Optical Engineering 49(4), 047007 (April 2010)

Sharing of multiple images: economically sized and fast-decoding approach based on modulus and Boolean operations

Kun-Yuan Chao
Ja-Chen Lin
National Chiao Tung University
Department of Computer and Information Science
1001 Ta Hsueh Road
Hsinchu, 300 Taiwan
E-mail: kychao@cis.nctu.edu.tw

Abstract. Secret image sharing is a popular technology to secure digital images in storage and transmission. Traditionally, the technology transforms one secret image into several images called shadows or shares. Later, when the number of collected shadows reaches a specified threshold value, the decomposed image can be reconstructed. We propose a new sharing approach to transform *n* secret images into *n* shadows. Later, after gathering all the *n* shadows, all the *n* secret images can be retrieved error-free. No information in any secret image is revealed if one shadow is absent. The total size of *n* generated shadows is identical to the total size of *n* input secret images; hence, this approach does not waste storage space. Each pixel in each secret image is reconstructed using only one Boolean, one modulus, and two mathematical operations, so it is also a fast approach for reconstructing many secret images. Comparisons are included. [DOI: 10.1117/1.3407067]

Subject terms: multi-image sharing; modulus operation; Boolean operation; input/output size ratio; computational complexity.

Paper 090756R received Sep. 28, 2009; revised manuscript received Feb. 6, 2010; accepted for publication Mar. 1, 2010; published online Apr. 20, 2010.

1 Introduction

To secure an image file in storage or transmission, there are two well-known sharing approaches: polynomial secret sharing (PSS)[1] and visual cryptography (VC).[2] They can divide a secret image into several extremely noisy images called shadows (also called shares). Each participant can hold some of these shadows. Later, when the total number of shadows brought to a meeting reaches a specified threshold value, the shared secret image can be reconstructed. Several extended works of the introducing papers[1,2] have been reported. Examples include the reduction of memory cost for shadows,[3] extension of binary VC to grayscale[4] or color images,[5] even a two-in-one sharing method called VCPSS[6] that combined PSS and VC to create a tool whose decoding quality depends on the condition of whether a computer is available or not.

In real life, a project team often processes several secret images simultaneously. Therefore, some research shared multiple images in one encoding process. For example, the elegant PSS scheme[7] presented by Feng et al. used Lagrange interpolation to deal with multisecret images. Their method is an economical method, for it has a very high input/output (I/O) size ratio between 1/2 and 1, i.e., total input image size is at least 50% of the total output shadow size, and 100% is possible. But the computational complexity $O(\log^2 k)$ would be needed to reconstruct each secret pixel by using Lagrange interpolation from k required shadows. To the contrary, to save computational operations in the retrieval of secret images, visual cryptography (VC) schemes can be used. For example, Shyu et al. used two circular shadows to design a VC scheme[8] that can share more than two secret images. Feng et al. also presented a multisecret VC scheme,[9] and their shadows are in rectangular shape. By stacking the shadows (know as transparencies in the VC field), these VC schemes are very fast in revealing all secret images. The disadvantage of using VC methods is their low I/O size ratio (at least 1/2) due to the high pixel expansion rate (per$\geqq$2) in generating shadows. As for the disadvantage of the low contrast of the images recovered by stacking transparencies, it can be avoided if VC methods are implemented on a computer. When VC methods are implemented on a computer to reconstruct n original secret images error-free, the complexity to decode a pixel of a secret image would be $O(n)$ due to the high per.

Besides PSS and VC schemes, Alvarez, Encinas, and del Rey also developed a multisecret sharing scheme[10] for color images with different sizes based on modulus operations. Albeit their I/O size ratio is a very good value $n/(n+1)$ after sharing n secret images by $n+1$ shadows, their reconstruction in each secret pixel needs one modulus operation and many mathematical operations (addition or subtraction) whose computational complexity is $O(n)$. Muñoz-Rodríguez and Rodríguez-Vera developed three very fast methods[11–13] based on phase encoding of a secret image and a pattern image. In their methods, decoding is performed by using just a simple overlapping operation to retrieve each pixel of the input secret image. The methods are very fast, but their decryption creates approximated versions of the original image, rather than error-free recovery. Among the error-free (lossless) schemes[7–10] for multise-

0091-3286/2010/$25.00

crets, no one can simultaneously own the two advantages: 1. I/O size ratio is 1, and 2. only a constant number of operations is needed to reconstruct each secret pixel. To achieve these two advantages simultaneously, we propose here a novel sharing scheme for multiple images, by using modulus (MOD) and exclusive-OR (XOR) operations. The proposed method generates n extremely noisy shadows for n given binary/grayscale/color secret images (notably, the n given images all have the same size), and each shadow's size is identical to each given image. When the n shadows replace the n original secret images in the image database, since our I/O size ratio is always 1, we do not need extra storage space. Furthermore, after gathering all n shadows, our lossless decoding process only uses one XOR, one MOD, one addition (ADD), and one subtraction (SUB) operation (symbolized as $\oplus$, Mod, $+$, and $-$) to reconstruct each pixel's 8-bit value of each secret image. This holds for all values of n. Hence, no matter how many secret images are shared, the CPU time in decoding each secret image will not increase. In summary, the proposed lossless method is not only economical in storage space of shadows but also in constant-speed quickness in decoding.

The rest of the work is as follows. Section 2 describes two basic tools based on MOD and XOR operations, respectively, and these tools are used in the proposed method. Section 3 presents the method. Section 4 gives experimental result and comparisons. Security analysis is in Sec. 5. Summary and future work are in Sec. 6.

2 Two Basic Tools Used in the Proposed Scheme

To achieve the two advantages (higher I/O size ratio and fewer decoding operations) mentioned in Sec. 1, two basic tools are used in the proposed method. The first is the "MOD-based (2, 2) secret sharing tool" in Sec. 2.1, which can make our I/O size ratio 1. The other is the "XOR-based (n,n) shadows combination tool" in Sec. 2.2, which can make our (n,n) scheme only need constant operations to decode each secret pixel, no matter how large the value of n is.

2.1 *MOD-Based (2, 2) Secret Sharing Tool*

Thien and Lin proposed a modified version[14] of Shamir's (k,n)-threshold PSS approach[1] to reduce the size of shadows. They use polynomials to share a secret image A among n shadows $B_1,B_2,\ldots,B_n$; and each of them is k times smaller than A in size. A cannot be revealed unless k of the n shadows are gathered. In their encoding, to transform a sector $\{a_0,a_1,\ldots,a_{k-1}\}$ formed of k secret pixel values of A into n shadow pixel values $\{b_1,b_2,\ldots,b_n\}$, (where each $b_i \in B_i$), they use a prime number $p=251$ to create a polynomial,

$$q(x) = (a_0 + a_1 x + \ldots + a_{k-1}x^{k-1})_{\text{Mod } p}, \tag{1}$$

of degree $k-1$. Then we evaluate $b_1=q(1), b_2=q(2), \cdots, b_n=q(n)$. Later, using any k of the n produced pairs $\{(i,b_i)\}_{i=1}^{n}$, people can recover all k coefficients $a_0,a_1,\ldots,a_{k-1}$ in $q(x)$ by constructing the interpolation polynomial.

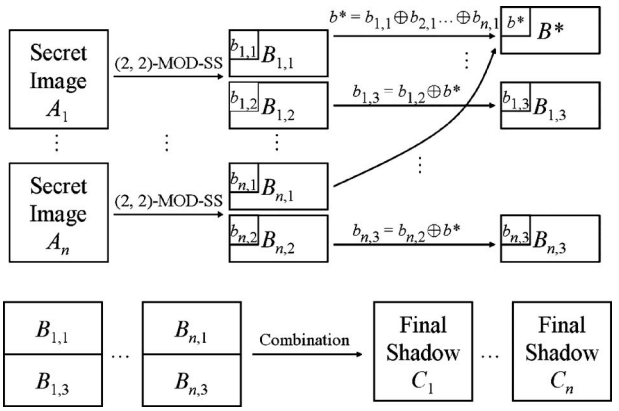


Fig. 1 Diagram of the proposed encoding algorithm.

To let our method have shadows with small size, we apply Ref. 14. More specifically, we apply Ref. 14 in a special manner ($k=2$, $n=2$). We call this (2, 2) scheme a "MOD-based (2, 2) secret sharing tool." The tool is described as follows.

Sharing phase. [See (2, 2)-MOD-SS in left-top part of Fig. 1.]

Step 1. Use a prime number key to generate a permutation sequence to permute all pixel positions of the given grayscale image A. Then attach the key in the permuted image $\tilde{A}$.

Step 2. Sequentially read in gray values $\{p_i\}$ of $\tilde{A}$, and then store in array E according to the rules.

Step 2.1. If $p_i<250$, then store p_i in E.
Step 2.2. If $p_i \geqq 250$, then split p_i into two values 250 and (p_i-250). Store these two values in E (first 250, then p_i-250).

Step 3. Sequentially grab two not-shared-yet elements a_0 and a_1 of E. Use the grabbed (a_0,a_1) to evaluate

$$b_1 = q(1) = (a_0 + a_1)_{\text{Mod } 251}, \tag{2}$$

$$b_2 = q(2) = (a_0 + a_1 \times 2)_{\text{Mod } 251}, \tag{3}$$

and then attach b_1 to shadow B_1, and attach b_2 to shadow B_2.
Step 4. Repeat step 3 until all elements of the array E are processed.

Notably, step 2.2 handles the gray values larger than 250 (see Ref. 14), which seldom happens for most natural images. Therefore, the size of E, which equals the total size of B_1 and B_2, is very close to size A. In other words, each created B_1 and B_2 is about two times smaller than A in size.

Reveal phase. [See Reveal of (2, 2)-MOD-SS in the bottom part of Fig. 2.]

Step 1. Take the first nonused pixel from each of the two shadows B_1 and B_2. Call the two values (b_1,b_2).

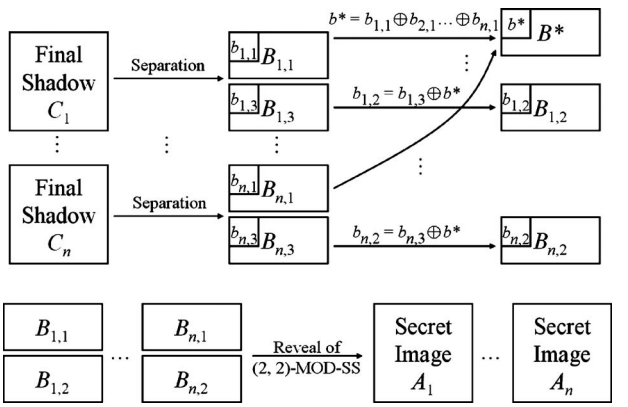


Fig. 2 Diagram of the proposed decoding algorithm.

Step 2. Use these two values (b_1, b_2) to recover the two coefficients (a_0, a_1) in Eqs. (2) and (3) by

$$a_1 = (b_2 - b_1 + 251)_{\text{Mod } 251}, \tag{4}$$

$$a_0 = (b_1 - a_1 + 251)_{\text{Mod } 251}. \tag{5}$$

The recovered (a_0, a_1) are the two corresponding values in E.
Step 3. Repeat steps 1 and 2 until all values of the two shadows B_1 and B_2 are processed.
Step 4. Sequentially grab an element p_i of E, then do:

Step 4.1. If $p_i < 250$, then store p_i in $\tilde{A}$. Now, delete p_i from E, because the information contained in p_i has been used.
Step 4.2. If $p_i = 250$, then read in p_{i+1} immediately. Then store the single value $(250 + p_{i+1})$ in $\tilde{A}$. Now, delete both p_i and p_{i+1} from E, because the information in p_i and p_{i+1} have both been used.

Step 5. Extract the prime number key from $\tilde{A}$ and apply the inverse-permutation operation to the permuted image $\tilde{A}$ to get back to the secret image A.

In this reveal algorithm, only three operations (one SUB, one ADD, and one MOD) are needed in Eqs. (4) or (5) to reconstruct each secret pixel of $\tilde{A}$. Because usually there are only a few pixels in A (and hence in $\tilde{A}$) whose gray values are above 250, the ADD operation in step 4.2 seldom occurs. Also, for each pixel, step 5 uses one mapping operation (indexing) rather than computational operation.

2.2 *Exclusive-OR-Based (n, n) Shadows Combination Tool*

Once all given n secret images $A_1, A_2, \ldots, A_n$ are processed by the MOD-based (2, 2) secret sharing tool described in Sec. 2.1, each secret image A_i $(1 \leq i \leq n)$ generates two half-size temporary shadows $B_{i,1}$ and $B_{i,2}$. To avoid any secret leaking when collecting less than n final shadows, we use the following steps of a combination phase to form the final shadow C_i.

Combination phase. (See right-top and bottom parts of Fig. 1.)

Step 1. Size-synchronization: if some shadows $B_{i,j}$ $(1 \leq i \leq n$ and $1 \leq j \leq 2)$ have distinct size due to the existence of pixels whose gray values are in 251 to 255 in some images A_i, then add a suitable number of dummy pixels in all shadows to make all shadows $B_{i,j}$ have the same size as the one with the largest size.
Step 2. Stacking: take the first not-yet-processed pixel $b_{i,1}$ from each shadow $B_{i,1}$ $(1 \leq i \leq n)$. Then evaluate the corresponding pixel b^* for a new image B^* by

$$b^* = b_{1,1} \oplus b_{2,1} \oplus \cdots \oplus b_{n,1}. \tag{6}$$

After all pixels of all $B_{i,1}$ are processed, the generation of image B^* is done, and its size is the same as $B_{i,1}$ (and $B_{i,2}$, too).
Step 3. Shifting $B_{i,2}$ to $B_{i,3}$: take next not-yet-processed pixel $b_{i,2}$ from each shadow $B_{i,2}$ $(1 \leq i \leq n)$, and at the same pixel position take the corresponding pixel $b^* \in B^*$. Then create the corresponding pixel value $b_{i,3}$ of a new image $B_{i,3}$ by

$$b_{i,3} = b_{i,2} \oplus b^* \quad (1 \leq i \leq n). \tag{7}$$

After all pixels in all $B_{i,2}$ are processed, all n images $B_{i,3}$ are generated. Notably, each $B_{i,3}$ is as large as each $B_{i,2}$.
Step 4. Physically gluing: for $1 \leq i \leq n$, physically combine each pair of $B_{i,1}$ and $B_{i,3}$ to generate their final shadow $C_i = (B_{i,1}; B_{i,3})$. Because $B_{i,1}$ and $B_{i,3}$ are both about two times smaller than A_i, each C_i is about as large as A_i.

When all n final shadows $C_1, C_2, \ldots, C_n$ are gathered, we can reconstruct all $B_{i,1}$ and $B_{i,2}$ $(1 \leq i \leq n)$ by using the following steps of a decomposing phase whose computation is very low.

Decomposition phase. [See top part of Fig. 2.]

Step 1. For $1 \leq i \leq n$, physically separate each $C_i = (B_{i,1}; B_{i,3})$ into two halves to get $B_{i,1}$ (front half) and $B_{i,3}$ (rear half).
Step 2. Take next not-yet-processed pixel $b_{i,1}$ from each $B_{i,1}$ $(1 \leq i \leq n)$. Then evaluate the corresponding pixel b^* in the image B^* by Eq. (6). After all pixels of all $B_{i,1}$ are processed, the image B^* is generated.
Step 3. Take next not-yet-processed pixel $b_{i,3}$ from each $B_{i,3}$ $(1 \leq i \leq n)$, and at the same pixel position, take the corresponding pixel b^* from B^*. Then evaluate the corresponding pixel $b_{i,2}$ for each image $B_{i,2}$ by

$$b_{i,2} = b_{i,3} \oplus b^* \quad (1 \leq i \leq n). \tag{8}$$

After all pixels of all $B_{i,3}$ are processed, all $B_{i,2}$ are recovered.

On average, only one XOR operation is needed to recover a pixel of a secret image, and this statement is true for each secret image A_i $(1 \leq i \leq n)$. The analysis is as fol-

lows. Assume size of each A_i is $w \times h$, then each $B_{i,1}$ in step 2 uses on average $[(n-1)/n] \times [(w \times h)/2]$ XOR operations in Eq. (6) to create B^*. And each $B_{i,3}$ in step 3 uses on average $(w \times h)/2$ XOR operations in Eq. (8) to recover $B_{i,2}$. Therefore, for each secret image A_i, the total number of XOR operations needed in the decomposition phase is $\{[(n-1)/n]+1\} \times [(w \times h)/2]$, which will be smaller than 1 after dividing by A_i's size $w \times h$. In other words, less than one XOR operation is needed in the decomposition phase to recover a pixel of a secret image A_i.

3 Proposed Method

3.1 *Encoding*

The encoding uses the sharing phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1, followed by the combination phase of the XOR-based (n,n) shadows combination tool in Sec. 2.2. The encoding creates n shadows $C_1, C_2, \ldots, C_n$ to replace the n given secret images $A_1, A_2, \ldots, A_n$. The main steps are as follows.

Encoding algorithm. (The diagram is in Fig. 1.)

Input: n input binary/grayscale/color secret images $A_1, A_2, \ldots, A_n$ of the same size.

Step 1. It does not matter whether the image is binary or gray or color, just treat each A_i $(1 \leq i \leq n)$ as a long byte-stream (so each element has 8 bits and can be considered as a gray-value pixel).
Step 2. Then, for each stream A_i $(1 \leq i \leq n)$, use the sharing phase of MOD-based (2, 2) secret sharing tool in Sec. 2.1 to create its half-size shadows $\{B_{i,1}, B_{i,2}\}$.
Step 3. Use all $\{B_{i,1}, B_{i,2}\}$ $1 \leq i \leq n$ in the combination phase of the XOR-based (n,n) shadows combination tool in Sec. 2.2 to generate n final shadows $C_1, C_2, \ldots, C_n$.

Notably, each final shadow C_i is (nearly) as large as A_i. As a result, even from a multisecrets view, the I/O size ratio is also 1, because total secret size $|\{A_1, A_2, \ldots, A_n\}|$ is identical to the total shadow size $|\{C_1, C_2, \ldots, C_n\}|$.

3.2 *Decoding*

To recover the n secret images $A_1, A_2, \ldots, A_n$ from the n final shadows $C_1, C_2, \ldots, C_n$, the decoding uses the decomposition phase of the XOR-based (n,n) shadows combination tool in Sec. 2.2, followed by the reveal phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1. The main steps are as follows.

Decoding algorithm. (See the diagram in Fig. 2.)

Step 1. Use the decomposition phase of the XOR-based (n,n) shadows combination tool in Sec. 2.2 to reconstruct the half-size images $\{B_{i,1}; B_{i,2}\} 1 \leq i \leq n$ from the n final shadows $\{C_1, C_2, \ldots, C_n\}$.
Step 2. For each pair $\{B_{i,1}; B_{i,2}\}$, where $1 \leq i \leq n$, use the reveal phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1 to recover the secret image A_i from $\{B_{i,1}; B_{i,2}\}$.

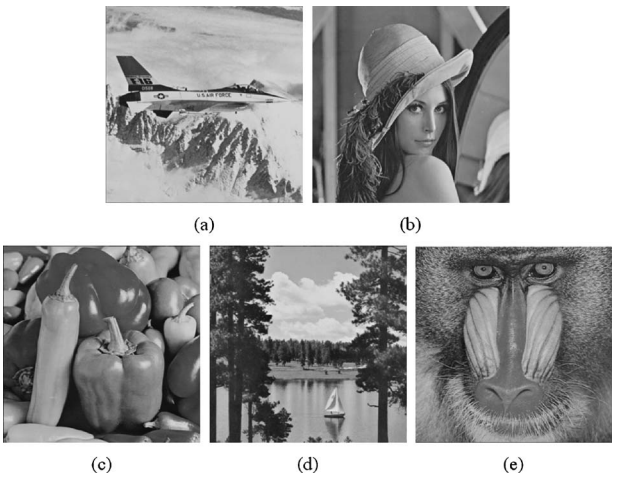


Fig. 3 Five input grayscale images in the $n=5$ case.

Step 3. If the original secret images are not gray-valued, then transform all n 8-bit images $A_1, A_2, \ldots, A_n$ to their binary/color equivalent.

On average, the decoding only needs one XOR, one MOD, one ADD, and one SUB operations (all are between bytes) to reconstruct each 1-byte (8-bit) pixel value of a secret image. (In step 1 of decoding, one XOR operation is needed to recover a 1-byte pixel value of $B_{i,2}$ for the decomposition phase of the XOR-based (n,n) shadows combination tool. In step 2, one MOD, one ADD, and one SUB operations are needed to recover a 1-byte pixel value of A_i in the reveal phase of the MOD-based (2, 2) secret sharing tool.)

4 Experimental Result and Comparisons

4.1 *Experimental Result*

In the experiment here, without the loss of generality, let $n=5$ and let the n input images $\{A_1, A_2, \ldots, A_5\}$ be the 512×512 grayscale images {Jet, Lena, Pepper, Scene, and Monkey} shown in Figs. 3(a)–3(e). Then, Figs. 4(a)–4(e) show the $n=5$ final shadows $\{C_1, C_2, \ldots, C_5\}$ generated in Sec. 3.1, and each 512×512 shadow is as large as each input image. Figures 5(a)–5(e) show five error-free recovered images $A_1, A_2, \ldots, A_5$ (Jet, Lena, Pepper, Scene, and Monkey) in Sec. 3.2 using all $n=5$ final shadows $C_1, C_2, \ldots, C_5$. In general, all recovered images in our method are error-free, i.e., the peak signal-to-noise ratio (PSNR) value is infinity (∞) for each of our recovered images. So each of the recovered images {Jet, Lena, Peppers, Scene, and Monkey} shown in Fig. 5 has PSNR$=\infty$. (The five images in Fig. 5 and the five images in Fig. 3 are exactly the same, rather than just similar.)

In addition, to show our constant decoding-time property in the retrieval, we also listed in Table 1 the number of operations on average needed to decode each one of the n secret images. It can be seen that, indeed, a constant number of operations is needed for each value of n. The actual CPU time taken in decoding is also checked in a computer. We still find that our decoding time does not vary as the value of n varies.

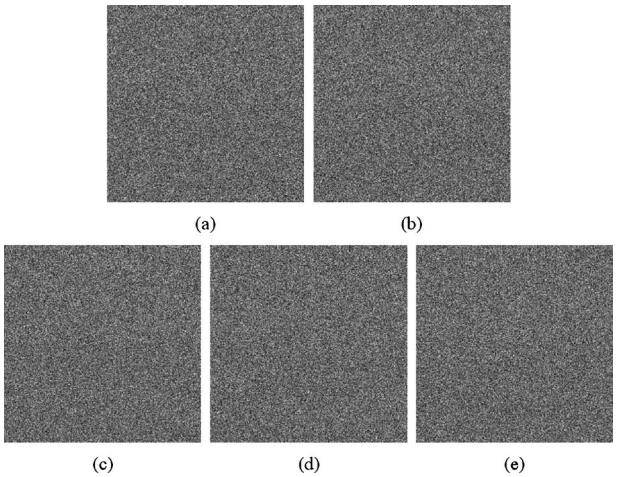


Fig. 4 The five generated noisy shadows $\{C_1, C_2, \ldots, C_5\}$ in the $n=5$ case. Their PSNRs are all around 8.0 if we compare them with the secret image Jet (all around 9.5 if we compare them with Lena; all around 9.2 if we compare them with Peppers; all around 8.6 if we compare them with Scene; all around 9.7 if we compare them with Monkey).

4.2 *Comparisons*

Table 2 compares our method with other multisecret schemes[7–10] in terms of two quantifiable measures: 1. I/O size ratio and 2. decoding computational complexity. For comparison, the same conditions hold for each scheme. They are: 1. there are n secret images, 2. every recovered image must be lossless, and 3. the computer versions of VC schemes[8,9] are implemented using an OR-like operation to simulate the stacking action of transparencies. From Table 2, we can see that our method not only keeps the I/O size ratio as large as 1, but also needs only constant decoding operations to reconstruct a secret pixel. Other schemes[7–10] either have I/O size ratio smaller than 1, or need nonconstant decoding complexity to reconstruct a secret pixel. For completeness, we also include in Table 2 the very fast methods[11–13] developed by Muñoz-Rodríguez and Rodríguez-Vera that use one secret image and one pattern image as input. Their I/O size ratios are as good as ours. Their decoding complexity is very economic, but their decoding only gives approximated versions (rather than error-free recovery) of the original image. So theirs and ours have different highlighting.

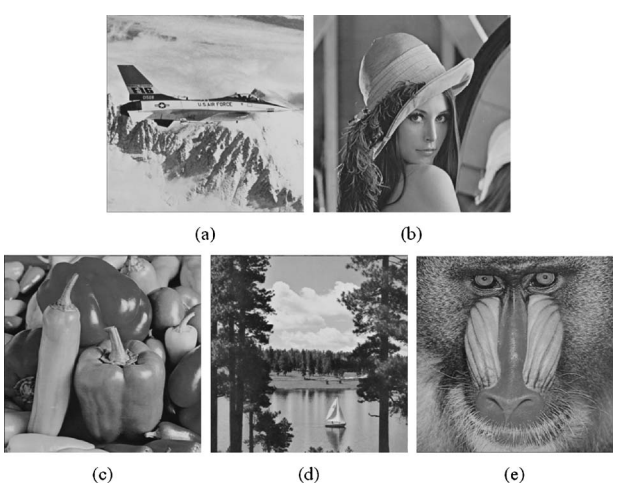


Fig. 5 The five error-free images (PSNR=∞) recovered by using all five shadows (Fig. 4) in the $n=5$ case.

Table 1 Number of operations needed to decode each one of the n secret images whose sizes are all 512×512.

Value of n	Number of operations needed to decode one 512×512 secret image
2	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
3	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
4	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
5	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
6	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
7	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
8	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
9	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations
…	512×512 XOR, 512×512 MOD, 512×512 ADD, 512×512 SUB operations

Table 2 I/O size ratio and decoding complexity. For I/O size ratio, total size of input images is divided by total size of shadows. Hence, larger is better. For decoding complexity, complexity is to decode a pixel of a secret image. For math operators, k is a user-specified threshold value, e.g., $k=0.5n$ or $k=2$. For Ref. 10, math operations: $+$, $-$, $\times$, $\div$. References 11–13 are very fast, but the decoding just gives approximated versions (rather than an error-free version) of the original secret image.

Schemes	I/O size ratio (larger is better)	Decoding complexity (smaller is better)
Ref. 7	1/2∼1	$O(\log^2 k)$ (math operations)
Ref. 8	1/4	$2\times n$ (OR-like operations)
Ref. 9	1/6	$3\times n$ (OR-like operations)
Ref. 10	$n/(n+1)$	1 (MOD operation) and $O(n)$ (math operations)
Refs. 11–13	1	1 overlapping operation
Our scheme	1	1 XOR, 1 MOD, 1 ADD, and 1 Substraction operations

5 Security Analysis

In this section, we analyze the security of the encrypted images (shadows).

5.1 *Probability*

Assume that only $n-1$ final shadows are available, and a shadow C_j is missing. Then, people cannot obtain B^* created by $b^*=b_{1,1}\oplus b_{2,1}\oplus\cdots\oplus b_{n,1}$ in Eq. (6), due to the lack of the $b_{j,1}$, which is in the front half of C_j. Wthout B^*, people cannot reconstruct $B_{1,2},B_{2,2},\ldots,B_{n,2}$ defined by $b_{i,2}=b_{i,3}\oplus b^*$ $(1\leqslant i\leqslant n)$ in Eq. (8). As a result, no secret image A_i can be recovered [see the reveal phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1] due to absence of all $B_{i,2}$ $(1\leqslant i\leqslant n)$.

Next we discuss the probability of obtaining some right secret images A_i through guessing. Without the loss of generality, assume that a betrayal party of $n-1$ participants gathers their $n-1$ final shadows $C_1,C_2,\ldots,C_{n-1}$, and try to recover some of the n secret images without the cooperation of the missing shadow C_n. From Eqs. (6) and (8), we have

$$b_{i,2}=b_{i,3}\oplus b^*=b_{i,3}\oplus b_{1,1}\oplus b_{2,1}\oplus\cdots\oplus b_{n,1}\quad(1\leqslant i\leqslant n).\tag{9}$$

Because of the lack of $C_n=[B_{n,1};B_{n,3}]$, for each pixel $b_{n,1}$ in $B_{n,1}$, the betrayal party will have to guess a value, and then they use this guessing value to get a set of $n-1$ pixel values $b_{1,2},b_{2,2},\ldots,b_{n-1,2}$ in $B_{1,2},B_{2,2},\ldots,B_{n-1,2}$, respectively, at the same pixel position (there is no $b_{n,2}$ value in $B_{n,2}$ due to lack of $b_{n,3}$). Then, the $2\times(n-1)$ secret pixels in $A_1,A_2,\ldots,A_{n-1}$ can be reconstructed by using $b_{i,1}$ and $b_{i,2}$ $(1\leqslant i\leqslant n-1)$ in the reveal phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1. This is just to reconstruct two pixels in each A_i $(1\leqslant i\leqslant n-1)$. This value guessing of one pixel, like $b_{n,1}$, will repeat $(w\times h)/2$ times if the size of each A_i is $w\times h$ (then, the size of $B_{n,1}$ is $0.5\times w\times h$).

From this description, we can evaluate the probability of obtaining some right grayscale images A_i of size $w\times h$ as follows:

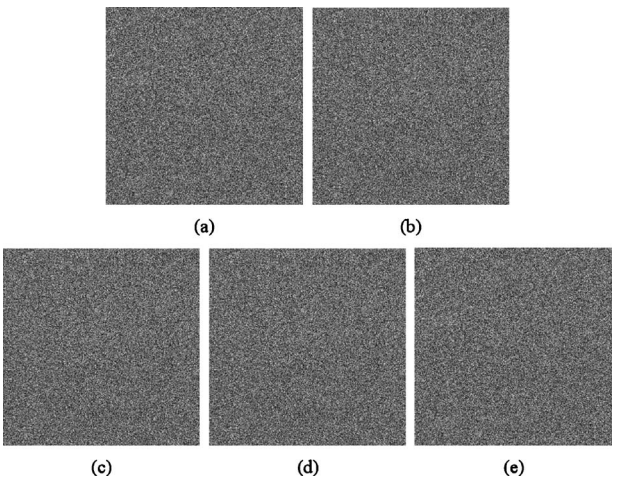


Fig. 6 The $n=5$ images A_1,A_2,A_3,A_4,A_5 recovered by using only four shadows C_1,C_2,C_3,C_4 [Figs. 4(a)–4(d)] and two guessed images $B_{5,1}$ and $B_{5,2}$ in the $n=5$ case.

$$\text{probability}=\left(\frac{1}{\text{values range}}\right)^{w\times h/2}=\left(\frac{1}{251}\right)^{w\times h/2},$$

which is $(1/251)^{512\times512/2}=(1/251)^{131,072}=10^{-314530}$ if each image size is 512×512. Here, $1/\text{values range}=1/251$ is the probability to guess successfully a pixel's value whose range is from 0 to 250; $w\times h/2$ is the number of pixels in $B_{n,1}$. In fact, for each secret image A_i $(1\leqslant i\leqslant n)$ in the encoding process [as we did in step 1 of the sharing phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1], we already use a prime number as a key (a seed) of a random number generator to rearrange all the pixel positions of A_i. Even if many pixel values in $B_{n,1}$ are guessed successfully, in step 4 of the reveal phase of the MOD-based (2, 2) secret sharing tool in Sec. 2.1, the recovered images $\tilde{A}_i$ $(1\leqslant i\leqslant n-1)$ are still extremely noisy. Therefore, the security guardian has double levels.

For instance, in the experimental example of Sec. 4.1, if we only get four shadows, say $C_1,C_2,\ldots,C_4$ in Figs. 4(a)–4(d), and then guess the pixel values of images $B_{5,1}$

Table 3 The entropy and PSNR values of the five generated noisy shadows $\{C_1,C_2,\ldots,C_5\}$ shown in Fig. 4.

The five shadows	C_1	C_2	C_3	C_4	C_5
Entropy values	7.999	7.999	7.999	7.999	7.999
PSNR values (Jet)	8.059	8.055	8.047	8.056	8.036
PSNR values (Lena)	9.473	9.514	9.506	9.501	9.495
PSNR values (Peppers)	9.202	9.204	9.206	9.186	9.213
PSNR values (Scene)	8.594	8.601	8.603	8.571	8.593
PSNR values (Monkey)	9.703	9.704	9.698	9.705	9.699

and $B_{5,2}$, then the five recovered images of $A_1, A_2, \ldots, A_5$ are the extremely noisy images shown in Fig. 6.

5.2 *Dissimilarity between Shadow Images*

In the case of sensitivity, the difference between two shadow images cannot be observed visually. For this matter, we check the dissimilarity between shadow images as follows. First, from Table 3, we can see that all generated shadows have very similar entropy (7.999) and very similar PSNRs (when they are all compared with a secret image, say Lena). Second, besides comparing a shadow (C_i) with a secret image, if a shadow C_i is compared with another shadow $C_j (1 \leq i < j \leq 5)$, then a PSNR value called $\mathrm{PSNR}(C_i, C_j)$ can be evaluated by the formula

$$\mathrm{PSNR}(C_i, C_j) = 10 \times \log_{10} \frac{255^2}{\sum_{u=1}^{512}\sum_{v=1}^{512}[C_i(u,v) - C_j(u,v)]^2/(512 \times 512)}, \tag{10}$$

and the computation result is as listed in Table 4. From this table, we can see that the $\mathrm{PSNR}(C_i, C_j)$ values are all very low (smaller than 9) for all pairs of shadows (C_i, C_j). This means that none of the shadows are alike. Third, none of the shadow C_i in $\{C_1, \ldots, C_5\}$ can be predicted in any part from another shadow C_i, because each difference image $|C_i - C_i|$ (see Fig. 7) between two shadows C_i and C_j $(1 \leq i < j \leq 5)$ still looks extremely noisy and has neither a meaningful pattern nor meaningful contour. From the end of Fig. 7, we also see that the ten histograms for $|C_i - C_j|_{1 \leq i < j \leq 5}$ all look alike (due to space limitation, only two of the ten are shown), and this fact implies that the ten entropies for the ten difference images $\{|C_i - C_j|\}_{1 \leq i < j \leq 5}$ are almost identical (the entropy of each difference image reads 7.673…, as shown in Table 4.) From Fig. 7 and Table 4, no shadow can be predicted in any part from another shadow.

Table 4 The entropy of the difference image $|C_i - C_j|$, and the $\mathrm{PSNR}(C_i, C_j)$ value. (The two shadow images C_i and C_j are taken from the five shadow images $\{C_1, \ldots, C_5\}$ shown in Fig. 4. So there are $5! \div [2!(5-2)!] = 10$ pairs.)

Difference image $\lvert C_i - C_j \rvert$	Entropy value of $\lvert C_i - C_j \rvert$	$\mathrm{PSNR}(C_i, C_j)$ for the pair (C_i, C_j)
$\lvert C_1 - C_2 \rvert$	7.673	7.728
$\lvert C_1 - C_3 \rvert$	7.673	7.875
$\lvert C_1 - C_4 \rvert$	7.673	8.140
$\lvert C_1 - C_5 \rvert$	7.673	7.728
$\lvert C_2 - C_3 \rvert$	7.673	7.792
$\lvert C_2 - C_4 \rvert$	7.673	7.748
$\lvert C_2 - C_5 \rvert$	7.673	7.824
$\lvert C_3 - C_4 \rvert$	7.673	8.016
$\lvert C_3 - C_5 \rvert$	7.673	7.974
$\lvert C_4 - C_5 \rvert$	7.673	7.734

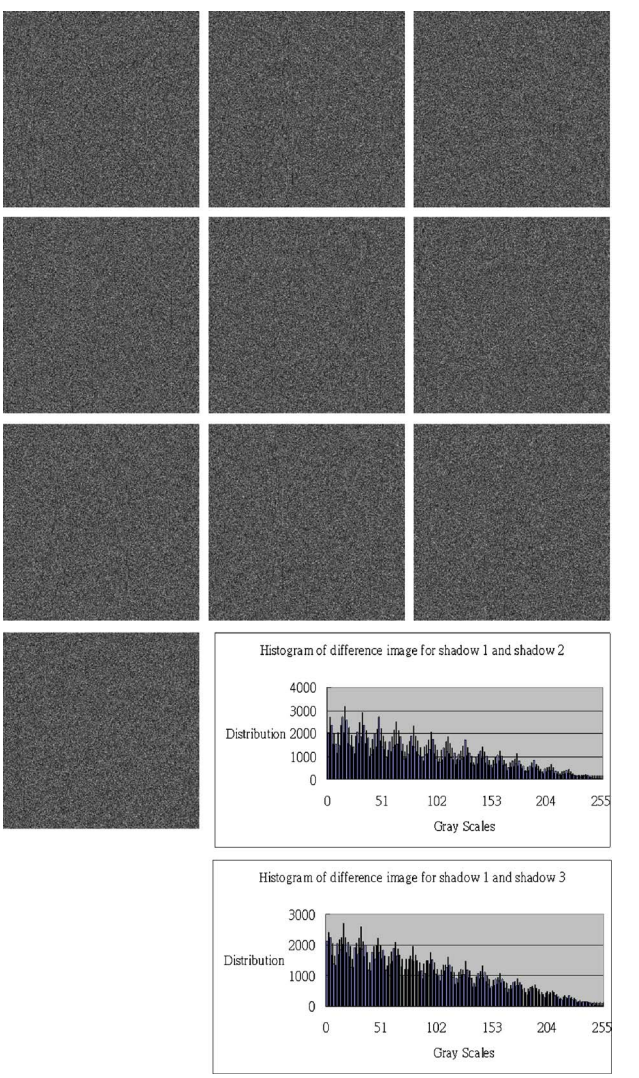


Fig. 7 The ten difference images $\{|C_1| - |C_2|, |C_1| - |C_3|, |C_1| - |C_4|, |C_1| - |C_5|, |C_2| - |C_3|, |C_2| - |C_4|, |C_2| - |C_5|, |C_3| - |C_4|, |C_3| - |C_5|, |C_4| - |C_5|\}$; and the two histograms for $|C_1 - C_2|$ and $|C_1 - C_3|$, respectively. The remaining $10 - 2 = 8$ histograms all look like these two histograms, so their entropies all read 7.673…, as shown in Table 4.

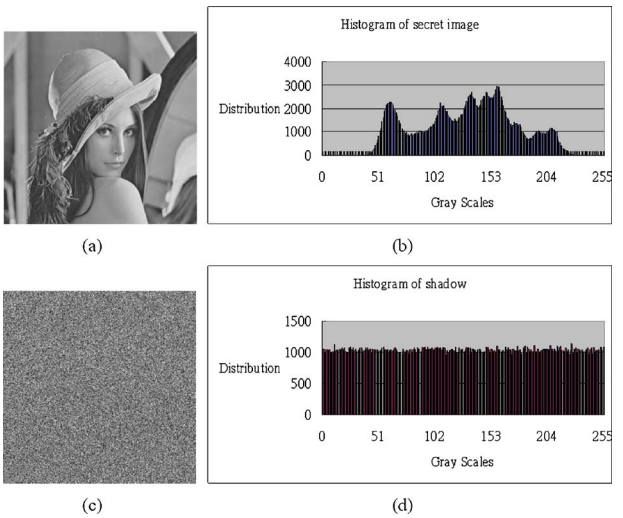


Fig. 8 Histograms of the original and encrypted images. (a) is one of the $n=5$ input secret images; (b) is the histogram of (a); (c) is a typical shadow image; and (d) is the histogram of (c).

5.3 *Statistical Analysis*

Many attackers use statistical analysis to analyze the encrypted images to trace back the original secret images. So we also apply statistical analysis to our shadows. It includes the following: the histogram of each shadow is inspected in Sec. 5.3.1; the correlations of two adjacent pixels of each shadow are examined in Sec. 5.3.2; and the number of pixels change rate (NPCR) and unified average changing intensity (UACI) values are examined in Sec. 5.3.3 to check the relationship between the original images (secrets) and the encrypted images (shadows).

5.3.1 *Histograms, entropies, and peak signal-to-noise ratios of the encrypted images*

We still inspect the five images in Fig. 3 whose image contents are of quite different styles (airplane, human face, vegetable, scenery, and animal). The histograms of the five input images and the histograms of the five created shadows are compared. A typical example of the comparison is shown in Fig. 8. The histogram of each encrypted image (shadow) is fairly uniform, and it is significantly different from the histogram of any input secret image (e.g., Lena). The same observation also exists for the shadows created using other sets of n secret images.

In general, besides the histogram, entropy is another useful value to measure the information distribution of an image. If an image has G gray levels ($G \leqslant 256$), and the probability of gray-level $k(0 \leqslant k \leqslant G-1)$ is $P(k)$, then the image's entropy H_e is

$$H_e = -\sum_{k=0}^{G-1} P(k)\log_2[P(k)]. \tag{11}$$

The entropy value of the secret image Lena in Fig. 8(a) is $H_e(\text{Lena})=7.327$, and the entropy value of the shadow image in Fig. 8(c) is $H_e(\text{shadow image})=7.999$. Note that 7.999 is very close to the perfect entropy value

$$8 = -\sum_{k=0}^{256-1} \frac{1}{256}\log_2\left(\frac{1}{256}\right) = -\log_2\left(\frac{1}{256}\right),$$

which is the entropy value for an ideal image whose gray values are of exactly uniform distribution. H_e (shadow image) is almost 8, because the gray values of each shadow are almost uniformly distributed. In view of security, shadow images with entropy values very close to 8 are good, because it means that the original secret information is now widely and uniformly spread over the shadow images. We have also checked the entropy values of the shadows created from other sets of secret images; and the entropy values of the shadows are still very close to 8.

In terms of PSNR, we include in Table 3 the PSNR values of the five generated shadow images printed in Fig. 4. As shown in Table 3, if we randomly pick a shadow C_i ($i=1, 2, 3, 4, 5$, because $n=5$) in Fig. 4, and compare this randomly chosen shadow with the secret image Jet in Fig. 3, then the PSNR value is in a very concentrated range (between 8.036 and 8.059, as shown in row 3 of Table 3). Likewise, if we compare the randomly chosen shadow with the secret image Lena, then the PSNR value is still in a very concentrated range (between 9.473 and 9.514, as shown in row 4 of Table 3). Similar observation exists in the remaining rows of Table 3, if the secret image is being compared with Peppers (or Scene or Monkey). The phenomenon that shadows $\{C_1, \ldots, C_5\}$ are so alike in both entropy and PSNRs indicates that the shadows are very similar in pixel distribution. Also note that the PSNR values appeared in Table 3 are all very low (less than 10), and this indicates that each of these extremely noisy shadows $\{C_1, \ldots, C_5\}$ in Fig. 4 looks completely different from any of the five original secret images shown in Fig. 3. Thus, the hackers can hardly guess what the original secret images are.

5.3.2 *Correlation of two adjacent pixels*

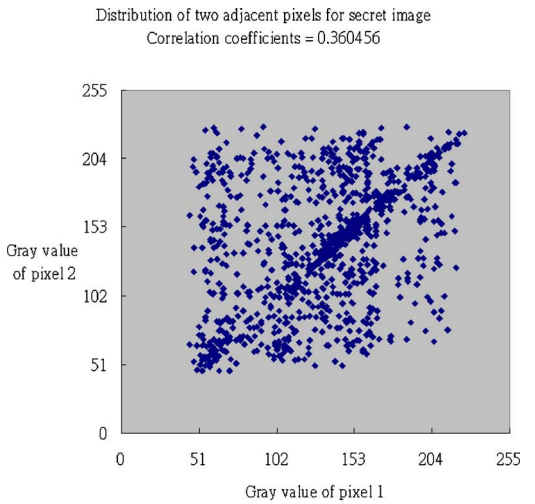
In the subsection, we examine the correlation between two (vertically/horizontally/diagonally) adjacent pixels. As quoted from Ref. 15, for a given image, its correlation coefficient between adjacent pixels is computed according to the formula

$$r_{xy} = \frac{\text{cov}(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}. \tag{12}$$

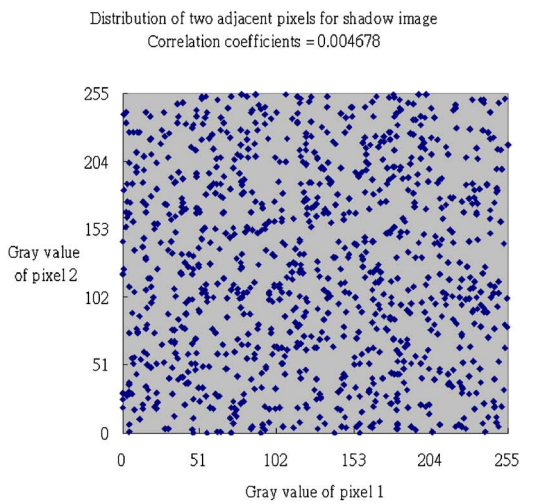
Here, the symbol (x,y) represents the random variable of the gray values of two adjacent pixels. Notably, for an image, if there are N pairs of (x,y), then we compute

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad E(y) = \frac{1}{N}\sum_{i=1}^{N} y_i, \tag{13}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N} [x_i - E(x)]^2, \quad D(y) = \frac{1}{N}\sum_{i=1}^{N} [y_i - E(y)]^2, \tag{14}$$



(a)



(b)

Fig. 9 Distribution of the pairs of adjacent pixels. (a) is for the secret image Lena sketched in Fig. 8(a), whereas (b) is for the shadow image sketched in Fig. 8(c).

$$\mathrm{cov}(x,y)=\frac{1}{N}\sum_{i=1}^{N}[x_i-E(x)][y_i-E(y)], \tag{15}$$

to get r_{xy}. Figure 9(a) shows the distribution of two adjacent pixels (x,y) for the secret image Lena in Fig. 8(a). Then, Fig. 9(b) shows the distribution for the shadow image sketched in Fig. 8(c). The correlation coefficients are 0.360456 and 0.004678, respectively. In general, the correlation coefficient is high for each secret image, and very low for each shadow.

5.3.3 *Differential analysis*

Here, the number of pixels change rate (NPCR) and unified average changing intensity (UACI) are checked. In general, when a pixel value is changed in one of the n secret images, two measures NPCR and UACI can be utilized to describe the impact to the n shadow images. Details are as follows.

Assume that we change a pixel value in one of the n input secret images—for example, change a pixel value for the Lena image shown in Fig. 3(b). Before this one-pixel change, let the n corresponding shadow images be $\{C_1,\ldots,C_n\}$. Then, after this one-pixel change of the secret image Lena, let the n generated shadow images be $\{C_1',\ldots,C_n'\}$. For the two versions C_k and C_k' of the k'th shadow, let $C_k(i,j)$ and $C_k'(i,j)$ denote their grayscale values at position (i,j), respectively. The NPCR_k for shadow k is defined as

$$\mathrm{NPCR}_k=\frac{\Sigma_{i,j}D_k(i,j)}{W\times H}\times 100\%, \tag{16}$$

where W and H are the width and height of a shadow. In the evaluation of NPCR_k for shadow k, the comparison record D_k is defined as the binary matrix in which $D_k(i,j)=0$ if $C_k(i,j)=C_k'(i,j)$, and $D_k(i,j)=1$ if $C_k(i,j)\neq C_k'(i,j)$. Notably, NPCR_k measures the percentage of the altered pixels for shadow k (0% means the two versions C_k and C_k' of shadow k are identical everywhere).

The UACI_k for shadow k is defined as

$$\mathrm{UACI}_k=\frac{1}{W\times H}\left[\sum_{i,j}\frac{|C_k(i,j)-C_k'(i,j)|}{255}\right]\times 100\%, \tag{17}$$

which measures the average intensity of the differences between the two versions (C_k and C_k') of shadow k.

One of the performed tests is when the one-pixel change is on the Lena image of Fig. 3(b). Then the range of $\{\mathrm{NPCR}_1,\ldots,\mathrm{NPCR}_n\}$ for the n shadows is from 99.60 to 99.63%, and the range of $\{\mathrm{UACI}_1,\ldots,\mathrm{UACI}_n\}$ for n shadows is from 33.20 to 33.40%. The average of $\{\mathrm{NPCR}_1,\ldots,\mathrm{NPCR}_n\}$ is $\mathrm{NPCR}_{\mathrm{average}}=99.6111\%$, and the average of $\{\mathrm{UACI}_1,\ldots,\mathrm{UACI}_n\}$ is $\mathrm{UACI}_{\mathrm{average}}=32.2980\%$. When the one-pixel change is on any other image of Fig. 3, the average value of $\{\mathrm{NPCR}_1,\ldots,\mathrm{NPCR}_n\}$ is still larger than 99.6%; and the average value of $\{\mathrm{UACI}_1,\ldots,\mathrm{UACI}_n\}$ is still larger than 33.2%. Therefore, a one-pixel change in any secret image can cause a significant change to all n shadow images. Hence, our shadows can resist differential attack in which the opponents, as mentioned in Refs. 15 and 16, make a slight change in the secret image, and then observe the result in the shadow images. In this way, the opponents try to find a meaningful relationship between the pixels of the secret image and the pixels of the shadows. In our system, since one minor change in the secret image can cause a significant change in all shadows, the differential attack would become useless.

6 Summary and Future Works

A novel lossless sharing scheme for multiple secret images is designed using MOD and XOR operations. Our advantages are: 1. the total size of n given secret images is the same as the total size of the n final shadows (hence our I/O size ratio is 1); 2. we only need one XOR, one MOD, one ADD, and one SUB (all are byte-to-byte) operations to get a lossless recovery of each secret pixel's 8-bit value for the

given binary/grayscale/color images (so our decoding computational complexity for each pixel is a constant and independent of n).

Our method is better than other lossless multisecret schemes in terms of two quantity measures: I/O size ratio and decoding time. But Ref. 7–10 have features of their own. 1. Feng et al.'s[7] uses generalized access structures to achieve higher flexibility, and each qualified set of the access structure is able to share secret images independently. 2. If the shadows are printed in transparencies, then the VC schemes[8,9] can also reveal images by doing physical stacking of transparencies although true-color images will be very hard for VC to retrieve error-free by stacking transparencies. 3. Alvarez et al.'s method[10] can process n given secret images of nonequal sizes. To improve our multisecret sharing scheme in the future, the advantages of these others will be our main guidelines.

Acknowledgments

This work is supported by National Science Council, Taiwan, under grant NSC 97-2221-E-009-120-MY3. The authors also thank the reviewers for valuable suggestions.

References

1. A. Shamir, "How to share a secret," *Commun. ACM* **22**(11), 612–613 (1979).
2. M. Naor and A. Shamir, "Visual cryptography," in *Adv. Cryptolog. EUROCRYPT'94*, **950**, 1–12 (1995).
3. R. Z. Wang and C. H. Su, "Secret image sharing with smaller shadow images," *Pattern Recogn. Lett.* **27**, 551–555 (2006).
4. C. C. Lin and W. H. Tsai, "Visual cryptography for gray-level images by dithering techniques," *Pattern Recogn. Lett.* **24**, 349–358 (2003).
5. Y. C. Hou, "Visual cryptography for color images," *Pattern Recogn.* **36**, 1619–1629 (2003).
6. S. J. Lin and J. C. Lin, "VCPSS: A two-in-one two-decoding-options image sharing method combining visual cryptography (VC) and polynomial-style sharing (PSS) approaches," *Pattern Recogn.* **40**, 3652–3666 (2007).
7. J. B. Feng, H. C. Wu, C. S. Tsai, and Y. P. Chu, "A new multi-secret images sharing scheme using Largrange's interpolation," *J. Syst. Softw.* **76**, 327–339 (2005).
8. S. J. Shyu, S. Y. Huang, Y. K. Lee, R. Z. Wang, and K. Chen, "Sharing multiple secrets in visual cryptography," *Pattern Recogn.* **40**, 3633–3651 (2007).
9. J. B. Feng, H. C. Wu, C. S. Tsai, Y. F. Chang, and Y. P. Chu, "Visual secret sharing for multiple secrets," *Pattern Recogn.* **41**, 3572–3581 (2008).
10. G. Alvarez, L. H. Encinas, and A. M. del Rey, "A multisecret sharing scheme for color images based on cellular automata," *Inf. Sci.* **178**, 4382–4395 (2008).
11. J. A. Muñoz-Rodríguez and R. Rodríguez-Vera, "Image encryption based on moiré pattern performed by computational algorithms," *Opt. Commun.* **236**(4–6), 295–301 (2004).
12. J. A. Muñoz-Rodríguez and R. Rodríguez-Vera, "Image encryption based on a grating generated by a reflection intensity map," *J. Mod. Opt.* **52**(10), 1385–1395 (2005).
13. J. A. Muñoz-Rodríguez and R. Rodríguez-Vera, "Image encryption based on phase encoding by means of a fringe pattern and computational algorithms," *Revista Mexicana de Física* **52**(1), 53–63 (2006).
14. C.-C. Thien and J.-C. Lin, "Secret image sharing," *Computers and Graphics* **26**(5), 765–770 (2002).
15. H. El-din, H. Ahmed, H. M. Kalash, and O. S. Farag Allah, "Encryption efficiency analysis and security evaluation of RC6 block cipher for digital images," *Intl. J. Computer, Info. Syst. Sci. Eng.* **1**(1), 33–39 (2007).
16. G. Chen, Y. B. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons Fractals* **21**(3), 749–761 (2004).

Kun-Yuan Chao received his BS degree in computer and information science in 1996 from National Chiao Tung University, Taiwan. Then he received his MS degree in computer science and information engineering in 1999 from National Taiwan University, Taiwan. He is currently a PhD candidate in the Department of Computer and Information Science, National Chiao Tung University, Taiwan. His recent research interests include secret image sharing, visual cryptography, and image processing.

Ja-Chen Lin received his BS degree in computer science and MS degree in applied mathematics, both from National Chiao Tung University, Taiwan. He received his PhD degree in mathematics from Purdue University, Lafayette, Indiana. He joined the Department of Computer and Information Science at National Chiao Tung University in 1988, and then became a professor there. His research interests include pattern recognition and image processing.