Chapter 2

Fault-Tolerant and Progressive Image Sharing Using Vector Quantization

In this chapter, we proposed two methods about image sharing. One is the fault-tolerant version of vector quantization (VQ) style image sharing; and the other is the lossless progressive version of VQ-style image sharing. In the first method, the secret image is compressed by VQ and shared among *n* shadows. The *n* shadows are hidden into the last bit-plane of codebooks, which look like ordinary images. After collecting *r* shadows ($r <= n$), the codebooks and the code indices of the secret image can be computed. The secret image can then be recovered. No information can be revealed if less than *r* shadows are gotten. In the second method, a lossless progressive image sharing method is proposed. As the number of collected shadows increases, the quality of recovered secret image will be improved, too. Finally, a lossless secret image can be revealed when all *n* shadows are acquired. The organization of this chapter is described as follows. Section 2.1 presents the introduction, which contains the motivation and some sharing methods nowadays. Section 2.2 presents the proposed methods. Section 2.3 shows experimental results. And the summary is in Section 2.4.

2.1 Introduction

There are many methods of secret image sharing nowadays. For example, the user can share the secret image to some shadows by the polynomial equations [1]; or can use the method of visual cryptography to share the transparencies [10]. Vector quantization [2] is one of the approaches used in secret image sharing. By the codebook in VQ, user computes the code indices of secret image. Because the size of code indices is smaller than original secret image, secret image can be compressed such that user can hide it more easily than hide the pixel values of the image directly. Besides, because there are only two colors (black and white) can be used in the visual cryptography, the quality of the recovered image using VQ , which can use many different colors, is more acceptable than the one using visual cryptography. In the literature, there are two major ways about VQ-style image sharing. The first one is that the user provides some host images, and the codebooks for VQ are constructed using the first *m* bit-planes of these host images [21] (assume that the gray levels of the pixels of the host images are in the range 0~255, i.e., there are 8 bit-planes in every host image). The value of *m* depends on the number of the host images and the sizes of the secret image and host images. Then the user compresses the secret image using these codebooks to get code indices. Finally, the code indices are hidden into the last (8-*m*) bit-planes of these host images. In the recovery step, the user must collect all of the host images. By computing the code indices of the secret image, which lie in the last (8-*m*) bit-planes of the host images, and the codebooks, which are the first *m* bit-planes of the host images, the user can retrieve the secret image. Because the host images are only modified the last (8-*m*) bit-planes, the impact to the host images is small; and besides, the quality of retrieved secret image is also not bad. However, there is a drawback in this method, namely, if one of the host images is damaged or lost, the user cannot get codebook, and hence, recovering the secret image becomes impossible. The other VQ-style sharing method is that the user shares the codebook among the *n* shadow images by (*r*, *n*) threshold scheme proposed by Shamir [20]; and the code indices of secret image are kept in a local storage unit. Thus if the user gets any r of the n shadow images, the codebook can be recovered by these r shadows. Using the recovered codebook and the code indices saved in the local storage unit, the user can get the secret image. Unlike the first method, the second method is fault-tolerant. The codebook can always be recovered as long as the number of broken shadows is no more than n-r. However, the quality of the secret image recovered using in [20] is not good as the one using in [21]. Therefore, by modifying the first method, we propose here a VQ-style sharing method which is fault-tolerant and with good recovered quality.

Progressive image sharing is another useful sharing method. To begin our design, we review the progressive VQ introduced in [14] and the progressive image sharing method proposed in [15]. By computing and hiding the difference of the original image and the compressed image, the method in [14] can achieve the effect of progressive reconstruction. Then, after collecting all shadow images, the secret image was revealed losslessly in [14]. However, the order of collecting the shadow images was important in [14], since the difference was saved in order. For example, if the second and the third shadow images are gotten, nothing can be acquired because there exist only the information about difference. In [15], a progressive image sharing method by using the property of DCT frequency domain is proposed. According to the frequency bands, user can share the data of different frequency bands in different threshold. When more shadows are collected, the more information of frequency can be gotten, that makes the better quality of secret image. However, this method is not lossless. The secret image is still damaged though all the shadows are received, and the adjustment of the threshold is not convenient, too. Thus we propose here a method that the rough version of the secret image can be gotten when *any* r_1 shadows are collected. Then, with the increasing of the number of the shadows being collected, the quality of the recovered secret image will be better and better. Finally a secret image will be recovered losslessly when all of the shadows are received.

2.2 The Proposed Secret Image Sharing Method of VQ-style

In Section 2.2.1, we propose the fault-tolerant VQ-style sharing method, which includes the hiding phase and reconstruction phase. In Section 2.2.2, the lossless progressive image sharing method of VQ-style is presented.

2.2.1 Fault-tolerant VQ-style Image Sharing

Without loss of generality, suppose that there are 5 shares in this method, and the number of codebooks needed in the method is 3. That is, the *n* in Thien's (*r*, *n*) threshold scheme [7] equals 5; and the value of *r* is 3. In other words, collecting only 3 images will be enough reveal the secret image. In the *n*=5 host images, which all look like normal images, only the first three of them are using as the codebooks. Just like the original method in [21] did, the code indices of secret image are then computed using these three codebooks. And the code indices are shared by Thien's (*r*, *n*) threshold scheme. Then the five generated shares are hidden in the last (8-*k*) bit-planes of the five host images, respectively. As for our fourth and fifth host images, which are not codebooks, stored not only the shares of the code indices, but also the mixed information of the three codebooks. The mixed information of the three codebooks is calculated from the original three codebooks using the exclusive-OR operations

$$X_{1,j} = A_j \oplus B_j, \tag{2.1}$$

$$X_{2,j} = B_j \oplus C_j, \tag{2.2}$$

$$Y_{1,j} = A_j \oplus B_j \oplus C_j. \tag{2.3}$$

(Note that A, B and C denote the first, second and third codebooks, respectively, and j means the serial number of the code index in its own codebook. For example, A_{36} denotes the 36th code index in the first codebook; X_1 and X_2 denote the two numbers to be hidden in the fourth host image; correspondingly, Y_1 denote the information to be hidden in the fifth host image.)

The exclusive-OR operations can be set arbitrarily as long as they guarantee that the three codebooks can be recovered when any 3 shadow images are received. For example, the information generated using Equations (2.4) ~ (2.7), instead of Equations (2.1) ~ (2.3), can also be used as the hidden data in the fourth and fifth host images.

$$X_{1,j} = A_j \oplus B_j \tag{2.4}$$

$$X_{2,j} = B_j \oplus C_j \tag{2.5}$$

$$Y_{1,j} = A_j \oplus C_j \tag{2.6}$$

$$Y_{2,j} = B_j \oplus C_j \tag{2.7}$$

After hiding the information of codebooks, the fourth and the fifth host images will look like the original host images if the size of codebook is not too huge. For example, in a 512 x 512 host image X, it only need about one bit-plane to hide the mixed information ((2.4) and (2.5)) of the codebooks {A, B, C} each of which contains 256 code words and each code word is 4x4=16 dimensions. The generations of the codebooks, the code indices, and the mixed information of the codebooks are shown in Figures 2.1 and 2.2.

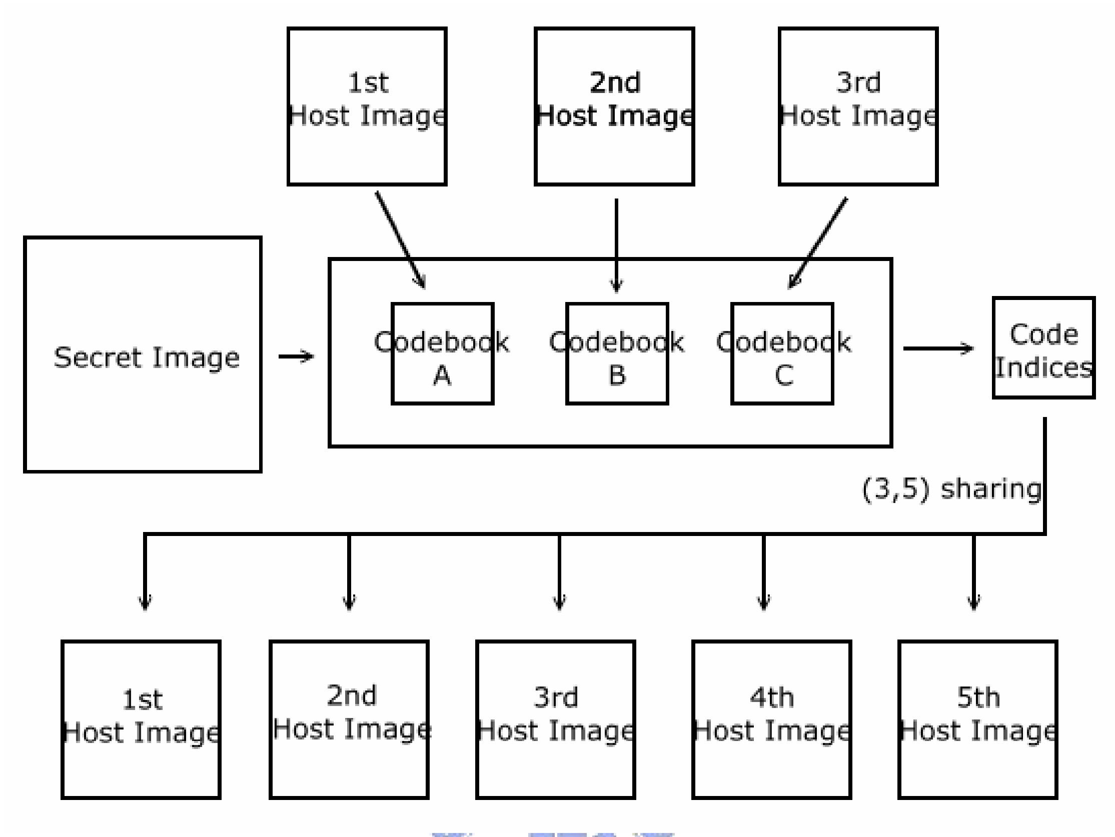


Fig. 2.1 The generations of the codebooks and the code indices.

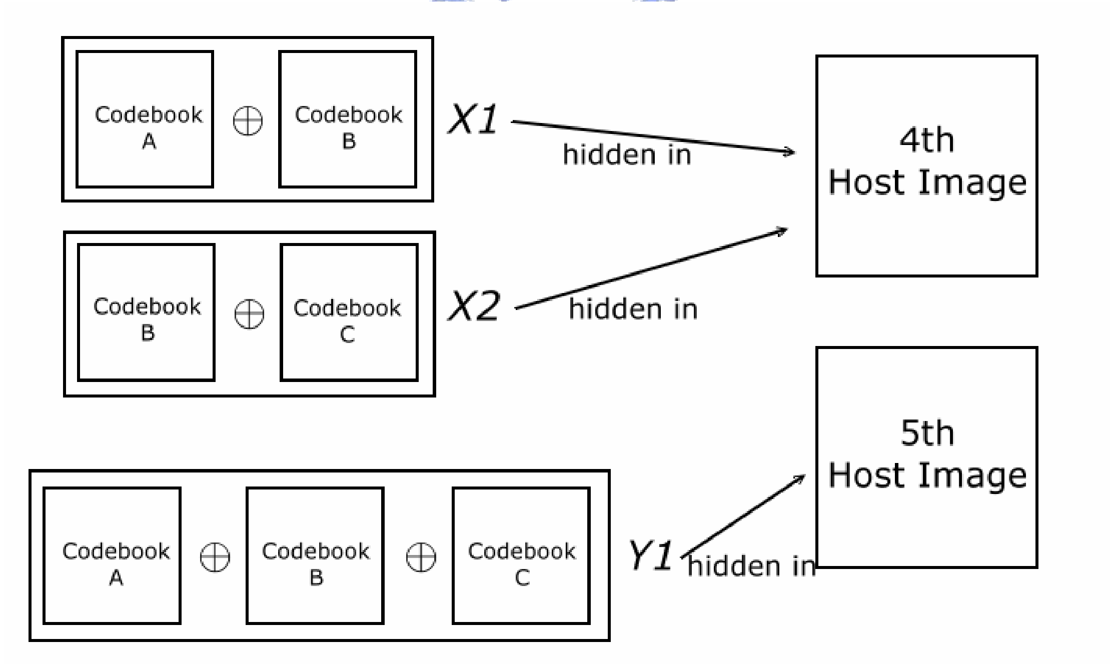


Fig. 2.2 The generation and hiding of the mixed information.

In the recovery part, only three of the five shares are needed to recover the secret image. If we collect three shadows which do not include the fourth and the fifth shadow (recalling that the fourth and the fifth shadows are the shadows that contain the mixed information of the codebooks, and the mixed information is calculated by the exclusive-OR operations), the code indices can then be revealed by the last (8-*k*) bit-planes. Then, using these code indices and the three codebooks which are the first *k* bit-planes of the three shadows, the secret image can be recovered. However, if one of the three collected shadows is the fourth or fifth shadow, then only two codebooks can be generated directly using the first *k* bit-planes of the two collected codebook shadows. As for the third codebook, it is computed by applying the exclusive-OR operations on these three shadows. For example, if the exclusive-OR operations are the equations (2.1) ~ (2.3) and the three collected shares are the first, the second and the fourth shadows, respectively, then the operation that generates the third codebook is

$$C_j = B_j \oplus X_{2,j}. \tag{2.8}$$

If the three collected shares are the second, the third and the fifth shadows, then the operation to calculate the first codebook becomes

$$A_j = B_j \oplus C_j \oplus Y_{1,j}. \tag{2.9}$$

Moreover, if two of the three collected shadows are the fourth shadow and the fifth shadows, then only one codebook can be generated directly. Then, the other two codebooks are calculated by the exclusive-OR operations similarly. Thus all three codebooks are gotten (either directly, or after some computation), and the code indices can be extracted from the last (8-*k*) bit-planes. The secret image can be recovered in the same way. As a remark, if only one or two shadows are collected; then the code indices cannot be computed even if the information of the codebooks is sufficient, so nothing about the secret image will be revealed.

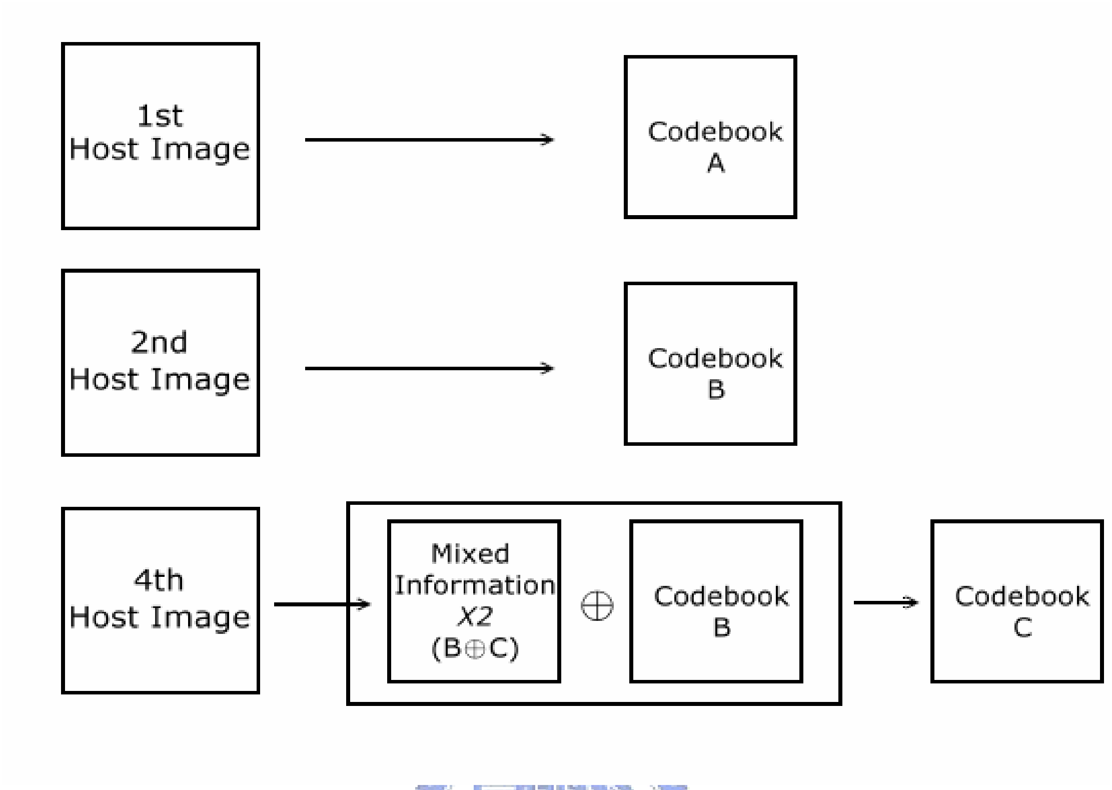


Fig. 2.3 The operations when receiving the 1st, 2nd, and 4th shadows.

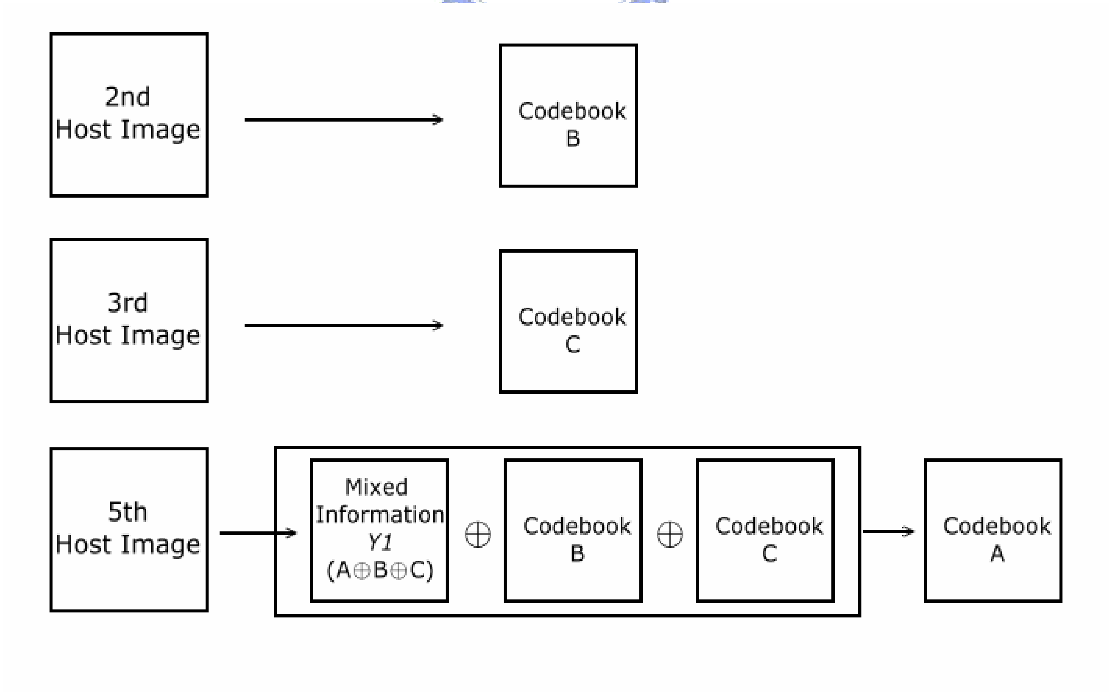


Fig. 2.4 The operations when receiving the 2nd, 3rd, and 5th shadows.

2.2.2 Lossless Progressive Image Sharing by Vector Quantization

Like the original VQ sharing method in [14], the code indices of the secret image are first calculated using a given codebook "c". We call the set of the code indices as i_1 (with the letter i representing the word indices), and the lossy secret image reconstructed by i_1 as s_1. To achieve the goal of recovering secret image by *any* r_1 shadows, the set i_1 is shared to create *n* shadows by Thien's (*r*, *n*) secret image with $r = r_1$. The *n* shares are then hidden in *n* host images. At the second stage, the difference image d_2 of the secret image and the reconstructed image s_1 is computed. The image d_2 is processed by VQ again using a shrunk codebook $\tilde{c}$ ($\tilde{c}$ is obtained by shrunk all codeword values of original codebook c by 10 in every dimension), and the code indices data of d_2 is denoted as i_2. Let s_2 denoted the approximated image of d_2 using code indices data i_2 and the codebook $\tilde{c}$. As mentioned above, the indices data i_2 is also shared among n shadows by Thien's method with threshold $r = r_2$. The value of r_2 can be any value more than r_1 for the purpose of progressive sharing. Then the n shadows of this stage (Stage 2) are hidden in the same *n* host images where the *n* shadows at the first stage are hidden. Similarly, at the third stage, the difference of d_2 and s_2 is evaluated and called d_3. Then, i_3, s_3 and the shadows of i_3 are generated by the same method (r_2 must be smaller than r_3, too; and the codebook being used is still the shrunk codebook $\tilde{c}$). The remaining stages are executed in the same manner. After implementing the all stages (assuming there are *k* stages in total), the last difference image which assigned as d_k is gotten at the final stage. In the final stage (the k_{th} stage), the difference image d_k is no longer processed by VQ. Instead, it is only shared by Thien's method, where the threshold r_k equals *n*, and the *n* shadows are hidden in the same *n* host images mentioned above. Lastly, the *n* host images are completed and they can be shared to internet or saved in different institutions, etc.

In the recovery phase, if any r_1 host images are acquired, the code indices i_1 can be computed. Thus a lossy secret image s_1 can be generated. With the increasing of the number of the shadow images being collected, if the number is fulfilled r_2, i_2 can be computed, which generates s_2. By adding s_1 and s_2, it will produce a better quality of secret image than s_1. When the incoming number of the shadow images comes to r_3, the recovery method of the secret image becomes to adding s_1, s_2, and s_3, which is generated from i_3 (i_3 can be recovered by using the incoming r_3 shadow images). By adding s_3, it improves the quality of the secret image by adding s_1 and s_2 alone. The methods are similar as above when there are r_4 r_5, ..., and r_{k-1} shadow images being gotten. Finally, by all of the shadow images (namely, n shadow images), d_k can be computed (recalling that d_k is shared to the n host images directly instead of applying VQ in the sharing phase). And the lossless secret image is generated by adding from s_1 to s_{k-1} and d_k. We will briefly describe every stage as follows. Fig. 2.5 shows the flowchart of the proposed method in the sharing phase.

Nomenclature:	
i	the calculated code indices image
s	the image generated by i
r	the threshold in Thien's sharing method
d_2	the difference of secret image and s_1
d_m $(m=3,4,...,k)$	the difference of d_{m-1} and s_{m-1}

Sharing Phase

I. Computer the code indices i_1 of the secret image, where i_1 can recover a lossy secret image s_1, and share i_1 to the n host images with the threshold r_1 by Thien's method.

II. Calculate the difference of the original secret image and s_1. Denominate the difference d_2 and computer the code indices i_2 of d_2. Share i_2 to the same n host images as *I.* with the threshold r_2 (note that $r_2 > r_1$), and compute the image s_2 generated by i_2.

III. Calculate d_3, which is the difference of d_2 and s_2. Then compute i_3 and s_3 as mentioned in *II.* Then, i_3 is shared to the n host images with the threshold r_3 ($r_3 > r_2$ and the n host images are still the same ones).

IV. Repeat the method mentioned above to get d, i, s, and r until the stage before the final one.

V. After calculating d_k share it to the n host images directly with the threshold n instead of doing VQ.

Recovery Phase

I. Collect r_1 shadow images to compute i_1, which can get s_1 by VQ.

II. Gather r_2 shadow images to compute i_2, which can be revealed s_2. Get a better quality of secret image than s_1 by adding up s_1 and s_2.

III. Repeat *II.* with the increasing number of the shadow images gotten.

IV. Recovery the lossless secret mage by adding from s_1 to s_{k-1} and d_k if the number of the collected shadow images arrives at n.

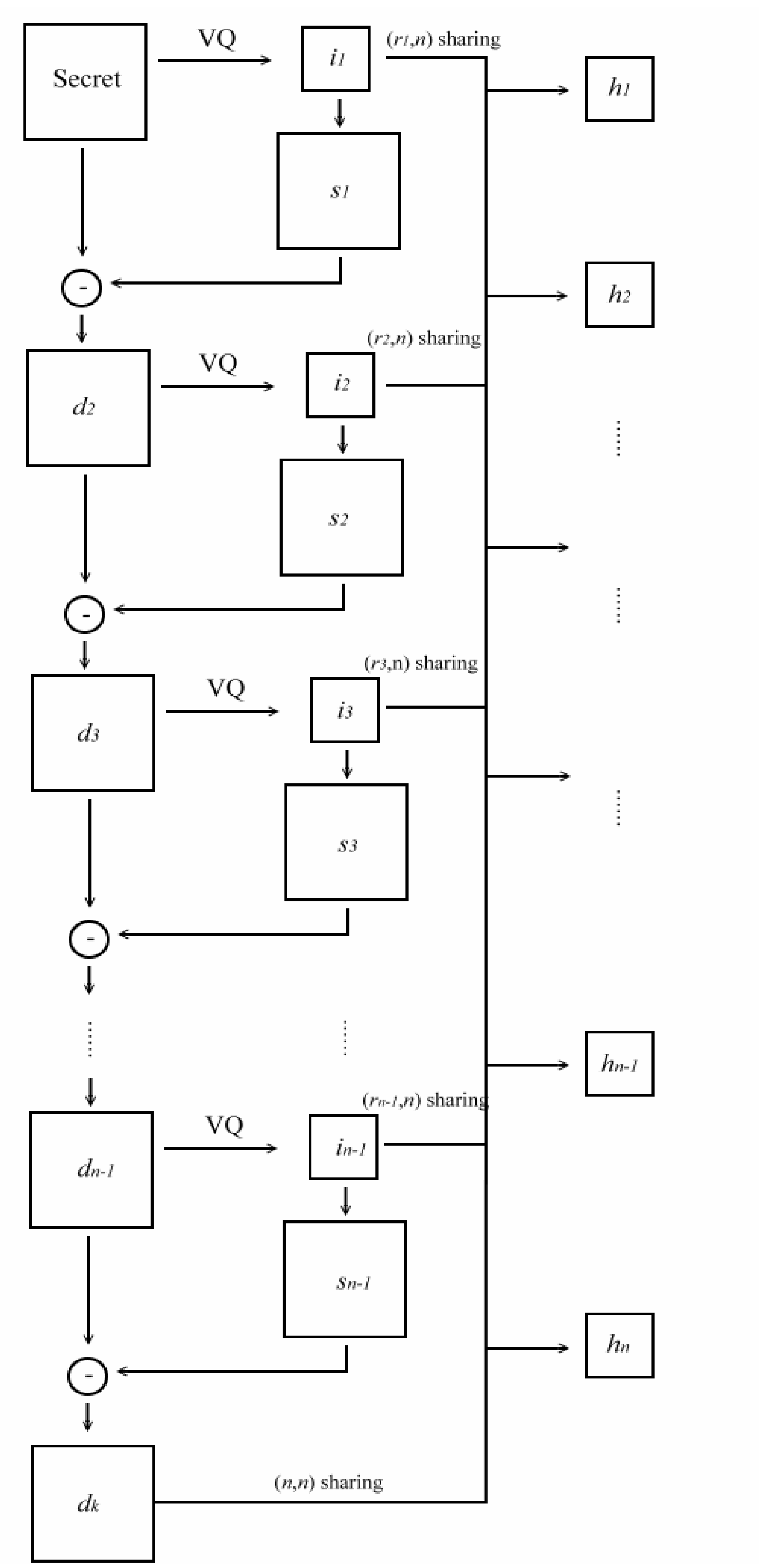


Fig. 2.5 The flowchart of lossless progressive image sharing in sharing phase.

2.3 Experimental Results

2.3.1 Fault-tolerant Image Sharing by Vector Quantization

The *r* equals 3, and *n* equals 5 in the example. Fig. 2.6 shows the original secret image "Lena" whose data size is 1024 x 1024 bytes. Fig. 2.7 (a) ~ (e) are the five host images and each one has size 512 x 512. The shadow images in which the code indices of secret image have been hidden in the last bit-plane are shown in Fig. 2.8 (a) ~ (e). Fig. 2.8 (d) and (e) contain the mixed information of three codebooks generated by Fig. 2.8 (a) ~ (c) (the information is calculated by the exclusive-OR operations). Fig. 2.9 compares the original secret image (i.e. Fig. 2.6) and the recovered secret image using any 3 shadow images. The quality of the recovered image is estimated by the peak signal-to-noise ratio (PSNR), which is defined as $PSNR = 10\log_{10}\frac{255^2}{MSE}$, where the MSE (mean-square error) is defines as $MSE = \frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}(x_{ij} - \tilde{x}_{ij})$ (note that x_{ij} and $\tilde{x}_{ij}$ are the original pixel value and the recovered pixel value, respectively, in an m x n image). The PSNR of the recovered secret image is about 34.01 dB, whose visual quality is not too bad. And the PSNRs of the 5 shadow images are 52.70, 52.71, 52.69, 48.87 and 50.34 dB, respectively.

2.3.2 Lossless Progressive Image Sharing by Vector Quantization

The secret image "Jet" of size 512 x 512 in this example is shown in Fig. 2.10. The five 512 x 512-bytes host images are shown in Fig. 2.11 (a) ~ (e). Fig. 2.12 displays the five shadow images in which the information of the secret image has been hidden. The PSNRs are 40.21, 40.41, 40.89, 40.80 and 40.87 dB, respectively. Figures 2.13, 2.14, 2.15, and 2.16 show the recovered secret images with different numbers of shadow images being received. After collecting *any* 2 shadow images, a secret image with poor quality can be revealed, which is shown in Fig. 2.13. With any 3 shadow images, a better quality of secret image can be gotten, as displayed in Fig. 2.14. Figures 2.15 and 2.16 are two recovered secret images using 4 and 5 shadow images, correspondingly. The PSNRs of the secret images recovered by any 2, 3, and 4 shadow images are 28.00, 32.30 and 35.96 dB, respectively, and the secret image recovered by using 5 shadow images is “lossless”.

2.4 Summary

In this chapter, two method of VQ-style secret image sharing are proposed. One is the fault-tolerant image sharing, and the other is the lossless progressive image sharing. In the first method, $n-r$ shadow images can be destructed since the information of codebooks and code indices can be recovered by the other r shadow images. Nothing about the secret image can be gotten when there are insufficient number of shadow images being collected. It provides a flexible method in the secret image sharing. In the other method that we proposed, the more the shadow images being gotten, the better the quality of the secret image being recovered. There is no restriction about which shadow images are gotten since the information is hidden in every shadow image. It is convenient that we don’t need to care about *which* shadow images we get, and just need to care about *how many* shadow images we collect. Besides, the shadow images are not downgraded too much (about 38 dB) after hiding the sharing information. Also note that the secret image can be revealed in a loss free manner when all shadows are collected.

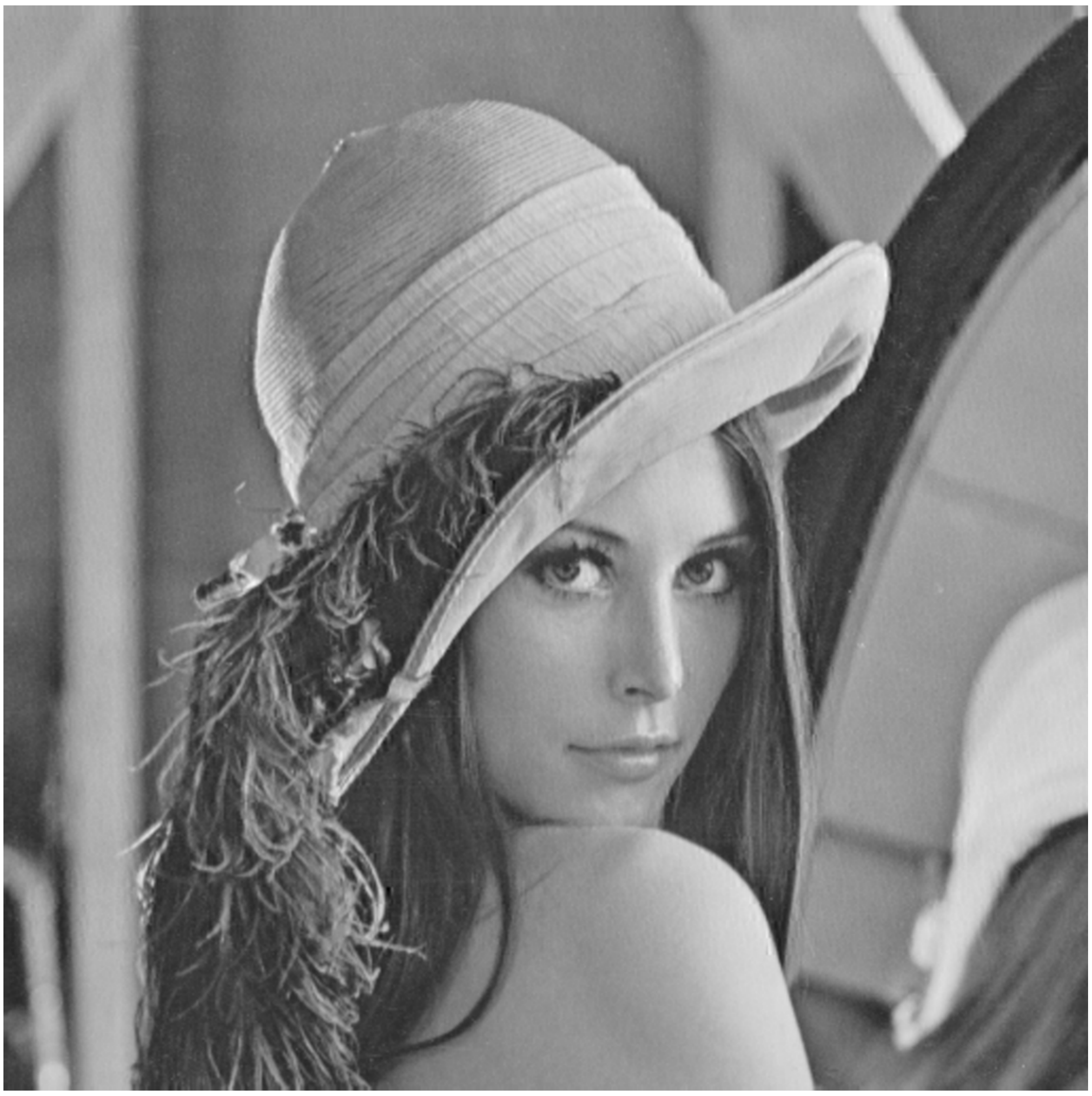
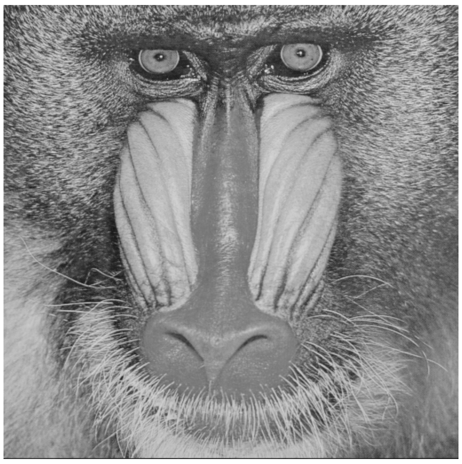
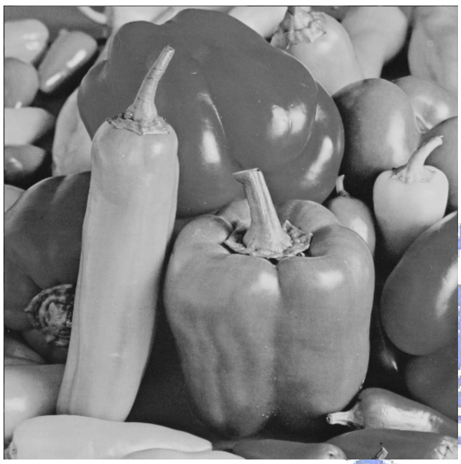


Fig. 2.6 The original 1024x1024 secret image “Lena”.

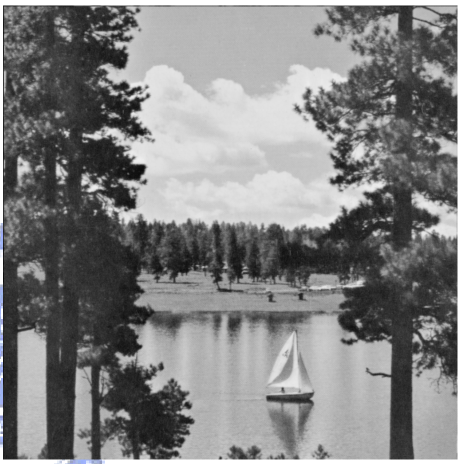
(a)



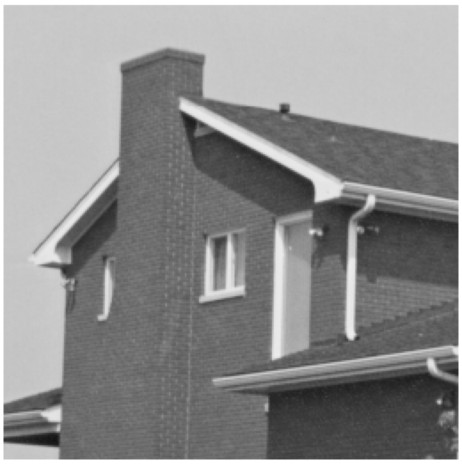
(b)



(c)



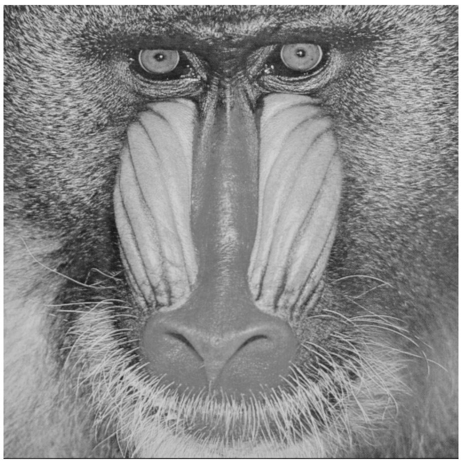
(d)



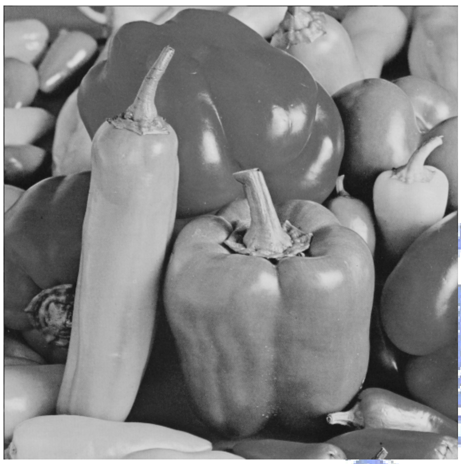
(e)

Fig. 2.7 The five host images of size 512x512.

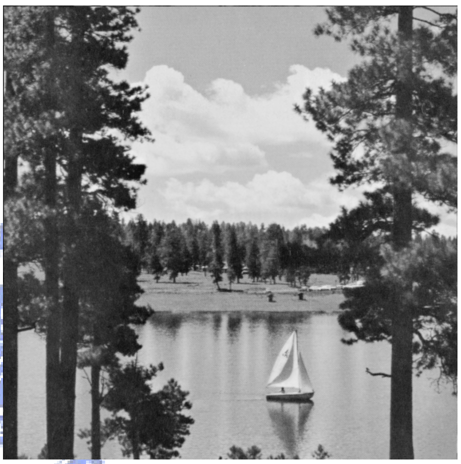
(a)



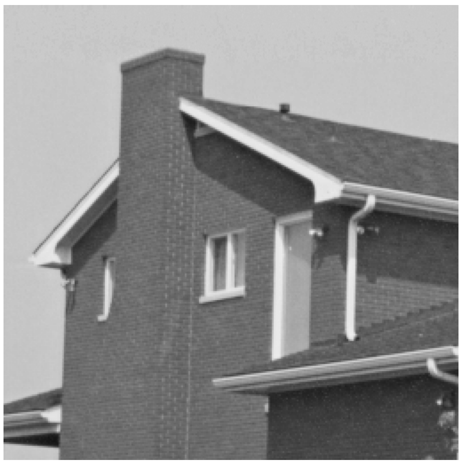
(b)



(c)



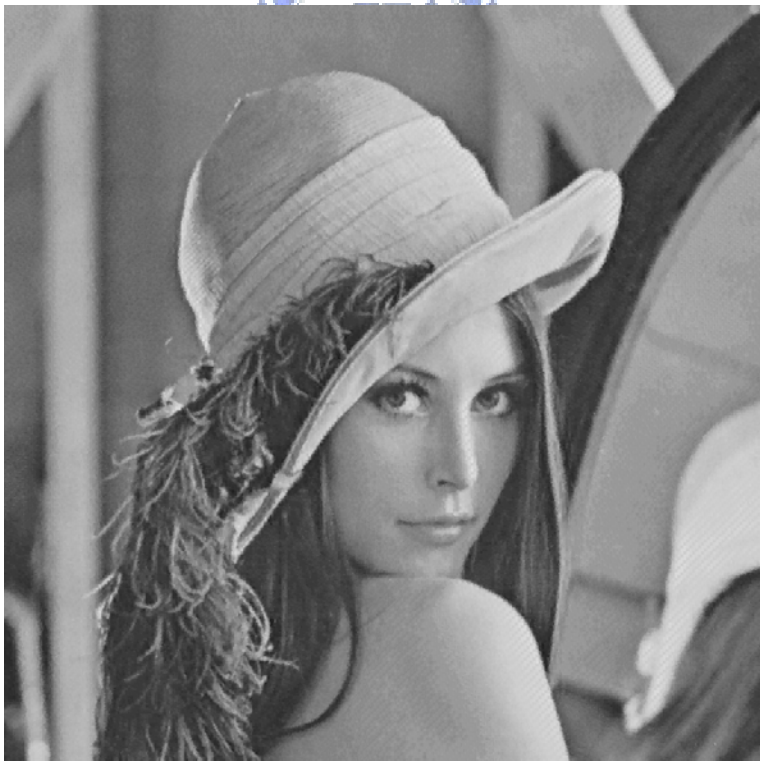
(d)



(e)

Fig. 2.8 The five shadow images, which are the modified versions of Fig. 2.7 (a) ~ (e), respectively (the corresponding PSNRs are 52.70, 52.71, 52.69, 48.87 and 50.34 dB).

(a)



(b)

Fig. 2.9 The recovered result. (a) is the original secret image (i.e. Fig. 2.6), while (b) is the 34.01 dB image recovered using any 3 shadow images.

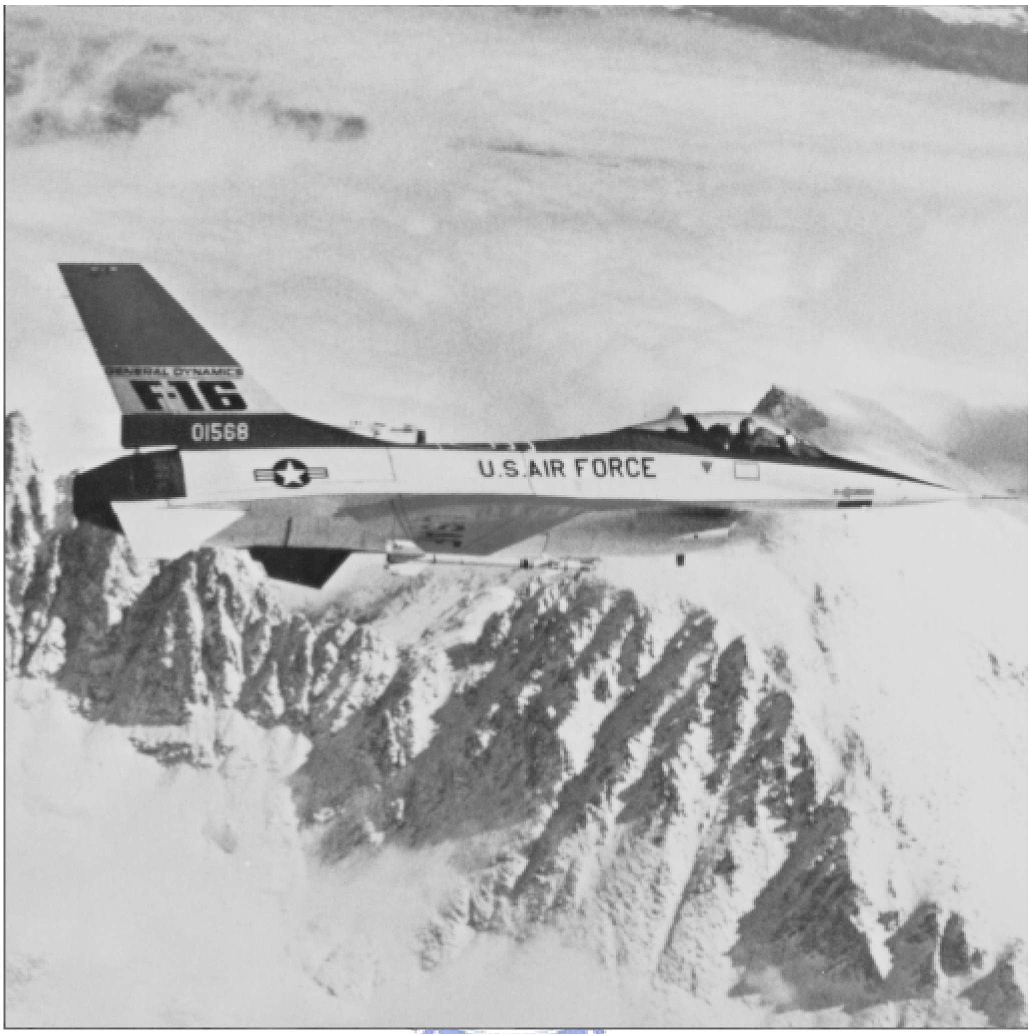


Fig. 2.10 The original 512x512 secret image “Jet”.

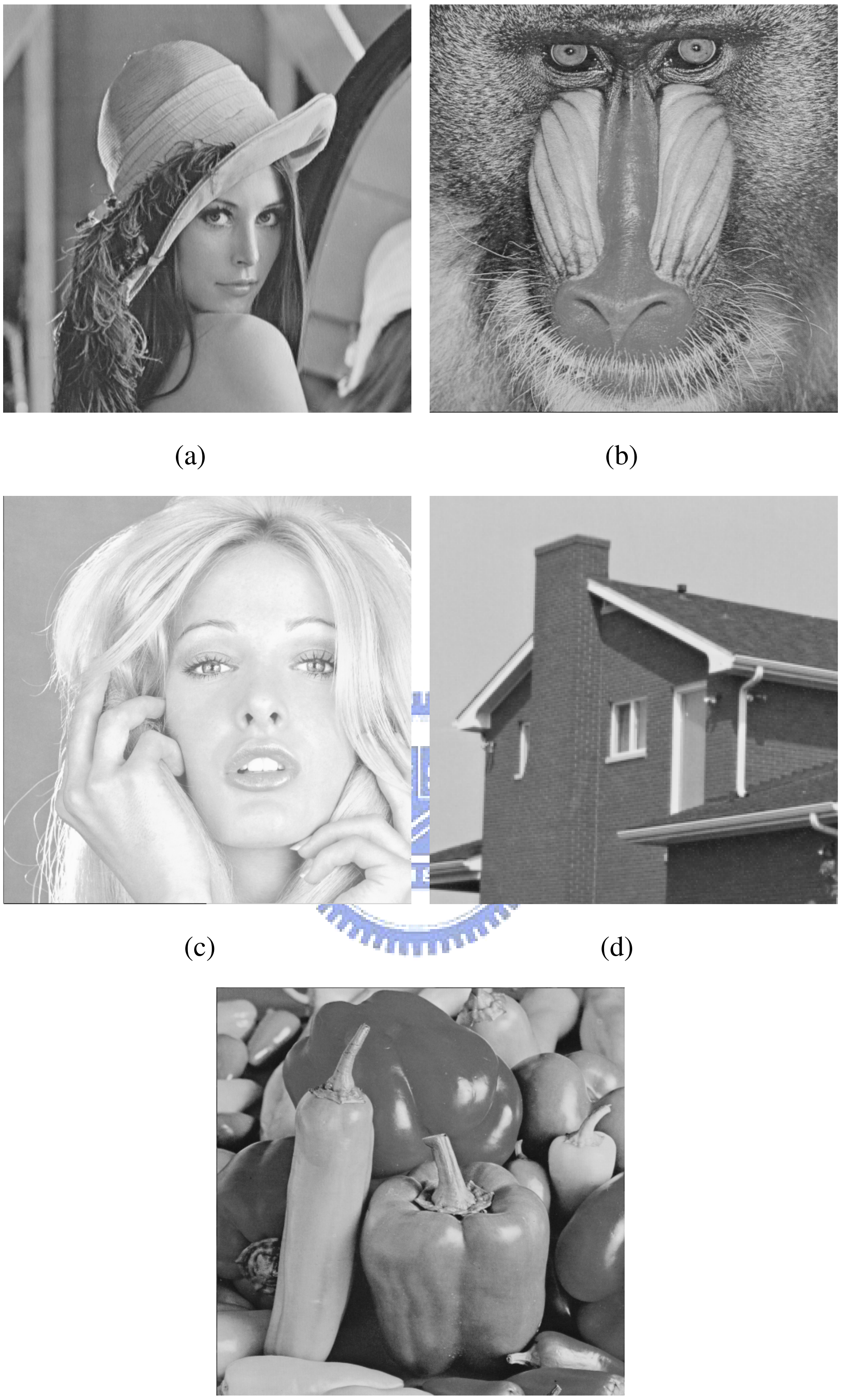
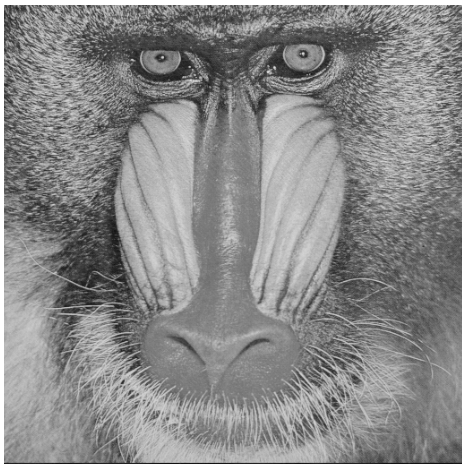
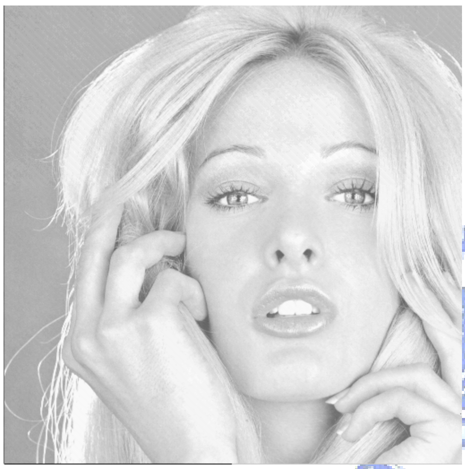


Fig. 2.11 The five host images of size 512 x 512.

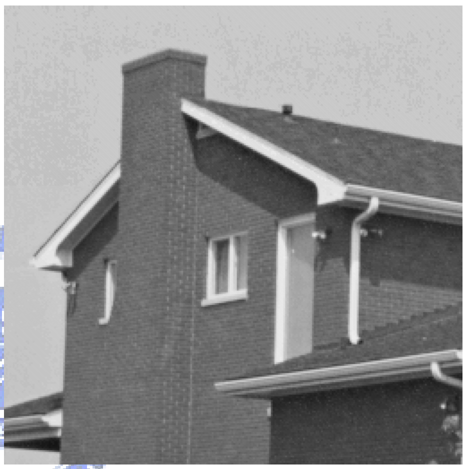
(a)



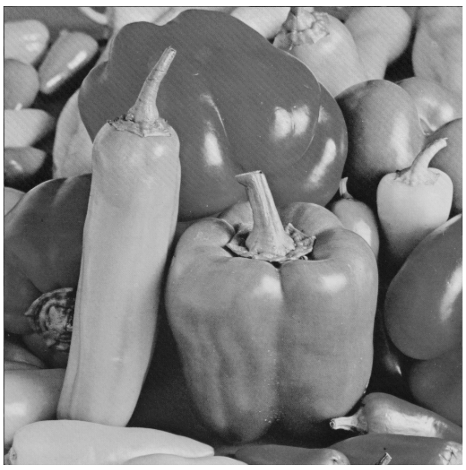
(b)



(c)



(d)



(e)

Fig. 2.12 The five shadow images of size 512 x 512. The PSNRs of (a) ~ (e) are 40.21, 40.41, 40.89, 40.80 and 40.87 dB, respectively.

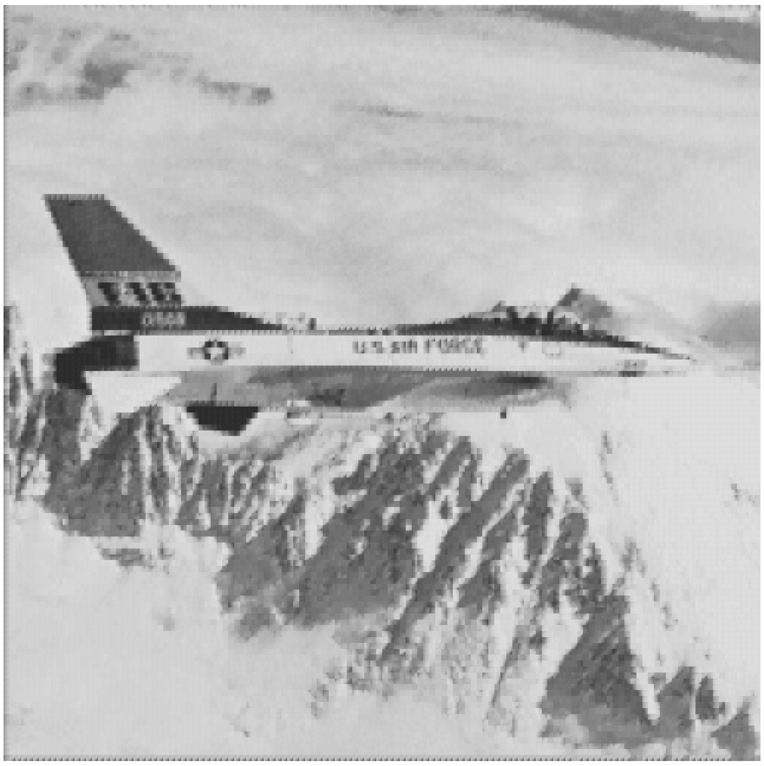


Fig. 2.13 The recovered secret image (PSNR = 28.00 dB) using any 2 shadows in Fig. 2.12.

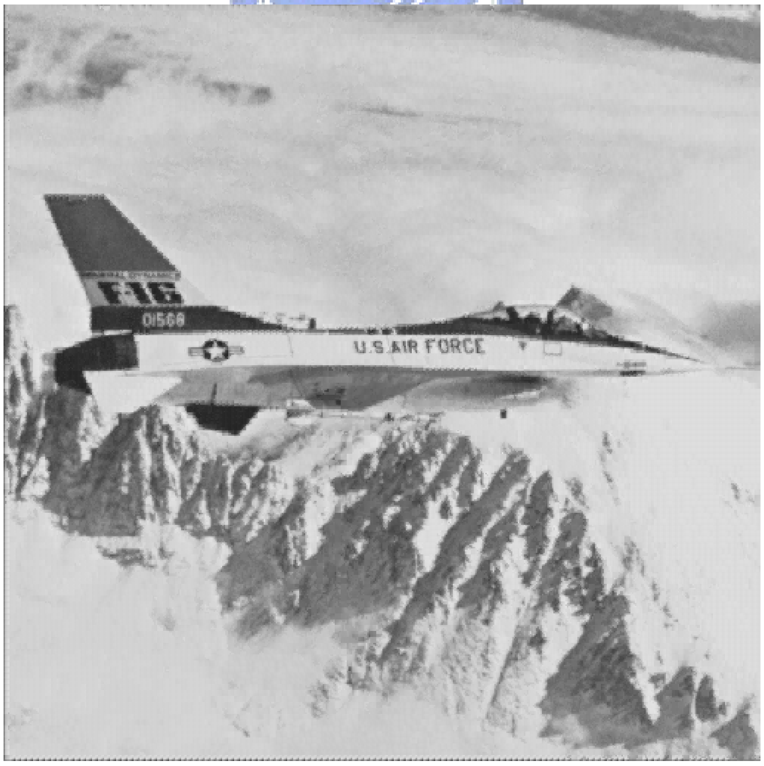


Fig. 2.14 The recovered secret image (PSNR = 32.30 dB) using any 3 shadows in Fig. 2.12.

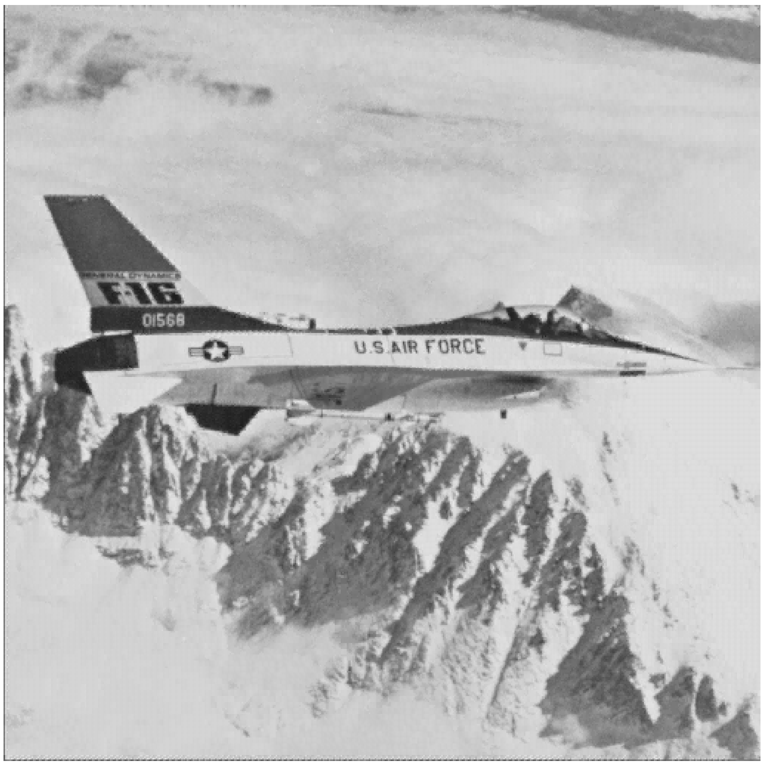


Fig. 2.15 The recovered secret image (PSNR = 35.96 dB) using any 4 shadows in Fig. 2.12.

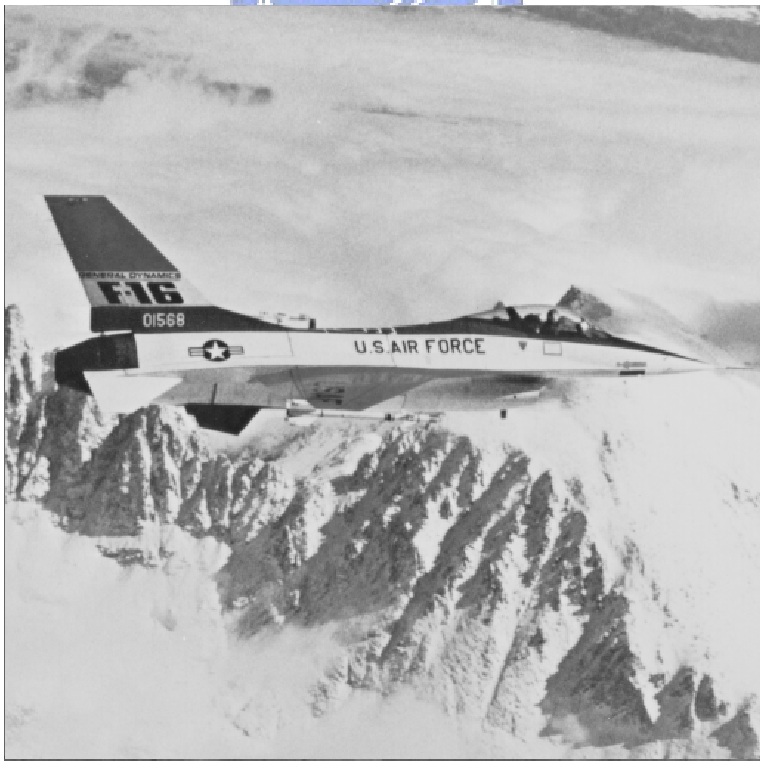


Fig. 2.16 The recovered "lossless" secret image using all 5 shadows in Fig. 2.12.