)$, for transparency S_1 . Analogously, columns (α_2) and (β_2) are for transparency S_2 ; (α_L) and (β_L) are for the stacked result "LENA"; (α_p) and (β_p) are for the stacked result "PEPPERS". Note that the stacked result are $\alpha_1 \oplus \alpha_2 = \alpha_L, \beta_1 \oplus \beta_2 = \beta_L, \alpha_1 \oplus \beta_2^{turn-over} = \alpha_p,$ and $\beta_1 \oplus \alpha_2^{turn-over} = \beta_p$ (to understand, see Fig. which corresponds to the last row (B,B,B,B)here).....16

Fig. 2.5 The experiment of the turn-over-style visual cryptography. (a) and (b) are the two original black-and-white images; (c) and (d) are the two generated transparencies S_1 and S_2 ; whereas (e) and (f) are the two recovered images ((e) is the result of stacking the pair $\{(c), (d)\}$, while (f) is the result of stacking the pair $\{(d)^{turn-over}, (c)\}$).....17

Fig.2.6. Three kinds of perturbation operation to get some other combinations of pattern blocks whose stacked results $(\alpha_1 \oplus \alpha_2 = \alpha_L, \beta_1 \oplus \beta_2 = \beta_L, \alpha_1 \oplus \beta_2^{turn-over} = \alpha_p, \beta_1 \oplus \alpha_2^{turn-over} = \beta_p)$ are all (W,W,W,W). (a) is the first row

(the (W,W,W,W) row) of Fig. (b) is obtained from (a) by switching α_1 with α_2 , then followed by switching β_1 with β_2 . (c) is obtained from (a) by permuting the 3 rows of α_1 , then permuting the 3 rows (using the same permutation order) for each of α_2 , β_1 and β_2 . (d) is obtained using a new design (not necessarily related to (a)).....21

Fig.2.7. The experiment (perturbed version). (a) and (b) are the two original images; (c) and (d) are the two generated transparencies S_1 and S_2 ; whereas (e) and (f) are the two recovered images ((e) is from stacking $\{(c),(d)\}$, while (f) is from stacking $\{(d)^{\text{turn-over}}, (c)\}$).....22

Fig.2.8. Another experiment for the perturbed version. (a) and (b) are the two original images; (c) and (d) are the two generated transparencies S_1 and S_2 ; whereas (e) and (f) are the two recovered images ((e) is from stacking $\{(c),(d)\}$, while (f) is from stacking $\{(d)^{\text{turn-over}}, (c)\}$).....23

Fig.3.1. The six types of fundamental blocks (Types 0-5).....30

Fig.3.2. The pairs whose stacking results represent white blocks.....32

Fig.3.3 The pairs whose stacking results represent black blocks.....32

Fig.3.4 Hiding the confidential text file (see Steps 1 and 2 of the encoding algorithm).....35

Fig.3.5 Sharing the secret image Lena (see Step 3 of the encoding algorithm) by filling in the undetermined parts of Fig. 3.4.....35

Fig.3.6 An example illustrating the proposed method. (a) is the original halftone secret image "Lena"; (b) and (c) are the two transparencies generated by our method; (d) is the image "LENA" obtained by stacking (b) and (c) (note that (b) and (c) can also be used to extract the hidden confidential text); (e) is a

faked halftone image "Girl" owned by a hacker; (f) is a faked transparency; (g) is the faked image "GIRL" obtained by stacking (b) and (f). (However, by extracting the hidden text from (b) and (f), our ally can know that (g) is faked.).....	37
Fig.3.7. Double-decoding without using a decoding-computer. (a) and (b) are the two transparencies T_1 and T_2 ; (c) is the result of stacking (a) and (b); (d) is the result of stacking T_1 with "shifted T_2 ".	49
Fig 4.1 The partitioning of a section.....	53
Fig.4.2. A sharing result for $n=8$.(a) is the important image Lena; (b) is the modified image Lena*, which contains the image U, and all a_i extracted from it are in 0-250 (see Eq. (4.1)); (c) is the 8 shares that can recover (b) together. The top-most share in (c) is the universal share, which is identical to the image U.	55
Fig.4.3. Another sharing result for $n=8$. (a) is the important image Monkey;(b) is the modified image Monkey*;(c) is the 8 shares that can recover (b) together; and the top-most share is identical to the top-most share in Fig. 4.2(c).....	56
Fig 4.4. An experiment for a color image Lena. (a) is the important image Lena; (b) is the modified image Lena*. (c) is the $n=8$ shares that can recover (b) together. The top-most share in (c) is the universal share.....	57
Fig.5.1. Flowchart of the proposed method.....	63
Fig.5.2. An example of split. (a) is the original, (b ₁) is the upper part, and (b ₂) is the lower part.....	64
Fig.5.3. Step 1 of the ($k=3,n=3$) example.....	66
Fig.5.4. Step 2 of the ($k=3,n=3$) example.....	67

Fig.5.5. The basis matrices C_0 and C_1 used for the (2,6) system in the example....70

Fig.5.6. Step 1 for the ($r=2, n=6$) example. Here, (a)-(f) are, respectively, the six shares $S_1 - S_6$73

Fig.5.7. Step 2 for the ($r=2, n=6$) example. Here, (a)-(f) are, respectively, the six shares $S_1 - S_6$. Darker elements were determined earlier in Step 1, and hence cannot be changed now in Step 2.....76

Fig.5.8. An ($r=2, n=2$) experiment using our method. (a) is the input gray-value image; (b) and (c) are the two gray-value shares (the size of each share is just $n/(n+1)=2/3$ of that of (a)); (d) is the restored error-free result (identical to (a)) using (b) and (c).....76

Fig.5.9. An ($r=2, n=2$) experimental result in Ref. [LP2005]. (a) is the input grey-value image; (b-c) are the two grey-value shares (each share is $2 \times 2=4$ times bigger than (a)); (d) is the restored error-free result (identical to (a)) using (b) and (c).....77

Fig.5.10. An ($r=2, n=6$) experiment using our method. (a) is the input gray-value image; (b) is one of the six gray-value shares (each share is $(n/(n+1)) \times (2 \times 2) = (2 \times 2) \times 6/7 = 3.43$ times greater than (a)); (c) is the restored error-free result (identical to (a)) using any two of the six shares.....78

Fig.5.11. An ($r=2, n=6$) experimental result in Ref. [LP2005]. (a) is the input grey-value image; (b) is one of the six grey-value shares (each share is $2 \times 2=4$ times bigger than (a)); (c) is the restored error-free result (identical to (a)) using any two of the six shares.....78

Chapter 1

Introduction

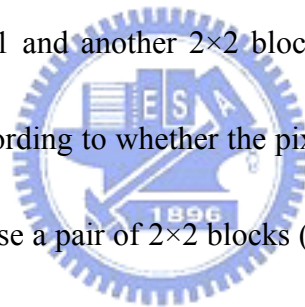
The motivation is stated first in Sec. 1.1. The background knowledge for the related topics is then introduced in Sec 1.2. Overview of the proposed methods is in Sec. 1.3. The organization of the dissertation is in Sec 1.4.

1.1 Motivation

Image sharing is used when nobody alone can be trust. No one can get the secret image without enough shares. While being applied in practice, the sharing procedure may encounter problems including forge shares, difficult management, and slow decoding. In this dissertation, we try to find solutions to these problems. We can check the shares' validity by means of the first two image sharing styles introduced in this dissertation, namely, the turnover style and two-level style. In the two-level style design, the decoding has two levels, and the two levels are of different importance. Level 1's decoding is easier and less confidential than Level 2's. Also, using the universal share designed in this dissertation, a company leader no longer has the problem of difficult management due to the increasing amounts of shares. Finally, using the image sharing technique designed in our fast-decoding style, the shared images can be decoded quickly.

1.2 Notations and Background for the related topics

Visual cryptography (VC) was introduced by Naor and Shamir [NS1995] to protect secret images. The important feature is that in visual cryptography, people can use their eyes (rather than the computer's computation) to recover a secret image by simply stacking the corresponding shares, while any share alone cannot reveal the given secret image. To know the simplest design of VC, the readers may inspect Fig. 1.1 where they can find some very simple pattern blocks that can be used repeatedly to decompose a black-and-white secret image. Note that each pixel of the secret image yields a 2×2 block in Share 1 and another 2×2 block in Share 2. In fact, for every pixel of the secret image, according to whether the pixel is black or white, people just use Fig. 1.1 to randomly choose a pair of 2×2 blocks (related to that pixel's brightness) to paint the corresponding position in Shares 1 and 2. For example, if the given pixel is white, then use one of the two upper rows to do the painting; if the given pixel is black, then use one of the two lower rows. It is obvious that stacking these two blocks yields the corresponding block shown in the rightmost column of Fig. 1.1. Notably, after stacking, if all four elements in the resulting 2×2 stacked block are black, then the input pixel of the secret image must be a black pixel; on the other hand, if just two of the four elements in the resulting stacked block are black, then the input pixel must be a white pixel. Therefore, the resulting recovered image ($2 \times 2 = 4$ times larger than



the original secret image) can be utilized to unveil the original secret image easily.

There are many extended reports or modified works [ABSS2001, BSS1999, NP1997, Shamir1998, BSN2002, HLC1999, HCL1999, Hou2003, NS1997, BC1994, CC2002, ABSS1996, BS1998] of Naor and Shamir’s work[NS1995], including the extension from black-and-white to gray-valued secret images (the extension to color images was reported less frequently [LT2003, Stinson1997, ABSS1996]).













A pixel in secret image	Corresponding blocks in Share 1	Corresponding blocks in Share 2	Resulting block from stacking the two shares
□			
			
■			
			

Fig. 1.1 Some sharing blocks found in Ref. [NS1995] (not used here in the dissertation).

1.3 Overview of the proposed methods

We briefly describe below each of the methods proposed in this dissertation.

1.3.1 Visual cryptography in turnover style

We propose a brand new type of visual cryptography (VC), namely, the VC in turnover style. For any two given secret images, two corresponding transparencies S_1 and S_2 , also known as shares, can be produced. Both transparencies look noisy. However, if we stack the front views of both transparencies, then the first secret image is unveiled. On the other hand, if we stack the front view of S_1 with the back view (the turnover) of S_2 , then the second secret image is unveiled.

1.3.2 Visual Cryptography in Two-level style: The VC with Extra Ability of Hiding Confidential Data.

Chapter 3 presents a two-in-one Visual Cryptography scheme, which not only shares an image of moderate confidentiality between two noisy transparencies, but also hides in these two transparencies a more confidential text file describing the image. None of the transparencies alone can reveal anything about the image or text. Later, people can view the image by simply stacking the two transparencies; on the other hand, after certain simple computations, the more confidential text data can also be extracted. We also introduce here an alternative version in which the decoding of both the image and text needs no computer.



1.3.3 Universal share for the sharing of multiple images

To share numerous grey-valued images (or numerous color-valued images), chapter 4

presents a system with a universal share. A company organizer can use this universal share to attend the recovery meeting of any shared image. No storage space is wasted; i.e. for each shared image, the total storage space occupied by all generated shares (including the universal share) is identical to the image size.

1.3.4 Fast-decoding sharing of a gray-value image: by using bit-level sharing and economic size shares

Chapter 5 proposes a method to distribute images over network, or store images among several storage places. The reconstruction of the images is lossless and fast. The size of each transmitted share is also competitive, as compared with an earlier work which is also bit-level based.



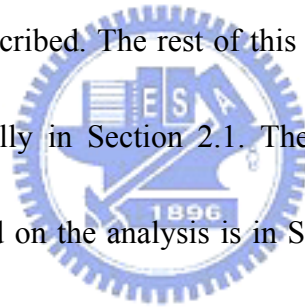
1.4 Dissertation Organization

In the rest of this dissertation, Chapter 2 discusses the visual cryptography in turnover style. Chapter 3 discusses another visual cryptography with extra ability of hiding confidential data. Chapter 4 proposes the universal share for the sharing of multiple images. By using bit-level sharing and economic size shares, Chapter 5 introduces a fast-decoding type sharing for gray-value images. Finally, the conclusions and suggestions for future works are in Chapter 6.

Chapter 2

Visual Cryptography in Turnover Style

In this Chapter, we propose a visual cryptography scheme of turnover style. Traditionally, only one secret image can be unveiled when two shares are collected.[ABSS1996, ABSS2001, BC1994, BS1998, BSN2002, BSS1999, CC2002, HLC1999, HCL1999, Hou2003, LT2003, NP1997, NS1995, NS1997, Shamir1998, Stinson1997] In this chapter, how the two shares generated by our method can unveil two secret images will be described. The rest of this chapter is organized as follows. The problem is stated formally in Section 2.1. The analysis of the problem is in Section 2.2. Our design based on the analysis is in Section 2.3. Experimental results are shown in Section 2.4. The perturbed version of the design is in Section 2.5. Finally, the discussion is presented in Section 2.6.



The notation is given below. Notably, when a pair of pixels (one from secret image 1, and the other from secret image 2) is decomposed to generate a pair of blocks (one in Share 1 and the other in Share 2), the block size of both blocks will always be 3-by-3 in our method. Therefore, the stacked blocks are also 3-by-3.

Notation:

S_1 and S_2 : the two generated transparencies (shares).

(x, y) : the 3-by-3 block at location (x, y) of each share. Note that $1 \leq x \leq 256$ and

$1 \leq y \leq 256$ if the given secret images are 256-by-256 (because then there will be 256×256 blocks for each share).

B: a black block. (In the recovered images obtained from stacking shares, a black block is a 3-by-3 block consisting of 8 black elements and 1 white element).

W: a white block. (In the recovered images obtained from stacking shares, a white block is a 3-by-3 block consisting of 6 black elements and 3 white elements).

2.1 The goal

In this chapter, we will try to design two transparencies S_1 and S_2 meeting the following two requirements: if we stack S_1 and S_2 directly, then the stacked result will be the first secret image (say, "LENA"); if we turn over the transparency S_2 , and then stack it with S_1 (S_1 is without turnover), then the stacked result will become the second secret image (say, "PEPPERS"). The idea is illustrated in Fig 2.1.

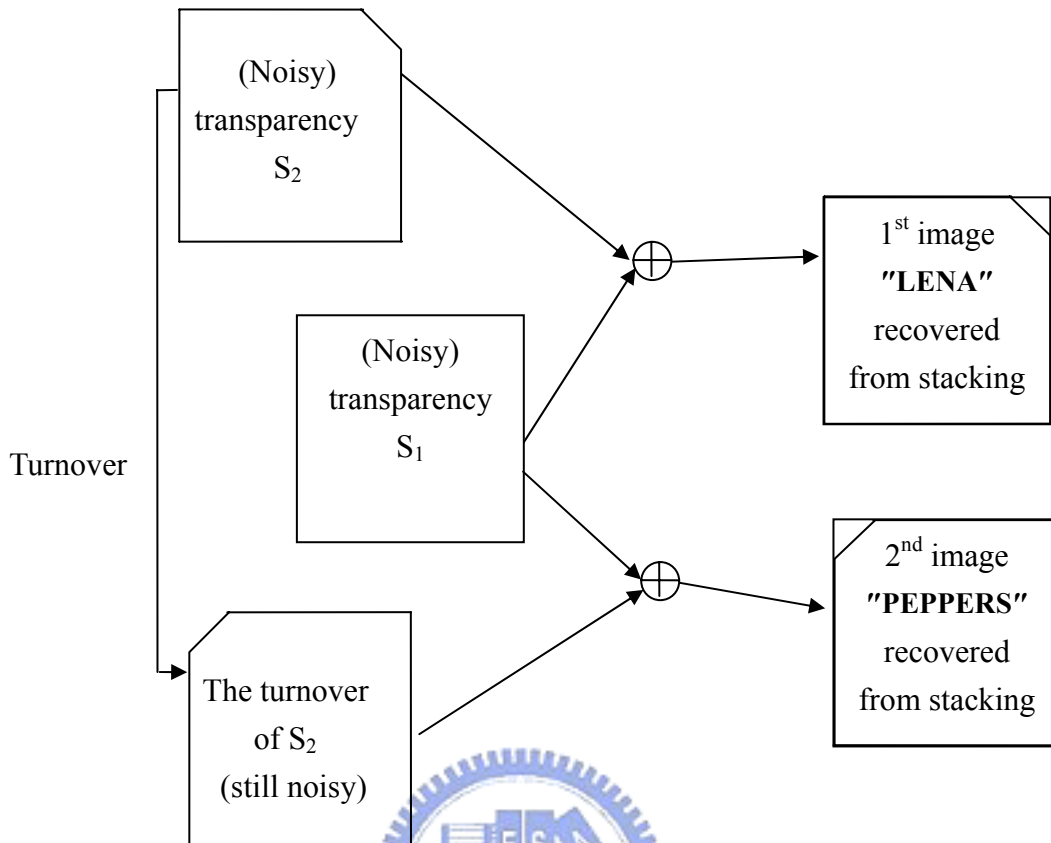


Fig. 2.1. Visual Cryptography in turnover style

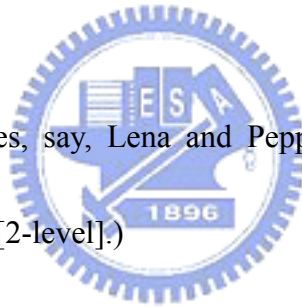
Before designing our method, we have to define the problem more precisely.

Note that in visual cryptography, it is hard to design the transparencies without size-expansion. [ABSS1996, ABSS2001, BC1994, BS1998, BSN2002, BSS1999, CC2002, HCL1999, HLC1999, Hou2003, LT2003, NP1997, NS1995, NS1997, Shamir1998, Stinson1997] In our design, if the two input images Lena and Peppers are both 256-by-256 in size, then each of our two transparencies will be 768-by-768 (where $768=3\times 256$), and each of the two recovered images LENA and PEPPERS will also be 768-by-768. This is because we always use

blocks of size 3-by-3 to expand pixels. Also, we have to define the so-called black blocks and white blocks in the recovered images LENA and PEPPERS. In our approach, as listed in the notation part of this chapter, each 3-by-3 block in the recovered images will be called a white block if its 9 elements consist of 6 black elements and 3 white elements. On the other hand, it will be called a black block if it consists of 8 black elements and 1 white element. Now we can state the problem precisely, as follows:

The Problem

Given: two input images, say, Lena and Peppers. (Both are 256-by-256 and black-and-white [2-level].)



Goal: to generate two 768-by-768 transparencies (S_1 and S_2) so that stacking S_1 and S_2 will give us the recovered image LENA (defined below); while stacking S_1 and \tilde{S}_2 (the back-view of the S_2) will give us the recovered image PEPPERS (defined below).

Recovered images: a 768-by-768 "LENA" and a 768-by-768 "PEPPERS". Note that each recovered image is a 768-by-768 black-and-white image consisting of $768/3 \times 768/3 = 256 \times 256$ non-overlapping blocks. Each 3-by-3 block (x,y) in LENA is a black block if and only if the corresponding

(1-by-1) pixel (x,y) in the original 256×256 image Lena is a black pixel.

Each 3-by-3 block (x,y) in PEPPERS is required to meet the same statement (except that the term "Lena" is replaced by the term "Peppers").

2.2. Analysis

2.2.1 The turnover operation

Fig. 2.2 below easily describes the turnover operation of a transparency. The dashed line is a rotation axis to rotate the image in order to get the turnover version of a transparency.

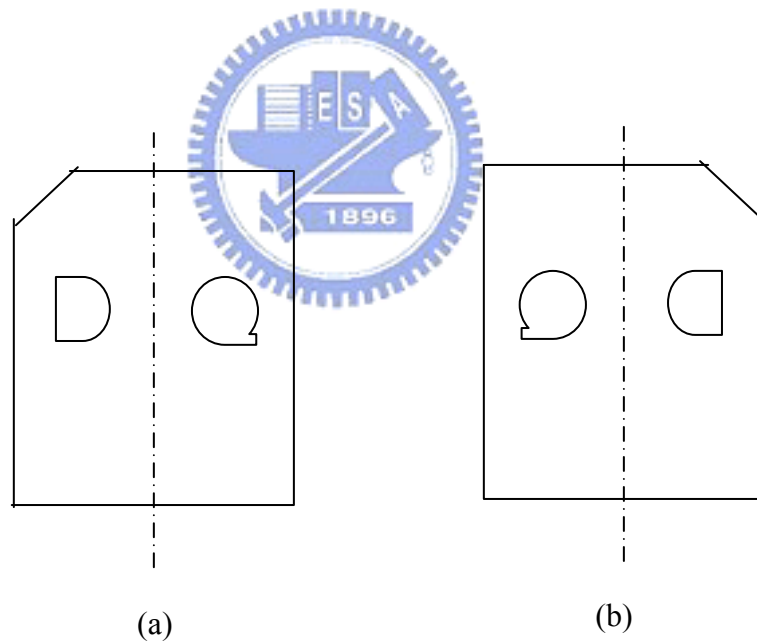
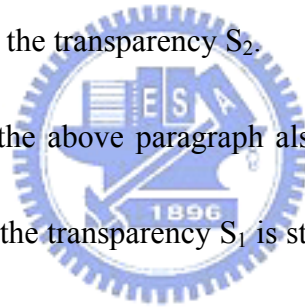


Fig. 2.2 The "turnover" operation of a transparency. (a) is a transparency, and (b) is its turnover version (the back view of the transparency in (a)).

2.2.2 Stacking the two transparencies

We may use Fig. 2.3 to illustrate how two transparencies S_1 and S_2 are stacked to get

LENA and PEPPERS. When we stack S_1 and S_2 , i.e. when we stack S_1 with the front view of S_2 , then, each 3-by-3 block (x, y) of S_1 is stacked precisely on the corresponding block (x,y) of S_2 , and becomes the block (x, y) of the recovered image LENA. This statement holds for all $1 \leq x \leq 256$ and $1 \leq y \leq 256$. On the other hand, when we turn over the transparency S_2 and stack it with S_1 (note that S_1 is without turnover) to get the recovered image PEPPERS, the block (x,y) of PEPPERS is in fact the stacked result of stacking the block (x,y) of the transparency S_1 with the turnover version (for example, the symbol "E" becomes "Э") of a block which was originally the block $(256-x,y)$ of (the front view of) the transparency S_2 .



An argument similar to the above paragraph also shows that, when we stack S_1 and S_2 , the block $(256-x,y)$ of the transparency S_1 is stacked with the block $(256-x,y)$ of the transparency S_2 to get the block $(256-x,y)$ of the recovered image LENA. On the other hand, when we turn over the S_2 and stack it with the transparency S_1 to get the recovered image PEPPERS, the block $(256-x,y)$ of PEPPERS is in fact the stacked result of stacking the block $(256-x,y)$ of the transparency S_1 with the turnover version of a block which was originally the block (x,y) of (the front view of) the transparency S_2 .

Based on the observations stated in the above two paragraphs, we know that (see Fig. 2.3):

$$\alpha_1 \oplus \alpha_2 = \alpha_L , \quad (2.1)$$

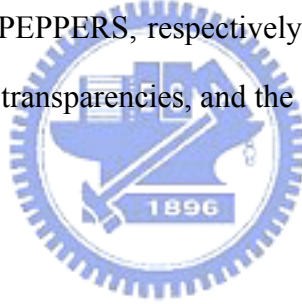
$$\beta_1 \oplus \beta_2 = \beta_L , \quad (2.2)$$

$$\alpha_1 \oplus (\beta_2^{turn-over}) = \alpha_p , \quad (2.3)$$

and

$$\beta_1 \oplus (\alpha_2^{turn-over}) = \beta_p . \quad (2.4)$$

Here, " \oplus " means the stack operation; α_i means the block (x, y) of the transparency S_i (i=1 and 2); while α_L and α_p mean the block (x,y) of the recovered image LENA and PEPPERS, respectively. Similarly, β_i means the block (256-x, y) of the transparency S_i (i=1 and 2); while β_L and β_p means the block (256-x, y) of the recovered images LENA and PEPPERS, respectively. The above four equations (1)-(4) give us the rules to design our transparencies, and the design is discussed in next section.



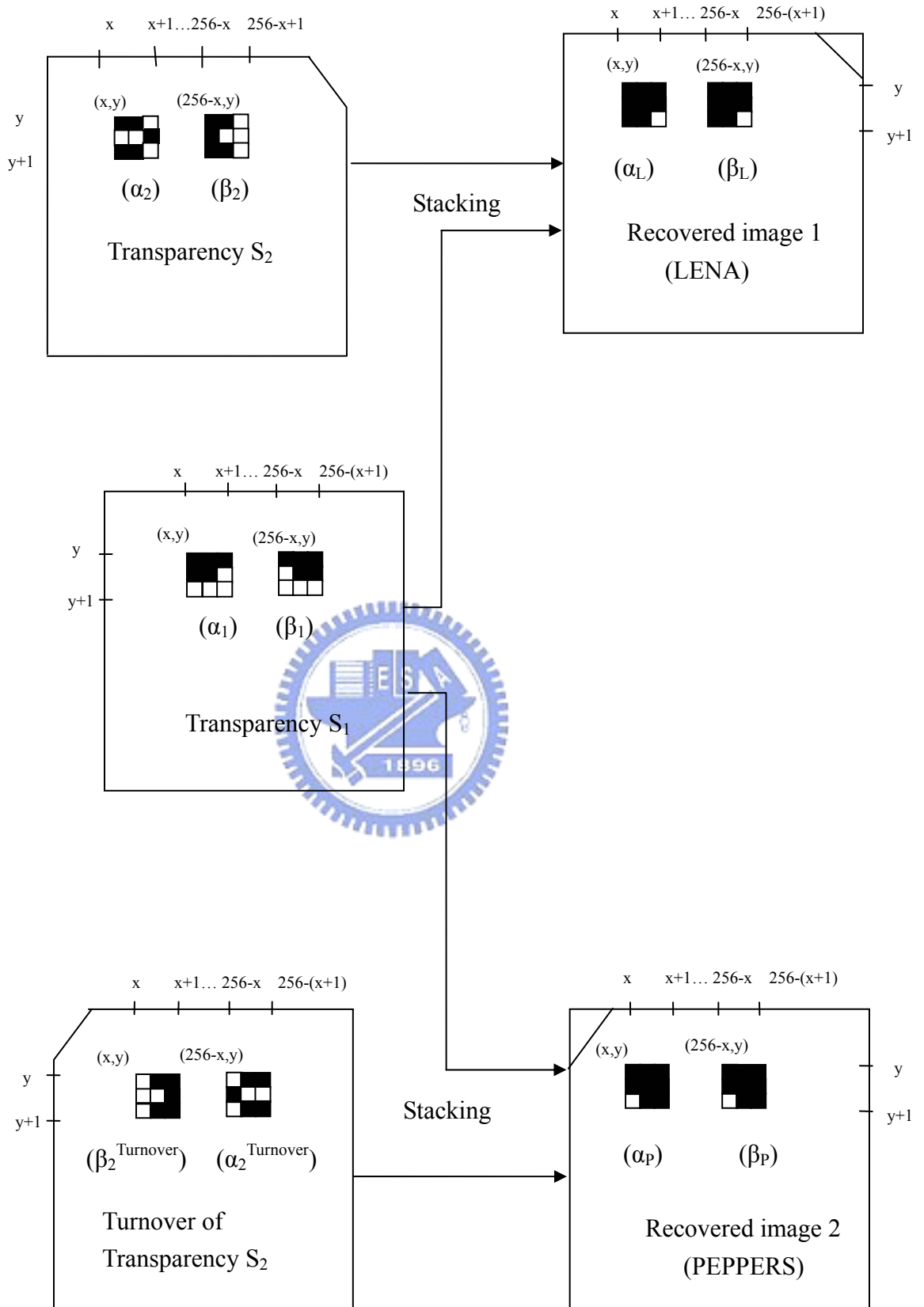
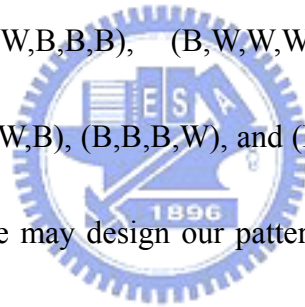


Fig. 2.3 Two types of stacking.

2.3. The design of the shares (transparencies) S_1 and S_2

Using the requirement (1)-(4), we may now design our two transparencies S_1 and S_2 . Note that if we inspect the blocks α_L , β_L , α_P , and β_P , and keep in mind that each of these four stacked blocks is either a black block or a white block (because we require that, say, block α_L is a black block if and only if the corresponding pixel (x,y) in the original black-and-white image Lena is a black pixel), we know that there are 16 possible cases for the blackness-whiteness of the four stacked blocks (α_L , β_L , α_P , β_P), namely, (W,W,W,W), (W,W,W,B), (W,W,B,W), (W,W,B,B), (W,B,W,W), (W,B,W,B), (W,B,B,W), (W,B,B,B), (B,W,W,W), (B,W,W,B), (B,W,B,W), (B,W,B,B), (B,B,W,W), (B,B,W,B), (B,B,B,W), and (B,B,B,B). Therefore, according to these 16 possible cases, we may design our pattern blocks and use these pattern blocks to "paint" the two transparencies S_1 and S_2 . The pattern blocks are sketched in Fig.2.4.



To illustrate how to use Fig. 2.4, without the loss of generality, assume that the two pixels (x,y) and $(256-x,y)$ in the original image 1 (Lena) are both black, and the two pixels (x,y) and $(256-x,y)$ in the original image 2 (Peppers) are both black too. In other words, assume that we have a (B,B,B,B) case here. We can look up the first column of the table in Fig.2.4 and find that the (B,B,B,B) case is shown in the last row. As a result, we can paint the two corresponding blocks (x,y) and $(256-x,y)$ of the

transparency S_1 by the two pattern blocks sketched in the last row of the two columns marked as (α_1) and (β_1) , respectively. Similarly, we can paint the two corresponding blocks (x,y) and $(256-x,y)$ of the transparency S_2 using the two pattern blocks sketched in the last row of the two columns (α_2) and (β_2) , respectively. It is obvious that the stacked results $\{\alpha_L$ (which is $\alpha_1 \oplus \alpha_2$) , β_L (which is $\beta_1 \oplus \beta_2$) , α_P (which is $\alpha_1 \oplus (\beta_2^{turn-over})$), β_P (which is $\beta_1 \oplus (\alpha_2^{turn-over})$) } really meet the expected blackness/whiteness values, namely, they are all black blocks, as shown in the last columns of the last row of Fig.2.4.



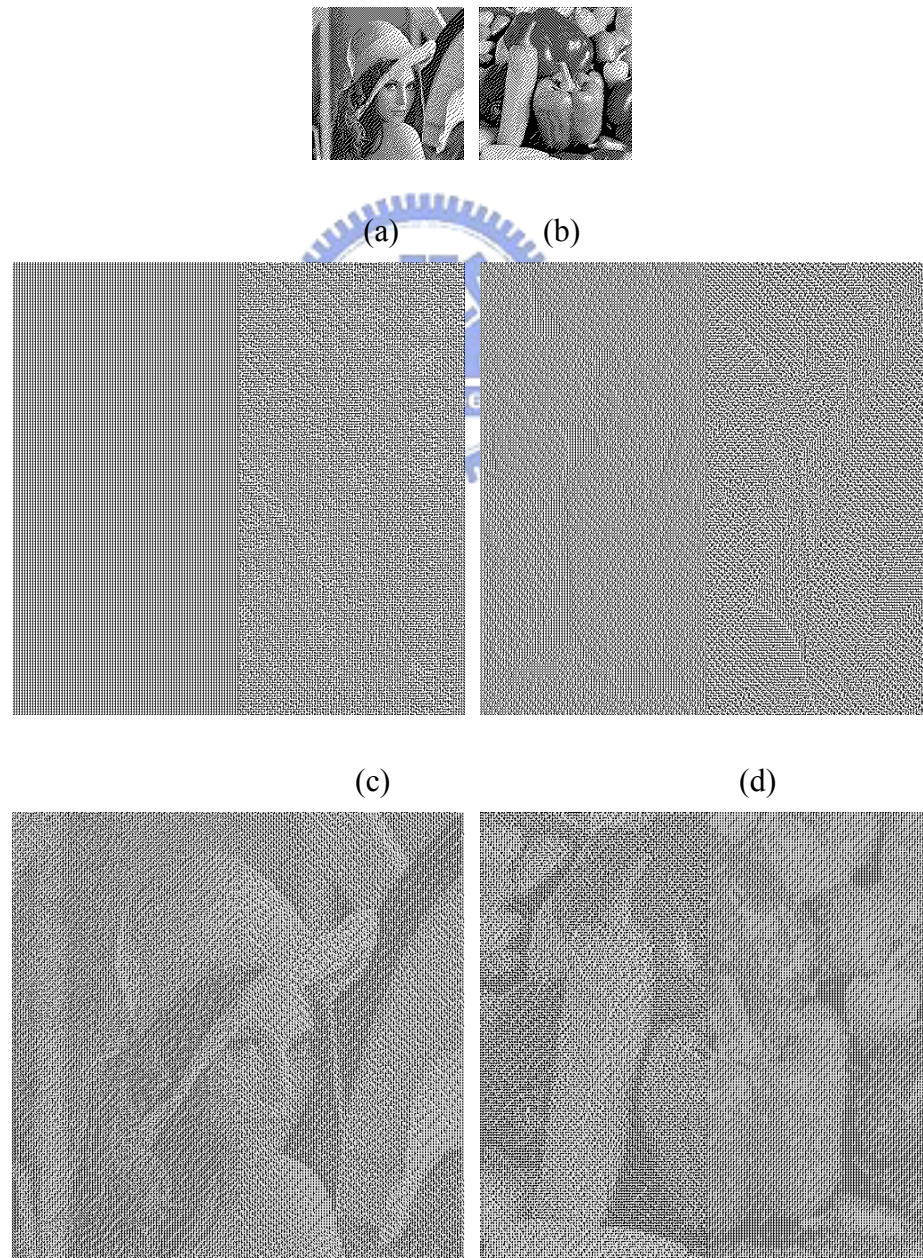
Our goal for the stacked results	Design of the two transparencies		Stacked results really meet our goal
	Transparency	Transparency	
	S ₁	S ₂	
(W,W,W,W)			
(W,W,W,B)			
(W,W,B,W)			
(W,W,B,B)			
(W,B,W,W)			
(W,B,W,B)			
(W,B,B,W)			
(W,B,B,B)			
(B,W,W,W)			
(B,W,W,B)			
(B,W,B,W)			
(B,W,B,B)			
(B,B,W,W)			
(B,B,W,B)			
(B,B,B,W)			
(B,B,B,B)			

$(\alpha_1) (\beta_1)$ $(\alpha_2) (\beta_2)$ $(\alpha_L) (\beta_L) (\alpha_P) (\beta_P)$

Fig.2.4. The pattern blocks $\{\alpha_1, \beta_1, \alpha_2, \beta_2\}$ used in our method to paint the two transparencies. Columns (α_1) and (β_1) are, respectively, the pattern blocks in position $\alpha=(x,y)$ and position $\beta=(256-x,y)$, for transparency S₁. Analogously, columns (α_2) and (β_2) are for transparency S₂; (α_L) and (β_L) are for the stacked result "LENA"; (α_P) and (β_P) are for the stacked result "PEPPERS". Note that the stacked result are $\alpha_1 \oplus \alpha_2 = \alpha_L, \beta_1 \oplus \beta_2 = \beta_L, \alpha_1 \oplus \beta_2^{turn-over} = \alpha_P$, and $\beta_1 \oplus \alpha_2^{turn-over} = \beta_P$ (to understand, see Fig. 2.3, which corresponds to the last row (B,B,B,B) here).

2.4. Experiment

Fig. 2.5 shows the experimental result. The two binary images (a) and (b) in Fig. 2.5 are the original secret images (Lena and Peppers), both are 256-by-256. We then use the pattern listed in Fig.2.4 to create the two transparencies S_1 and S_2 ((c) and (d)). If we stack S_1 and S_2 , the stacked result is (e). If we stack S_1 and the back-view of S_2 , that is, stack (c) and the turnover version of (d), then we get another stack image (f).



(e)

(f)

Fig. 2.5 The experiment of the turnover-style visual cryptography. (a) and (b) are the two original black-and-white images; (c) and (d) are the two generated transparencies S_1 and S_2 ; whereas (e) and (f) are the two recovered images ((e) is the result of stacking the pair $\{(c),(d)\}$, while (f) is the result of stacking the pair $\{(d)^{\text{turnover}}, (c)\}$).

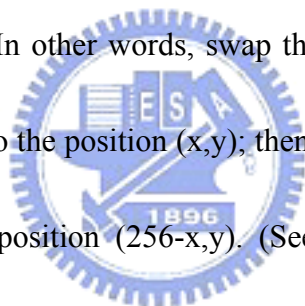
2.5 Perturbed version of the design

If we inspect Fig. 2.5(d) carefully, we can see that some rough contours of Lena appear in this transparency. The reason of this phenomenon might be that sometimes the same case (e.g. the (B,W,B,B) case) repeats along one side of the original contour of the input secret image Lena. For example, Lena's (139,36) is black, Lena's (256-139,36) is white; Peppers' (139,36) is black, and Peppers' (256-139,36) is black. Then, again, Lena's (139,37) is black, Lena's (256-139,37) is black; Peppers' (139,37) is black, Peppers' (256-139,37) is black. We will then repeatedly use the same pattern blocks listed in row 12 (the (B,W,B,B) row) of Fig.2.4 to paint transparencies S_1 and S_2 on the corresponding positions. The above analysis also holds on the other side of the original contour of the input secret image Lena. Therefore, the contour might also appear in the produced transparencies. In order to reduce this phenomenon, we use the three rules shown in Fig. 2.6 (b-d) to perturb each row of Fig.2.4 to obtain more combinations of pattern blocks $\{\alpha_1, \beta_1, \alpha_2, \beta_2\}$. In summary, the general guideline is to avoid using a single combination of pattern blocks repeatedly to represent a case (say,

the (B,W,B,B) case).

Below, we discuss how to perturb each row of Fig. to obtain more combinations. At least three possible ways exist. Notably, although Fig. 2.6 only shows how the three kinds of operation can be applied to perturb the case in which the stacking result is (W,W,W,W), these three types of operations can be attempted to operate each of the $2 \times 2 \times 2 \times 2 = 16$ cases listed in Fig.2.4 to generate more combinations.

Among the three types of operations, the first type is the so-called "external interchanging", i.e. interchanging between transparencies S_1 and S_2 in a position-by-position manner. In other words, swap the (α_1) and (α_2) columns in Fig. 2.4 because both correspond to the position (x,y) ; then swap the (β_1) and (β_2) columns because both correspond to position $(256-x,y)$. (See Fig. 2.6 (b) for the external interchanging.) The second type of operation that can be used is the so-called "internal permutation", i.e. permute the 3 rows of the elements for each 3-by-3 pattern block in (α_1) , then do the same permutation of rows for each pattern block in (β_1) , (α_2) and (β_2) . (See Fig. 2.6 (c) for the internal permutation.) The third type of operation that may be used is to re-design the pattern blocks without referring to the pattern blocks listed in Fig.2.4. For example, if our goal is that the stacked result appearing on LENA's (x,y) , LENA's $(256-x,y)$, PEPPERS' (x,y) , and PEPPERS' $(256-x,y)$, are W, W, W, and W, respectively; then we can see that not only 2.6 (a)-(c) but also 2.6 (d)



meet our goal: the stacked results ($\alpha_1 \oplus \alpha_2 = \alpha_L$, $\beta_1 \oplus \beta_2 = \beta_L$, $\alpha_1 \oplus \beta_2^{\text{turn-over}} = \alpha_p$, $\beta_1 \oplus \alpha_2^{\text{turn-over}} = \beta_p$) are indeed (W, W, W, W).

Finally, as a very specific remark to benefit the readers, although internal permutation always guarantee that the output case is always the same as the input case (e.g. if the input case is (W, W, B, W), the output case must also be (W, W, B, W)), the external permutation does not guarantee this. For example, when the external-interchanging operation is applied to the (W, W, B, W) case of Fig. 2.4, the generated combination will be suitable for the (W, W, W, B) case, but not the (W, W, B, W) case. On the other hand, when the external interchanging operation is applied to the (W, B, W, W) case of Fig. 2.4, the generated combination will still be suitable for (W, B, W, W) case. In fact, for 8 of the 16 cases in Fig. 2.4, each will switch to another case when external-interchanging operation is applied. So, only $16-8=8$ cases can use external-interchanging; and these 8 cases are (W, B,W, W),(B, W, W, W),(W, W,W, W),(B, B, W, W),(W, W, B, B),(B, B, B, B).

After perturbing each row of Fig. 2.4 using the three operations mentioned above (or equivalently, mentioned in Fig. 2.6 (b)-(d)), we will have more choices when we want to paint the two transparencies S_1 and S_2 . We just randomly select one of the many choices to do the painting. For example, for the (W, W, W, W) case, sometimes we use Fig. 2.6(b), sometimes we use Fig. 2.6 (c), which has $3!=6$ possible

combinations (one of them is shown in Fig. 2.6(a)), and sometimes we use

Fig. 2.6(d). Then, the generated transparencies S1 and S2 are as shown in Fig. 2.7

(c)-(d). Fig. 2.8 is another experiment for this perturbed version.



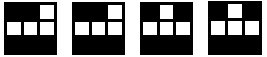

Original combination	Modification	Modified combinations
<p>2.6 (a):</p>  <p>(α_1) (β_1) (α_2) (β_2)</p>	<p>Interchanging between the two transparencies S1 and S2</p>	<p>2.6 (b):</p>  <p>(α_1) (β_1) (α_2) (β_2)</p>
	<p>Row-by-row permutation</p>	<p>2.6 (c):</p>  <p>(α_1) (β_1) (α_2) (β_2)</p>
	<p>Other new design</p>	<p>2.6 (d):</p>  <p>(α_1) (β_1) (α_2) (β_2)</p>

Fig. 2.6. Three kinds of perturbation operation to get some other combinations of pattern blocks whose stacked results ($\alpha_1 \oplus \alpha_2 = \alpha_L$, $\beta_1 \oplus \beta_2 = \beta_L$, $\alpha_1 \oplus \beta_2^{turn-over} = \alpha_p$, $\beta_1 \oplus \alpha_2^{turn-over} = \beta_p$) are all (W,W,W,W). (a) is the first row (the (W,W,W,W) row) of Fig. (b) is obtained from (a) by switching α_1 with α_2 , then followed by switching β_1 with β_2 . (c) is obtained from (a) by permuting the 3 rows of α_1 , then permuting the 3 rows (using the same permutation order) for each of α_2 , β_1 and β_2 . (d) is obtained using a new design (not necessarily related to (a)).

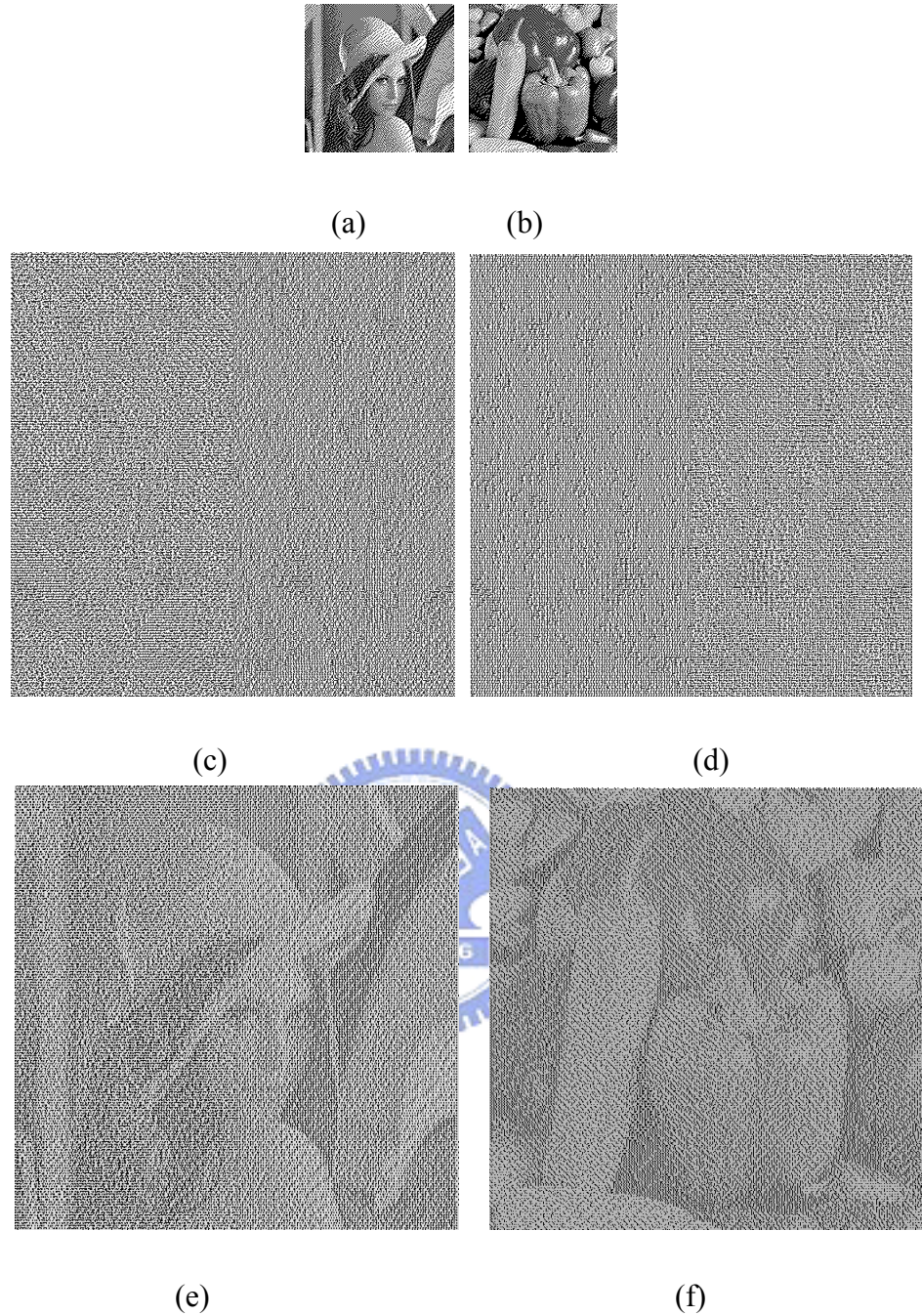


Fig. 2.7 The experiment (perturbed version). (a) and (b) are the two original images; (c) and (d) are the two generated transparencies S1 and S2; whereas (e) and (f) are the two recovered images ((e) is from stacking $\{(c),(d)\}$, while (f) is from stacking $\{(d)^{\text{turnover}}, (c)\}$).

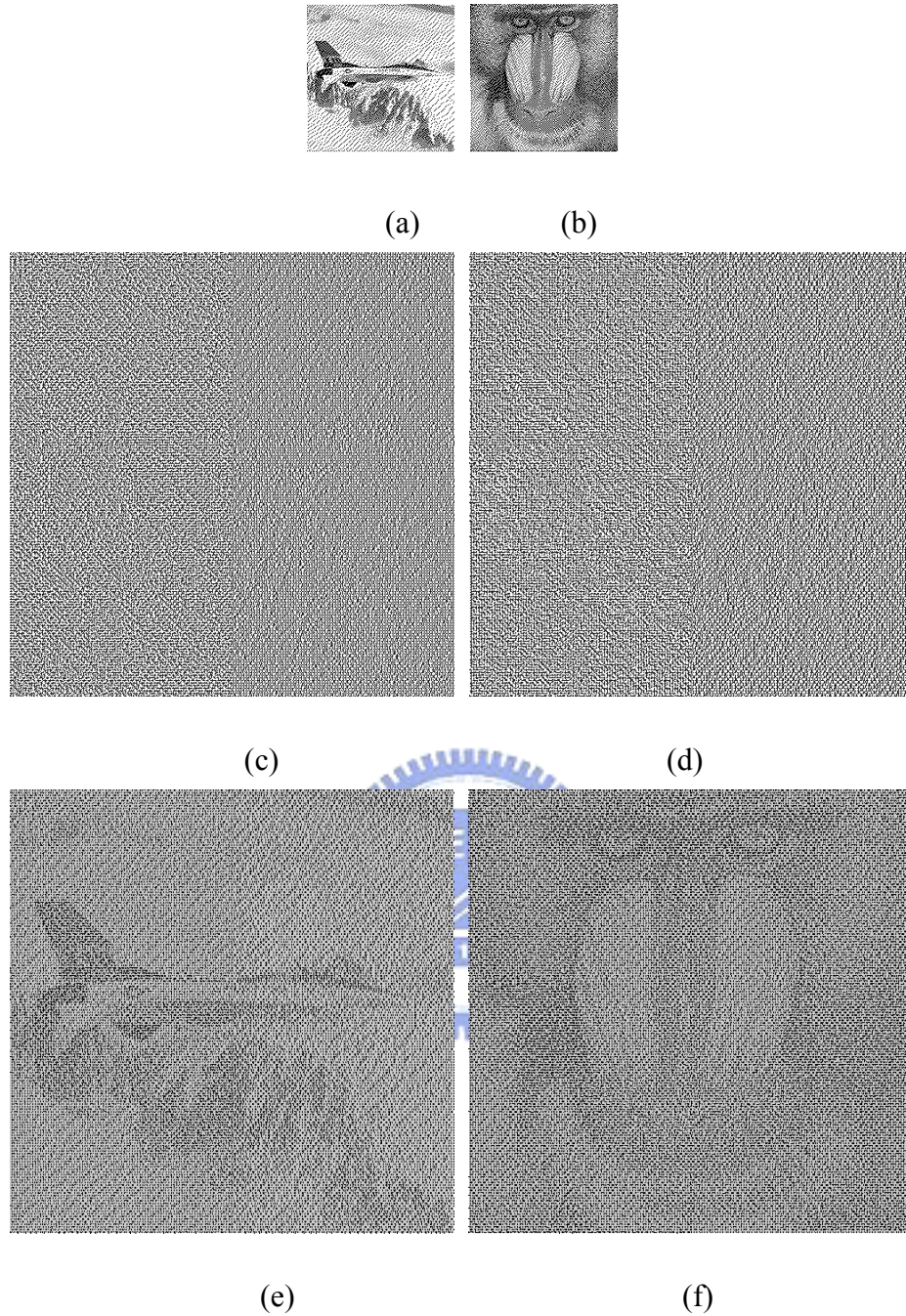
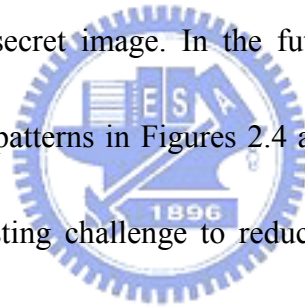


Fig. 2.8 Another experiment for the perturbed version. (a) and (b) are the two original images; (c) and (d) are the two generated transparencies S_1 and S_2 ; whereas (e) and (f) are the two recovered images ((e) is from stacking $\{(c),(d)\}$, while (f) is from stacking $\{(d)^{\text{turnover}}, (c)\}$).

2.6. Summary

A new style of visual cryptography, called the visual cryptography in the

turnover style, has been proposed in this chapter. After the analysis using Fig. 2.3, we set up Equations (1)-(4) as the requirement for the design. Then we designed in Fig.2.4 the 16 basic combinations of pattern blocks to handle the 16 possible cases WWW, WWWB,...,BBBB. In order to reduce the original-image-related contours from appearing in transparencies, we used the perturbation technique (Fig. 2.6) to get a perturbed version. The final result was shown in Fig. 2.7 and 2.8. The major contributions of this chapter might be stated as: a) the back-view of the transparency was also used; and b) the design showed that two noisy transparencies (S1 and S2) could share more than one secret image. In the future, we may try to find more efficient ways to perturb the patterns in Figures 2.4 and 2.6 to deal with the contour problem. It is also an interesting challenge to reduce the left-to-right discontinuity behavior across the vertical middle line $\{(x,y) \mid x=128 \text{ and } y=1,2,\dots,256\}$ of the stacked results (assuming that the input images are 256-by-256).



Below, we explain why all the blocks used in this chapter are 3-by-3 instead of 2-by-2. Let P_T be the number of black elements in a transparency block (i.e. a block in either transparency S1 or S2), P_b be the number of black elements in a recovered image's black block (i.e. a black block in LENA or PEPPERS), and P_w be the number of black elements in a recovered image's white block (i.e. a white block in LENA or PEPPERS). For example, if the blocks being used are 3-by-3, then $P_T=5$, $P_w=6$, and

$P_b=8$ in Fig. 2.4. Now, if the blocks used in the method are all n -by- n , then, regardless of the value of n , the constraints stated in Eq. (2.5)-(2.6) below must always be satisfied.

Since the number of black elements in any n -by- n block must not exceed $n \times n$, we have

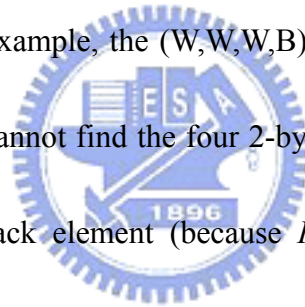
$$0 < P_T < n \times n \quad (2.5)$$

In Eq. (2.5), the value of P_T cannot be 0 or $n \times n$. If P_T is 0, then all of the blocks in the whole transparency has no black elements at all, so the whole transparency S_1 is white everywhere (so is S_2). If one stacks the two transparencies, one will only yield an image which is white everywhere. Therefore, P_T cannot be 0. Analogous argument also explains why P_T cannot be $n \times n$. Now, consider what will happen when a transparency block of S_1 is stacked with a transparency block of S_2 . If none of the P_T black elements of S_1 overlap the P_T black elements of S_2 , then the stacked result will have $2P_T$ black elements. On the other hand, if all black elements overlap in pairs when the two transparency blocks are stacked, then the stacked result will have only P_T black elements. Therefore, after stacking, the black elements appearing in the resulting block must be in the range between P_T and $2P_T$. As a result, the values of P_b and P_w must satisfy

$$P_T \leq P_w < P_b \leq 2P_T. \quad (2.6)$$

Here, $P_w < P_b$ is due to the natural requirement that the number of black elements appearing in a white block of the stacked results is always smaller than that appearing in a black block.

With the requirements (2.5) and (2.6) above, if we use 2×2 blocks in the method, then P_T cannot be 0 and 4 by Eq. (2.5). Since P_T is 1, 2, or 3, Eq. (2.6) implies that the only possible cases of (P_T, P_w, P_b) are (1,1,2), (2,2,3), (2,2,4), (2,3,4), and (3,3,4). After checking Eq. (2.1)-(2.4) carefully, we find that none of the (P_T, P_w, P_b) -triples in the candidate set $\{(1,1,2), (2,2,3), (2,2,4), (2,3,4), (3,3,4)\}$ can generate all the 16 expected cases listed in Fig. 2.4. For example, the (W,W,W,B) case cannot be generated when $(P_T, P_w, P_b) = (1,1,2)$, i.e. we cannot find the four 2-by-2 blocks $\{\alpha_1, \beta_1, \alpha_2, \beta_2\}$ such that each block has only one black element (because $P_T = 1$), and the stacked results defined by Eq. (2.1)-(2.4) are W,W,W, and B, respectively. We therefore abandoned 2×2 blocks and used 3×3 blocks in this chapter. With the decision of using 3×3 blocks, we have tried several other kinds of settings to satisfy Eq. (2.5)-(2.6) (for example, using $P_T = 6, P_w = 7, P_b = 8$). However, we found that designing the pattern blocks to satisfy Eq. (1)-(4) for all 16 cases was hard for most of these settings. For example, the $(P_T = 7, P_w = 7, P_b = 9)$ setting cannot generate the pattern blocks needed for the (W,W,W,B) case. As for the $(P_T = 6, P_w = 7, P_b = 8)$ setting, although we did obtain some pattern blocks (for each of the 16 cases) that meet all the requirements in Eq.



(2.1)-(2.4), the recovered images are darker and have smaller contrast values than those appeared in Fig. 2.5, 2.7, and 2.8. We therefore still used $P_T=5$, $P_w=6$, and $P_b=8$ in our design.



Chapter 3

Visual Cryptography in Two-level Style: the VC with Extra Ability of Hiding Confidential Data

We try to design in this chapter a VC scheme whose decoding has two levels, and these two levels are of distinct importance. The proposed VC scheme shares an image Lena of moderate confidentiality using two transparencies; and these two transparencies together can also extract a more confidential text data file describing the image Lena. More specifically, simply stacking the two transparencies can show the image Lena directly; meanwhile, after certain computations (or after shifting one of the transparencies before stacking), one can also extract the more sensitive text data. Of course, none of the transparencies alone can reveal anything about the image Lena or the more sensitive text.

There are several kinds of application for this scheme. Perhaps the most direct application is to store or transmit an image of moderate confidentiality (for example, the photo of an employee, a suspect, or a VIP member) using the two generated transparencies, whose stacking together can yield the image. In addition, the background description of the image (for example, the employee's name, address,

birthday, nationality, salary, and professional training) can only be extracted from joint information of the two transparencies by a high-ranking officer who knows the decoding key discussed in Sec. 3.3. As a result, if the two shares are collected, both of the lower rank officers and their commander can view the image; however, only the commander can decrypt the background text.

Another application is to authenticate the transparencies generated by the VC system so that the combination containing faked transparencies can be identified, as will be discussed later in Sec. 3.3.

The rest of this chapter is organized as follows: the method is stated in Sec. 3.1; the experimental results are in Sec. 3.2; the application to the authentication of the transparencies is in Sec. 3.3; the security concern is discussed in Sec. 3.4; an alternative version, whose decoding does not need any computer, is given in Sec. 3.5; finally, the conclusions are in Sec. 3.6. Throughout the chapter, "Lena" is an input image of size 256-by-256, whereas "LENA" is the corresponding 512-by-512 stacking result obtained from stacking the two transparencies (both are 512-by-512) generated by the proposed method.

3.1. The method

We describe here how to create the two transparencies. Before doing so, we will

need to introduce six fundamental blocks (Type 0 - Type 5, see Fig. 3.1) and the combinations to expand a white pixel or a black pixel (see Fig. 3.2 and 3.3, respectively) of a given image Lena.

The simplest design is to use 2x2 as the expansion rate, i.e. each share is 2x2 times greater than the input image Lena in size. Then, to define the fundamental blocks, we consider all possible cases in which a 2-by-2 block has 2 white elements and 2 black elements. Therefore, we obtain $C(4,2)=4 \times 3 / 2 = 6$ types of fundamental blocks, called Types 0,1,2,3,4,5, as shown in Fig. 3.1.

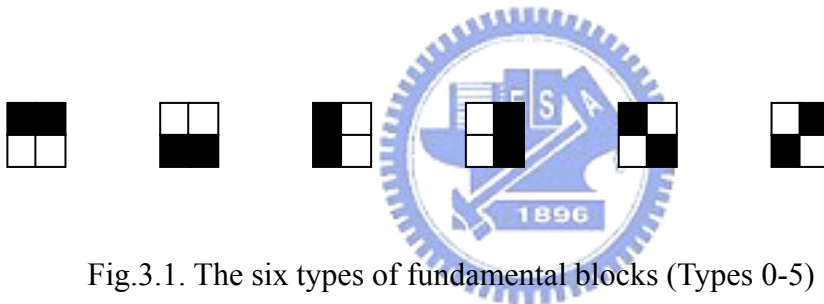
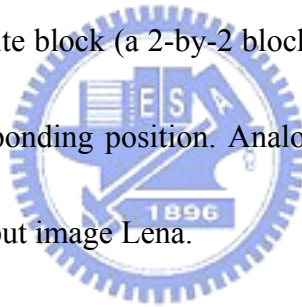


Fig.3.1. The six types of fundamental blocks (Types 0-5)

When we stack two shares and obtain the stacking result, it is an image 2x2 times greater than the input image Lena; for each pixel of Lena is expanded to a 2-by-2 block of the stacking image LENA. In order to maintain the black-and-white distribution (and hence the visual appearance) of the input image, the number of black elements in each black block of the stacking result LENA (a 2-by-2 block expanded from a black pixel of Lena) should be more than that of each white block of the stacking result LENA (a 2-by-2 block expanded from a white pixel of Lena). For simplicity of the design, we assume that each black block has four black elements,

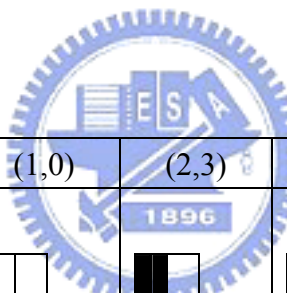
and each white block has two white elements. (This obviously meets the rule that each black block has more black elements than each white block has.) Under this assumption, it is easy to see that if we want to share a white pixel of Lena, we may use one of the six combinations $\{(0,0),(1,1),(2,2),(3,3),(4,4),(5,5)\}$ sketched in Fig. 3.2 to paint the corresponding position of the two shares. For example, the combination (3,3) means that, when we encode a white pixel, we may paint both Shares 1 and 2 (at the block position corresponding to the white pixel being processed) using the fundamental block of Type 3. Then in the decoding process, stacking these two shares will generate a white block (a 2-by-2 block having 2 white elements and 2 black elements) at the corresponding position. Analogously, we may use Fig. 3.3 to encode a black pixel of the input image Lena.



Also note that, in order to hide the more sensitive text file, we first transfer the text file to a sequence of digits of base 6, since we have six types of fundamental blocks. (Therefore, all digits are in the range 0-5.) Then this sequence is hidden.

Combinations	(0,0)	(1,1)	(2,2)	(3,3)	(4,4)	(5,5)
Blocks in Share 1						
Blocks in Share 2						
Stacking result						

Fig.3. 2 The pairs whose stacking results represent white blocks.



Combinations	(0,1)	(1,0)	(2,3)	(3,2)	(4,5)	(5,4)
Blocks in Share 1						
Blocks in Share 2						
Stacking result						

Fig. 3.3 The pairs whose stacking results represent black blocks.

Below is the algorithm of the proposed method. The more sensitive text file is

hidden first using Part I of the algorithm; the input image Lena is then hidden using Part II.

Encoding algorithm.

Part I. (To hide the confidential text data $\mathbf{d} = (d_1, d_2, d_3, \dots, d_8, d_9, \dots)$ where each d_i is a digit in the range 0-5.)

Step 1. Split each digit d_i into two parts x_i and y_i so that if a person only acquires one of the two numbers in $\{x_i, y_i\}$, he cannot extract d_i . More precisely, we randomly select a number x_i in the range 0-5, then find the corresponding y_i in the range 0-5 so that $(x_i + y_i) = d_i \pmod{6}$. For example, if the confidential text data $\mathbf{d} = (d_1, d_2, d_3, \dots, d_8, d_9, \dots) = (1035004234\dots)$, then we may decompose \mathbf{d} as

$$1=3+4 \pmod{6},$$

$$0=1+5 \pmod{6},$$

$$3=1+2 \pmod{6},$$

$$5=0+5 \pmod{6},$$

$$0=2+4 \pmod{6},$$

$$0=0+0 \pmod{6},$$

etc.

Step 2. In an interleave manner, the process sequentially generates half of the pixels in

each of the two transparencies. For instance, $1=3+4 \pmod 6$, $0=1+5 \pmod 6$, and $3=1+2 \pmod 6$ in the above example, so the 2-by-2 fundamental blocks of Types 3, 1, 1 are the 1st, 3rd, and 5th blocks of Transparency 1, while the 2-by-2 fundamental blocks of Types 4, 5, 2 are the 2nd, 4th, and 6th blocks of Transparency 2. As for the remaining blocks, they are to be determined later in Part II to share the image Lena. In other words, as is shown in Fig. 3.4, right now we have

Transparency 1 : 3 ? 1 ? 1 ? 0 ? 2 ? 0 ? ... ,

Transparency 2: ? 4 ? 5 ? 2 ? 5 ? 4 ? 0 ... ,

Part II. (To share the input image Lena)

Step 3. To determine the block types of the remaining blocks (the question marks written at the end of Step 2 above) of the two transparencies, we sequentially read the pixel values of the input image Lena. Without the loss of generality, assume that, according to the scanning order, the pixel values are BBWBWB.... Since the desired visual decoding property requires that, after stacking, the 2-by-2 blocks had better also appear in the sequence BBWBWB..., thus the block types of the "?" can be determined by looking up the table in Fig. 3.2 (or Fig. 3.3) if a "W" (or a "B") is the current pixel value of Lena. This results in the two transparencies shown in Fig. 3.5; in other words, we now have:

Transparency 1: 3 5 1 4 1 3 ...,
 Transparency 2: 2 4 1 5 1 2 ...,
 Expected stacking: B B W B W B ...

—END of the Algorithm—

Transparency T1		?		?		?
Transparency T2	?		?		?	

Fig. 3.4 Hiding the confidential text file (see Steps 1 and 2 of the encoding algorithm).

The desired stacking result	B	B	W	B	W	B
Transparency T1						
Transparency T2						

Fig. 3.5 Sharing the secret image Lena (see Step 3 of the encoding algorithm) by filling in the undetermined parts of Fig. 3.4.

In the decoding phase, directly stacking the two transparencies can reveal the (enlarged) image LENA without any computation, while the hidden text file $\mathbf{d} = (d_1, d_2, d_3, \dots, d_8, d_9, \dots) = 1035004234\dots$ can be extracted by the formula $(x_i + y_i) = d_i \pmod 6$ from the two transparencies. Here, x_i is the block type of the $(2i-1)$ -th block of Transparency 1, and y_i is the block type of the $(2i)$ -th block of Transparency 2.

3.2 Experimental results

The experimental results are shown in Fig. 3.6. The size of each transparency is 512×512 , which is 2×2 times larger than the 256×256 input image Lena. The transparencies are noise like. If we stack them, then we can get the visible image directly as shown in Fig 3.6(d). The two transparencies can also cooperate together to extract a sequence of 128×256 digits (each digit is in the range 0-5) hidden earlier, and this sequence is the confidential text for the purpose of validity-verification or background-description. The contrast (defined in Ref. [HKS2000]) of our stacked result (Fig. 3.6(d)) is always $(4-2)/(2 \times 2) = 50\%$ because each 2-by-2 block of our stacked result always has 4 black elements if it is a black block, and 2 black elements if it is a white block. Notably, according to Ref [HKS2000], the contrast is good if it reaches 50%. Therefore, the visual quality of our stacked result is not worse than those obtained using reported VC methods, while we have the extra advantage to hide the confidential text describing the unveiled image.

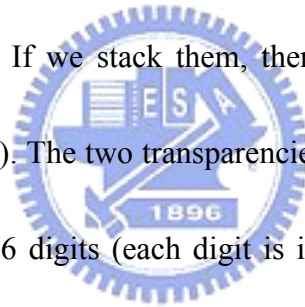




Fig. 3.6 An example illustrating the proposed method.

(a) is the original halftone secret image "Lena"; (b) and (c) are the two transparencies generated by our method; (d) is the image "LENA" obtained by stacking (b) and (c) (note that (b) and (c) can also be used to extract the hidden confidential text); (e) is

a faked halftone image "Girl" owned by a hacker; (f) is a faked transparency; (g) is the faked image "GIRL" obtained by stacking (b) and (f). (However, by extracting the hidden text from (b) and (f), our ally can know that (g) is faked.)

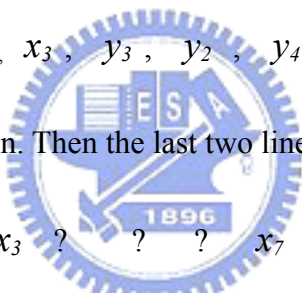
3.3 Application to the authentication of the transparencies

As mentioned in Sec. 3.1, one of the applications of the proposed scheme is to authenticate the transparencies so that the combination of faked transparencies can be identified. This is explained below. We only have to transform an authentication message to a sequence d of digits in the range 0-5, and then hide this sequence according to Part I of the algorithm to obtain Shares 1 and 2. Now, assume that a hacker intercepts Share 1 (Fig 3.6(b)). Using Share 1 and his own faked input image "Girl" (Fig. 3.6(e)), the hacker may create a faked share (Fig. 3.6(f)), called Share 2', so that stacking Shares 1 and 2' will yield the faked stacking result "GIRL" (Fig. 3.6(g)). However, the hacker cannot fool our agent/ally who is waiting at the receiver end of our network, since the faked pair {Share 1, Share 2'} cannot extract d (and hence, cannot extract our authentication message).

3.4 Techniques to improve the security

Because Shares 1 and 2 are transmitted using two distinct channels or stored in two different places, the chance that both shares are intercepted is very low. However, to

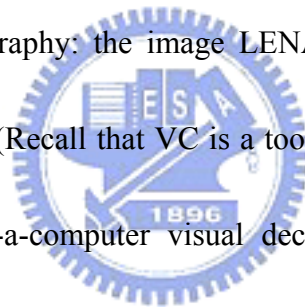
reduce the damage that might be caused by a super-hacker (or an internal betrayer of a company) who might intercept or access both Shares 1 and 2, the security department can modify our algorithm by using their own manner to reassign the locations of the $\{x_i\}$ in Share 1 and $\{y_i\}$ in Share 2. ($\{x_i\}$ are not necessarily in the odd positions of Share 1, and $\{y_i\}$ are not necessarily in the even positions of Share 2 [see the final sentence of Step 2 of the algorithm].) For example, to paint the two shares (also called transparencies), the sequence $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5, \dots)$ can be permuted by a random-position generator using a security seed. For instance, it becomes $(x_1, x_2, x_3, y_3, y_2, y_4, x_7, y_1, y_5, x_4, x_8, x_9, \dots)$ after the permutation. Then the last two lines of Step 2 becomes



Transparency 1: $x_1 \quad x_2 \quad x_3 \quad ? \quad ? \quad ? \quad x_7 \quad ? \quad ? \quad x_4 \quad x_8 \quad x_9 \quad \dots,$
 Transparency 2: $? \quad ? \quad ? \quad y_3 \quad y_2 \quad y_4 \quad ? \quad y_1 \quad y_5 \quad ? \quad ? \quad ? \quad \dots$

In the hidden-message extraction phase, the inverse of the random position generator is applied using the same seed in order to reverse the sequence $(x_1, x_2, x_3, y_3, y_2, y_4, x_7, y_1, y_5, x_4, x_8, x_9, \dots)$ back to $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5, \dots)$. Then, as usual, $(x_i + y_i) = d_i \pmod 6$ are utilized to generate (d_1, d_2, d_3, \dots) . Because the super-hacker does not know how the designer distributes the $\{x_i\}$ and $\{y_i\}$, two things can be assured even if both Shares 1 and 2 are intercepted. First, the super-hacker is very unlikely to extract the confidential text d .

Second, the super-hacker is very unlikely to use the information grabbed from the two intercepted shares {Shares 1 and 2} to produce two faked shares {Shares 1' and 2'} which can also pass our authentication test. (In addition, stacking them yield a faked image similar to the "GIRL" image in Fig. 3.6(g).) Of course, if the pixels of the input image Lena are also permuted using a random-position generator, then the image LENA itself is also free from being viewed by the super-hacker intercepting both Shares 1 and 2. But this operation (permuting pixels of Lena) is seldom used unless the image Lena itself is also highly confidential, because this operation reduces the advantages of visual cryptography: the image LENA can no longer be viewed by simply stacking the shares. (Recall that VC is a tool that balances between security and the benefit of "without-a-computer visual decoding" of images. If we only permute the locations of $\{x_i, y_i\}$, the disclosure of the image LENA still does not need a computer; although the disclosure of the confidential text \mathbf{d} will then need a computer.)



3.5 When a decoding-computer is not available

In the above approach, the decoding of the Lena image by VC does not need a computer, but the decoding of the confidential text \mathbf{d} does. Therefore, the method can be used in an environment in which the decoding-computer is not always available. For example, consider a system in which a single decoding-computer serves, say, 100

teams, in turn. Then, each team can decode its own moderately-confidential image immediately (and begin the within-team discussion about the visually-decoded image immediately); although each team leader must wait (until it is his turn) for the computer server to help him to decode the confidential text.

Below, we discuss how to modify the design so that the two new transparencies (generated by the modified design) can decode both Lena and the confidential text without using a decoding-computer.

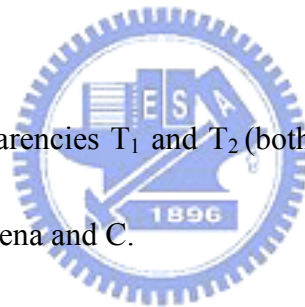
Apparently, if the confidential text can also be visually decoded, then there is no need to use a decoding-computer. In this kind of approach, we have two images to deal with: the Lena image, and its background description image. What we can do is to create two transparencies T_1 and T_2 having the following two properties: 1) stacking T_1 with T_2 yields Lena; 2) after the stacking mentioned in 1, if we fix T_1 , and then shift T_2 by u units horizontally and v units vertically, then the stacking result becomes the confidential text image. Notably, the values of the u and v are kept by the higher-ranking officer. If the higher-ranking officer thinks that it is not easy to shift a transparency by u units horizontally and v units vertically in the decoding phase, then he may use an auxiliary "non-noisy" transparency T_3 prepared earlier, in which only the boundaries of the transparency T_1 and T_2 are sketched.

An encoding algorithm for this modified approach is given below, followed by a

description example and an experiment. In the algorithm, the image C is either a confidential text image describing Lena, or a logo image for authentication purpose. The coordinate (i,j) indicates a pixel at location (i,j) of the image Lena (or the image C), or equivalently, the 2-by-2 block at location (i,j) of the transparency T_1 (or the transparency T_2).

The Modified Encoding Algorithm (No decoding-computer is needed later)

Input: Two natural numbers u and v (u is the Horizontal-shift-amount, and v is the Vertical-shift-amount), an image Lena of size 256-by-256, and a confidential text image C.



Output: two noise-like transparencies T_1 and T_2 (both are of size 512-by-512) useful in visually decoding of both Lena and C.

Steps:

Step 1: For each coordinate (i,j) in the "easier-to-construct" areas, namely, $\{(i,j):0 \leq i < u,$

$0 \leq j < 255\}$ and $\{(i,j): u \leq i < 255, 0 \leq j < v\}$, do the following:

1a. Randomly assign one of the six fundamental block types to $T_1(i,j)$;

1b. If $Lena(i,j)$ is a white pixel, then copy the block type of $T_1(i,j)$ to $T_2(i,j)$;

else copy the complement of the block type of $T_1(i,j)$ to $T_2(i,j)$.

Step 2: (for the remaining area whose construction is a little more complicated):

2a: Initially, let the value of the counter k be 0 .

2b: For each (i,j) satisfying $u \times k \leq i < u \times (k+1)$ and $0 \leq j < 255-v$, do the following:

2b-1: If $C(i,j)$ is a white pixel, then copy the block type of $T_2(i,j)$ to $T_1(i+u,j+v)$;

else copy the complement of the block type of $T_2(i,j)$ to $T_1(i+u,j+v)$.

2b-2: If $Lena(i+u,j+v)$ is a white pixel, then copy the block type of $T_1(i+u,j+v)$

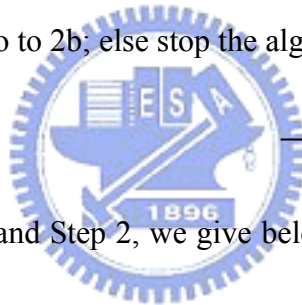
to $T_2(i+u,j+v)$; else copy the complement of the block type of $T_1(i+u,j+v)$

to $T_2(i+u,j+v)$.

2c:

2c-1: Add 1 to the value of k .

2c-2. If $k < (256/u) - 1$, then go to 2b; else stop the algorithm.



—End of the Algorithm—

To help the readers understand Step 2, we give below an example. In the example, assume that $T_1 \oplus T_2$ means stacking T_1 with T_2 , i.e. " \oplus " is the stacking operator.

Example.

To make the explanation easier, we assume that the horizontal-shift-amount is $u=3$, and the vertical-shift-amount is $v = 0$. Without the loss of generality, assume that the pixel values of the input image *Lena* and confidential text image *C* are

Lena: BWBWBBBWWWBW...

Image *C*: WWBWBWBWB...

Because the horizontal-shift-amount is $u=3$, we split Lena and C into sectors of 3 pixels each. Therefore, we have

Lena: BWB WBB BWW WBW...

Image C : WWB WBW BWB ...

Iteration 0a. Initially, because $u=3$, randomly assign the block types to the three blocks of the first sector of T_I ; for example, assign (0,4,3). So we have

Transparency T_I : 0 4 3 ??? ??? ??? ...

Transparency T_2 : ??? ??? ??? ??? ...

Iteration 0b. In order that the first three blocks of the stacking result ($T_I \oplus T_2$) can be (B,W,B), which are the 3 blocks in the first sector of Lena, the first sector of T_2 should be ($\tilde{0}$, 4, $\tilde{3}$), due to the fact that the first sector of T_I is (0,4,3). Here, $\tilde{0}$ is the complement of the block type 0, i.e. $\tilde{0}=1$. (See Fig. 3.3 for understanding; note that stacking block type 0 with block type 1 yields a black block.) Similarly, $\tilde{1}=0$, $\tilde{2}=3$, $\tilde{3}=2$, $\tilde{4}=5$, and $\tilde{5}=4$. Anyway, we now know the first sector of T_I , and the first sector of T_2 . i.e.

T_I : 0 4 3 ??? ??? ??? ...
 ↓ ↓ ↓
 T_2 : 1 4 2 ??? ??? ??? ...

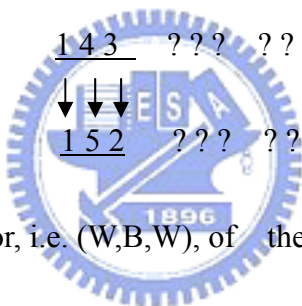
Iteration 1a. In order that the three blocks of the first sector in ($T_I \oplus T_2^{shifted}$) can be

(W,W,B), which are the first 3 blocks of the confidential text image C , let the three blocks of the second sector of T_I be of types (1,4,3). Again, block type 3 is the complement of block type 2 implies $(1,4,3) \oplus (1,4,2)=(W,W,B)$. So we have

$$\begin{array}{ccccccc} \text{Transparency } T_I: & 0 & 4 & 3 & \underline{1} & 4 & 3 & ??? & ??? & \dots \\ & & & & \nearrow & \nearrow & \nearrow & & & \\ \text{Transparency } T_2: & \underline{1} & 4 & 2 & ??? & ??? & ??? & ??? & \dots \end{array}$$

Iteration 1b. Because the second sector of Lena is (W,B,B), we let the second sector of

T_2 be $(1, \tilde{4}, \tilde{3}) = (1,5,2)$. So we have

$$\begin{array}{ccccccc} \text{Transparency } T_I: & 0 & 4 & 3 & \underline{1} & 4 & 3 & ??? & ??? & \dots \\ & & & & \downarrow & \downarrow & \downarrow & & & \\ \text{Transparency } T_2: & 1 & 4 & 2 & \underline{1} & 5 & 2 & ??? & ??? & \dots \end{array}$$


Iteration 2a. Read in next sector, i.e. (W,B,W), of the confidential text image C .

Then, in order that $(T_I \oplus T_2^{shifted})$ can be (W,B,W) at current location, the 7th, 8

th, 9th blocks of T_I should be (1,4,2), since $(1, \tilde{5}, 2) = (1,4,2)$. Now we have

$$\begin{array}{ccccccc} \text{Transparency } T_I: & 0 & 4 & 3 & 1 & 4 & 3 & \underline{1} & 4 & 2 & ??? & \dots \\ & & & & & & & \nearrow & \nearrow & \nearrow & & \\ \text{Transparency } T_2: & 1 & 4 & 2 & \underline{1} & 5 & 2 & ??? & ??? & ??? & \dots \end{array}$$

Iteration 2b. Because the third sector of the input image Lena are (B,W,W), let the


third sector of T_2 be $(\tilde{1}, 4, 2) = (0,4,2)$, so we have

$$\begin{array}{ccccccc} \text{Transparency } T_I: & 0 & 4 & 3 & 1 & 4 & 3 & \underline{1} & 4 & 2 & ??? & \dots \\ & & & & & & & \downarrow & \downarrow & \downarrow & & \\ \text{Transparency } T_2: & 1 & 4 & 2 & 1 & 5 & 2 & \underline{0} & 4 & 2 & ??? & \dots \end{array}$$

Iteration 3a. Read in next sector, i.e. (B,W,B), of the confidential text image C . Then, in order that $(T_1 \oplus T_2^{shifted})$ can be (B,W,B) at current location, the 10th,11th,12th blocks of T_1 should be $(\tilde{0},4,\tilde{2}) = (1,4,3)$, so we have

$$\begin{array}{rcccccccc}
 \text{Transparency } T_1: & 0 & 4 & 3 & & 1 & 4 & 3 & & 1 & 4 & 2 & & \underline{1} & 4 & 3 & & \dots \\
 & & & & & & & & & & & & & & \nearrow & \nearrow & \nearrow & \\
 \text{Transparency } T_2: & 1 & 4 & 2 & & 1 & 5 & 2 & & \underline{0} & 4 & 2 & & & ? & ? & ? & \dots
 \end{array}$$

Iteration 3b. because the fourth sector of the input image Lena are (W,B,W), let the fourth sector of T_2 be $(1,\tilde{4},3) = (1,5,3)$, so we have

$$\begin{array}{rcccccccc}
 \text{Transparency } T_1: & 0 & 4 & 3 & & 1 & 4 & 3 & & 1 & 4 & 2 & & \underline{1} & 4 & 3 & & \dots \\
 & & & & & & & & & & & & & & \downarrow & \downarrow & \downarrow & \\
 \text{Transparency } T_2: & 1 & 4 & 2 & & 1 & 5 & 2 & & 0 & 4 & 2 & & & \underline{1} & 5 & 3 & \dots
 \end{array}$$


Remaining iterations: similar to above iterations.

—End of the example—

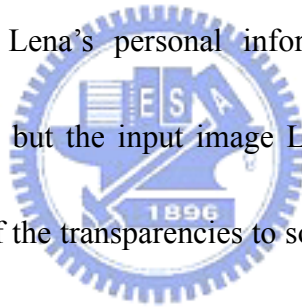
An experiment using the modified encoding algorithm is done here, and the results are shown in Fig. 3.7. No computer is used in the decoding process. Only stacking operation is used to get Fig. 3.7 (c) and (d).

3.6 Summary

In this chapter, we have proposed a two-in-one method that has two decoding levels. The method not only visually shares an input image of moderate

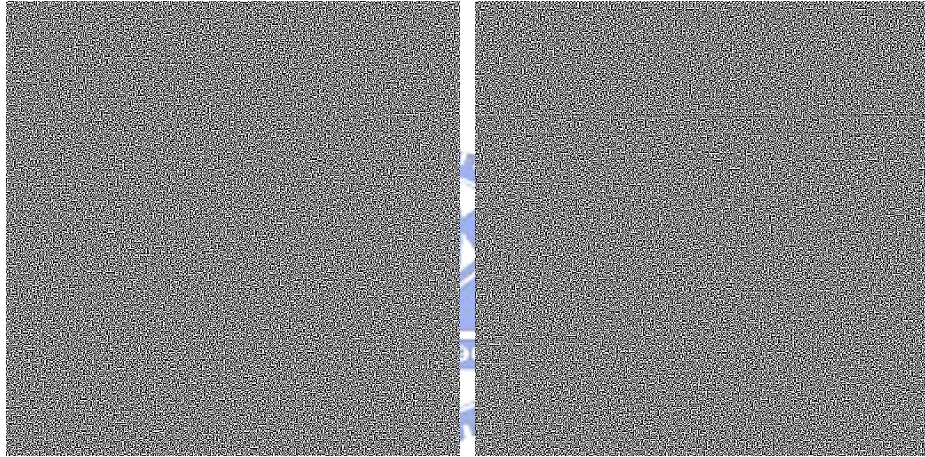
confidentiality, but also hides extra text file which is more confidential. The extra hiding ability is achieved without giving any artificial appearance in the unveiled image. The unveiled LENA is lossless, in the sense that we can sequentially map from the 2-by-2 blocks of LENA back to the "exact" pixel values of Lena. The method can also be applied to the authentication of the transparencies. In this chapter, some other versions with improved security or easier decoding are also discussed.

Notably, in the design introduced in Sec. 3.2, if the more confidential text is too long (for example, if Lena's personal information text needs more than $256 \times 256 / 2$ digits to express, but the input image Lena is only 256×256), then we may enlarge the block size of the transparencies to solve the problem. (When we use 2-by-2 blocks [as we did so far], there are only six types of blocks. By using 3-by-3 blocks, there are many more block types that can be used.)



For possible future works, people may consider the following two topics: 1) If there is no computer available for decoding, then people can still use the proposed visual cryptography method (Sec. 3.6) to handle both Lena image and the confidential text image; however, due to the natural limitation of visual resolution, the hidden text is much shorter (as compared with the data amount carried by the transparencies when a computer is available for decoding). Therefore, it is an

interesting topic to find a solution to increase the length of the text being carried, assuming that no computer can be used for decoding. 2). So far, our method deals with binary images. If we apply the method to the eight bit-planes of gray-value images, there might exist some new applications.



(a)

(b)



(c)

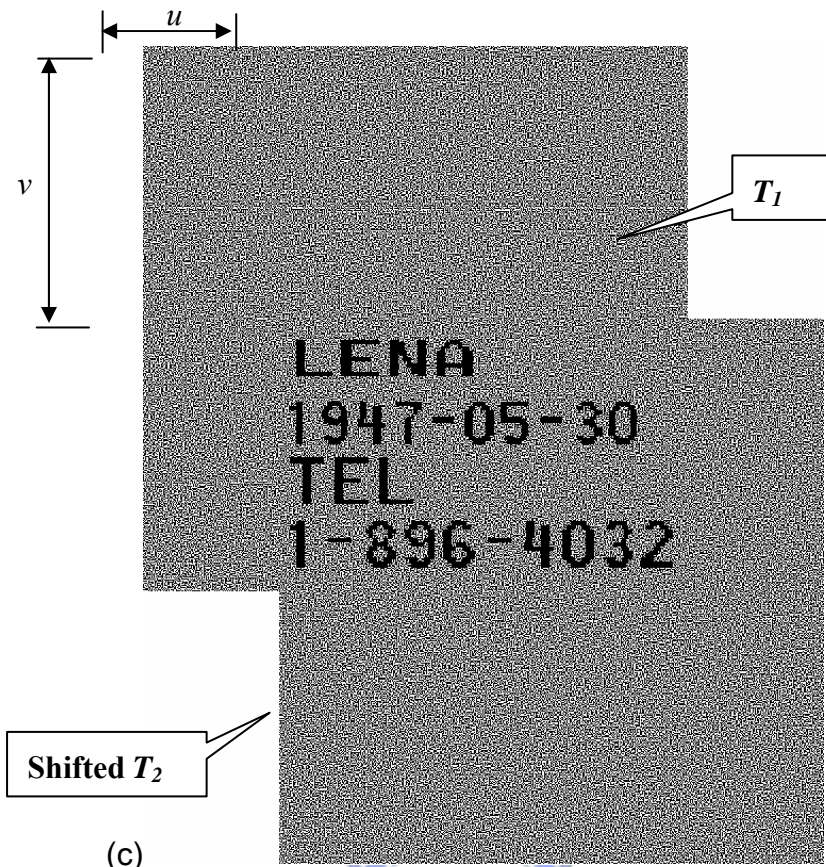


Fig. 3.7. Double-decoding without using a decoding-computer. (a) and (b) are the two transparencies T_1 and T_2 ; (c) is the result of stacking (a) and (b); (d) is the result of stacking T_1 with "shifted T_2 ".

Chapter 4

Universal share for the sharing of multiple images

In an (n, n) image sharing system [TLIEEE2003, TL2002, WTL2004], n shares $\{L_1, L_2, \dots, L_n\}$ are created for a given image, say, Lena. (For simplicity, temporarily assume the image is grey-valued; although our method can be applied to color images, too.) The image can be unveiled when all n shares are received, while less than n shares reveal nothing about the image. With sharing, nobody (including the company's organizer) can unveil the image without attending a public meeting. Therefore, sharing is a safety process useful in a company where no employee/investor alone should be trusted. Notably, the original image can be discarded after the sharing; moreover, each of the n shares is $1/n$ times smaller than the given image. Therefore, the sharing process causes no storage-space waste.

To share another image Monkey (which is grey-valued if Lena is grey-valued), another n shares $\{M_1, M_2, \dots, M_n\}$ will be created likewise. Each employee/partner of a company can thus get a share from each image related to his job/investment. As a result, if a company's organizer gets 1 share from each of the 100 important images being shared, the management of the 100 shares will be a large burden to him. As the number of images increases, the management of the shares becomes more difficult. By

combining the techniques of sharing with that of hiding [LC1999,MB2004, TLPR2003,WLL2001, WL2003], we design here an "universal" share for company's organizer; and he only has to take this special share (single share with compact size) to attend any image's recovery meeting.

Notably, although the method show below deals with grey images; by processing the three color components one by one, our method can also be applied to color images.

4.1. The method

4.1.1 Embedding U in the LSB of the secret image

Let $p \times q$ denote be the standard size of each image to be shared. The company organizer randomly grabs or creates an extra image U of size $p \times q/n$ (all pixel values of U are less than 251, as they are used later in Eq. (4.1)). This image U , whether noisy or not, has $8pq/n$ bits, which are embedded in the $p \times q$ image Lena by the least-significant-bits (LSB) replacement method [LC1999, MB2004, TLIEEE2003, TL2002, TLPR2003, WLL2001]. For instance, if $n \geq 8$, then Image U has at most $p \times q$ bits, so U can be hidden using the least-significant-bits (1 bit per pixel) of Lena, which has $p \times q$ pixels. If $4 \leq n \leq 7$, then U is hidden using the last two bits of Lena's pixels. After embedding, Lena becomes a distorted image called Lena*. Since only some less-important bits of Lena are replaced (assuming $n \geq 4$), the distortion is invisible when comparing Lena* and Lena.

4.1.2 Partitioning each sector

Decompose Lena* into non-overlapping sectors of n pixels each ($8n$ bits per sector).

Then share the $8n$ bits of each sector among the n shares. We assume $n=8$ below; other values of n are handled analogously.

(1.2.i). The LSBs of the $n=8$ pixels of the sector form an 8-bit number, called a_0 . Notably,

$$a_0 < 251 \text{ by Sec. 2.1.}$$

(1.2.ii). The remaining $8 \times 7 = 56$ bits of the sector are then partitioned into another 7

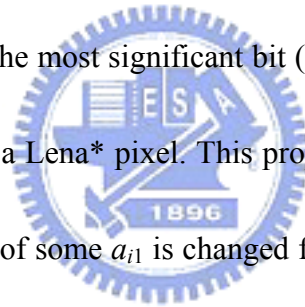
numbers $\{a_1, \dots, a_7\}$ of 8 bits each (see Fig. 4.1); i.e., $\{a_i = (a_{i1}, a_{i2}, \dots, a_{i8}) \mid 1 \leq i \leq 7\}$. For

$1 \leq i \leq 7$, Fig. 4.1 ensures that the most significant bit (MSB) a_{i1} of every $a_i = (a_{i1}, a_{i2}, \dots, a_{i8})$

is the bit next to the LSB of a Lena* pixel. This property avoids visible damage to the

image Lena* if the bit value of some a_{i1} is changed from 1 to 0 so that all a_i stay in the

0–250 range before utilizing Eq. (4.1) (as noted below Eq. (4.1)).



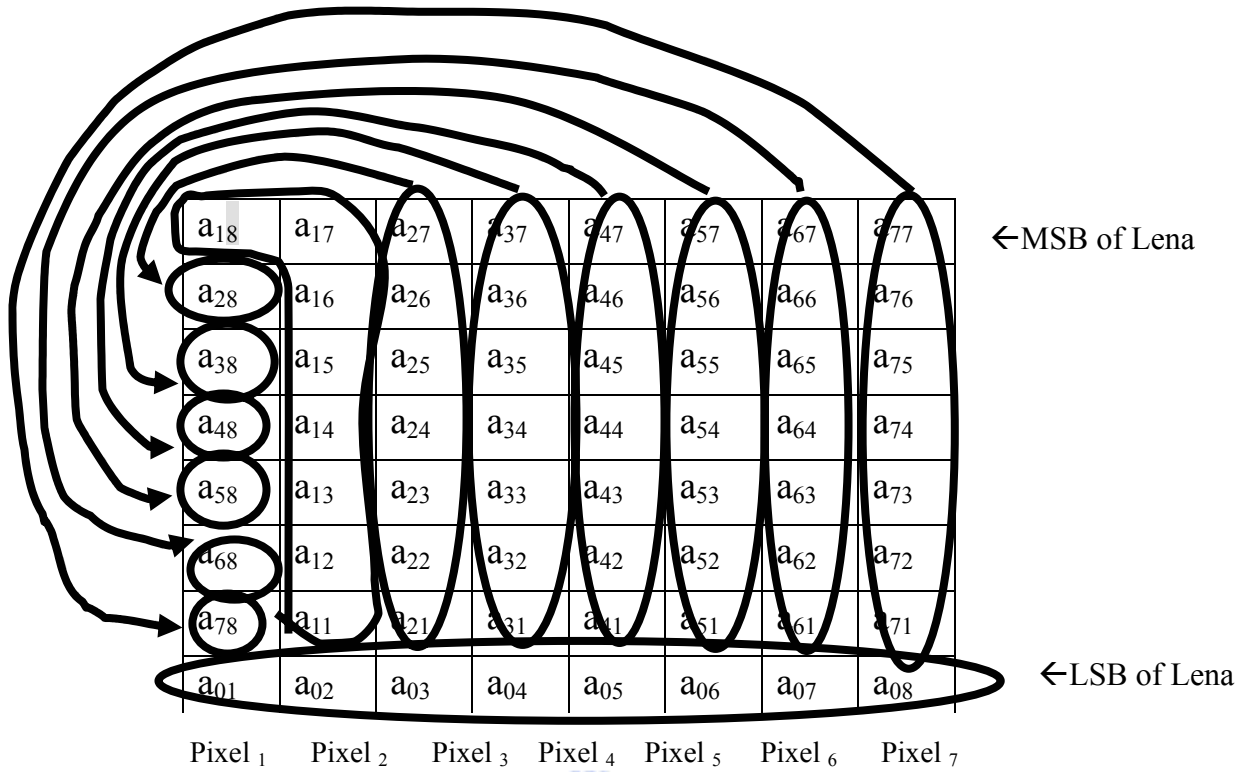


Fig 4.1 The partitioning of a section



4.1.3 Sharing

We already have $\{a_0, a_1, \dots, a_7\}$. Affix to each Share k , where $0 \leq k \leq n-1=7$, a value

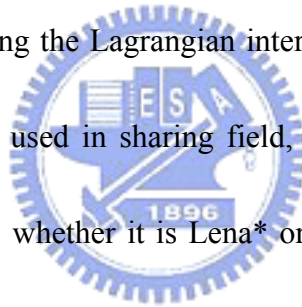
$$f(k) = (a_0 + a_1k + a_2k^2 + \dots + a_{n-1}k^{n-1}) \bmod 251 \quad (4.1)$$

where 251 is the prime number suggested in [TL2002]. In Eq. (4.1), for recovery purpose, each a_i must be in the range 0–250. Therefore, some bits in the sector might need adjustment before (4.1) can be applied. The image after this minor adjustment is still called Lena*. Since each n -pixel sector only contributes a value $f(k)$ to Share k , each share is n times smaller than the image Lena* (and hence Lena). The total size of the n shares is therefore identical to that of Lena; hence, no storage space is wasted.

4.1.4 Using the universal share U

The company organizer keeps Share 0, whose value is $f(0) = a_0$ for each sector. Since the set $\{a_0\}$ is formed of the LSB of image Lena*, it is also formed of image U , because U is embedded in Lena's LSB to obtain Lena* in Sec. 4.1.1. Hence, Share 0 is identical to U , the image created earlier by the organizer. This statement is true even if Lena is replaced by any other image (e.g., Monkey). Share 0, i.e. image U , is thus the desired universal share.

During the recovery phase, when all n shares are collected, recover the values $\{a_0, \dots, a_{n-1}\}$ from $\{f(0), \dots, f(n-1)\}$ using the Lagrangian interpolation polynomials, which is an ordinary and routine procedure used in sharing field, as described in Ref. [WTL2004]. The modified image, regardless whether it is Lena* or Monkey*, can thus be recovered sector-by-sector.



4.2. Experiments

Assume that $n=8$. Fig. 4.2(a) depicts the input image Lena, and the company organizer arbitrarily creates his own share (the top-most noisy image U in Fig 4.2(c), whose pixel values are all below 251). Fig. 4.2(b) is the modified image Lena*, which not only hides the entire image U in its LSB, but also has the property that all a_i extracted from it are in the range 0–250 (see the explanations in Sec. 4.1.1 and 4.1.2). Then, Lena* is shared. Share 0 is the given image U , and the remaining $n-1=7$ shares are generated using

$k=1,2,\dots,n-1$ in Eq. (4.1). All 8 shares are shown in Fig. 4.2(c). These 8 shares can together recover the Lena* displayed in Fig. 4.2(b). Fig. 4.3 shows another experiment in which Lena is replaced by Monkey. Its Share 0 (the top-most noisy image in 4.3(c)) is identical to Share 0 in Fig. 4.2(c), because both are identical to U . Notably, as stated earlier, by processing the three color components one by one, our method can also be applied to color images. In Fig. 4.4, we show the result of using our method on an input color image Lena.

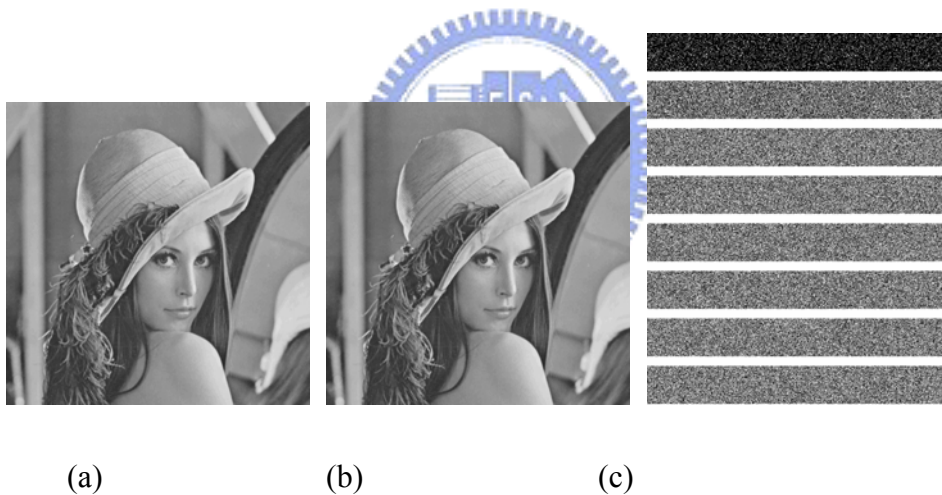


Fig 4.2 A sharing result for $n=8$.

(a) is the important image Lena;

(b) is the modified image Lena*, which contains the image U , and all a_i extracted from it are in 0-250 (see Eq. (4.1));

(c) is the 8 shares that can recover (b) together. The top-most share in (c) is the

universal share, which is identical to the image U.

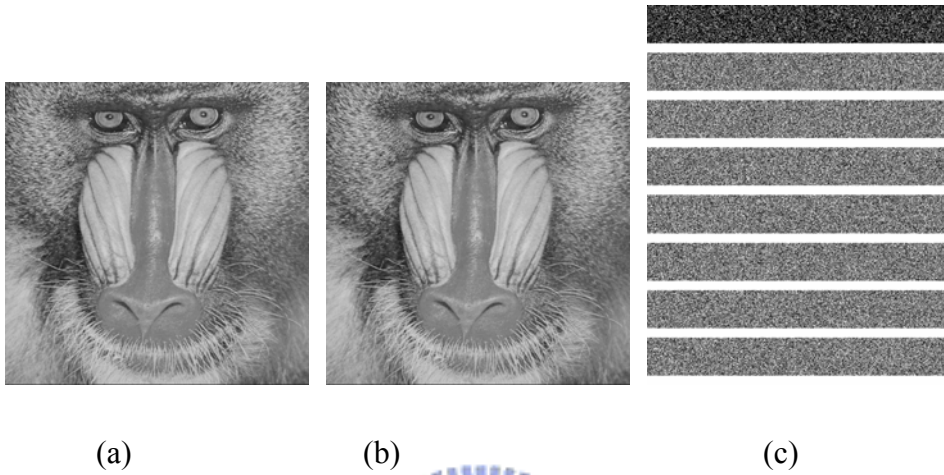


Fig 4.3 Another sharing result for $n=8$.

(a) is the important image Monkey;

(b) is the modified image Monkey*;

(c) is the 8 shares that can recover (b) together; and the top-most share is identical to the top-most share in Fig. 4.2(c).

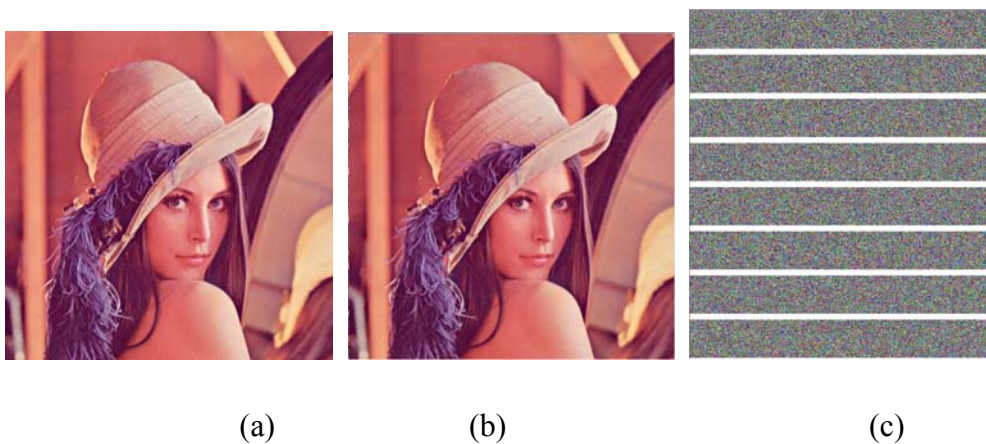


Fig 4.4 An experiment for a color image Lena.

(a) is the important image Lena;

(b) is the modified image Lena*.

(c) is the $n=8$ shares that can recover (b) together. The top-most share in (c) is the universal share, which is identical to the image U.

4.3. Comparison

Below we compare our method with Ref. [TCC2002, FWTC2005] (but not with Ref. [CC2005], because there is no experiment there). Tsai et al [TCC2002] proposed an elegant method using visual cryptography and LSB hiding to deal with multiple secrets images. For $c(4,2)=6$ secret images of size 200×200 each, Ref. [TCC2002] hid the corresponding $6 \times 200 \times 200$ bytes using just 4 stego images (each is of size 600×600 and about 42.5 dB in PSNR). Any two of the four can be combined to extract one of the six secrets. For our universal approach, assuming $n=4$ shares are created for each image. Before hiding the generated shares, our storage space is $200 \times 200 \times ((1/4) \times 1 + (3/4) \times 6) = 4.75 \times 200 \times 200$ bytes where $(1/4) \times 1 = 1/n$ is for the universal share, and $(3/4) \times 6 = ((n-1)/n) \times 6$ is for the non-universal shares of the six secret images. Ours thus saves more space than [TCC2002] does (4.75 : 6, before hiding). If we also hide the generated shares, the total space for the stego images are $4 \times 4.75 \times 200 \times 200 = 19 \times 200 \times 200$

bytes to obtain stego images of PSNRs much better than 42.5 dB; while theirs is $4 \times 600 \times 600 = 36 \times 200 \times 200$ bytes to obtain their 42.5dB stego images. However, their secret image can be recovered losslessly, ours is lossy. In their system, there is no super share (the company's organizer's share), but we have one. So, ours is suitable when the boss of a company wants to control every secret, while [TCC2002] (and [FWTC2005] mentioned below) is suitable for team work in which every teammate is of equal importance.

By summing up the variables used in the sharing polynomials, Feng *et al* proposed in [FWTC2005] another gorgeous sharing method for multiple secret images. They can use, say, 5 shares {a,b,c,d,e}, and unveil secret image 1 using {a,b,c}, unveil secret image 2 using {a,d,e}, etc. In general, before hiding the sharing result, their total size of the sharing result is 1 to 2 times larger than the total size of the input secret images. So, size expansion occurs. To the contrary, ours has size reduction effect, because our total size of the sharing result is even smaller than the total size of the input secret images. More precisely, our size is only $(1 - (1 - S^{-1})/n) \times 100\%$ of the input, where S is the number of secret images, and n is the number of shares for each secret image. So, our benefit is again the economic size. However, just like [TCC2002], the method in [FWTC2005] can get lossless recovery of the secret images.

4.4. Summary

In summary, the proposed sharing method is space-saving and with a convenient

universal share. The advantages are achieved by tolerating an invisible distortion in the recovered images. For instance, the recovered images in Fig. 4.2(b) and 4.3(b) are 52.5 dB in PSNR, when being compared with the original images in Fig. 4.2(a) and 4.3(a). As a remark, our program can be run repeatedly to handle any number of secret images, for instance, 1000 secret images, without the need for reprogramming. Additionally, the universal share \mathbf{U} can be non-noisy, because \mathbf{U} can be any kind of images, including ordinary photos. Hence, only non-universal shares, which always look noisy, need post-processing hiding.



Chapter 5

Fast-decoding sharing: by using bit-level sharing and economic size shares

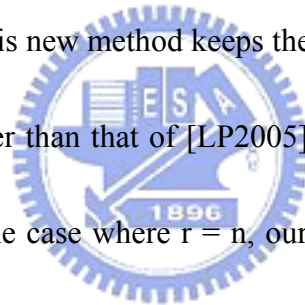
Image sharing can deal with the need of transmitting or storing an image against interceptor. In an (r, n) image sharing system ($r \leq n$), the given image is shared among n shares. Each share is like a random noise image, and some people might hide these noisy shares in other less important images to increase security further. The n shares are then transmitted through (or stored in) n of the many existing channels. If there are 100 channels, then there are n "used" channels, and the remaining $100-n$ channels are just channels not used by this image. To recover the image, any r of the n used channels can cooperate to recover it, but less than r used channels cannot. As a result, sharing among n of the many existing channels can balance between fault-tolerance (up to $n-r$ used channels can be disconnected) and security (up to $r-1$ of the n used channels can be intercepted, not to mention that each of the much-more-than- n channels is usually filled with many other coy or ordinary images not related to the given important image).

To share an image, a possible way is to use polynomial-style sharing (PSS) (see Ref. [TLIEEE2003, TL2002, WTL2004]). Using PSS can get shares of smaller size, but it is "extremely" time-consuming in the decoding phase to recover the image from the r

received shares. There is another way to share an image, as stated below. In Ref. [LP2005], Lukac and Plataniotis successfully applied visual-cryptography (VC) techniques in a bit-level manner (using VC on each bit-planes) and obtain another type of image sharing method having the following good properties:

1. real-time decoding, as opposed to PSS approach;
2. lossless recovery of images (PSS is also lossless).

The only disadvantage in [LP2005] is that each (grey or colour) share is several times bigger than the given image. In the current paper, we propose an alternative method that is also bit-level based. This new method keeps the above advantages of [LP2005], and uses shares of size smaller than that of [LP2005] (thus reduce their transmission time and storage space). In the case where $r = n$, our share-size is even smaller than the size of the input image; while the share-size in [LP2005] is, say, 4 times greater the input image. (As for the (r,n) case (with $r < n$), our share-size is a little smaller than that in [LP2005].)



5.1. The method

For a given $H \times W$ gray-value or color-value image G , and for a given pair of parameters (r,n) , our method to create the n expected shares is as follows (also see Fig. 5.1):

- 1 Physically split the secret image G , whose size is $H \times W$, into two parts: upper

parts and lower parts (see Fig. 5.2). For the given parameter pair (r, n) , the upper part is the first $\lceil n/(n+1) \rceil \times H \times W$ pixels of the input image; while the lower part is the remaining $\lfloor 1/(n+1) \rfloor \times H \times W$ pixels. (So, the upper part is always bigger than the lower part, since $n > 1$.)

2 Notably, there are 8 bit-planes for a grey image (or, 24 bit-planes if color).

Each bit-plane $B = B_U \cup B_L$ also has its own upper part B_U and lower part B_L . The upper part B_U is the first $\lceil n/(n+1) \rceil \times H \times W$ bits of plane B ; while the lower part B_L is the remaining $\lfloor 1/(n+1) \rfloor \times H \times W$ bits.

3. Sequentially pick a not-yet-processed bit-plane $B = B_U \cup B_L$ to process, until all 8 (or 24) bit-planes are processed. Each bit-plane is processed by two sub-steps to generate n binary-value shares $\{S_1, \dots, S_n\}$ for this bit-plane. The two sub-steps are:

(3-i) Share the lower part.

(3-ii) Then, share the upper part.

(The details of (3-i) and (3-ii) are described later in Sec. 5.2.1 for the case $r=n$; and then described again in Sec. 5.2.2 for the case $r < n$.)

4. For each share-index m in $\{1, 2, \dots, n\}$, combine the 8 binary-value shares (or 24 binary-value shares) of the same share-index m (but from 8 or 24 distinct bit-planes) to form a grey (or color) share. Therefore, we get n grey (or color)

shares.

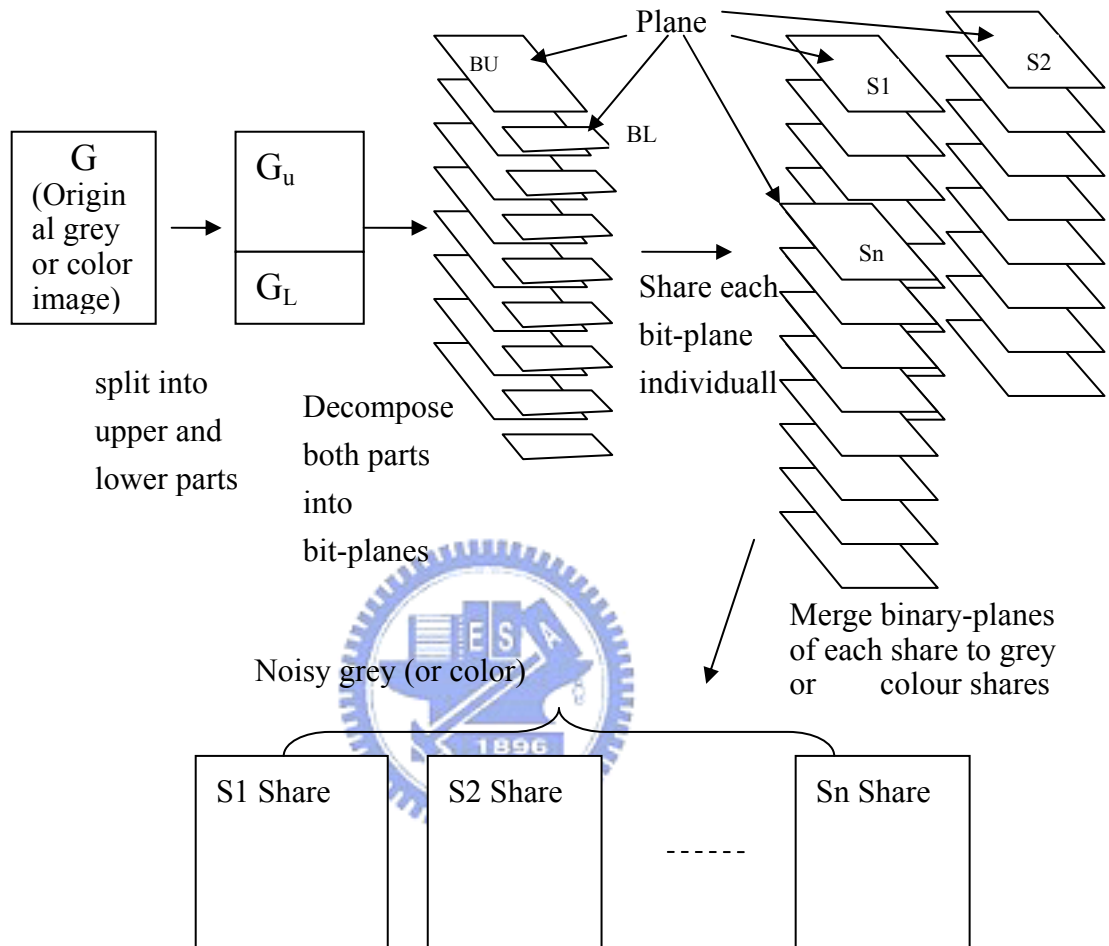


Fig .5.1 Flowchart of the proposed method.

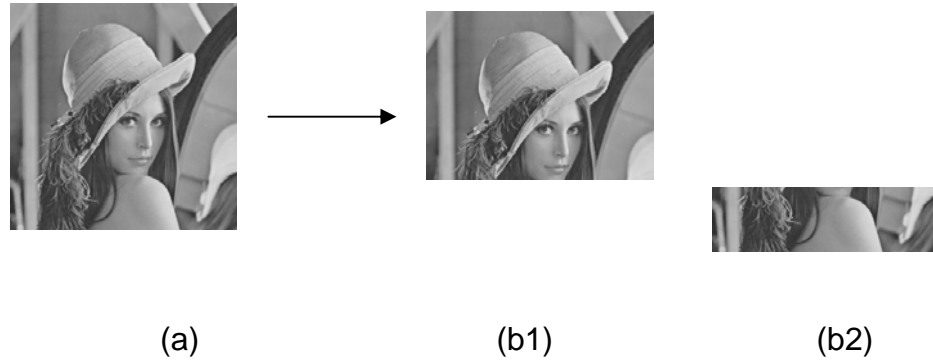


Fig. 5.2. An example of split. (a) is the original, (b1) is the upper part, and (b2) is the lower part.

5.1.1. Sharing a bit plane B in the (n,n) case.

Algorithm (n,n) : Sharing a "bit-plane" B which has H rows and W columns.

INPUT: The $H \times W$ bits of a bit-plane B .

OUTPUT: The n binary-value shares $\{S_1, S_2, \dots, S_n\}$. Each share has $H \times W \times \frac{n}{n+1}$ "bits".

PRE-PROCESSING: Split the bit-plane B into upper part B_U and lower part B_L .

(The first $\frac{n}{n+1}HW$ bits of B are the B_U . The remaining $\frac{1}{n+1}HW$ bits are B_L).

Note that B_U is n times bigger than B_L .

MAIN STEPS:

Step 1. (Sharing the Lower part (B_L) of B .)

For each share S_m , where $m = 1, 2, \dots, n-1$, randomly assign its bits located at $\{n \times i + m$:

$i=0,1,2,\dots\}$. As for the last share, i.e. Share S_n , we do not randomly assign its bits located at $\{n \times i+n; i=0,1,2,\dots\}$. Instead, we use the lower part (B_L) of B to compute the bit-values of the share S_n at location $\{n \times i+n; i=0,1,2,\dots\}$. More specifically, for each $i=0,1,2, \dots$, we compute the value of $S_n(n \times i+n)$ by

$$S_n(n \times i+n) = [S_1(n \times i+1) \oplus S_2(n \times i+2) \oplus \dots \oplus S_{n-1}(n \times i+[n-1])] \oplus B_L(i)$$

where \oplus is the exclusive-OR operator. Note that $S_n(n \times i+n)$ is computed this way because later we can recover the lower part of B by using

$$B_L(i) = S_1(n \times i+1) \oplus S_2(n \times i+2) \oplus \dots \oplus S_{n-1}(n \times i+[n-1]) \oplus S_n(n \times i+n)$$

Step 2. (Sharing the Upper part (B_U) of B .)

Use data B_U (the upper part of B) to determine the remaining bits of all shares. The requirement is very simple: at each position t , we require that

$$B_U(t) = S_1(t) \oplus S_2(t) \oplus \dots \oplus S_n(t).$$

In other words, we only requires that: there are "odd" number of 1s appearing in the n -bits set $\{S_1(t), S_2(t), \dots, S_n(t)\}$ if and only if $B_U(t)=1$. ◆◆

Example (k=3,n=3)

Assume the upper $n/(n+1)=3/(3+1)=3/4$ part of the original bit-plane B is $B_U=1010101\dots$; and assume the lower $1/(n+1)=1/(3+1)=1/4$ part of B is $B_L=\underline{1110}\dots$. Then, we show how to use the above algorithm to produce the $n=3$ shares $\{S_1, S_2, S_3\}$,

where each share has $(3H/4) \times W$ bits when the bit-plane B has $H \times W$ bits.

Step 1. For Share S_1 , randomly assign its bits at $\{1,4,7,\dots\} = \{3i+1: i=0,1,2,\dots\}$.

For Share S_2 , randomly assign its bits at $\{2,5,8,\dots\} = \{3i+2: i=0,1,2,\dots\}$. However,

for the last share, i.e. Share S_3 (because $n=3$), we do not randomly assign its bits at

$\{3,6,9,\dots\} = \{3i+3: i=0,1,2,\dots\}$. Instead, we use the lower part (BL) of B to compute

the values of these bits at location $\{3,6,9,\dots\}$ of Share 3. More specifically, for each

$i=0,1,2,3,4,\dots$, we require that

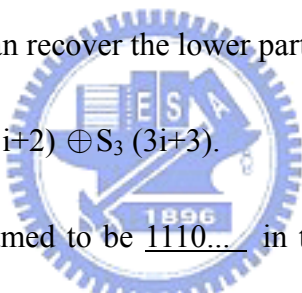
$$S_3(3i+3) = S_1(3i+1) \oplus S_2(3i+2) \oplus B_L(i)$$

for the purpose that later we can recover the lower part of B by using

$$B_L(i) = S_1(3i+1) \oplus S_2(3i+2) \oplus S_3(3i+3).$$

For example, since B_L is assumed to be 1110... in this example, the above idea of

Step 1 can be illustrated by Fig. 5.3.



S_1 (Random values at 1,4,7, ...)	1	?	?	1	?	?	0
S_2 (Random values at 2,5,8, ...)	?	0	?	?	1	?	?
Data B_L (Lower part of B)			<u>1</u>			<u>1</u>	
S_3 (Compute bits 3,6,9,.. by \oplus)	?	?	0	?	?	1	?

Fig. 5.3. Step 1 of the $(k=3,n=3)$ example.

Step 2. Use data B_U (the upper part of B) to determine the remaining bits of all

shares. The requirement is very simple: at each position t , we require that

$$B_U(t) = S_1(t) \oplus S_2(t) \oplus S_3(t).$$

In other words, there are "odd" number of 1s appearing in these three bits $\{S_1(t), S_2(t),$

$S_3(t)\}$ if and only if $B_U(t)=1$. One of the many solutions is shown in Fig. 5.4. Note

that all three shares have been created after Step 2.

S_1		1	?=0	?=1	1	?=0	?=1	0
S_2		?=0	0	?=0	?=1	1	?=0	?=0
S_3		?=0	?=0	0	?=0	?=0	1	?=1
data B_U (upper part of B)		1	0	1	0	1	0	1

Fig. 5.4. Step 2 of the $(k=3, n=3)$ example.

5.1.2. Sharing a bit plane B in the (r, n) case, i.e. when $r < n$.

(r,n) Algorithm (The (r, n) algorithm that shares a bit-plane B which has $H \times W$ bits.)

INPUT: The $H \times W$ bits of a bit-plane B .

OUTPUT: The n shares $\{S_1, S_2, \dots, S_n\}$. Each share has $H \times W \times n / (n+1)$ "blocks"

rather than $H \times W \times n / (n+1)$ "bits".

PRE-PROCESSING: As before, split the bit-plane B into upper part B_U and lower part B_L . (The first $\frac{n}{n+1}$ HW bits of B are the B_U . The remaining $\frac{1}{n+1}$ HW bits are B_L). Again, B_U is n times bigger than B_L .

MAIN STEPS:

Step 0: Create a pair of basis matrices C^0 and C^1 for the (r,n) system. (The creation of C^0 and C^1 is introduced in many other papers, see Ref. [NS1994] for example, we do not introduce the detail here.) No matter how they are created, this pair must have the following properties:

- Each matrix has n rows. Each entry of the two matrices is just a single bit whose value is either 0 or 1; each 1 means a black dot while each 0 means a white dot.
- "Stacking" r of the n rows of C^0 (or C^1) is defined as getting a row whose i^{th} element is the result of using the "OR" operator (not "exclusive-OR") on the i^{th} elements of the corresponding r rows.
- Stacking any r rows of C^0 together always get a row whose number of 1s are less than the number of 1s obtained from stacking any r rows of C^1 .

Step 1 (To share the information of B_L , i.e. the lower part of B).

(1-i) Initially, let $q=1$.

(1-ii) Let p be the q -th bit of the string B_L , i.e. $p=B_L(q)$.

(1-iii) For $m=1,2,\dots,n$, use the m -th row of C^p to paint the $([q-1]n+m)$ -th block of the share S_m ($m=1,2,\dots,n$).

(1-iv). If q reaches $\frac{1}{n+1}H$, then Step 1 ends here, and we go to Step 2. Else, q

$\leftarrow q+1$ and go to (1-ii).

Step 2 (To share the information of B_U , i.e. the upper part of B).

(2-i) Initially, let $q=1$.

(2-ii) Let p be the q -th bit of the string B_U . Also, let $m=q \pmod n$

(2-iii-A) If the m^{th} row of C^p is identical to the q^{th} block of the share S_m , then, for all $k=1,\dots,n$, paint the q^{th} block of the share S_k using the k^{th} row of C^p .

Then go to Step (2-iv).

(2-iii-B) If the m^{th} row of C^p is different from the q^{th} block of the share S_m , then,

we need to permute the columns of C^p to get a temporary matrix $C^{p'}$

whose m^{th} row is identical to the q^{th} block of the share S_m . (This

permutation subroutine is quite easy to design, so we omit it here.) Then,

for all $k=1,\dots,n$, paint the q^{th} block of the share S_k using the k th row of

$C^{p'}$. Then go to Step (2-iv).

(2-iv). If q reaches $\frac{n}{n+1}H$, then go to post-processing. Else, let $q \leftarrow q+1$ and go

to (2-ii).

POST-PROCESSING: We already have n shares, and each share is a sequence of

$\frac{n}{n+1}H \times W$ blocks. Now, convert each sequence of blocks from 1-dim block-string to

its 2-dim image version. In other words, for each share, divide its $\frac{n}{n+1}H \times W$ blocks

into $\frac{n}{n+1}H$ rows. (The first W blocks form the first row; the next W blocks form

the second row; etc.)



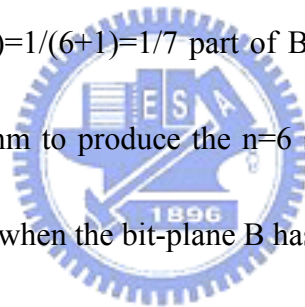
Example (k=2,n=6)

Assume the upper $n/(n+1)=6/(6+1)=6/7$ part of the original bit-plane B is $B_U=1001\dots$;

and assume the lower $1/(n+1)=1/(6+1)=1/7$ part of B is $B_L=1011\dots$. Then, we show

how to use the above algorithm to produce the $n=6$ shares $\{S_1, \dots, S_6\}$, where each

share has $(6H/7) \times W$ "blocks" when the bit-plane B has $H \times W$ bits.



Step 0. Create a pair of basis matrices C^0 and C^1 for the (2,6) system. Since $n=6$, each

matrix has 6 rows. The two matrices in Fig. 5 obviously meet the requirements stated

in the Step 0 of the algorithm.

C^0 :

1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

C^1 :

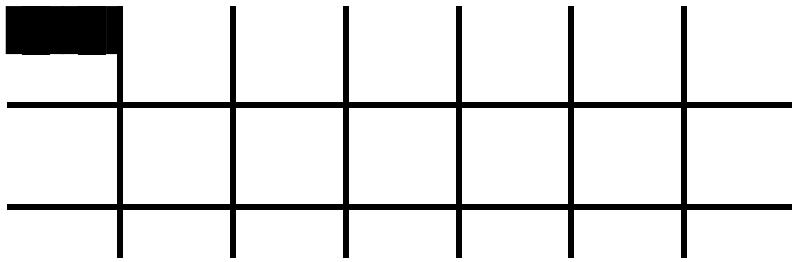
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1

Fig. 5.5. The basis matrices C^0 and C^1 used for the (2,6) system in the example.

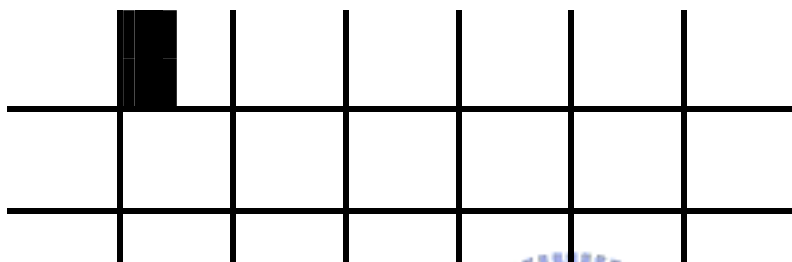
Step 1: (To share $B_L=1011\dots$, i.e. to share the lower part of B.)

See Fig. 5.6 to understand Step 1. Because the 1st bit of B_L is 1, We look up the matrix C^1 . Then, use the 1st row of C^1 , i.e. use 1100 to paint 1st block of share S_1 . (Since 1100 means BBWW, we may use the first two entries (BB) to paint the upper half of the block, then use the next two entries (WW) to paint the lower half of the block.) Analogously, the share S_2 uses 1010 (the 2nd row of C^1) to paint its 2nd block. The share S_3 uses 1001 (the 3rd row of C^1) to paint its 3rd block. The share S_4 uses 0110 (the 4th row of C^1) to paint its 4th block. The share S_5 uses 0101 (the 5th row of C^1) to paint its 5th block. The share S_6 uses 0011 (the 6th row of C^1) to paint its 6th block. Then the cycle repeats itself again using next bit of B_L . Since $B_L(2)=0$, we use C^0 now. The rows 1-6 of C^0 are copied respectively (one row per share), to 7th block of S_1 , 8th block of S_2 , 9th block of S_3 , 10th block of S_4 , 11th block of S_5 , and 12th block of S_6 . This is Cycle 2. Third cycle uses $B_L(3)=1$ to grab C^1 to paint the $(12+i)$ th block of the share S_i ($i=1,2,..6$). The process repeats again and again until all bits of B_L are used.

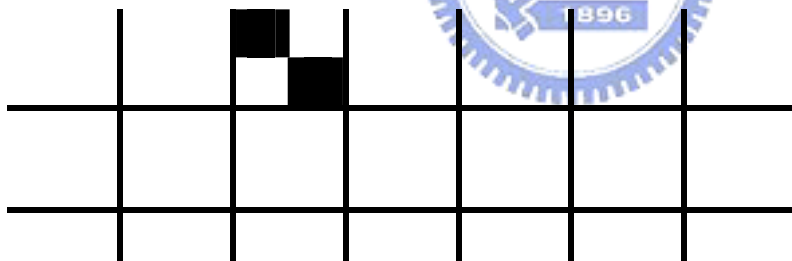




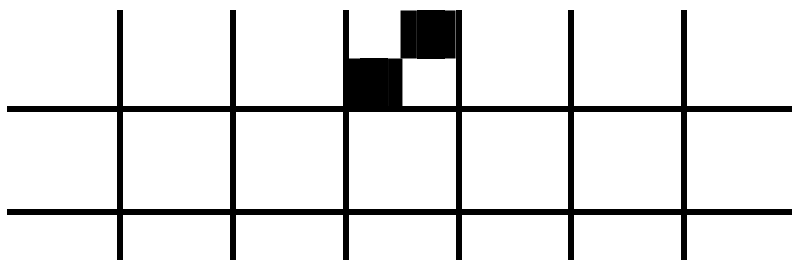
(a)



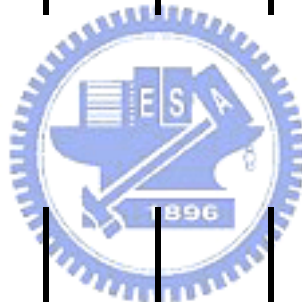
(b)

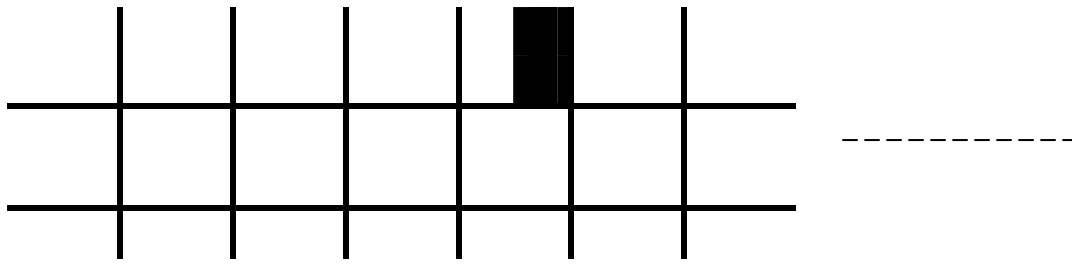


(c)

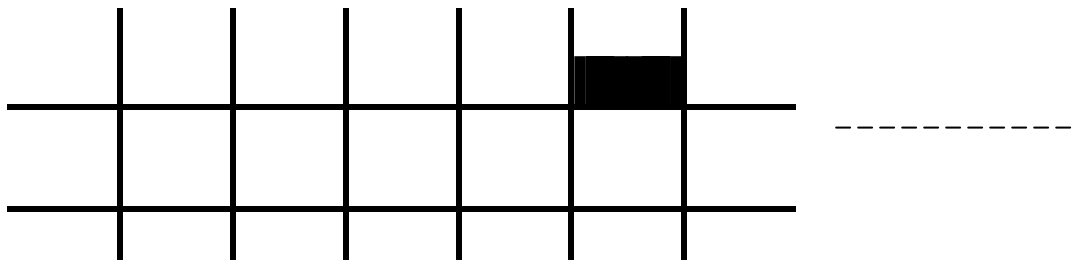


(d)



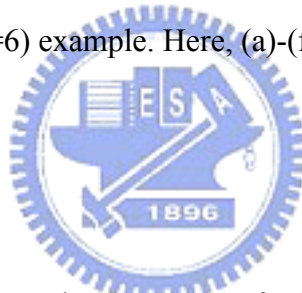


(e)



(f)

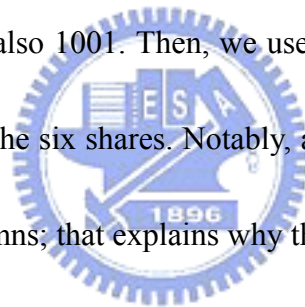
Fig. 5.6 Step 1 for the ($r=2, n=6$) example. Here, (a)-(f) are, respectively, the six shares $S_1 - S_6$.

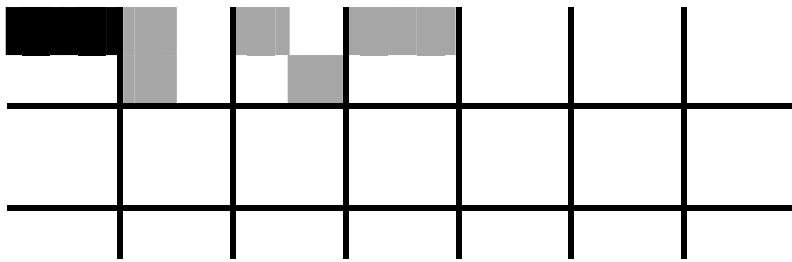


Step 2: (To share B_U , i.e. to share the upper part of B.)

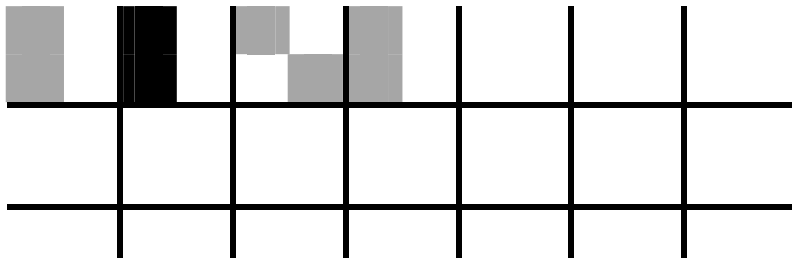
See Fig. 5.7. to understand Step 2. Note that the upper part of B is 1001...; so, we look up first C^1 , then C^0 , then C^0 again, then C^1 again, and so on. First, because $B_U(1) = 1$, we use the six rows of C^1 to paint the 1st blocks of shares $S_1 - S_6$. In this painting, the painted pattern of the 1st block of the 1st Share (S_1) has no contradiction with what it had been painted earlier in Step 1. (The block had been painted earlier in Step 1 as 1100 (i.e. BBWW), so, no contradiction if we use the 1st row of C^1 to paint it.) Therefore, the 1st blocks of all six shares are done using the six rows of C^1 . Now we

proceed to $B_U(2)$. Since $B_U(2)=0$, we use the six rows of C^0 to paint the 2nd blocks of shares S_1 - S_6 . Again, this painting is accepted because the painted pattern of the 2nd block of the 2nd share (S_2) has no contradiction with what it had been painted earlier in Step 1. Now we proceed to $B_U(3)$. Because $B_U(3)=0$, we also try to use the six rows of C^0 to paint the 3rd blocks for Shares S_1 - S_6 . However, we find that this will cause the 3rd block of the 3rd share (S_3) has contradiction with what it is already painted in Step 1 earlier. (The block had been painted in Step 1 as 1001 (i.e. BWWB), rather than 1010 (i.e. BWBW)). Therefore, we permute the columns of C^0 to get a temporary matrix $C^{0'}$ whose 3rd row is also 1001. Then, we use the six rows of the new matrix $C^{0'}$ to paint the 3rd blocks of the six shares. Notably, all six rows of $C^{0'}$ become 1001 after this permutation of columns; that explains why the 3rd blocks of all six shares are painted as 1001 (BWWB) in Fig. 7. Now, we proceed to the 4th bit of B_U and find that $B_U(4)=1$, so we use the six rows of C^1 to paint the 4th blocks for Shares S_1 - S_6 . In the painting, the painted pattern of the 4th block of the 4th Share (S_4) has no contradiction with what it had been painted in Step 1 earlier. (The block had been painted earlier in Step 1 as 0110 (i.e. WBBW), so, no contradiction if we use 4th row of C^1 to paint it.) Therefore, the 4th blocks of the six shares are done using the six rows of C^1 . Our explanation ends here, for the remaining process are similar.

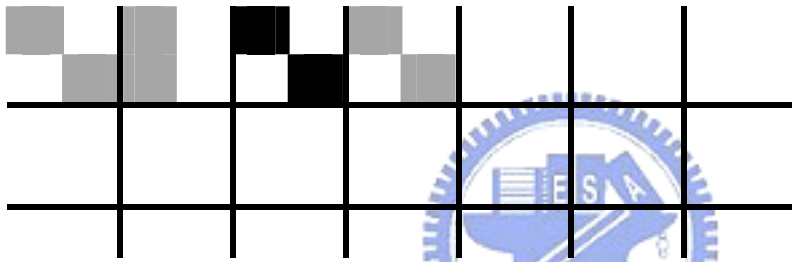




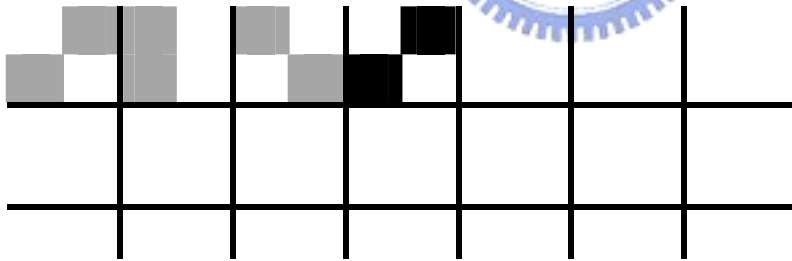
(a)



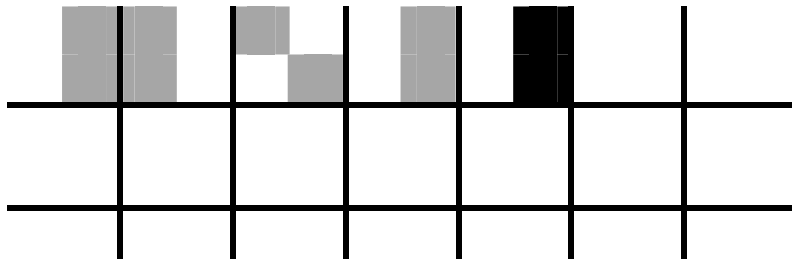
(b)



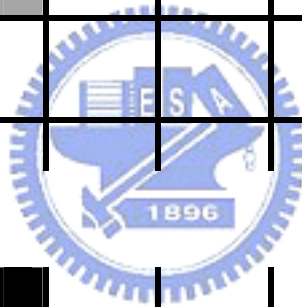
(c)

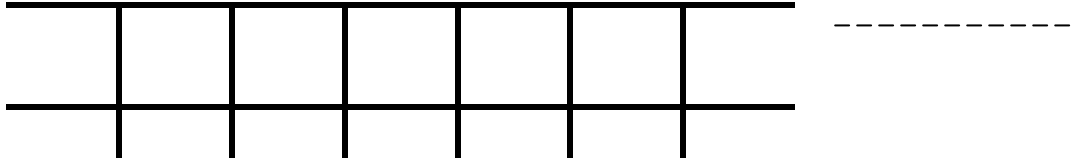


(d)



(e)





(f)

Fig. 5.7 Step 2 for the $(r=2, n=6)$ example. Here, (a)-(f) are, respectively, the six shares $S_1 - S_6$. Darker elements were determined earlier in Step 1, and hence cannot be changed now in Step 2.

5.2. Experiments

In the first experiment, the (r,n) is $(2,2)$. The result is shown in Fig 5.8, of which (a) is the input gray-value image Lena; (b) and (c) are the two gray-value shares (the size of each share is just $n/(n+1)=2/3$ of that of (a)); (d) is the restored error-free result using (b) and (c). The result can be compared with Fig. 5.9, which is a result appeared in Ref. [LP2005]. In Fig. 5.9, their recovery is also error-free (just like ours), but each of their shares is 4 times larger than the input image, while each of our shares is $n/(n+1)=2/3$ times smaller than the input image. In other words, their share-size is $(2 \times 2) \times (3/2) = 6$ times bigger than ours if the input image is the same.

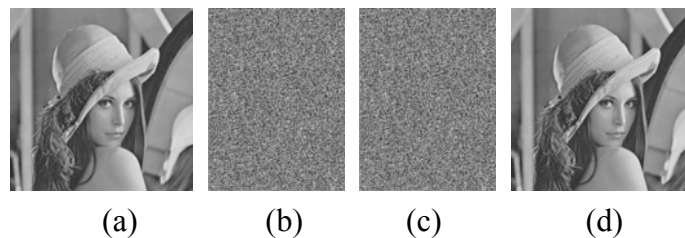


Fig. 5.8. An $(r=2,n=2)$ experiment using our method. (a) is the input gray-value image; (b) and (c) are the two gray-value shares (the size of each share is just $n/(n+1)=2/3$ of that of (a)); (d) is the restored error-free result (identical to (a)) using (b) and (c).

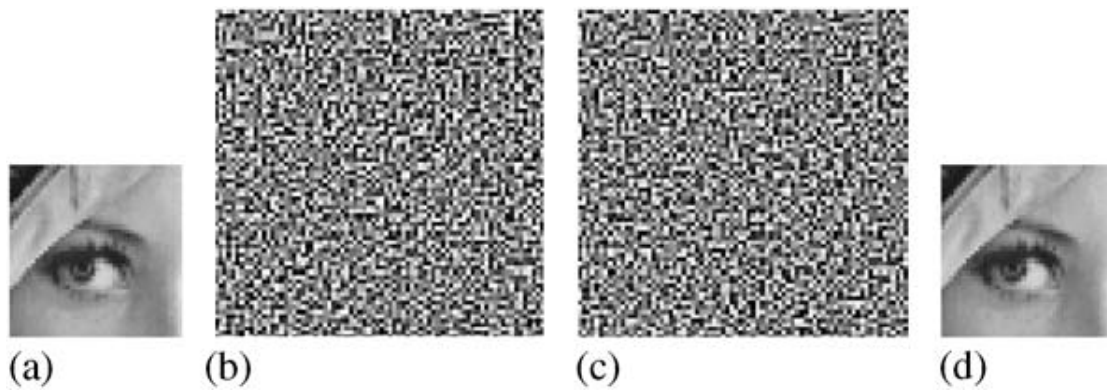
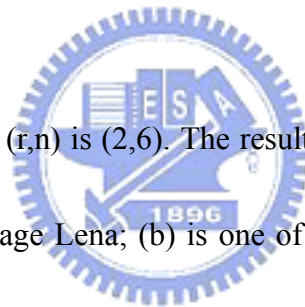


Fig. 5.9. An $(r=2, n=2)$ experimental result in Ref. [LP2005]. (a) is the input grey-value image; (b-c) are the two grey-value shares (each share is $2 \times 2 = 4$ times bigger than (a)); (d) is the restored error-free result (identical to (a)) using (b) and (c).



In the second experiment, the (r,n) is $(2,6)$. The result is shown in Fig 5.10, of which (a) is the input gray-value image Lena; (b) is one of the six gray-value shares (each share is $(n/(n+1)) \times (2 \times 2) = (6/7) \times (2 \times 2) = 3.43$ times greater than (a)). Using any two of the six shares, we can get the error-free recovery of the input image (identical to (a)). The result can be compared with Fig. 5.11, which is a result appeared in Ref. [LP2005]. In Fig. 5.11, their recovery is also error-free (just like ours), and each of their shares is 4 times lager than the input image, while each of our shares is $(2 \times 2) \times 6/7 = 3.43$ times larger than the input image. Therefore, their share-size is $(n+1)/n = 7/6$ times bigger than ours if the input image is the same.

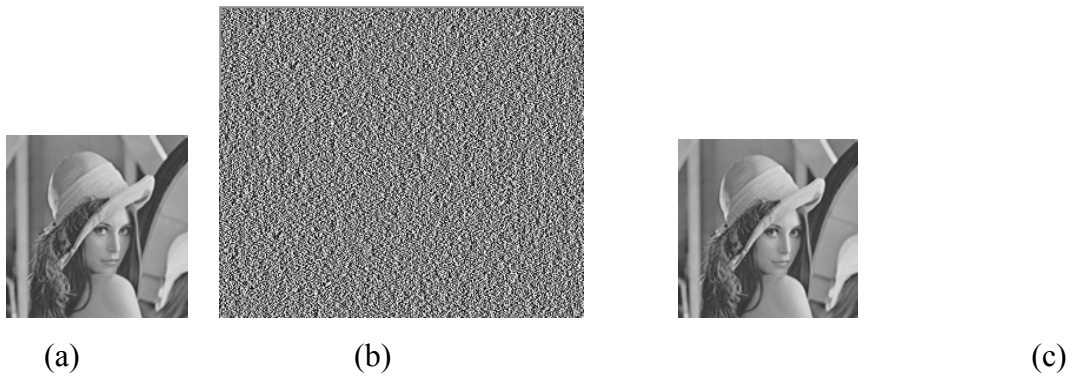


Fig. 5.10. An $(r=2, n=6)$ experiment using our method. (a) is the input gray-value image; (b) is one of the six gray-value shares (each share is $(n/(n+1)) \times (2 \times 2) = (2 \times 2) \times 6/7 = 3.43$ times greater than (a)); (c) is the restored error-free result (identical to (a)) using any two of the six shares

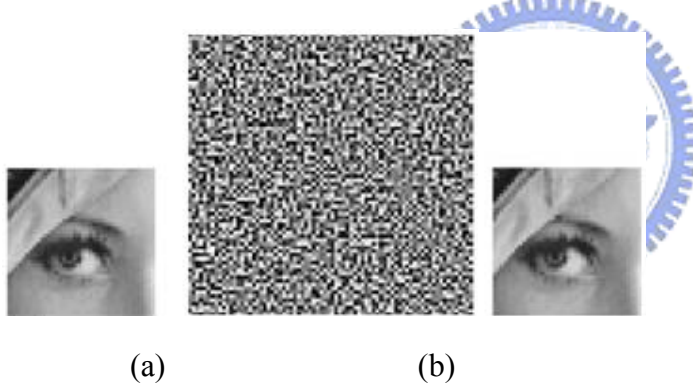


Fig. 5.11. An $(r=2, n=6)$ experimental result in Ref. [LP2005]. (a) is the input grey-value image; (b) is one of the six grey-value shares (each share is $2 \times 2 = 4$ times bigger than (a)); (c) is the restored error-free result (identical to (a)) using any two of the six shares.

In general, in the (n,n) cases, e.g. the $(2,2)$, or $(3,3)$, or $(4,4)$ cases, each of their shares is $2 \times 2 \times (n+1)/n$ times larger than ours. Note that $2 \times 2 \times (n+1)/n$ is a number

between 4 and 6. We can therefore save more transmission time or storage space than the method in [LP2005] does. On the other hand, in the (r,n) cases, then each of their shares is $(n+1/n)$ times larger than ours. Note that $(n+1)/n$ is a number between 1 and $4/3=1.33$ (for $n>2$ in the (r,n) case, because r cannot be 1.) Of course, as compared with [LP2005], we still have a little advantage of space-saving or communication-time-saving in the (r,n) cases, although the advantage is not as sharp as in the (n,n) cases.

Table 5.1 provides a look of the processing time. The computer being used is a Pentium IV PC, and the image being shared is a 256×256 gray-value image. The unit used is millisecond (0.001 seconds per unit; therefore, 3.7 means 0.0037 seconds). When it is (r,n) case, our performance is similar to [LP2005], no matter it is encoding or decoding. But, when it is (n,n) case, our method is obviously faster than [LP2005], no matter it is encoding or decoding. The reason is that, in the (n,n) case, we did not use any kind of blocks (for example, the 2×2 blocks) to expand any pixel. For readers' interest, we also list the processing time of the polynomial-style-sharing (PSS, see [TL2002]). Its decoding time is definitely much slower than bit-level methods (ours and [LP2005]). As for its encoding time, PSS beats us and [LP2005] in the (r,n) case, but lose to us and [LP2005] in the (n,n) case. The reason is that in the (n,n) case our system is extremely simple.

In summary, in the (n,n) case, we lead both [LP2005] and PSS [TLIEEE2003], no matter it is encoding or decoding. In the (r,n) case, PSS takes the lead in encoding, but falls far behind ours and [LP2005] in decoding.

Table 5.1. Comparison of the processing time; the unit is "millisecond". ([LP2005] did not give the detail about (3,3) case, so we did not list data here, although we know that the time for (3,3) is longer than that in the (2,2) case.)

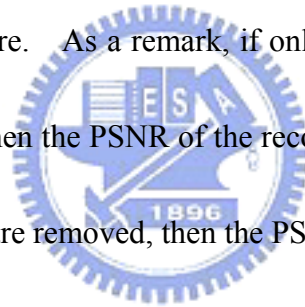
	encoding	encoding	encoding	decoding	decoding	decoding
(r,n) (or (n,n))	(2,2)	(3,3)	(2,6)	(2,2)	(3,3)	(2,6)
Ours	11.7	11.8	234.2	3.7	4.4	13.1
[LP 2005]	243.7	N/A	243.7	17.4	N/A	17.4
Polynomial	34.3	182.9	115.5	185.5	418.8	206.3

5.3. Summary

In this chapter, we have proposed a sharing method for grey or color images. The decoding speed, just like the one in [LP2005], is real-time. The recovered image is lossless; so is [LP2005]. But our method uses shares of size smaller than those used in [LP2005], and hence, transmission time and storage space can be saved. This advantage is particularly obvious in the (n,n) systems, i.e. when r=n. In that case, each grey/colour share is 4 to 6 times smaller than that used in [LP2005], and the

processing speed (no matter it is encoding or decoding) is also much faster than in [LP2005].

Both [LP2005] and our method are bit-level based; and each bit-plane is processed independently. Therefore, if the transmission time (or storage space) is too limited, people may discard some less important bit-planes. For example, discard the last 2 bit-planes and only use the most important $8-2=6$ bit-planes, then each grey-value share is actually a physical-combination of six bit-plane shares rather than a combination of eight bit-plane shares. Therefore, each share is reduced in size to $6/8$ of the original grey-value share. As a remark, if only the least important one of the eight bit-planes is removed, then the PSNR of the reconstructed image is about 52 db. If two least important planes are removed, then the PSNR is at least 39 db.



Chapter 6

Conclusions and Future works

6.1 Conclusions

Four new types of image sharing have been proposed in this dissertation, and they are: turnover, two-level, universalizing, and fast decoding. The first two types can both be used in the authentication of the shares, or used in the background description of the secret image being shared. The universalizing sharing is for easy management of the shares, and the fast-decoding sharing is for the real-time reconstruction of the shared secret images.



In the turnover style, for any two given secret images, two corresponding noisy transparencies are produced. If we stack the front view of both transparencies, then we can see the first secret image. On the other hand, if we stack the front view of Transparency 1 with the back view (the turnover) of Transparency 2, then the second secret image is unveiled. We have also analyzed why 3-by-3 extension, rather than 2-by-2, is needed in turn-over design.

In the two-level style, we have presented a two-in-one visual cryptography scheme, which not only shares an image of moderate confidentiality between two noisy transparencies, but also hides in these two transparencies a more confidential

text file, which is either the information for authentication purpose or the information describing the image. The method can therefore be applied to the authentication of the transparencies. Notably, the decoding is of two levels. In level 1, we stack the two transparencies without any shift, and we can see the secret image. In level 2, we shift Transparency 1 to a predefined amount before stacking with Transparency 2, and we can see some other information of more confidential level in the shifted stacking. We have also provided another version in which the decoding in Level 2 uses a computer. The information carried in Level 2 can be more complicated in this version.

In the universalizing style, we have designed a so-called universal share. This share can be either noisy or just an ordinary image. A company's organizer can hold this special share to attend any unveiling meeting of any secret image shared in his company. No matter how many secret images are shared in his company, the organizer only have to hold this special share, rather than thousands or millions of shares. This reduces the burden of the organizer. Also, the same program can be run repeatedly to handle any number of secret images, for instance, 1000 secret images, without the need for reprogramming.

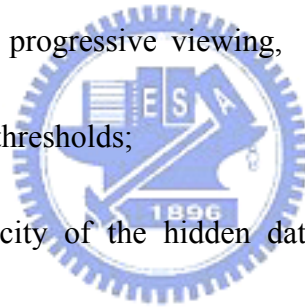
In the fast-decoding style, we have used bit-plane decomposition to design a scheme to share gray-value secret images. The decoding speed is fast. The method is particularly useful if the threshold r in the (r, n) sharing scheme equals n . In this case,

as shown in Table 5.1, not only the decoding speed, but also the encoding speed, is extremely fast. Its share size is also better than a fast-decoding method [LP2005] reported recently. For example, in the case $(r; n) = (2, 2)$, the size of each share is just $n/(n+1) = 2/3$ of the input image, while each share in [LP2005] is 4 times larger than the input image.

6.2 Future works

In the future, we aim to add some features to the proposed methods, as follows:

1. Try to add the feature of fault tolerance to the turnover style and two-level style.
2. Try to add the ability of progressive viewing, when the number of collected shares reach a predefined thresholds;
3. Try to increase the capacity of the hidden data disclosed in Level 2 of the two-level sharing.



References

[ABSS1996]G. Ateniese, C. Blundo, A. De Santis, and D.R. Stinson, "Visual cryptography for general access structures", *Inform. Comput.* Vol.129, pp. 86-106,1996.

[ABSS2001]G. Ateniese, C. Blundo, A. De Santis, D.R. Stinson, "Extended schemes for visual cryptography, "*Theoretical Computer Science*, Vol. 250, pp.137-161, 2001.

[Anderson1996] R.J. Anderson, "Information hiding: 1st International Workshop," *Lecture Notes in Computer Science*, Vol. 1174. Berlin, Germany: Springer-Verlag, 1996.

[BBGHPP20000]W. Bender, W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, and S. Pogreb, "Applications for data hiding," *IBM System Journal*, Vol. 39(3-4), pp. 547-568, 2000.

[BBP2002]. A. Benazza-Benyahia and J.C. Pesquet, "A unifying framework for lossless and progressive image coding," *Pattern Recognition*, Vol. 35, pp. 627-638, 2002.

[BD2003]N. Bourbakis and A. Dollas, "SCAN-based compression-encryption-hiding for video on demand," *IEEE Multimedia Magazine*, Vol. 10, pp. 79-87, 2003.

[Blakley1979]. G.R. Blakley, "Safeguarding cryptographic keys," *Proceedings AFIPS 1979 National Computer Conference*, Vol. 48, pp. 313-317, New York, USA, June, 1979.

[BS1998]C. Blundo, A. de Santis, "Visual cryptography schemes with perfect reconstruction of black pixels, " *Computers & Graphics*, Vol. 22, pp. 449-455 ,1998.

[BSS1999]C. Blundo, A. De Santis, D.R. Stinson, " On the contrast in visual cryptography schemes " . *J. Cryptology*, Vol. 12, pp. 261-289, 1999.

- [BS2001]. A. De Bonis and A. De Santis, "Randomness in secret sharing and visual cryptography schemes," *Theoretical Computer Science*, Vol. 314, pp. 351-374, 2001.
- [BSN2002]C. Blundo, A. D. Santis, M. Naor, "Visual cryptography for gray level images," *Information Processing Letters*, 75, pp. 255-259 ,2002.
- [CAM 2000] B. Carlo, D. S. Alfredo and N. Moni, "Visual cryptography for grey level images," *Information Processing Letters*, Vol. 75(6), pp. 255-259, 2000.
- [CC2002]C.C. Chang and J.C. Chuang, " An image intellectual property protection scheme for gray-level images using visual secret sharing strategy " , *Pattern Recognition Letters*, Vol. 23, pp. 931-941 ,2002.
- [CC2004]C.K. Chan and L.M. Cheng, "Hiding data in images by simple LSB substitution", *Pattern Recognition*, Vol. 37(3), pp. 469-474 ,2004.
- [CC2005] C.W. Chan, and C.C. Chang, "A scheme for threshold multi-secret sharing," *Applied Mathematics and Computation*, Vol.166, No.1, pp. 1-14, 2005.
- [CCL2004]. C.C. Chang, G.M. Chen, M.H. Lin, "Information hiding based on search-order coding for VQ indices," *Pattern Recognition Letters*, Vol. 25, pp. 1253-1261, 2004.
- [CH1998]. C. C. Chang and R. J. Hwang, "Sharing secret images using shadow codebooks," *Information Sciences*, Vol. 111, pp. 335-345, 1998.
- [CJC1998]. C.C. Chang, J.J. Jau and T.S. Chen, "A fast reconstruction method for transmitting image progressively," *IEEE Transactions on Consumer Electronics*, Vol. 44(4), pp. 1225-1233, 1998.
- [CT2001]. K.L. Chung and S.Y. Tseng, "New progressive image transmission based on quadtree and sharing approach with resolution control," *Pattern Recognition Letters*, Vol. 22, pp. 545-1555, 2001.
- [CTC1999]C.C. Chang, C.S. Tsai and T.S. Chen, "A technique for sharing a secret color image", *Proceedings of the Ninth National Conference on Information Security*,

Taichung, May 1999, pp. LXIII-VLXXII.

[EMSMSZ2003]. A.M. Eftekhari-Moghadam, J. Shanbehzadeh, F. Mahmoudi, H. Soltanian-Zadeh, "Image retrieval based on index compressed vector quantization," *Pattern Recognition*, Vol. 36, pp. 2635-2647, 2003.

[FL2006]W.P. Fang and J.C. Lin, "Visual cryptography with extra ability of hiding confidential data", *J. Electronics Imaging*, Vol. 15,4, online, 2006.

[FWTC2005]J.B. Feng, H.C. Wu, C.S. Tsai and Y.P. Chu, , "A new multi-secret images sharing scheme using Lagrange's interpolation", *J. System & Software*, Vol.76, pp. 327-339, 2005.

[HCL1999]Y.C. Hou, C.Y. Chang and F. Lin, "Visual cryptography for color images based on color decomposition", *Proceedings of the Fifth Conference on Information Management*, Taipei, pp. 584-591, November 1999.

[HC2001] R. J. Hwang and C. C. Chang, "Hiding a picture in two pictures," *Optical Engineering*, Vol. 40, pp. 342-351, 2001.

[HCL2004] H. C. Hsu, T. S. Chen and Y. H. Lin, "The Ringed Shadow Image Technology of Visual Cryptography by Applying Diverse Rotating Angles to hide the Secret Sharing," *Proc. of the 2004 IEEE ICNSC*, pp. 996-1001, 2004.

[HKS2000]T.Hofmeister, M. Kruse, and H. U. Simon, "Contrast-optimal k out of n secret sharing scheme in visual cryptography," *Theoretical Computer Science*, 240, pp. 471-485, 2000.

[HLC1999]Y.C. Hou, F. Lin, C.Y. Chang, "Improvement and implementation of the secret color image sharing technique", *Proceedings of the Fifth Conference on Information Management*, Taipei, pp. 592-597, November 1999.

[HLC1999]Y.C. Hou, F. Lin and C.Y. Chang, "A new approach on 256 color secret image sharing technique", *MIS Review*, No. 9, December 1999, pp. 89-105.

[Hou2003]Y.C. Hou, " Visual cryptography for color images " , *Pattern Recognition*,

36, pp. 1619-1629, 2003.

[Jamil1999]T. Jamil, "Steganography: the art of hiding information in plain sight," *IEEE Potentials*, Vol. 18, No. 1, 1999.

[LC1999] W.N.Lie and L.C. Chang, "Data hiding in images with adaptive numbers of least significant bits based on the human visual system," *International Conf. on Image Processing*, Kobe, Japan, Oct., Vol. 4, pp. 286-290, 1999.

[LP2005]R.Lukac and K.N. Plataniotis, "Bit-level based secret sharing for image encryption," *Pattern Recognition* , Vol. 38, pp. 767-772, 2005.

[LT2003]C.C. Lin and W.H. Tsai, "Visual cryptography for gray-level images by dithering techniques," *Pattern Recognition Letters*, 24, 349-358, 2003.

[MB2004]S.S. Maniccam and N. Bourbakis, "Lossless compression and information hiding in images," *Pattern Recognition*, Vol. 37(3), pp. 475-486,2004.

[MBR1999]L.M. Marvel, C.G. Boncelet, and C.T. Retter, "Spread spectrum image steganography," *IEEE Transactions on Image Processing*, Vol.8(8), pp.1075-1083 ,1999.

[MR1998]L.M. Marvel and C.T. Retter, "Hiding information in images," *International Conference on Image Processing*, Boston, USA. Vol. 2, pp. 396-398, 1998,

[NP1997]M. Naor and B. Pinkas, " Visual authentication and identification " , in: B. Kaliski, Jr. (Ed.), *Advances in Cryptology --CRYPTO 97, Lecture Notes in Computer Science*, Vol. 1294, Springer, Berlin, pp. 322-336, 1997.

[NS1995]M. Naor and A. Shamir, "Visual cryptography," *Advances in Cryptology --- Eurocrypt 94, Lecture Notes in Computer Science*, Vol. 950, Springer-Verlag, Berlin, pp. 1-12, 1995.

[NS1996]M. Naor and A. Shamir, in: M. Lomas (Ed.), "Visual Cryptography, II: Improving the Contrast via the Cover Base", Presented at Security in Communication

Networks, AmalE, Italy, September 16-17, 1996. *Lecture Notes in Computer Science*, Vol. 1189, Springer, Berlin, pp. 197-202, 1997.

[PAK1999]F.A.P. Petitcolas, R.J. Anderson, M.G. Kuhn, "Information hiding – a survey," *Proceedings of the IEEE*, Vol. 87(7), pp. 1062-1078, July 1999.

[RP1996]V. Rijmen and B. Preneel, "Efficient colour visual encryption for shared colors of Benetton", Eurocrypt'96, Rump Session, Berlin, 1996.

[Rubin1996]A.D. Rubin, "Independent one-time passwords", *Comput. Systems*, Vol. 9, pp. 15-27, 1996.

[Shamir1979]. A. Shamir, "How to share a secret," *Communication of the ACM*, Vol. 22(11), pp. 612-613, 1979.

[Shamir1998]A. Shamir, "Visual cryptanalysis", *Proceedings of the Eurocrypt '98*, Espoo, pp.201-210, 1998.

[Stinson1997]D.R. Stinson, "An introduction to visual cryptography", presented at *Public Key Solutions '97*, Toronto, Canada, April 1997.

[TCC2002]C.S. Tsai, C.C. Chang and T.S. Chen, "Sharing Multiple Secrets in Digital Images," *J. System & Software*, Vol. 64, No. 2, pp. 163-170, 2002.

[TCP2002]Y.C. Tseng, Y.Y. Chen and H.K. Pan, "A secure data hiding scheme for binary images", *IEEE-T-Communications*, Vol. 50(8), pp.1227-1231, 2002.

[TL2002] C.C. Thien and J.C. Lin, "Secret image sharing," *Computers & Graphics*, Vol. 26, pp. 765-770, 2002.

[TLIEEE2003]. C.C. Thien and J.C. Lin, "An Image-Sharing Method With User-Friendly Shadow Images," *IEEE-T. Circ. & Syst. Video Tech.*, Vol. 13, No. 12, pp. 1161-1169, 2003.

[TLPR2003]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(12), pp. 2875-2881, 2003.

[WC2005]. H. C. Wu and C. C. Chang, "Sharing visual multi-secrets using circle shares," *Computer Standards & Interfaces*, Vol. 28, pp. 123-135, 2005.

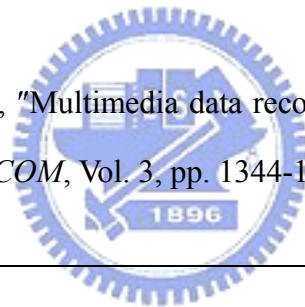
[WLL2001]R.Z. Wang, C.F. Lin and J.C. Lin, , "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern Recognition* Vol.34, No.3, pp. 671-683, 2001.

[WL2003]M. Wu and B. Liu, "Data hiding in image and video .I. Fundamental issues and solutions," *IEEE-T- Image Processing*, pp. 685- 695,2003.

[WTL2004]Y.S. Wu, C.C. Thien and J.C. Lin, "Sharing and hiding secret images with size constraint," *Pattern Recognition*, Vol. 37, No.7, pp. 1377-1385 , 2004.

[YC2005] C. N. Yang and T. S. Chen, "Aspect ratio invariant visual secret sharing schemes with minimum pixel expansion," *Pattern Recognition Letters*, Vol. 26(2), pp. 193-206, 2005.

[YY2000]H.H. Yu and P. Yin, "Multimedia data recovery using information hiding," *Proceedings of IEEE GLOBECOM*, Vol. 3, pp. 1344-1348, 2000.



Vita

Wen-Pinn Fang received his BS degree in mechanical engineering in 1993 from National Sun-Yet-Sen University and his MS degree in mechanical engineering in 1998 from National Chiao Tung University. Since 1999 he has been studying toward a PhD degree in the Computer Science Department of National Chiao Tung University. His recent research interests include visual cryptography and data hiding.



Publication List of Wen-Pinn Fang

A. Journal papers (published/accepted)

1. Wen-Pinn Fang and Ja-Chen Lin, "Visual cryptography with extra ability of hiding confidential data" *Journal of Electronic Imaging*, Vol. 15(2), 2006.
2. Wen-Pinn Fang and Ja-Chen Lin, "Multi-channel Secret Image Transmission with Fast Decoding: by using Bit-level Sharing and Economic-size Shares" *International Journal of Computer and Network Security*, Vol. 6(6), pp. 228-234, 2006.
3. Wen-Pinn Fang, "A Data Hiding Method for Multi-layer Protection Using Sensitive Transformation", *WSEAS Transactions on Information Science and Application*, accepted and to appear.
4. Wen-Pinn Fang, "A Multiple-Shots Method to Overcome Metering Problem for Ordinary Digital Cameras" *International Journal on Graphics, Vision and Image Processing*, accepted and to appear.
5. Chih-Ching Thein, Wen-Pinn Fang and Ja-Chen Lin, "Sharing Secret images by using base-transform and Small-size host images" *International Journal of Computer Science and Network Security*, accepted and to appear.
6. Wen-Pinn Fang and Ja-Chen Lin, "Progressive viewing and sharing of sensitive images", *Pattern Recognition and Image Analysis*, accepted and to appear.

B. Conference papers (published/accepted)

7. Wen-Pinn Fang and Ja-Chen Lin, "Adjustable inverse halftoning," *Computer Vision, Graphics & Image Processing Conference*, Hwu-Len, Taiwan, 2004 August.
8. Wen-Pinn Fang and Ja-Chen Lin, "A content protection method for video-secret sharing approach" *Computer Vision, Graphics & Image Processing Conference*, Taipei, Taiwan, 2005 August.
9. Wen-Pinn Fang and Ja-Chen Lin, "Progressive transmission of vector-quantized images with security and fault-tolerance" *International Conference on Signal Processing, Computational Geometry & Artificial Vision (ISCGA'06)*, Crete Island, Greece, 2006 August 18-20.