

Chapter 1

Introduction

The motivation for this dissertation 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

Traditionally, in progressive viewing of an image, the information contained in the image is partitioned into several parts. The most significant part is viewed or transmitted first, while the least significant part is done last. If the most significant part is damaged or lost, the recovered image will be significantly degraded. In most researches [CT 01, BBP 02, CJC 98] about progressive reconstruction on one end of the transmission channel, the receiver can immediately stop the transmission of the image, if the reconstructed rough version of the image shows that the image being reconstructed (for example, a jet airplane) is not the one the receiver is interested (for example, Lena). Notably, these methods are not fault-tolerant or unbiased: some parts are more important than the others, and the important parts cannot be lost. To develop a new progressive viewing system that is fault-tolerant and unbiased, using sharing technique might be one of the possible solutions.

Therefore, a progressive image viewing method using sharing is introduced in

Chapter 2. In that method, an user can set several thresholds, namely, the k thresholds:

$r_1 \leq r_2 \leq \dots \leq r_k = r$. If less than r_1 shared results are present, nothing can be revealed.

However, if r_1 shared results are present; a rough version of the image can be revealed.

Of course, for each $s > 1$, if r_s shared results are received, the quality of the recovered

image is, at least, not worse than the one using r_{s-1} shared results. Finally, if r_k shared

results are received; the image can be recovered losslessly.

When lossless recovery of the images is not so critical, we can reduce the size of each share, and hence reduce the transmission time or storage space, by developing a

vector-quantized version that is still progressive. In this version, each shared result is

much smaller than the one generated in Chapter 2, although receiving r_k or more

shared results can only reveals lossy image. More specifically, a progressive method

of vector-quantized images is proposed in Chapter 3. In the proposed $(r_1, r_2, \dots,$

$r_k)$ -thresholds method, when the generated n shares are stored or transmitted using n

distinct channels, the interception of up to r_1-1 channels will not let the thief has a

chance to peep a clue of the vector-quantized image; and the vector-quantized image

can be viewed progressively as long as the numbers of received shares reach some of

the preset threshold values $\{r_1, r_2, \dots, r_k\}$, where $r_1 \leq r_2 \leq \dots \leq r_k$. Finally, if the

number of shares that survive in an attack is not less than the given threshold value r_k ,

then the vector-quantized image recovered by our system is exactly the one recovered

by traditional VQ.

Chapters 2 and 3 are both based on the secret sharing scheme proposed by Blakley [Blakley 01] and Shamir [Shamir 02]. Another famous sharing scheme is Visual Cryptography (VC). When a computer is not available, VC is an alternative solution for sharing a secret image. The secret image shared by VC scheme can be decoded instantly by stacking the generated transparencies together. In Chapter 4, we will propose a VC scheme for progressive viewing of a secret image. Our VC scheme is a weighted version. The quality obtained from stacking two lower-weight transparencies is worse than the quality from stacking two higher-weight transparencies. Moreover, stacking additional transparency on the stacked transparencies always yields better quality.



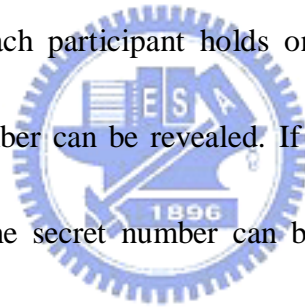
In Chapter 5, the proposed VC-based system is modified to share multiple secret images. The transparencies T_0, T_1, \dots, T_k with weights w_0, w_1, \dots, w_k are generated to encode the secret images S_1, \dots, S_k . Stacking T_i and T_j ($j < i$), the secret image S_i is revealed. Therefore, T_0 can be held by the most important person who can view all secret images S_1, \dots, S_k by stacking T_0 on T_1, \dots, T_k , respectively. This property makes the tool applicable to the management of multiple secret images in a company. Another application is the so-called “game cards”. When a player plays the game, he can get game card T_0 initially. When he passes the test at stage i ($i > 0$), he is granted

the game card T_i . The player can view the secret image S_i by stacking T_i and T_j together (j is the number corresponds to an earlier stage).

1.2 Background Review

1.2.1 Secret image sharing

Blakley [Blakley 01] and Shamir [Shamir 02] first proposed the idea of secret sharing, known as the (r, n) threshold scheme. In their (r, n) threshold scheme, the system is formed of a dealer and n participants, and the dealer distributes a secret number into n shares and each participant holds one share. Later, if r shares are received, then the secret number can be revealed. If less than r shares are received, then no information about the secret number can be revealed. This secret sharing scheme is fault-tolerant in the sense that $n-r$ shares can be lost during the reconstruction (because only r shares are needed).



Thien and Lin [TL 03] applied the idea of Ref.[Shamir 02] to share a secret image and to generate n shadow images. The size of shadow image is only $1/r$ of that of the original secret image. The secret image can be recovered if any r of the n shadow images are received. Because of the smaller size (the $1/r$ size) property of the shadow images, the total transmission time needed to recover an image (by receiving r shadow images) will not increase.

The method to share a secret image is based on Shamir's [Shamir 02] polynomial threshold scheme with a prime number p . The image is divided into several non-overlapped sectors, and each sector has r pixels. For each sector j , the r coefficients a_0, a_1, \dots, a_{r-1} of the corresponding polynomial

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

are assigned as the r gray values of the r pixels in the sector. The w -th shadow image is the collection $\left\{ q_j(w) \mid j = 1, 2, \dots, \frac{\text{original image size}}{r} \right\}$. Because each sector j , which has r pixels, contributes only one pixel $q_j(w)$ to the w -th shadow image, the size of the w -th shadow image is only $1/r$ of the secret image. This property holds for every $w \in \{1, 2, 3, \dots, p-1\}$.



1.2.2 Vector quantization

Vector Quantization [Gray 84, NK 88, Mielikainen 02, LBG 80, MM02, EMSMSZ 03, CCL 04] is a technique for compressing digital images. In the encoding phase of VQ, a given image is divided into several blocks, and then each block is mapped to its closest codeword chosen from a given codebook. The indices of the codewords are kept in a file to record the mapping sequence. In the decoding phase of VQ, the indices are used for finding the corresponding codewords to reconstruct the given image.

1.2.3 Visual cryptography

Visual cryptography (VC) [NS 95, HKS 00, YC05, BSS 96, CAM 00, LT 03, HC 01, WC 05, HCL 04, Hou 03] is a kind of secret image sharing scheme that decodes a given secret image by human visual system without any computation. In the encoding phase of VC, n transparencies, also called shares, are generated. In the decoding phase of VC, the secret image is revealed by stacking the shares together. Naor and Shamir [NS 95] introduced the so-called (r, n) -threshold visual cryptography. Stacking at least r of the n generated shares can decode the secret completely, but stacking less than r shares gives no information about the secret.

Now we review the definition of “contrast” first. In VC, each pixel of the secret image is extended to a block for each of the n generated shares. Let $p \times q$ be the block size. (Therefore, each share will be $p \times q$ times bigger than the secret image in size.) In the stacking result, if at least d_b black elements exist in each black block, and if at most d_w black elements exist in each white block, then the contrast of the stacking result was defined in Ref. [HKS 00] as $\frac{d_b - d_w}{p \times q}$.

1.2.4. Multi-secret image sharing and access structure.

Multi-secret images sharing system is an interesting research area for fault-tolerantly protecting many secret images. Wu and Chang [WC 05] embedded two secret images into two circle transparencies, called shares. If two shares are stacked together, the content of the first secret image can be revealed. Moreover,

rotating one of the two shares by a given angle degree, the other secret image can be revealed then. Their method is more flexibility than traditional VC scheme. However, so far, only two secret images can be applied by their method.

Tsai et al [TCC 02] and Feng et al [FWTC 05] proposed sharing methods for multiple images. In [TCC 02], they adopted XOR computing for embedding and extracting the secret images, and in [FWTC 05], they adopted Lagrange's interpolation that is also applied in Thien and Lin [TL 02] for generating several shares in order to fault-tolerantly recover the single secret image. The method of [TCC 02] and [FWTC 05] should use computer to extracting the secret images. Although the method in [TCC 02] and [FWTC 05] is very convenient in network environment, but when in war, a computer may be not easy for use; the multi-secret image sharing must have other approaches, like VC. Moreover; in both [TCC 02] and [FWTC 05], when some particular share is lost or damaged, more than one secret images can not be revealed forever. In our method, each share is lost or damaged, at most one secret image can not be revealed forever.

Visual Cryptography for general access structure was introduced in [ABSS 96, ABSS 01, BS 01]. Stacking each qualified subset of transparencies together can reveal the secret image, but stacking other, forbidden, sets of transparencies together has no information on the secret image. In [ABSS 96, ABSS 01, BS 01], only single secret

image is applied. Notably, in [FWTC 05], their method can be applied on general access structure; however, as mentioned above, their method is not for VC, thus, can not reveal secret images by stacking the shares together.

1.3 Overview of the proposed methods

We briefly describe below each of the proposed methods.

1.3.1 Progressive viewing of images: a sharing approach

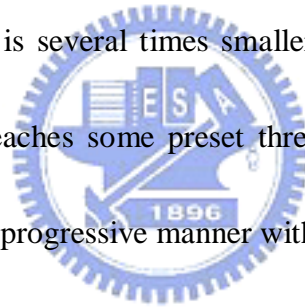
In our system, the original image is rearranged and then shared by an $((r_1, r_2, \dots, r_k), n)$ threshold scheme. The n shared results are then hidden into n host images to form n stego images. When r_k stego images are collected ($r_k < n$), the image can be recovered losslessly. Therefore, the loss of $n - r_k$ stego images will not affect the lossless recovery. On the other hand, if less than r_k , but not less than r_1 , stego images are received, people can still view the rough version of the original image in a progressive manner. Therefore the advantages of the proposed scheme are as follows:

(1). Unlike most progressive image recovery methods, the original image is divided into several parts of equal importance. Therefore, there is no need to worry about which part is lost or received first. (2). If the original image is a secret image, i.e. if security is of big concern, then the image can be transmitted using n distinct channels (one shared result per channel). Then the interception of some channels by the enemy (up to r_1-1 channels) will not reveal the secret. On the other hand, the disconnection

of some channels (up to $n-r_k$ channels) will not affect the lossless recovery of the secret image.

1.3.2 Progressive viewing of vector-quantized images: a sharing approach

This chapter introduces a technique of progressive viewing of vector-quantized images. The $((r_1, r_2, \dots, r_k), n)$ -threshold sharing scheme is applied to the index file of a given quantized image and n shares are generated. The loss or damage of up to $n-r_k$ shares does not affect at all the quality of the image reconstructed by VQ. The n shares can be stored or transmitted using up to n distinct channels to increase the survival rate, and each share is several times smaller than the index file. When the number of received shares reaches some preset threshold values (r_1, r_2, \dots, r_k) , the VQ-image can be viewed in a progressive manner with fault-tolerance.



1.3.3 Progressive viewing of a secret image : a VC approach

A VC approach for progressive viewing of images is designed in this chapter. n weighted transparencies $\{t_1, t_2, \dots, t_n\}$, also called shares, are generated to share a secret image. No single share can reveal any information about the secret image. However, when we stack at least two shares, the secret image is revealed to an extent; and the revealed contrast level is $(\text{SUM}\{weights\}-\text{MAX}\{weights\})/(p \times q)$. (Here, both SUM and MAX operators are evaluated using the set of the shares being stacked, which is a subset of $\{t_1, t_2, \dots, t_n\}$, and $p \times q$ means each share is $p \times q$ times greater than

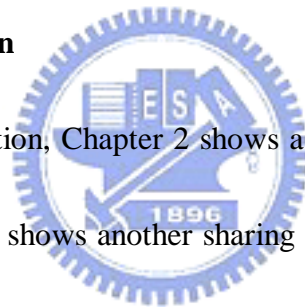
the secret image in size.)

1.3.4 Progressive viewing of multiple secret images: a VC approach

A VC-based system for sharing multiple secret images is proposed. Several weighted transparencies are generated so that people can reveal multiple secret images by stacking some transparencies together. The transparency with larger weight decides which secret image will be revealed. The proposed method has the following characteristics: decoding without computation, multiple secret images recovery, max-weight dominance, and fault-tolerance.

1.4 Dissertation Organization

In the rest of this dissertation, Chapter 2 shows a sharing method for progressive viewing of images. Chapter 3 shows another sharing method for progressive viewing of vector-quantized images. Chapters 4 and 5, respectively, propose the Visual Cryptograph systems for progressive viewing of single secret image and multiple secret images. Chapter 6 summarizes the relations between Chapters 2-5. Finally, the conclusions and suggestions for future works are in Chapter 7.



Chapter 2

Progressive viewing of images: a sharing approach

In this chapter, a sharing approach of progressive viewing of images is proposed. The user can set several thresholds, namely, the k thresholds: $r_1 \leq r_2 \leq \dots \leq r_k = r$. If less than r_1 shared results are received, nothing can be revealed. However, if r_1 shared results are received; a rough version of the original image can be revealed. Then, for each $s > 1$, if r_s shared results are received, the quality of the recovered image is better than or equal to the one using r_{s-1} shared results. Finally, if r_k shared results are received; the image can be recovered losslessly.

Notably, if the original image is an important (secret) image, then, since the content of each shared result always looks noisy, an attack from hackers is more likely. Therefore, a data-hiding method [WTL 04] is utilized to hide the shared results in some host images to form stego images which look ordinary instead of being noisy, and hence avoid attracting the hackers' attention.[BD 03, PAK 99, BBGHPP 00]

Sec. 2.1 shows the encoding phase; and Sec. 2.2 shows the decoding phase. Experimental results are in Sec. 2.3. Sec. 2.4 discusses the design to control quality. Finally, the conclusions are stated in Sec. 2.5.

2.1 Encoding

The scheme is illustrated in Fig. 2.1. Firstly, a bit-plane scanning method is

adopted to rearrange the gray value information of the original image. Then, the rearranged data is shared. Finally, the shared results are hidden into some host images.

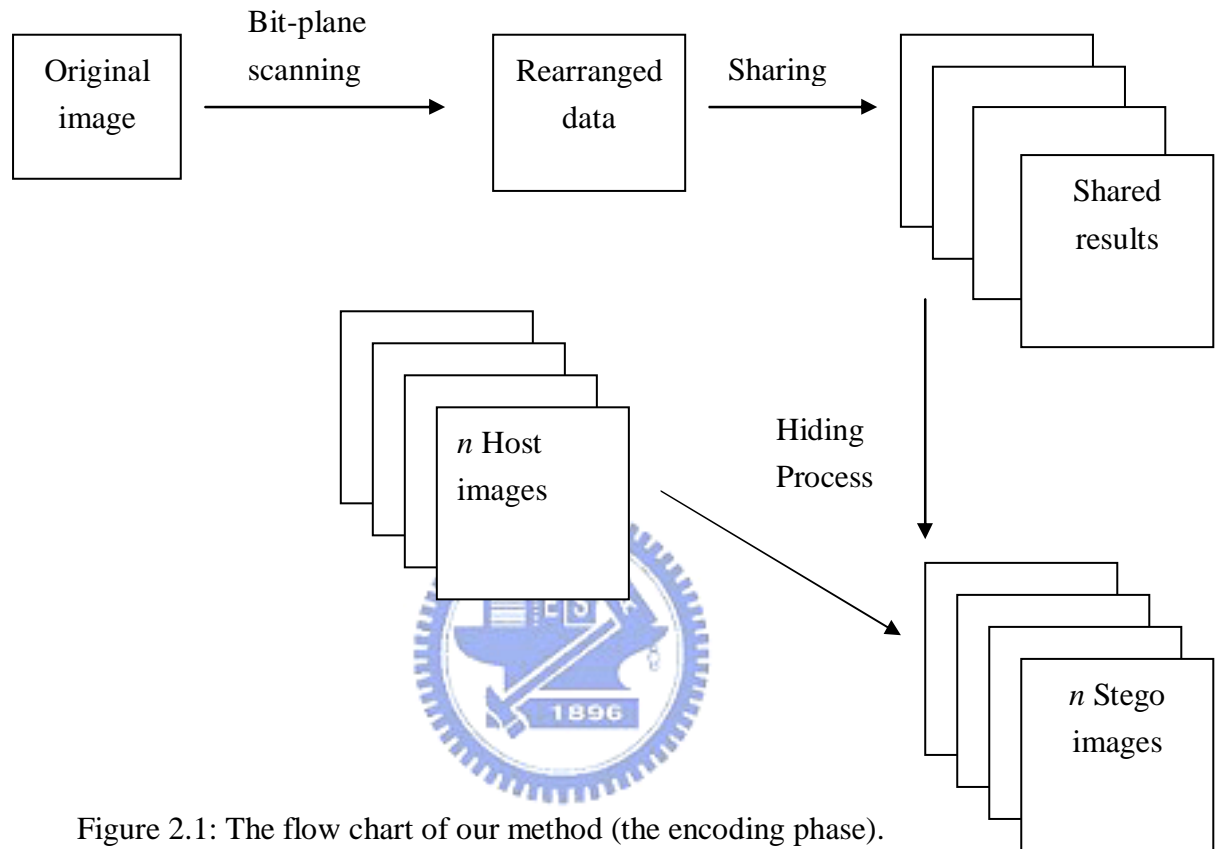


Figure 2.1: The flow chart of our method (the encoding phase).

Step1. Bit plane scanning to rearrange data

First, let the k threshold values r_1, r_2, \dots, r_k be assigned so that $r_1 \leq r_2 \leq \dots \leq r_k = r$. Here the k thresholds denote the distinct number of shared results needed to recover the image with distinct quality levels. For example, when r_1 shared results are received, a quite rough image can be recovered. Then, when more shared results are received, the image quality becomes better and better. Finally, If r_k shared results are received, the image can be recovered completely.

Below we discuss how to scan the image and generate the rearranged data. To begin the process, the original image is divided into several non-overlapped sectors, and the sectors are processed one by one. Each sector always has $RSUM$ pixels ($RSUM = r_1 + r_2 + \dots + r_k$). Since each pixel has 8 bits in gray-level image, each sector has $8(r_1 + r_2 + \dots + r_k)$ bits. We then rearrange these bits to get another $r_1 + r_2 + \dots + r_k$ values and each of them is still an 8-bit number ranging from 0 to 255. For example, if $k = 3$ and if we assign 2, 3, 4 as the k threshold values, then each sector contains $2 + 3 + 4 = 9$ pixels. Without the loss of generality, let the 9 pixels of the sector being discussed have the gray values 166, 167, 164, 166, 168, 165, 163, 166, and 168, respectively. By the rearranging process, shown in Fig. 2.2, the 9 transformed values of the sector will be 255, 128, 63, 224, 0, 143, 171, 76, and 140. The details are shown below. The nine input pixels $\{A, B, C, D, E, F, G, H, I\}$ are

$$A = 166 = (1010\ 0110)_2 = (A_1A_2A_3A_4\ A_5A_6A_7A_8)_2 ,$$

$$B = 167 = (1010\ 0111)_2 = (B_1B_2B_3B_4\ B_5B_6B_7B_8)_2 ,$$

$$C = 164 = (1010\ 0100)_2 = (C_1C_2C_3C_4\ C_5C_6C_7C_8)_2 ,$$

$$D = 166 = (1010\ 0110)_2 = (D_1D_2D_3D_4\ D_5D_6D_7D_8)_2 ,$$

$$E = 168 = (1010\ 1000)_2 = (E_1E_2E_3E_4\ E_5E_6E_7E_8)_2 ,$$

$$F = 165 = (1010\ 0101)_2 = (F_1F_2F_3F_4\ F_5F_6F_7F_8)_2 ,$$

$$G = 163 = (1010\ 0011)_2 = (G_1G_2G_3G_4\ G_5G_6G_7G_8)_2 ,$$

$$H = 166 = (1010\ 0110)_2 = (H_1H_2H_3H_4\ H_5H_6H_7H_8)_2 ,$$

$$I = 168 = (1010\ 1000)_2 = (I_1I_2I_3I_4\ I_5I_6I_7I_8)_2 .$$

Now, we scan these $8 \times 9 = 72$ bits according to the order specified in Fig. 2.2, i.e. the 9 most significant bits (MSB) first $(A_1, B_1, \dots, I_1)_2$, then the 9 second-most significant bits $(A_2, B_2, \dots, I_2)_2$, then the 9 third-most significant bits $(A_3, B_3, \dots, I_3)_2$, etc. We therefore obtain a rearranged 72-bit data set $(A_1, B_1, \dots, I_1, A_2, B_2, \dots, I_2, A_3, B_3, \dots, I_3, \dots, A_8, B_8, \dots, I_8)_2$. If we read these 72 bits (according to the above order), and explain them as 9 numbers (each one is an 8-bit number), then we obtain the rearranged result for this sector, namely, the following 9 values:

$$(A_1B_1C_1D_1\ E_1F_1G_1H_1)_2 = (1111\ 1111)_2 = 255,$$

$$(I_1A_2B_2C_2\ D_2E_2F_2G_2)_2 = (1000\ 0000)_2 = 128,$$

$$(H_2I_2A_3B_3\ C_3D_3E_3F_3)_2 = (0011\ 1111)_2 = 63,$$

$$(G_3H_3I_3A_4\ B_4C_4D_4E_4)_2 = (1110\ 0000)_2 = 224,$$

$$(F_4G_4H_4I_4\ A_5B_5C_5D_5)_2 = (0000\ 0000)_2 = 0,$$

$$(E_5F_5G_5H_5\ I_5A_6B_6C_6)_2 = (1000\ 1111)_2 = 143,$$

$$(D_6E_6F_6G_6\ H_6I_6A_7B_7)_2 = (1010\ 1011)_2 = 171,$$

$$(C_7D_7E_7F_7\ G_7H_7I_7A_8)_2 = (0100\ 1100)_2 = 76,$$

$$(B_8C_8D_8E_8\ F_8G_8H_8I_8)_2 = (1000\ 1100)_2 = 140.$$

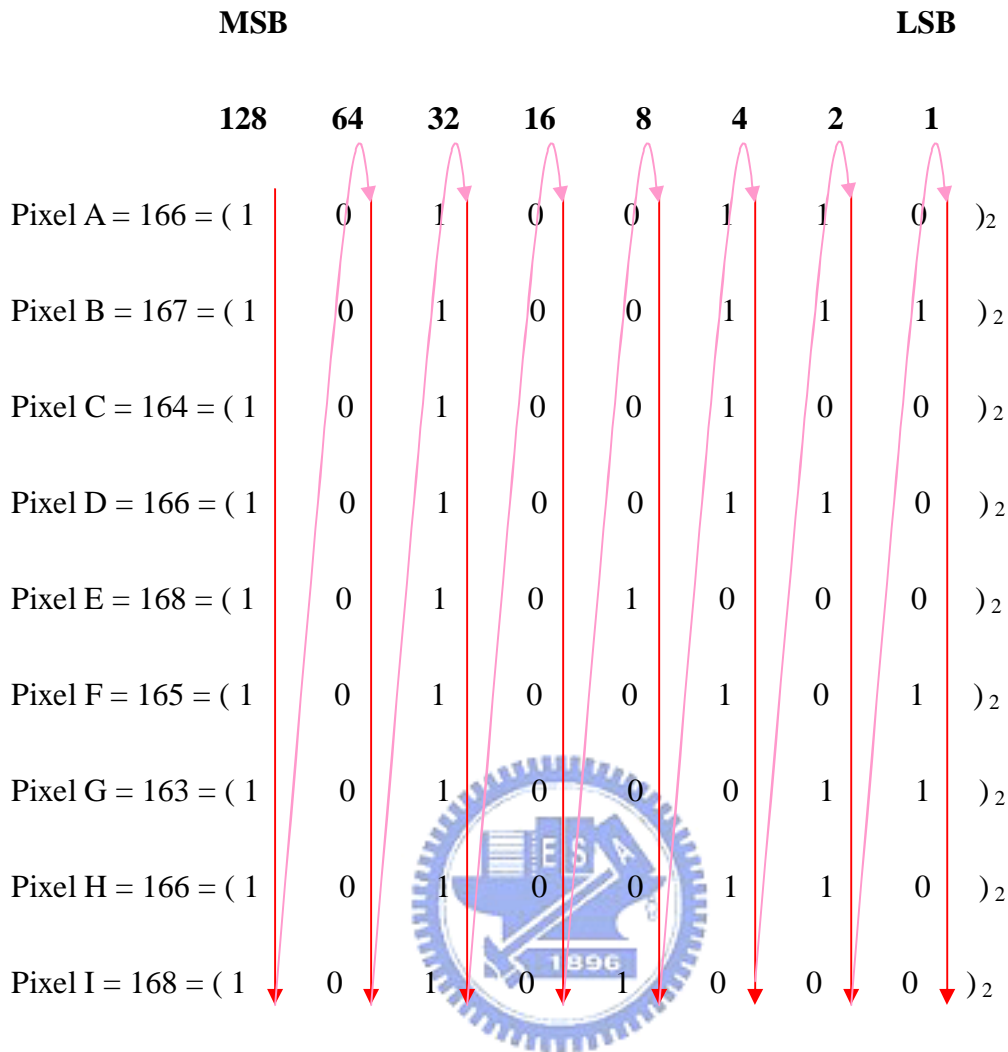


Figure 2.2: The sequence of scanning the 8 bit planes. Note that $RSUM = r_1 + r_2$

$+ \dots + r_k$ pixels are scanned for each sector.

Step2. Sharing

Recall that in our example above, we assumed that $r_1=2$, $r_2=3$, $r_3=4$, and therefore each sector has 9 pixels. In Step 1, we already transformed the 9 pixels {166, 167, 164, 166, 168, 165, 163, 166, 168} of the current sector (say, Sector j) into 9 new values {255, 128, 63, 224, 0, 143, 171, 76, 140}. Now, take the first $r_1=2$ transformed values {255, 128} to form the first polynomial

$$f_j^{(1)}(x) = (255 + 128x) \bmod 257 \quad (2.1)$$

for the current sector. Then take the next $r_2=3$ transformed values $\{63, 224, 0\}$ to form the second polynomial

$$f_j^{(2)}(x) = (63 + 224x + 0x^2) \bmod 257 \quad (2.2)$$

for the current sector. Finally, take the final $r_k=r_3=4$ transformed values $\{143, 171, 76, 140\}$ to form the final polynomial

$$f_j^{(3)}(x) = (143 + 171x + 76x^2 + 140x^3) \bmod 257 \quad (2.3)$$

for the current sector. For each participant $w \in \{1, 2, 3, \dots, 256\}$, the collection

$$\left\{ f_j^{(i)}(w) \mid j = 1, 2, \dots, \frac{\text{original image size}}{RSUM}, \text{ and } i = 1, 2, \dots, k \right\} \quad (2.4)$$

is called the w -th shared result $SR(w)$. Notably, since each sector has $RSUM = r_1 + r_2 + \dots + r_k$ pixels, the total number of sectors that we have is

$j_{MAX} = \frac{\text{original image size}}{RSUM}$. Also note that for each shared result $SR(w)$, it receives

only k numbers $f_j^{(1)}(w) \sim f_j^{(k)}(w)$ generated from Sector j which is an $RSUM$ -pixel

region of the original image. Therefore, the size of each shared result is

$\frac{k}{RSUM} = \frac{k}{r_1 + r_2 + \dots + r_k}$ of that of the original image. In the above example, this

ratio is $\frac{3}{2+3+4} = \frac{1}{3}$.

(Optional) Step 3: Hiding the shared results in some host images

The contents of the shared results look noisy. Therefore, in the special case that the original image is an important secret image, in order to avoid attracting an

attacker's attention, the shared results had better be hidden in some host images to form stego images which look ordinary (non-noisy). The hiding algorithm that we use here is one we developed earlier (similar to the one used in Sec. 2.3 of Ref. [WTL 04]). Note that the size of each shared result is about $\frac{k}{r_1 + r_2 + \dots + r_k}$ of the original image; therefore, the size of each stego image is about $\frac{2k}{r_1 + r_2 + \dots + r_k}$ of that of the original image.

2.2 Decoding

The recovery of the original image includes three steps: extracting the shared results from stego images, recovering the rearranged values from the shared results, and restoring the pixel values from the rearranged values.



Step 1: Extracting the shared results from stego images

This step only needs some simple operations such as division, addition and multiplication. The procedure is similar to the one used in Sec. 2.3 of Ref. [WTL 04], and hence omitted.

Step 2: Recovering the rearranged values from the shared results

To illustrate this step, let us inspect the example given in Equations (2.1)~(2.3) where $k = 3$ threshold values were used ($r_1=2, r_2=3, r_3=4$). According to Eq. (2.4), the shared result held by participant w is

$$SR(w) = \bigcup_j \{f_j^{(1)}(w), f_j^{(2)}(w), f_j^{(3)}(w)\} \quad (2.5)$$

where j ranges through all possible sectors contained in the original image. Now, in the decoding phase, assume that we receive two shared results, say, $SR(1)$ and $SR(4)$.

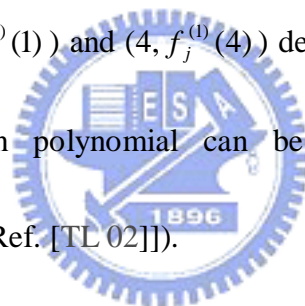
Then, since

$$SR(1) = \bigcup_j \{f_j^{(1)}(1), f_j^{(2)}(1), f_j^{(3)}(1)\}$$

and

$$SR(4) = \bigcup_j \{f_j^{(1)}(4), f_j^{(2)}(4), f_j^{(3)}(4)\},$$

we can know the values $f_j^{(1)}(1)$ and $f_j^{(1)}(4)$ for each sector j . Therefore, the coefficients 255 and 128 of the polynomial $f_j^{(1)}(x)$ defined in Eq. (2.1) can be determined (two points $(1, f_j^{(1)}(1))$ and $(4, f_j^{(1)}(4))$ determine a line; and the equation of this unique interpolation polynomial can be found by using Lagrange's interpolation [see Sec. 3.2 of Ref. [TL 02]]).



In the process of progressive transmission, assume that we receive one more shared result, say, besides $SR(1)$ and $SR(4)$, we also receive $SR(5)$. Then, since

$$SR(5) = \bigcup_j \{f_j^{(1)}(5), f_j^{(2)}(5), f_j^{(3)}(5)\}$$

we can know the values $f_j^{(2)}(1)$, $f_j^{(2)}(4)$, and $f_j^{(2)}(5)$. Again, by using Lagrange's interpolation, we can find the unique interpolation polynomial through the three points $(1, f_j^{(2)}(1))$, $(4, f_j^{(2)}(4))$, $(5, f_j^{(2)}(5))$. The three coefficients 63, 224, and 0 of $f_j^{(2)}(x)$ defined in (2) are therefore obtained. Therefore, $\{255, 128; 63, 224, 0\}$ are all known when we received three shared results $SR(1)$, $SR(4)$, and $SR(5)$.

An analogous argument shows that we can know all 9 coefficients in Eq. (2.1)~(2.3) if we receive four shared results, say, $SR(1)$, $SR(4)$, $SR(5)$, and $SR(8)$.

Step 3: Restoring the pixel values from the rearranged values

After obtaining the rearranged values in the previous step, the values can be transformed back to restore the pixels of the original image. For example, if $\{255, 128\}$ and $\{63, 224, 0\}$ are recovered in the previous step (assuming that three shared results are received), then, $255=(1111\ 1111)_2$, $128=(1000\ 0000)_2$, $63=(0011\ 1111)_2$, $224=(1110\ 0000)_2$, $0=(0000\ 0000)_2$ together form a sequence of 40 bits, i.e. $(1111\ 1111\ 1000\ 0000\ 0011\ 1111\ 1110\ 0000\ 0000\ 0000)_2$. If we restore these 40 bits according to the scan order listed in Fig. 2.2, we can restore at least $\left\lfloor \frac{40}{9} \right\rfloor = 4$ most significant bits of the 9 pixels A~I. In fact $40-9 \times 4 = 4$ implies that the first four pixels (A~D) can recover one more bit each. Therefore the $\left\lceil \frac{40}{9} \right\rceil = 5$ most significant bits of the pixels A, B, C, D are revealed to be $(10100)_2$, $(10100)_2$, $(10100)_2$, and $(10100)_2$, respectively; while the $\left\lfloor \frac{40}{9} \right\rfloor = 4$ most significant bits of the pixels E, F, G, H, I are revealed to be $(1010)_2$, $(1010)_2$, $(1010)_2$, $(1010)_2$, $(1010)_2$, respectively (see Fig. 2.2).

2.3 Experimental results

The experimental result is shown in Figures 2.3-2.5 and Table 2.1. The input is the image Lena shown in Fig. 2.3, which is shared by our progressive scheme. In the experiment, we use $k = 4$ thresholds, which are $(r_1=2) < (r_2=3) < (r_3=4) < (r_4=r_k=5)$.

We generate, say, $n = 6$ shares, and then the six shared results are hidden in six host images to generate six stego images. Fig. 2.4 shows the stego images. The PSNRs of them range from 34.20 to 34.49. Notably, the size of each host image and each stego image in this experiment is 388×388 , because $388 \times 388 \approx 2 \times \left(512 \times 512 \times \frac{4}{2+3+4+5} \right)$ where $512 \times 512 \times \frac{4}{2+3+4+5}$, i.e. original image size times $\frac{k}{r_1 + r_2 + \dots + r_k}$, is the size of each shared result. The factor 2 is due to the fact that the stego image size is two times greater than the data (the shared result) hidden inside.

Fig. 2.5 shows the images recovered from various numbers of the stego images. Fig. 2.5(a) shows the recovered image when “any” two of the stego images in Fig. 2.4 are available. The recovered image is with bad quality because PSNR is only 14.57db. Fig. 2.5(b) shows the recovered image when any three of the stego images are available. The recovered image has a better quality (29.28db). Fig. 2.5(c) shows the recovered image when any four of these stego images are available. The recovered image is with much better quality (48.46db). Fig. 2.4 shows the recovered image when any five of the six stego images are available. The recovered image is lossless.





Figure 2.3. The 512×512 original image.



(a)



(b)



(c)



(d)



(e)



(f)

Figure 2.4. The six 388×388 stego images (the PSNRs range from 34.20 to 34.49).



Figure 2.5. The recovered images revealed from various numbers of stego images. (a) from any 2 stego images (PSNR=14.57); (b) from any 3 stego images (PSNR=29.28); (c) from any 4 stego images (PSNR = 48.46); (d) from any 5 stego images (lossless).

2.4 Quality Control Design

It is possible to control the quality of the image recovered from a small number of shared results. In some applications, if people receive, say, only 3 shared results, we hope that image quality is poor (for example, a cable system whose image quality depends on the amount of money paid by the viewer). In some other applications, however, we hope that only 3 shared results can still provide good-quality recovery (for example, the hot line between two police stations). To control quality, we may

repeat some threshold values in the design. For example, consider a design where the threshold values are $\{3, 4, 5\}$. If we only use three threshold values $\{r_1=3, r_2=4, r_3=5\}$, we get the results shown the middle column of Table 2.1, i.e. 20.01 db and 42.7 db when we received 3 and 4 shared results, respectively. (Note that $3+4+5=12$ pixels per sector in this case and we construct 3 polynomials for each sector and these 3 polynomials have 3, 4, and 5 coefficients, respectively.) Now, in some business application, the image owner may hope to reveal only poor quality image when less than 5 shared results are purchased. Thresholding $\{r_1=3, r_2=4, r_3=5, r_4=5, r_5=5\}$ is then a solution for this, because only about $\frac{3+4}{3+4+5+5+5} \times 8 = \frac{7}{22} \times 8 \approx 2.5$ of the 8 bits is revealed for each pixel when 4 shared results are received. (Note that there are $3+4+5+5+5=22$ pixels per sector in this thresholding case, and in the encoding phase we build up 5 polynomials, and the number of coefficients in them are 3, 4, 5, 5, 5, respectively. Receiving 4 shared results can recover only the first two polynomials (because $3 \leq 4$ and $4 \leq 4$), and therefore recover only $3+4=7$ of the 22 coefficients of the polynomials.) As shown in Table 2.1, the image quality is quite poor (13.15 db or 24.00 db) when the image is recovered by 3 or 4 shared results. Finally, when fast revealing of the good-quality image is required, e.g., between police stations, thresholding $\{r_1=3, r_2=3, r_3=3, r_4=4, r_5=5\}$ can achieve the goal, because

$$\frac{3+3+3}{3+3+3+4+5} \times 8 = \frac{1}{2} \times 8 = 4 \text{ of the 8 bits is revealed for each pixel when only 3}$$

shared results are received. (Note that there are $3+3+3+4+5=18$ pixels per sector in this thresholding case, and in the encoding phase we build 5 polynomials, and the number of coefficients in them are 3, 3, 3, 4, 5, respectively. Receiving 3 shared results can recover the first three polynomials, and therefore recover $3+3+3 = 9$ of the 18 coefficients). As shown in Table 2.1, the image quality is acceptable (31.11 db when the image is recovered by 3 shared results. Moreover, the image quality is good (46.38 db) when the image is recovered by 4 shared results.

Table 2.1. Three kinds of thresholding that are all composed

of 3, 4, and 5 shared results.

PSNR	Thresholding		
	3, 4, 5 (12 pixels/sector)	3, 4, 5, 5, 5 (22 pixels/sector)	3, 3, 3, 4, 5 (18 pixel/sector)
Number of stego images			
3	20.01	13.15	31.11
4	42.70	24.00	46.38
5	lossless	lossless	lossless

2.5 Summary

In the proposed method, there are several characteristics: (1) the scheme is fault-tolerant (allowing $n-r$ stego images to be lost or damaged); (2) the shadow results are equally important, thus, there is no need to worry about which part is lost

(or which part is transmitted first) during the transmission; (3) the scheme is secure

(less than r_1 shared results cannot reveal any information about the image). (4)

Quality-control design is possible (as explained in Sec. 2.4).



Chapter 3

Progressive viewing of vector-quantized images: a sharing approach

In this chapter, we propose a sharing approach of progressive viewing of any image already quantized by a vector quantization (VQ) process. The approach is an $((r_1, r_2, \dots, r_k), n)$ -threshold sharing system: n shares are created to replace the index file of a given quantized image, and the loss or damage of up to $n-r_k$ shares does not affect at all the quality of the image reconstructed by VQ. The n shares can be stored or transmitted using up to n distinct channels to increase the survival rate, and each share is several times smaller than the index file. When some communication channels are faster than the others, the proposed method also provides some “rough” versions of the image quickly (before the expected VQ-image is recovered completely), when some of the required r_k shares arrive much later than the other shares do.

The remaining of the chapter is organized as follows. Sec. 3.1 describes the encoding method. Sec. 3.2 describes the decoding method. Sec. 3.3 shows the experimental results. Finally, Sec. 3.4 presents the conclusions.

3.1 Encoding

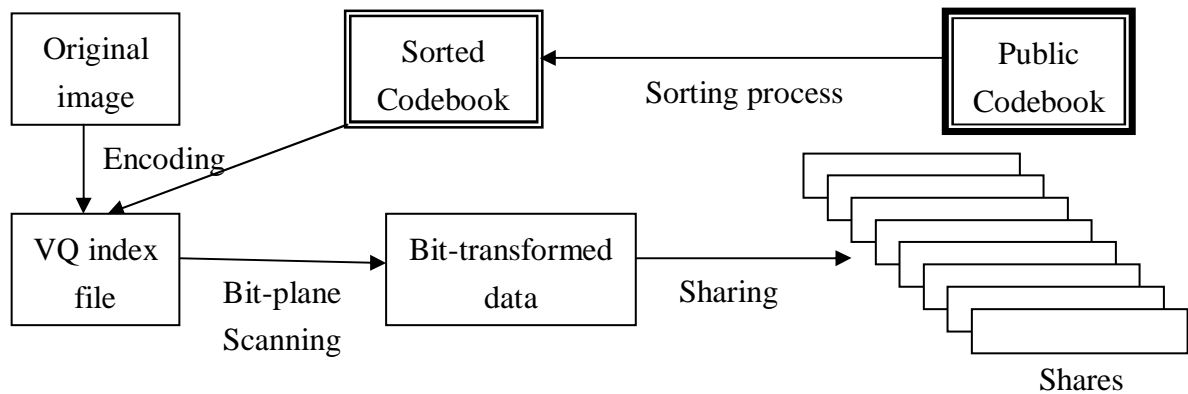


Figure 3.1. The encoding phase.

The encoding phase of the proposed method is illustrated in Fig. 3.1. Firstly, a sorted codebook is generated via sorting a given public codebook so that the average gray-values of the codewords are in a non-decreasing order (for each codeword, take an average among all dimensions of the codeword). The reason that the codebook must be sorted will be explained later at the end of Sec. 3.2. Right now let us still focus on how to use this sorted codebook to quantize an image. If we have been given the index file created earlier for the given image using the “non-sorted” codebook, then, in order to obtain the new index file corresponding to the sorted codebook, we simply modify the content of that given index file. This is just an easy job of switching names, for we only have to apply the 1-to-1 mapping (which maps the codewords of the non-sorted codebook to that of the sorted codebook) to the indices of the old index file. On the other hand, if no index file is given, then, in order to get

the index file, we directly quantize the original image using the sorted codebook. In both cases, an index file is generated, and it is corresponding to the sorted codebook.

Now, let the index file be bit-transformed further by a bit-plane scanning method introduced below in Sec. 3.1.1. After that, the bit-transformed indices are shared to generate the n shares, as explained in Sec. 3.1.2.

3.1.1 Bit-transform of the indices.

Let the k given threshold values be $r_1 \leq r_2 \leq \dots \leq r_k$, as stated earlier. The r_1 is the minimal threshold value to reveal the image roughly, and r_k is the threshold value to reveal the image identical to the one recovered by traditional VQ. Then, we perform the bit-transform of the index file, as described below. Firstly, the index file is divided into non-overlapping sectors so that each sector has RSUM ($= r_1 + r_2 + \dots + r_k$) indices. The indices of each sector are then processed by a bit-plane scanning. For example, assume that RSUM=5, the five indices of a sector are (738, 770, 751, 721, 643), and each index has t bits ($t = 10$ if there are 1024 codewords; $t = 9$ if there are 512 codewords; and so on). An example of the bit-transformed process using $t = 10$ is shown in Fig. 3.2, and the $\frac{5 \times t}{t} = 5$ bit-transformed values (still t bits per value) of the sector are (1000, 758, 642, 132, 935), according to the appearance order. Of course, the first several transformed values of the sector consist of the significant bits of the original indices. For instance, 1000 and 758 together record the four ($\lceil 2 \times t / 5 \rceil =$

$\lfloor 2 \times 10 / 5 \rfloor = 4$) most significant bits (MSB) of the original indices in the above example.

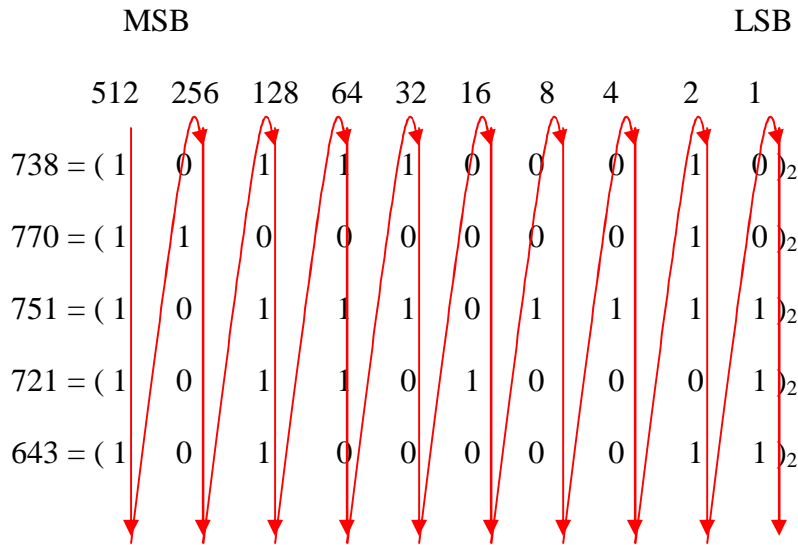


Figure 3.2. An example showing the bit-transform process.

3.1.2 Sharing the bit-transformed indices.

As stated above in Sec. 3.1.1, every $r_1 + \dots + r_k$ indices are grouped together to form a sector, and the $r_1 + \dots + r_k$ indices inside each sector are bit-transformed to another $r_1 + \dots + r_k$ values. Now, these values are to be shared. In the sharing process, the $r_1 + \dots + r_k$ values inside each sector are divided further into k sub-sectors (SS) so that each SS_i has r_i values. Then, at each SS_i , the r_i values are shared using a polynomial $P_i(x)$ of degree $r_i - 1$, and its r_i coefficients are in fact the r_i values being shared. (Therefore, each sector is equipped with k polynomials ($P_1(x), P_2(x), \dots, P_k(x)$) of its own: polynomial P_1 is to share the first r_1 transformed values; polynomial P_2 is to share the next r_2 transformed values; and so on.) Finally, each Share j is just a record of the values ($P_1(j), P_2(j), \dots, P_k(j)$) recorded in a sector-by-sector manner. An

example is given below to help the readers to understand the sharing process.

Example. (An example illustrating how to share the bit-transformed indices.)

Without the loss of generality, assume that there are $k = 2$ threshold values $\{r_1 = 2, r_k = r_2 = 3\}$, and the $r_1 + \dots + r_k = r_1 + r_2 = 2 + 3 = 5$ bit-transformed indices of the current sector are (1000, 758, 642, 132, 935). These 5 bit-transformed indices are to be shared. Therefore, create two polynomials $P_1(x) = 1000 + 758x \pmod{1031}$ and $P_2(x) = 642 + 132x + 935x^2 \pmod{1031}$ for this sector. The reason we use 1031 in the mod-function is because it is the prime number closest to, but still larger than, $2^{10} = 1024$ when indices are all 10-bits. (Similarly, the prime number 521 should be used when indices are all 9-bits, for $521 > 2^9 = 512$.) Now, without the loss of generality, assume that $n = 4$ shares are to be generated (n is the number of shares to be generated, and its value must be not less than any threshold value [therefore, $n \geq \text{Max}\{2,3\}=3$ is required here]). Then, Share 1 keeps a record of the pair $(P_1(1), P_2(1))$; Share 2 keeps $(P_1(2), P_2(2))$; Share 3 keeps $(P_1(3), P_2(3))$; and Share 4 keeps $(P_1(4), P_2(4))$.

The decoding phase will be given in next section. Right now, we just explain why people can use the shares created above to recover the image in a “progressive” manner. Assume that only two of the four created shares (for example, Shares 2 and 4)

arrive at the receiver end of a network. Then, the recovery of the three coefficients $(c_1, c_2, c_3) = (642, 132, 935)$ for $P_2(x)$ are still not possible, since (c_1, c_2, c_3) cannot be solved from the only two equations related to P_2 , i.e. from $P_2(2) = (c_1 + 2c_2 + 4c_3) \bmod 1031$, and $P_2(4) = (c_1 + 4c_2 + 16c_3) \bmod 1031$. However, the two coefficients $(a, b) = (1000, 758)$ of $P_1(x)$ can be recovered immediately by solving the linear set

$$P_1(2) = (a + 2b) \bmod 1031$$

$$P_1(4) = (a + 4b) \bmod 1031 ,$$

using the so-called Lagrangian interpolation polynomials (the solving detail can be found in [CH 98] stated by Chang and Hwang). As a result, the $\lceil 2 \times 10 / 5 \rceil = 4$ most significant bits of the original five indices in the sector can be revealed. Therefore, the five indices can be roughly estimated; then people can see a rough version of the image. On the other hand, if 3 ($=r_2=r_k$) of the 4 generated shares are received, for example, Shares 1, 2, and 4 are received; then we have $(P_1(1), P_2(1))$ from Share 1; $(P_1(2), P_2(2))$ from Share 2; and $(P_1(4), P_2(4))$ from Share 4. As a result, we may use the two equations $P_1(2) = (a + 2b) \bmod 1031$ and $P_1(4) = (a + 4b) \bmod 1031$ to solve for $\{a, b\}$ and obtain $\{a=1000, b=758\}$, as stated above. We also have the three equations $P_2(1) = (c_1 + 1c_2 + 1c_3) \bmod 1031$, $P_2(2) = (c_1 + 2c_2 + 4c_3) \bmod 1031$, and $P_2(4) = (c_1 + 4c_2 + 16c_3) \bmod 1031$ to solve for $\{c_1, c_2, c_3\}$ and obtain $\{c_1=642, c_2=132, c_3=935\}$. Therefore, not only $(1000, 758)$, but also $(642, 132, 935)$, can be



recovered. As a result, all five original indices (738, 770, 751, 721, 643) can be recovered completely without any error. Since the whole index file can be exactly recovered, the image obtained will be exactly the VQ-image reconstructed by the ordinary VQ method. ---End of the example.

Notably, after our bit-transform and sharing, the index file is replaced by n shares in our scheme, and these n shares can be transmitted using n distinct channels. Intercepting up to $r_1 - 1$ channels by the hackers will not reveal anything about the index file; and hence, will not reveal the image. The reason is quite obvious, as explained below. $P_1(x)$ only has r_1 coefficients; however, in order to recover back its r_1 coefficients, r_1 shares will be needed in order to provide the data $\{P_1(j)\}$ at r_1 distinct values of j . Similarly, $P_2(x)$ needs r_2 shares to recover its coefficients, and so on. Since $r_1 \leq r_2 \leq \dots \leq r_k$, if we get less than r_1 shares, none of the coefficients can be recovered, and this is true for each of the k polynomials.

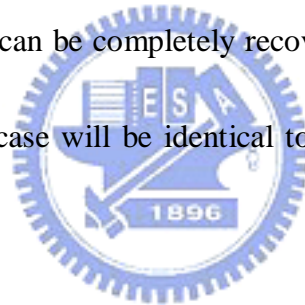
Besides the no-leaking property just mentioned, also note that, the disconnection of up to $n - r_k$ channels will not influence the recovery of the vector-quantized image. This is due to the fact that each of the k polynomials has no more than r_k coefficients; therefore, if we get r_k shares (recall that each share j keeps a record of $\{P_1(j), P_2(j), \dots, P_k(j)\}$), then we get enough information to solve for the r_i coefficients of each

polynomial $P_i(x)$.

3.2 Decoding

The sorted codebook discussed in the encoding phase is used again here in the decoding phase. Fig. 3.3 illustrates the decoding phase. There are three cases in the decoding phase, as explained below. If less than r_1 shares are collected, then nothing can be recovered. If less than r_k (but not less than r_1) shares are collected, then the bit-transformed data can be partially recovered, and so can the index file. A rough-quality image is obtained in this case. Finally, if at least r_k shares are collected, then the bit-transformed data can be completely recovered, and so can the index file.

The image recovered in this case will be identical to the one recovered by ordinary vector quantization algorithm.



Below we explain why the codebook should be sorted. When the number of received shares is less than r_k (but still not less than r_1), in order that “partial information” of the index file can still recover a rough image, the estimation of the indices is required. Of course, the image distortion due to wrong estimation certainly exists. Therefore, to reduce serious distortion, the public codebook is sorted in advance so that the neighboring codewords will have similar average gray-values. By doing so, we can alleviate the big image distortion introduced by using the neighboring codewords, when indices are estimated. To be able to estimate the indices

from their partial data (so that images can be recovered progressively), the bit-plane scanning method is also used earlier in the encoding phase to generate the bit-transformed data. By doing so, the first several elements created in each sector and stored in the transformed data, can be used to recover the most significant bits of the original indices.

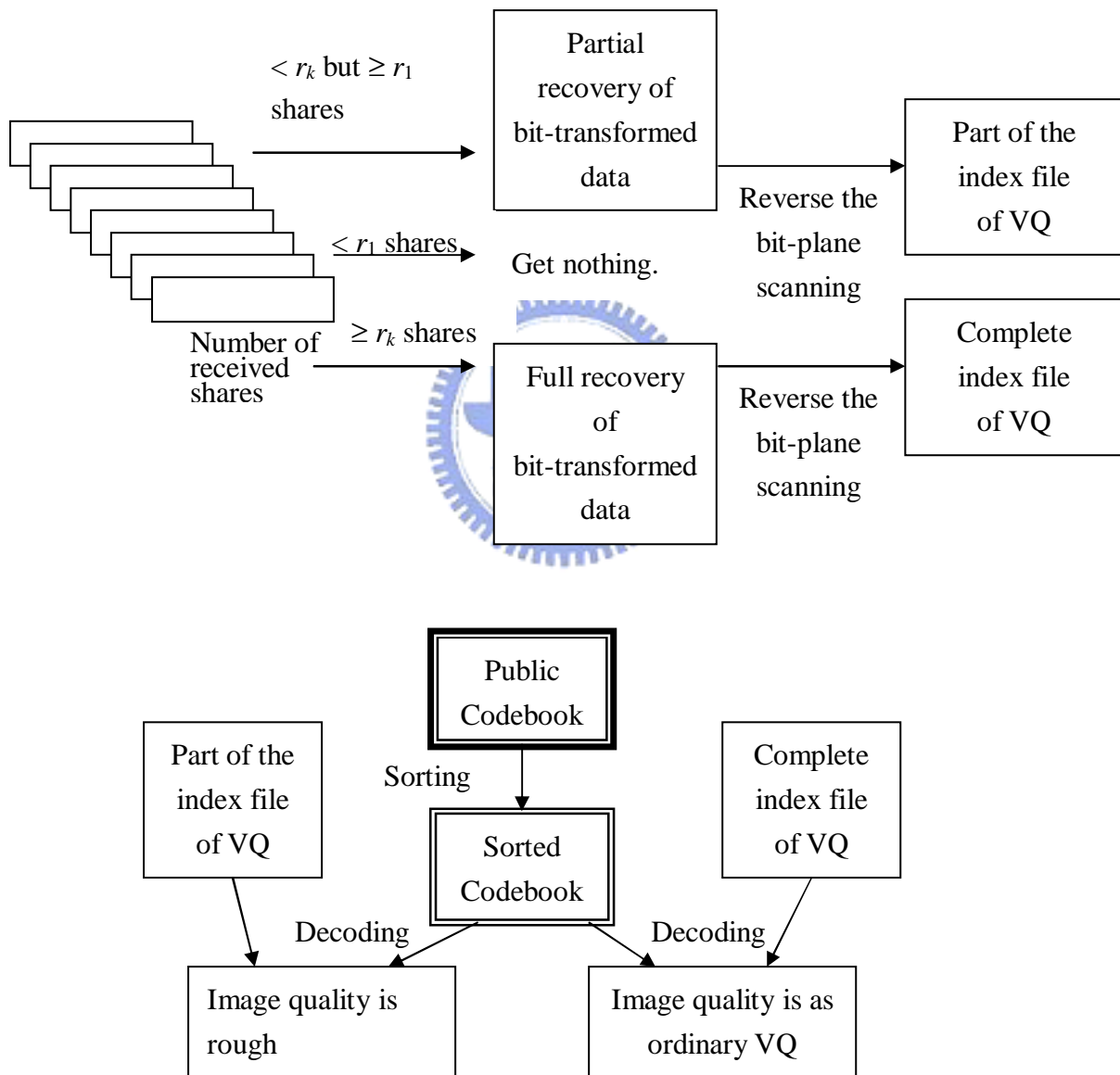
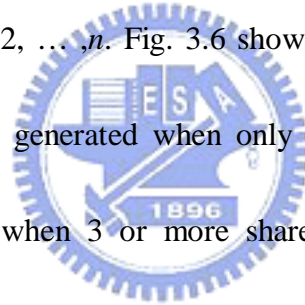


Figure 3.3. The decoding phase.

3.3 Experimental results

The experimental results are discussed in this section. Fig. 3.4 shows the original image Lena, which is vector-quantized to obtain an index file according to a public codebook, and the reconstructed VQ-image is shown in Fig. 3.5. So far, not a little bit of the proposed method is used. Then, when the codebook is sorted, the index file is also modified accordingly. Then, in the first experiment, $k = 2$ thresholds $\{r_1 = 2, r_2 = 3\}$ are used, and $n = 4$ shares are generated. Each share is about $\frac{k}{r_1 + \dots + r_k} = \frac{k}{r_1 + r_2} = \frac{2}{5}$ times smaller than the index file in size, for each sector of $r_1 + \dots + r_k$ transformed values only contributes k values $\{P_1(j), \dots, P_k(j)\}$ to Share j (see Sec. 3.2), true for any $j = 1, 2, \dots, n$. Fig. 3.6 shows the image recovered from the “partial” index file, which is generated when only 2 shares are received. Fig. 3.7 shows the recovered image when 3 or more shares are received. Note that this recovered image is identical to the traditional VQ-recovered image shown in Fig. 3.5, and it should be of no surprise, since $r_k = r_2 = 3$ implies that the index file is fully reconstructed when 3 or more shares are received. In the second experiment, $k = 3$ thresholds $\{r_1 = 3, r_2 = 4, r_3 = 5\}$ are used, and $n = 6$ shares are generated. Each share is $\frac{k}{r_1 + r_2 + r_3} = \frac{3}{12} = \frac{1}{4}$ times smaller than the index file in size. Figures 4.8 and 4.9 show the recovered images from the partial index files, which are generated when 3 and 4 shares are received, respectively. Fig. 3.10 shows the recovered image when 5 or more shares are received. Again, the image in Fig. 3.10 is identical to the ordinary



VQ-recovered image shown in Fig. 3.5, since $r_k = r_3 = 5$ implies that the index file is fully reconstructed when 5 or more shares are received.



Figure 3.4. Original image



Figure 3.5. The recovered image (PSNR = 31.60 dB) using ordinary VQ [LBG 80].



Figure 3.6. The recovered image when only 2 shares are received in the $(r_1, r_2) = (2, 3)$ case. The PSNR is 20.97 dB.



Figure 3.7. The recovered image (identical to the 31.60dB image in Fig. 3.5) when 3 or more shares are received in the $(r_1, r_2) = (2, 3)$ case.



Figure 3.8. The recovered image when 3 shares are received in the $(r_1, r_2, r_3) = (3, 4, 5)$ case. The PSNR is 19.24 dB.



Figure 3.9. The recovered image when 4 shares are received in the $(r_1, r_2, r_3) = (3, 4, 5)$ case. The PSNR is 22.37 dB.



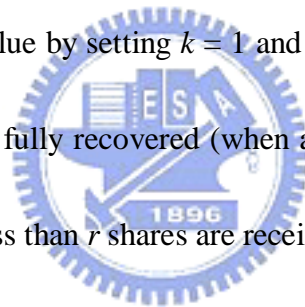
Figure 3.10. The recovered image (identical to the 31.60dB image in Fig. 3.5) when 5 or more shares are received in the $(r_1, r_2, r_3) = (3, 4, 5)$ case.

3.4 Discussion

In this chapter, we have proposed a method that generates n small shares from the index file of VQ, and then progressively recovers the vector-quantized image (VQ-image) in a fault-tolerant manner. The method is achieved by using the “sorted” codebook, and the sharing of bit-transformed indices.

The method can be treated as a post-processing tool for any VQ algorithm. With this post-processing, there are some extra properties added to the VQ algorithm being used: 1). The proposed method reduces the chance that the VQ-image is unveiled when the index file is intercepted, for the n shares can be stored or transmitted in n distinct channels, and the thief has to intercept at least r_1 channels before he can see a vague image. 2). It also increases the survival rate of the VQ-image after a sequence of attacks from the hackers, for up to $n - r_k$ shares can be lost (in fact, even if $n - r_1$ shares are lost, we can still get a vague version of the VQ-image). 3). When the n

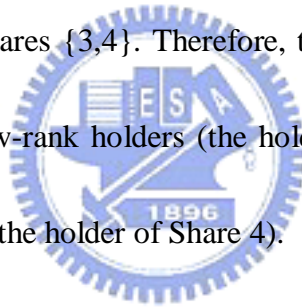
shares are received from n distinct channels, some shares might arrive earlier than the others, so the progressive reconstruction is useful. 4). Each share is equally important, so it is unnecessary to worry about which share is transmitted first in the transmission period, or worry about which share is lost in the recovery period. 5). Since the index file is already much smaller than the image itself (12.8 times smaller in our examples), the fact that each share is smaller than the index file means that the storage space of each share in any of the n distinct channels (one share per channel) is very compact. 6). If an image is quite confidential, then we may just give up the progressive property, and use only one threshold value by setting $k = 1$ and $r_1 = r_k = r$. This ensures that the VQ-image (Fig. 3.5) is either fully recovered (when at least r shares are received) or completely invisible (when less than r shares are received).



Chapter 4

Progressive viewing of a secret image: a VC approach

For a given image, a progressive viewing method using visual cryptograph (VC) approach is introduced in this chapter. The generated transparencies are with weights. By controlling the weights in the design, we can get some special effects. For example, if we create Shares 1-4, and their weights are $w_1 = 1$, $w_2 = 2$, and $w_3 = w_4 = 3$, respectively. Then, since $w_1 + w_2 + w_3 = w_3 + w_4$, the result of stacking shares $\{1,2,3\}$ is just like that of stacking shares $\{3,4\}$. Therefore, the holder of Share 3 can either ask for the help from two low-rank holders (the holders of Shares 1 and 2), or, the help from a high-rank holder (the holder of Share 4).



For the rest of this chapter, Sec. 4.1 describes the details of our scheme. Experimental results are in Sec. 4.2. Finally, the conclusions are stated in Sec. 4.3.

4.1 The proposed method

Let the n generated shares be $\{t_1, t_2, \dots, t_n\}$ with the corresponding weights being $\{w_1, w_2, \dots, w_n\}$ and $w_1 \leq w_2 \leq \dots \leq w_n$. In our method, the weights $\{w_1, w_2, \dots, w_n\}$ of the shares can be any positive integers, and each pixel in the black-and-white secret image will be expanded into blocks of size $p \times q$ (one block per pixel). In the above, p and q must meet the rule that $p \times q \geq w_1 + \dots + w_n$ due to the design used in Algorithm

4.1 and 4.2. For the purpose of attaining the best contrast, adjusting the parameters to make “ \geq ” become “=” is suggested (“=” can be achieved by adjusting $\{w_1, w_2, \dots, w_n\}$, or n , or $p \times q$).

Algorithm 4.1: The process of sharing a white pixel of the secret image among the corresponding blocks of the n shares.

Input : the n weights $w_1 \leq w_2 \leq \dots \leq w_n$.

Output: n blocks (one block per share, and each block has size $p \times q$).

Steps:

Step 1. Let $i = 1$ and $w_0 = 0$. Let the Painting-Guide be a set $Q = \{(j, k)\}_{\substack{1 \leq j \leq p \\ 1 \leq k \leq q}}$.

Step 2. (Create the corresponding Block B_i for Share i .)

2.1 Randomly select $w_i - w_{i-1}$ elements from Q . Then, delete these $w_i - w_{i-1}$ elements from Q .

2.2 Paint Block B_i according to the Painting-Guide Q . More precisely, an element of Block B_i is painted as white if that element appears (and as black if that element disappears) in Q .

Step 3. Raise the value of i by 1 and go to Step 2 until $i > n$.

For example, assume that there are $n = 4$ shares and the four weights are $(w_1, w_2, w_3, w_4) = (1, 2, 3, 3)$. The block size can be assigned as $p \times q = 3 \times 3$, since $3 \times 3 = 1 + 2 + 3 + 3$. To share a white pixel of the secret image, the corresponding expanded

block in each share is generated as in Fig. 4.1. In this figure, (a), (b), (c), and (d) are the expanded blocks in Shares 1, 2, 3, and 4, respectively. For Share 1, because $w_1 = 1$, a random position (say, (2, 3)) is selected (and then deleted) from $Q = \{(j, k) | 1 \leq j \leq 3, 1 \leq k \leq 3\}$. In Fig. 4.1(a), position (2, 3) of the block of Share 1 is set to black and all other positions are set to white. For Share 2, because $w_2 - w_1 = 1$, one more random position (say, (3, 2)) is selected (and then deleted) from Q . In Fig. 4.1(b), positions (2, 3) and (3, 2) of the block of Share 2 are set to black, and all other positions are set to white. For Share 3, because $w_3 - w_2 = 1$, an additional random position (say, (1, 2)) is selected (and then deleted) from Q . In Fig. 4.1(c), positions $\{(2, 3), (3, 2), (1, 2)\}$ of the block of Share 3 are set to black, and all other positions are set to white. For Share 4, because $w_4 - w_3 = 0$, no new random position is needed. In Fig. 4.1(d), the block of Share 4 is therefore the same as the block of Share 3. In this example, when we stack shares, at most 3 of the 9 positions in the block will be black if the block is a stacking result representing a white pixel of the secret image.

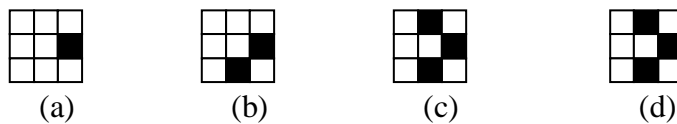


Figure 4.1. The expanded blocks of a white pixel.

Algorithm 4.2: The process of sharing a black pixel of the secret image among

the corresponding blocks of the n shares.

Input: the n weights $w_1 \leq w_2 \leq \dots \leq w_n$.

Output: n blocks (one block per share, and each block has size $p \times q$).

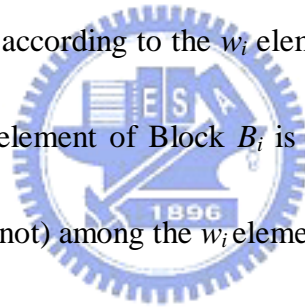
Steps :

Step 1. Let $i = 1$ and the Painting-Guide be a set $Q = \{(j, k)\}_{1 \leq j \leq p, 1 \leq k \leq q}$.

Step 2. (Create the corresponding Block B_i for Share i .)

2.1 Randomly select w_i elements from Q . Then, delete these w_i elements from Q .

2.2 Paint Block B_i according to the w_i elements selected in Step 2.1. More precisely, an element of Block B_i is painted as black (white) if that element is (is not) among the w_i elements selected in Step 2.1.



Step 3. Raise the value of i by 1, and go to Step 2 until $i > n$.

For explanation, assume again that $n = 4$, $(w_1, w_2, w_3, w_4) = (1, 2, 3, 3)$, and $p \times q = 3 \times 3$. Then, the expanded block of a black pixel in each share is generated as in Fig. 3.2. In this figure, (a), (b), (c), and (d) are the expanded blocks in Shares 1, 2, 3, and 4, respectively. For Share 1, because $w_1 = 1$, a random position (say, (1, 2)) is selected (and then deleted) from $Q = \{(j, k) \mid 1 \leq j \leq 3, 1 \leq k \leq 3\}$. In Fig. 3.2(a), Position (1, 2) of the block of Share 1 is therefore set to black, and the remaining positions are set to white. For Share 2, two new random positions (say, (2, 1) and (2,

3)) are selected (and then deleted) from the $9 - w_1 = 9 - 1 = 8$ existent positions in Q . In Fig. 3.2(b), Positions (2, 1) and (2, 3) of the block of Share 2 are therefore set to black, and the remaining positions are set to white. For Share 3, because $w_3 = 3$, three new random positions (say, (2, 2), (3, 1), and (3, 3)) are selected (and then deleted) from the $9 - w_1 - w_2 = 9 - 1 - 2 = 6$ existent positions in Q . In Fig. 4.2(c), positions (2, 2), (3, 1), and (3, 3) of the block of Share 3 are therefore set to black, and the remaining positions are set to white. At last, in Fig. 4.2(d), Positions $\{(1, 1), (1, 3), (3, 2)\}$ of the block of Share 4 are set to black, and the remaining positions are set to white, for $w_4 = 3$ and $\{(1, 1), (1, 3), (3, 2)\}$ are the only three possible positions (according to the current content of Q).

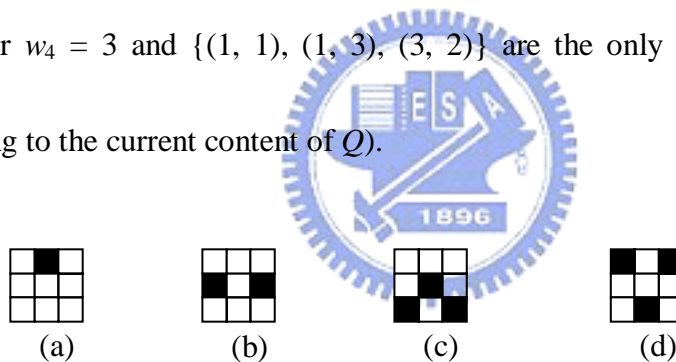


Figure 4.2. The expanded blocks of a black pixel.

Remarkably, when we stack the shares from a subset S of $\{t_1, \dots, t_n\}$, the stacked p -by- q block representing a black pixel will have $\sum_{t_i \in S} w_i$ black elements, and

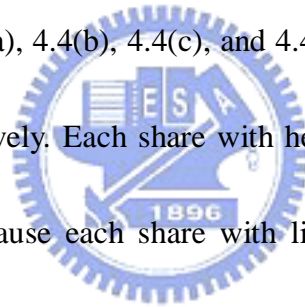
the stacked p -by- q block representing a white pixel will have $\text{Max}\{w_i \mid t_i \in S\}$ black

elements. Therefore, it is reasonable to use $\frac{\sum_{t_i \in S} w_i - \text{Max}_{t_i \in S} w_i}{p \times q}$ to represent the contrast

of the stacking result. This definition of contrast also agrees with the definition given in [HKS00].

4.2 Experimental results

The experimental results are shown in Figures 4.3-4.8. In each experiment, $n = 4$ shares are generated for each given image, and the weights of the four shares are $\{1, 2, 3, 3\}$. Since $1 + 2 + 3 + 3 = 9$, the block size $p \times q$ is set to 3×3 to meet the requirement $p \times q \geq \sum_{i=1}^n w_i$. When we stack all 4 shares (see Fig. 4.5(e) and 4.8(e)), the best contrast value $[(1+2+3+3)-\text{Max}\{1,2,3,3\}]/(3 \times 3) = 2/3$ for this $\{1,2,3,3\}$ -setting is obtained. In the first experiment, the input is the binary logo image shown in Fig. 4.3, and it is shared by the proposed scheme. Fig. 4.4 shows the $n = 4$ shares that look noisy. The shares in Fig. 4.4(a), 4.4(b), 4.4(c), and 4.4(d) are with weights $w_1 = 1$, $w_2 = 2$, $w_3 = 3$, $w_4 = 3$, respectively. Each share with heavier weight is darker than the one with lighter weight; because each share with lighter weight is generated with fewer black elements in the expanded blocks (see Figures 4.1 and 4.2). Therefore, the weights of the shares are roughly equivalent to their darkness. Fig. 4.5 shows the stacking results of various combinations. Fig. 4.5(a) is the result of stacking Fig. 4.4(a) and 4.4(b); Fig. 4.5(b) is the result of stacking Fig. 4.4(c) and 4.4(d); Fig. 4.5(c) is the result of stacking Fig. 4.4(a)-(c); Fig. 4.5(d) is the result of stacking Fig. 4.4(b)-(d); and Fig. 4.5(e)) is the result of stacking Fig. 4.4(a)-(d). An analogous experiment is done on a halftone [11] image Lena (Fig. 4.6). The experimental results are shown in Fig. 4.7-4.8. Recall that Shares 1, 2, 3, and 4 are with weights $w_1 = 1$, $w_2 = 2$, $w_3 = 3$,



and $w_4 = 3$, respectively. These 4 shares look just like the 4 shares in Fig. 4.4. Fig. 4.8 shows the stacking results for the halftone image Lena. Fig. 4.8(a) is when we stack Shares 1 and 2; Fig. 4.8(b) is when we stack Shares 3 and 4; Fig. 4.8(c) is when we stack Shares 1, 2, and 3; Fig. 4.8(d) is when we stack Shares 2, 3, and 4; and Fig. 4.8(e) is when we stack all 4 shares.

nctu
cis



Figure 4.3. A binary logo image.

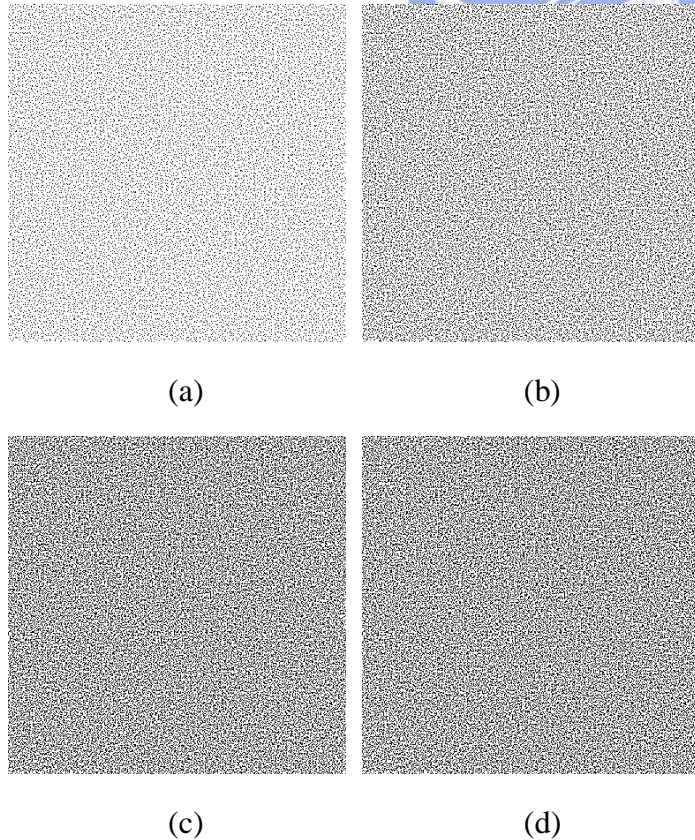


Figure 4.4. The $n = 4$ shares for the logo image.

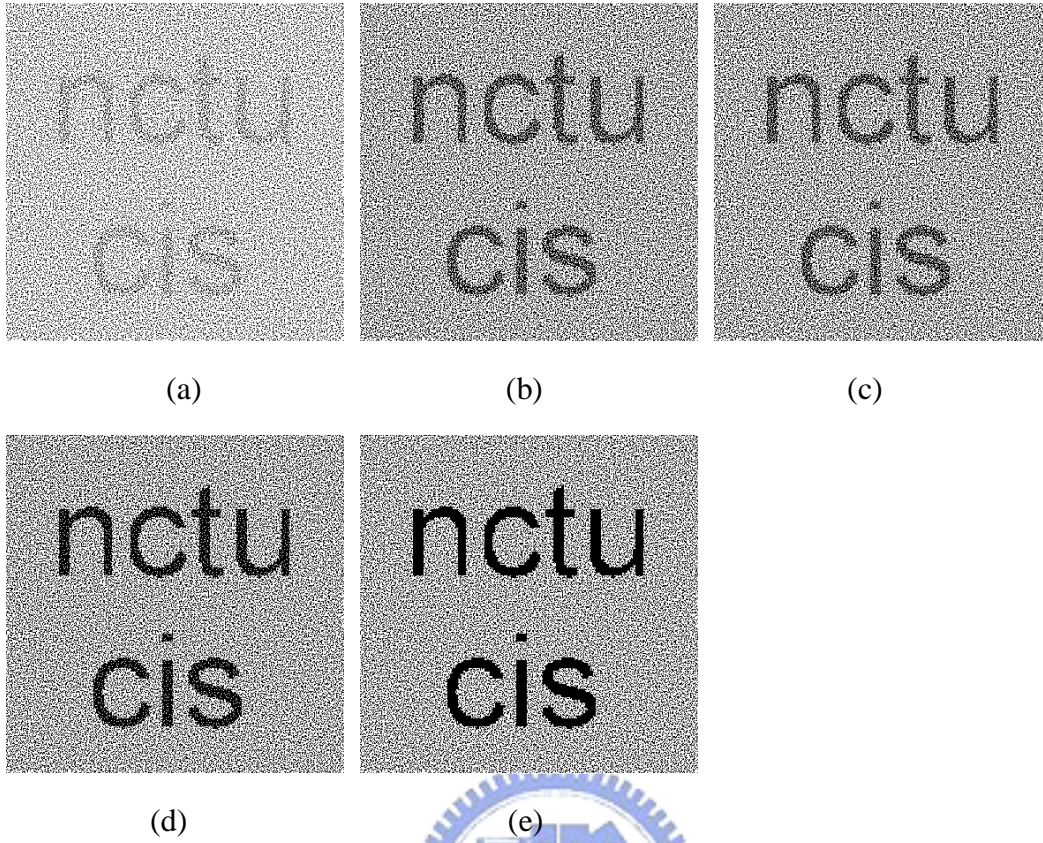
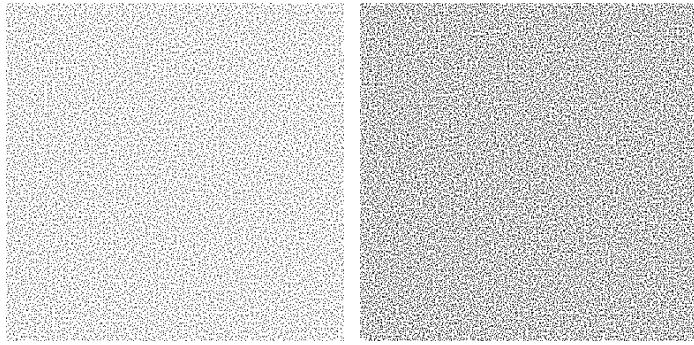


Figure 4.5. The stacking results for the logo image.

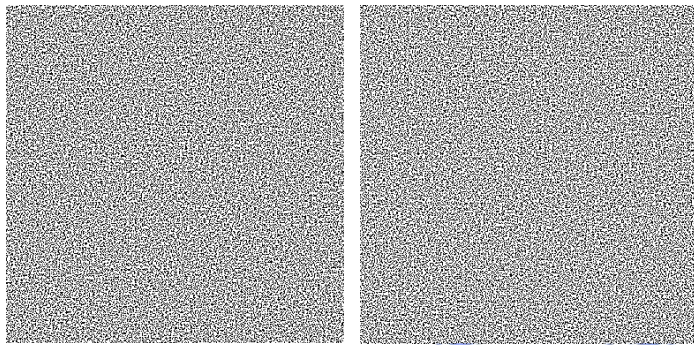


Figure 4.6. A halftone image Lena.



(a)

(b)

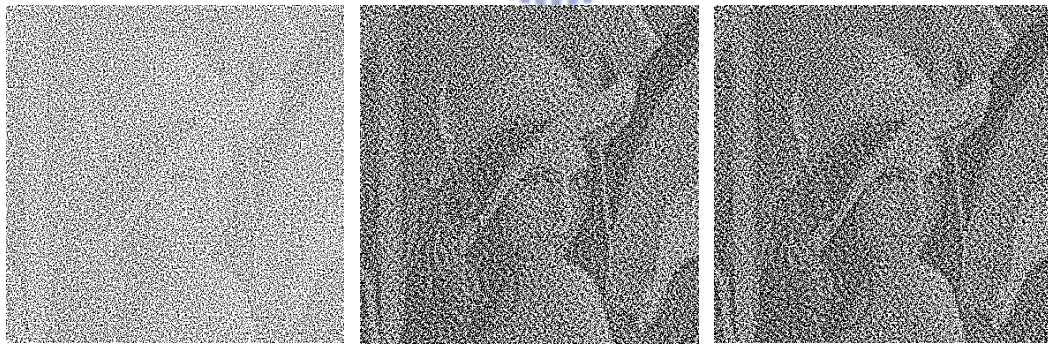


(c)

(d)



Figure 4.7. The $n = 4$ shares for the halftone image Lena.

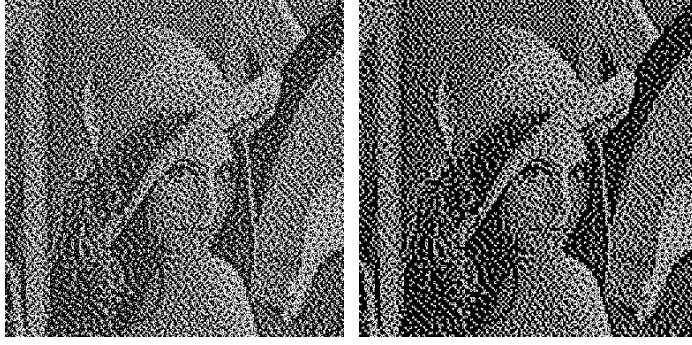


(a)

(b)

(c)

Figure 4.8. The stacking results for the halftone image Lena.



(d)

(e)

Figure 4.8. (Continued)

4.3 Discussion

The contrast of stacking the shares in a subset S is $\frac{\sum_{t_i \in S} w_i - \text{Max}_{t_i \in S} w_i}{p \times q}$. Therefore, if

each weight is set to 1, and all n shares are stacked, then the contrast can be as high as $\frac{n-1}{p \times q}$, which can be rewritten as $\frac{n-1}{n}$ if the block size $p \times q$ is designed to be n .

For example, if $n = 9$ shares are generated, the block size is 3×3 , and $1 = w_1 = w_2 = \dots$

$= w_9$, then the contrast of stacking all nine shares is $\frac{8}{9}$. Likewise, if $n = 4$ shares are

generated, the block size is 2×2 , and $1 = w_1 = w_2 = w_3 = w_4$, then the contrast of

stacking all four shares is $\frac{3}{4}$. Notably, according to the definition of the contrast value

(see [HKS 00]), no designer can get a contrast value higher than 1. In our design, as

the value of n increases, our contrast value $\frac{n-1}{n}$ can become very close to the

optimal value 1.

As a conclusion, there are several characteristics in the proposed VC scheme: (1)

fault-tolerant recovery, i.e. some shares can be absent; (2) adjustable contrast, i.e. the

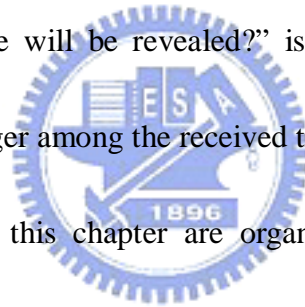
clarity of the stacking result is proportional to the difference between the maximal weight and the total weights (both measured using the shares currently available on the scene); (3) progressive contrast, i.e. it always gives a better contrast if we add one more not-yet-stacked share to the current set of the shares being stacked; Note that Property (1) can be found in many VC schemes; but Properties (2)-(3) are less common. Also note that: (4) the contrast formula $\frac{\sum_{t_i \in S} w_i - \text{Max}_{t_i \in S} w_i}{p \times q}$ is an explicit and convenient formula for each user of the proposed method, for he can use it to analyze and determine the weight values before designing his own version.



Chapter 5

Progressive viewing of multiple secret images: a VC approach

In this chapter, we propose a new VC scheme for progressive viewing, and the scheme shares k multi-secret images among $k+1$ transparencies. The method is a kind of VC using access structure. The forbidden set contains only subsets of which each contains single transparency. All other subsets, containing at least two transparencies, can reveal at least one of the secret images after stacking them. The answer to the question “which secret image will be revealed?” is totally up to the transparency whose weight is relatively larger among the received transparencies.



The remaining parts of this chapter are organized as follows: the proposed method is stated in Sec. 5.1; the experimental results are shown in Sec. 5.2; the conclusions are described in Sec. 5.3.

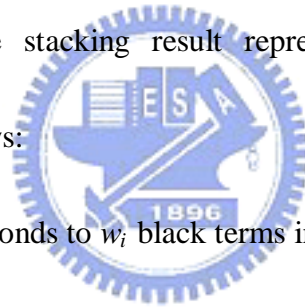
5.1 The proposed method

The proposed scheme shares k binary secret images S_1, S_2, \dots, S_k to generate a basic transparency and k weighted transparencies. The k transparencies T_1, \dots, T_k are with weights w_1, w_2, \dots, w_k , respectively, in increasing order: $w_1 < w_2 < \dots < w_k$. The basic transparency T_0 is with weight $w_0 (= w_1)$. Every corresponding binary pixels of the secret images are encoded into blocks t_0, t_1, \dots, t_k that belongs to T_0, T_1, \dots, T_k ,

respectively. Let the $p \times q$ be the block size. Therefore, each transparency will be $p \times q$ times bigger than the secret image in size.

In the max-weight dominance multi-secret image sharing system, stacking transparency T_i and some of T_j ($j < i$) together, the secret image S_i is revealed. Before stating the encoding algorithm, the stacking result representing the secret images should be defined, in detail, the original secret image pixel (“white” and “black”) should be defined as the number of black terms in the block of stacking results representing S_1, S_2, \dots, S_k .

For each block of the stacking result representing S_i , the corresponding brightness is defined as follows:



1. “White” pixel of S_i corresponds to w_i black terms in the block.

2. “Black” pixel of S_i corresponds to the following two cases:

($i < k$): at least (usually more than) w_i black terms and no more than w_{i+1} black terms.

($i = k$) at least (usually more than) w_i black terms and no more than $p \times q$ black terms.

For example, $\{S_1, S_2\}$ is the set of secret images. Therefore, T_0, T_1, T_2 are generated with weight $w_0(=w_1)$, w_1 , and w_2 . If the block of the stacking result of T_0 and T_1 has w_1 black terms, then it corresponds to a “white” pixel of S_1 . If the block of the stacking

result of T_0 and T_1 has w_2 black terms, then it corresponds to a “black” pixel of S_1 .

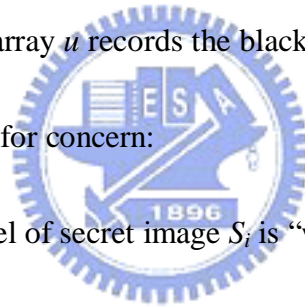
Now, we state the encoding process. For each corresponding blocks among T_0 , T_1 , ..., T_k , the block of basic transparency T_0 is generated first, the corresponding block of T_1 is generated then, ..., at last the corresponding block of T_k is generated.

Let t_0, t_1, \dots, t_k be an array to record each corresponding block of the transparencies T_0, T_1, \dots, T_k , respectively, and u be an array to record the accumulative terms during the work of generating the corresponding block of the transparencies T_0, T_1, \dots, T_k . First,

for generating t_0 , $w_0 (= w_1)$ terms are randomly assigned to black and others are assign

to white. After generating t_0 , array u records the black terms of t_0 . For generating $\{ t_i |$

$1 \leq i \leq k\}$, there are two cases for concern:



Case 1: the corresponding pixel of secret image S_i is “white”

Let g be a number of nonzero terms of u . For these corresponding terms of t_i , they are assigned to be black. Moreover, $w_i - g$ terms are randomly selected from zero-terms of u , and for these corresponding terms of t_i , they are assigned to black.

The rest undefined terms of t_i are assigned to white. After generating t_i , the corresponding terms of u increase according to the black terms of t_i .

Case 2: the corresponding pixel of secret image S_i is “black”

Case 2.1: $i < k$,

Assign the less w_i terms from the top w_{i+1} terms of u . For all

corresponding terms of t_i , they are assigned to black. After generating t_i ,

the corresponding terms of u increase according to the black terms of t_i .

Case 2.2: $i = k$

Assign the less w_i terms from u . For all corresponding terms of t_i , they are assigned to black.

The following example shows how to share three secret images S_1 , S_2 , and S_3 .

For each pixel position, we inspect the three corresponding pixel-values (b_1, b_2, b_3) taken from (S_1, S_2, S_3) , where b_i is from S_i , and create a block for each of the $1+3=4$

transparencies T_0, T_1, T_2 , and T_3 . Assuming the weights for these transparencies are $w_0 = w_1 = 3$, $w_2 = 5$, and $w_3 = 7$. Let the size expansion of each transparency be 3×3 .

The pixel-values (b_1, b_2, b_3) taken from (S_1, S_2, S_3) must be in the following 8 cases:

$(W, W, W), (W, W, B), (W, B, W), (W, B, B), (B, W, W), (B, W, B), (B, B, W),$ and $(B,$

$B, B)$. Here, W means “white” and B means “black”. Table 5.1 shows the eight cases

of this example. Without the loss of generality, we only discuss two cases (W, B, W)

and (B, W, B) .

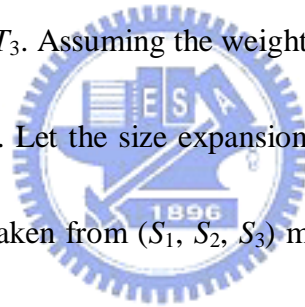



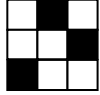
Table 5.1 Eight cases of encoding

S_1	S_2	S_3	t_0	t_1	t_2	t_3
W	W	W				
W	W	B				
W	B	W				
W	B	B				
B	W	W				
B	W	B				
B	B	W				
B	B	B				


In the case (W, B, W), t_0 is first generated by randomly selecting $w_0 = 3$ black

term, as  . and u records accumulative term as $\{0, 1, 0, 0, 0, 1, 1, 0, 0\}$. To

generate t_1 (with $w_1 = 3$), the black term of t_0 is followed. Then t_1 is generated as

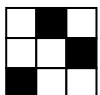
 , and then u records accumulative terms as $\{0, 2, 0, 0, 0, 2, 2, 0, 0\}$. To


generate t_2 , the less $w_2 = 5$ terms are selected among the top $w_3 = 7$ terms of u , thus, only four zero-terms can be selected and additional one term should be randomly

selected from value-2-terms of u . Therefore, t_2 is generated as  , and u records


accumulative terms as $\{1, 2, 0, 1, 1, 3, 2, 0, 1\}$. To generate t_3 , there should be $w_3 = 7$ black terms, and the number of non-zero-terms of u is equal to 7. Therefore, t_3 is

generated as  . Now, the stacking results are shown below: the stacking result

of “ t_0 and t_1 ” is  , only 3 black terms, thus it represents “white” to the pixel of


S_1 ; the stacking result of “ t_1 and t_2 ” and “ t_0, t_1 and t_2 ” is  , 7 black terms,

representing “black” to the pixel of S_2 (more than 5 black terms); the stacking result


of t_3 and subset (t_0, t_1, t_2) is  , 7 black terms, thus it represents “white” to the


pixel of S_3 .

In the case (B, W, B), t_0 is also generated by randomly selecting $w_0 = 1$ black term,

as  . and u records accumulative term as $\{0, 0, 0, 1, 0, 1, 0, 1, 0\}$. To generate

t_1 , the less $w_1 = 3$ terms are selected among the top $w_2 = 5$ terms of $u(j)$'s; thus, $w_2 - w_1 = 2$ terms are selected from zero-terms of u and remaining one term are selected from

value-1-term of u . Therefore, t_1 is generated as , and u records accumulative terms as $\{0, 1, 0, 2, 0, 1, 1, 1, 0\}$. To generate t_2 , there should be $w_2 = 5$ black terms, and the number of non-zero terms of u is equal to 5. Therefore t_2 is generated as

, and u records accumulative terms as $\{0, 2, 0, 3, 0, 2, 2, 2, 0\}$. To generate t_3 ,

the less $w_3 = 7$ terms are selected among the $u(j)$'s. Therefore, four zero-terms of u are selected, and then three terms are randomly selected from value-2-terms of u . The t_4 is

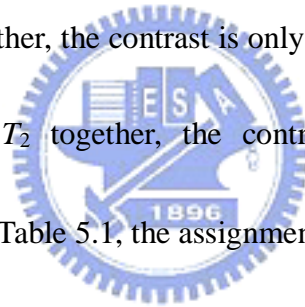
generated as .

Notably, our method can be designed with fault-tolerant property by assigning some of S_1, \dots, S_k to be the same images. For example, $S_1 = S_2 = S_3$, and $S_4 = S_5$. The damage or lost of any two of (T_0, T_1, T_2, T_3) never affect the revealing of S_1 , and the damage or lost of any one of (T_4, T_5) and any three of (T_0, T_1, T_2, T_3) never affect the revealing of S_4 .

If we adequately assign the weights, quality control is also available. Obviously, when T_0, T_1, \dots, T_i are stacked together, the contrast is $\frac{w_{i+1} - w_i}{p \times q}$ (if $i = k$, then $w_{i+1} =$

$p \times q$). Therefore, if the value of $w_{i+1} - w_i$ is larger, then quality of stacking result is better. For example, there is two secret images S_1 and S_2 and $p \times q = 3 \times 3 = 9$. If the stacking result representing S_1 (Lena) is not easier to be distinguish than that representing S_2 (NCTU-logo), the weights can be assigned as $w_0 = w_1 = 4$, and $w_2 = 8$. When stacking T_0 and T_1 together, the contrast is $\frac{w_2 - w_1}{p \times q} = \frac{8 - 4}{9} = \frac{4}{9}$. However, when stacking T_0 , T_1 and T_2 together, the contrast is only $\frac{p \times q - w_2}{p \times q} = \frac{9 - 8}{9} = \frac{1}{9}$. If the stacking result representing S_2 (Lena) is not easier to be distinguish than that representing S_1 (NCTU-logo), the weights can be assigned as $w_0 = w_1 = 4$, and $w_2 = 5$.

When stacking T_0 and T_1 together, the contrast is only $\frac{w_2 - w_1}{p \times q} = \frac{5 - 4}{9} = \frac{1}{9}$. However, when stacking T_0 , T_1 and T_2 together, the contrast is $\frac{p \times q - w_2}{p \times q} = \frac{9 - 5}{9} = \frac{4}{9}$. Moreover, for the example of Table 5.1, the assignment of weights $w_0 = w_1 = 3$, $w_2 = 5$, and $w_3 = 7$ is an average contrast assignment for all secret images.



5.2 Experimental results

In the experiment, there are three secret images, shown in Fig. 5.1(a)-(c). Then the transparencies with weights $w_0 = w_1 = 3$, $w_2 = 5$, and $w_3 = 7$, shown in Fig. 5.2(a)-(d), respectively, are generated. Stacking Fig. 5.2(a) and (b) together, the secret words of Fig. 5.1(a) can be revealed (see Fig. 5.3). Stacking Fig. 5.2(b) and (c) together, the secret words of Fig. 5.1(b) can be revealed (see Fig. 5.4). Stacking Fig. 5.2(d) and at least one image from Fig. 5.2(a)-(c), the secret words of Fig. 5.1(c) can

be revealed. Fig. 5.5 shows the result of stacking all generated transparencies in Fig.

5.2.

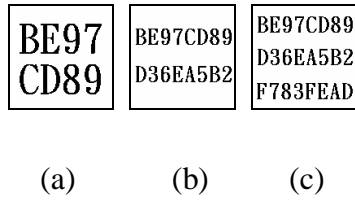


Figure 5.1. The secret images.

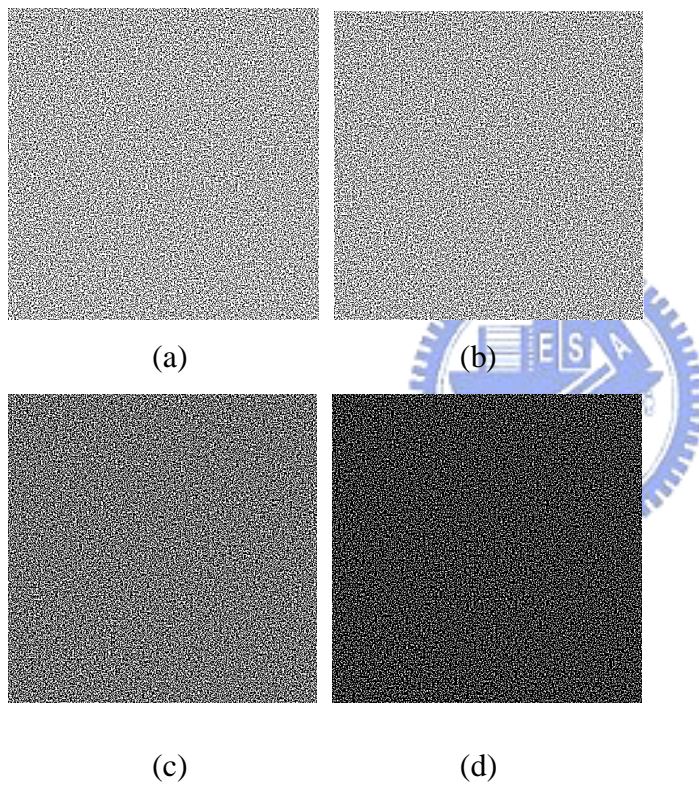


Figure 5.2. The generated transparencies ((a) basic transparency, (b) the transparency with weight = 3, (c) the transparency with weight = 5, (d) the transparency with weight = 7.)

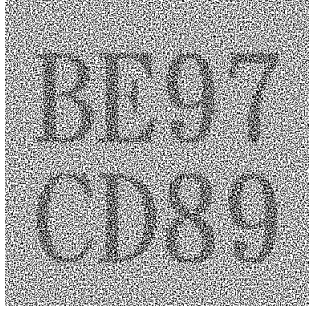


Figure 5.3. The stacking result of Fig. 5.2(a) and (b)

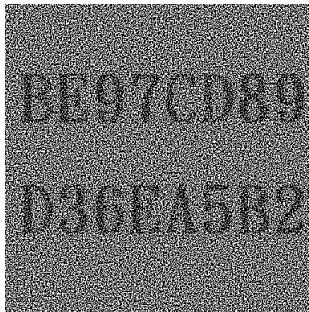


Figure 5.4. The result of stacking Fig. 5.2(b) and (c)

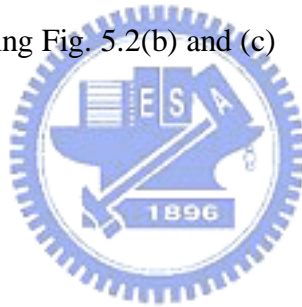


Figure 5.5. The result of stacking all transparencies in Fig. 5.2

5.3 Summary

In this chapter, we have proposed a computation-free multi-secret image sharing method. In the proposed method, the k secret images are shared using only $k+1$ transparencies. Stacking at least two transparencies together can reveal at least one of the secret images.

Chapter 6

Conclusions and Future works

6.1 Conclusions

This dissertation proposed progressive viewing of images, either by polynomial-style sharing or by VC-style sharing.

In Chapter 2, each digital image is divided into several shares of equal-significance. When the number of the available shares reaches a basic threshold r_1 , a rough version of the image can be revealed. When the number of the shares reaches another threshold r_t ($r_1 < r_t$), a better-quality image can be revealed. We can view the lossless image when r_k shares are received ($r_1 < r_k < n$, n is total number of parts). We need not worry about which part is lost or damaged, as long as r_k shares of them are alive.

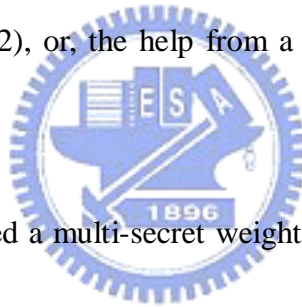
In Chapter 3, a progressive method to view vector-quantized images is proposed. When r_k ($r_1 < r_k < n$, n is total number of shares) shares are available, the recovered image is only of VQ-quality. However, each share is much smaller than the one generated in Chapter 2. In fact, the size of each share is only $\frac{1}{VQ \text{ compression rate}}$ of the one generated by Chapter 2.

In Chapter 4, we propose a Visual Cryptograph (VC) approach for progressive viewing of a secret image in order to handle the situation of power-failure or no

computer. The image is revealed by just stacking the shares (transparencies) together.

In this chapter, the shares are with their own weights. When all shares are of equal weight, all shares are equally significant to the viewing of the images; and this is just like the systems in Chapter 2 and Chapter 3. However, when weights are unequal, we can get some special effect. For example, if we create Shares 1-4, and their weights are $w_1 = 1$, $w_2 = 2$, and $w_3 = w_4 = 3$, respectively. Then, since $w_1 + w_2 + w_3 = w_3 + w_4$, the result of stacking shares $\{1,2,3\}$ is just like that of stacking shares $\{3,4\}$.

Therefore, the holder of Share 3 can either ask for the help from two low-rank holders (the holders of Shares 1 and 2), or, the help from a high-rank holder (the holder of Share 4).



In Chapter 5, we proposed a multi-secret weighted VC system. In this system, k secret images are shared using only $k+1$ transparencies. Stacking at least two transparencies together can reveal at least one of the secret images. Moreover, among the stacked transparencies, the one with the largest weight determines which image is revealed.

Below, we show the comparison of Chapter 2 and Chapter 3 in Table 6.1, the comparison of Chapters $\{2,3\}$ and Chapters $\{4,5\}$ in Table 6.2, and the comparison of Chapter 4 and Chapter 5 in Table 6.3.

Table 6.1. The comparison of Chapter 2 and Chapter 3.

	Chapter 2 Progressive viewing of images : a sharing approach	Chapter 3 Progressive viewing of vector-quantized images : a sharing approach
Similarity	<ol style="list-style-type: none"> 1. With fault-tolerance; 2. With unbiased property; 3. Quality-control is available. 	
Dissimilarity	<ol style="list-style-type: none"> 1. The best quality of the recovered image is lossless. 2. The size of each shared result is $\frac{k}{r_1 + r_2 + \dots + r_k}$ of the one of the original image. 	<ol style="list-style-type: none"> 1. The best quality of the recovered image is of VQ-quality, thus, lossy. 2. The size of each shared result is $\frac{k}{r_1 + r_2 + \dots + r_k} \times \frac{1}{\text{compression rate}}$ of the one of the original image.

Table 6.2. The comparison of Chapters 2&3 and Chapters 4&5.

	Chapters 2 & 3	Chapters 4 & 5
Similarity	<ol style="list-style-type: none"> 1. With fault-tolerant property 	
Dissimilarity	<ol style="list-style-type: none"> 1. Unbiased 2. The decoding process needs a computer 3. The size of each share is smaller than original gray image. 4. Need memory to store the shares 5. Application in image transmission via network. 	<ol style="list-style-type: none"> 1. Biased 2. The decoding process needs no computer 3. The size of each transparency is larger than original binary secret image. 4. Need physical space to store the transparencies. 5. Application in war and game.

Table 6.3. The comparison of Chapter 4 and Chapter 5.

	Chapter 4 Progressive viewing of a secret image : a VC approach	Chapter 5 Progressive viewing of multiple secret images : a VC approach
Similarity	<ol style="list-style-type: none"> 1. The decoding process needs no computer. 2. With fault-tolerant and biased property 	
Dissimilarity	<ol style="list-style-type: none"> 1. For single secret image 2. Stacking any ($r \geq 2$) transparencies together can have the contour of one image. 	<ol style="list-style-type: none"> 1. For multi-secret image 2. Stacking any ($r \geq 2$) transparencies together can have the contour of $r - 1$ images.

6.2 Future works

Below are some suggestions to extend the proposed methods in the future.

1. In the proposed methods, the watermarking technique might be applied on the shares to identify the ownership of legal user.
2. In the topic “Progressive viewing of a secret image: a VC approach”, $(2, n)$ has limited application. Therefore, try to extend $(2, n)$ to (r, n) for more general applications.
3. The verification of the shares to avoid faked shares might be a topic worthy of study.

References

- [ABSS 96]. G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, "Visual Cryptography for General Access Structures," *Information and Computation*, Vol. 129, pp. 86-106, 1996.
- [ABSS 01]. G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, "Extended capabilities for visual cryptography," *Theoretical Computer Science*, Vol. 250, pp. 143-161, 2001.
- [BBGHPP 00]. 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.
- [BBP 02]. A. Benazza-Benyahia and J.C. Pesquet, "A unifying framework for lossless and progressive image coding," *Pattern Recognition*, Vol. 35, pp. 627-638, 2002.
- [BD 03]. N. Bourbakis and A. Dollas, "SCAN-based compression-encryption-hiding for video on demand", *IEEE Multimedia Magazine*, Vol. 10, pp. 79-87, 2003.
- [Blakley 79]. G.R. Blakley, "Safeguarding cryptographic keys," *Proceedings AFIPS 1979 National Computer Conference*, Vol. 48, pp. 313-317, New York, USA, June, 1979.
- [BS 01]. A. De Bonis and A. De Santis, "Randomness in secret sharing and visual cryptography schemes," *Theoretical Computer Science*, Vol. 314, pp. 351-374, 2001.
- [BSS 96] C. Blundo, A. De Santis and D.R. Stinson, "On the contrast in visual cryptography schemes", *J. Cryptology*, Vol.12, pp. 261-289, 1996.
- [CAM 00] B. Carlo, D. S. Alfredo and N. Moni, "Visual cryptography for grey level images," *Information Processing Letters*, Vol. 75(6), pp. 255-259, 2000.

[CCL 04]. 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.

[CH 98]. C. C. Chang and R. J. Hwang, "Sharing secret images using shadow codebooks," *Information Sciences*, Vol. 111, pp. 335-345, 1998.

[CJC 98]. 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.

[CT 01]. 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.1545-1555, 2001.

[EMsmsz03]. 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.

[FWTC 05]. J. B. Feng, H. C. Wu, C. S. Tsai, and Y. P. Chu, "A new multi-secret images sharing scheme using Lagrange' s interpolation," *The Journal of Systems and Software*, Vol. 76(3), pp. 327-339, 2005.

[HC 01] R. J. Hwang and C. C. Chang, "Hiding a picture in two pictures," *Optical Engineering*, Vol. 40, pp. 342-351, 2001.

[HCL 04] 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.

[HKS 00] T. Hofmeister, M. Krause and H. U. Simon, "Contrast-optimal k out of n secret sharing schemes in visual cryptography," *Theoretical Computer Science*, Vol. 240(2), pp. 471-485, 2000.

[Hou 03] Y. C. Hou, "Visual cryptography for color images," *Pattern Recognition*, Vol. 36, pp.1619-1629, 2003.

[LBG 80]. Y. Linde, A. Buzo, and R. M. Gray. "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, Vol. 28, pp. 84-95, 1980.

[LT 03] C. C. Lin and W. H. Tsai, "Visual cryptography for gray-level images by dithering image," *Pattern Recognition Letters*, Vol. 24, pp. 349-358, 2003.

[MM 02]. D.Mukherjee and S.K. Mitra, "Successive refinement lattice vector quantization," *IEEE Trans. Image Processing*, Vol. 11(12), pp. 1337 – 1348, 2002.

[NK 88]. N. M. Nasrabadi and R.A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, Vol. 36(8), pp. 957-971, 1988.

[NS 95] M. Naor and A. Shamir, "Visual Cryptography", *Advances in Cryptology – Eurocrypt 94*, Springer, Berlin, pp. 1-12, 1995.

[PAK 99]. 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.

[Shamir 79]. A. Shamir, "How to share a secret," *Communication of the ACM*, Vol. 22(11), pp. 612-613, 1979.

[TCC 02]. C. S. Tsai, C. C. Chang, and T. S. Chen, "Sharing multiple secrets in digital images," *The Journal of Systems and Software*, Vol. 64(2), pp. 163-170, 2002.

[TL 02]. C.C. Thien and J.C. Lin, "Secret image sharing," *Computer & Graphics*, Vol. 26, pp. 765-770, 2002.

[WC 05]. H. C. Wu and C. C. Chang, "Sharing visual multi-secrets using circle shares," *Computer Standards & Interfaces*, Vol. 28, pp. 123-135, 2005.

[WTL 04]. Y. S. Wu, C.C. Thien, and J.C. Lin, "Sharing and hiding secret images with size constraint," *Pattern Recognition*, Vol. 37(7), pp. 1377-1385, 2004.

[YC 05] 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.



Vita

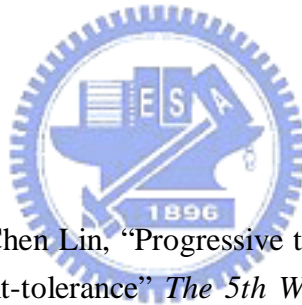
Shang-Kuan Chen was born in 1972 in Taiwan, Republic of China. He received his BS degree in applied mathematics in 1994 from Fu Jen Catholic University, Taiwan. In 1998, he received his MS degree in applied mathematics from National Chiao Tung University, Taiwan. Since 1999 he has been studying toward a PhD degree in the Computer Science Department of National Chiao Tung University. His research interests include visual cryptography, data hiding, and interconnecting network.



Publication List of Shang-Kuan Chen

A. Journal papers

1. Shang-Kuan Chen and Ja-Chen Lin, "Fault-tolerant and Progressive Transmission of vector-quantized images" *WSEAS Transactions on Signal Processing*, Vol. 2(5), pp.787-793, 2006.
2. Shang-Kuan Chen and Ja-Chen Lin, "Fault-tolerant and progressive transmission of images" *Pattern Recognition*, Vol. 38, pp. 2466-2471, 2005.
3. Shang-Kuan Chen, Frank Huang, and Yu-Chi Liu, "Some Combinatorial Properties of Mixed Chordal Rings," *Journal of Interconnection Networks*, Vol. 4(1), pp. 3-16, 2003.



B. Conference papers

1. Shang-Kuan Chen and Ja-Chen Lin, "Progressive transmission of vector-quantized images with security and fault-tolerance" *The 5th WSEAS International Conference on Signal Processing (SIP '06)*, Istanbul, Turkey, 27-29, 2006 May.
2. Shang-Kuan Chen, "Fault-tolerant and fast image transmission using VQ" *The 18th IPPR Conference on Computer Vision, Graphics and Image Processing*, Taipei, Taiwan, 2005 August.
3. Shang-Kuan Chen and Ja-Chen Lin, "Proactive Secret Image Sharing," *The 17th IPPR Conference on Computer Vision, Graphics & Image Processing*, Hwa-Len, Taiwan, 2004 August.
4. Frank Huang and Shang-Kuan Chen, "The 1.5-loop network and the mixed 1.5-loop network" *SIROCCO 2000*, Cattedra Bernardiniana Convento di S. Bernardino. Via V. Veneto, Italy, pp. 297-306, 2000 May.