User-friendly sharing of images: progressive approach based on modulus operations

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

Abstract. *Image sharing is a popular technology to secure important images against damage. The technology decomposes and transforms an important image to produce several other images called shadows or shares. To decode, the shared important image can be reconstructed by combining the collected shadows, as long as the number of collected shadows reaches a specified threshold value. A few sharing methods produce user-friendly (i.e., visually recognizable) shadows—in other words, each shadow looks like a replica of reduced visual quality of a given image, rather than completely meaningless random noise. This facilitates visual management of shadows. (For example, if there are 100 important images and each creates 2 to 17 shadows of its own, then it is easy to visually recognize that a stored shadow is from, say, a House image, rather than from the other 99 images.) In addition to visually recognizable shadows, progressive decoding is also a convenient feature: it provides the decoding meeting a convenient manner to view a moderately sensitive image. Recently, Fang combined both conveniences of visually recognizable shadows and progressive decoding [W. P. Fang, Pattern Recogn.,* **41***, 1410–1414 (2008)]. But that method was memory expensive because its shadows were too big. In order to save memory space, we propose a novel method based on modulus operations. It still keeps both conveniences, but shadows are two to four times smaller than Fang's, and the visual quality of each shadow can be controlled by using a simple expression.* © *2009 SPIE and IS&T.* [DOI: 10.1117/1.3206950]

1 Introduction

Sharing can be utilized to secure an image for storage and transmission. Usually, a sharing method shares an important image among several extremely noisy images called *shadows* or *shares*. By combining these shadows, one can reconstruct the image later. Several works have extended this fundamental concept.[1,2]. Examples include reduction of memory cost for shadows,[3] fast decoding and small pixel expansion rate,[4] and extension of binary visual cryptography (VC) to grayscale images.[5]

Some other extensions are "application-oriented," such as user-friendly shadows[6,7] for easier management of shadows, and progressive decoding[7,8] of an image that is moderately sensitive but needs to be processed frequently. Among these methods, Thien and Lin[6] first introduced the idea of using user-friendly (visually recognizable) shadows, Jin *et al.*[8] developed a progressive technique for grayscale/color images with three types of decryptions to enable recovery in varying qualities, and Fang[7] utilized user-friendly shadows and progressive decoding simultaneously.

From the viewpoint of shadow management, to classify or locate a shadow, attaching a name tag to each shadow in advance is needed if each shadow looks like random noise. (Most reported methods have this kinds of shadows.) Another method is to use visually identifiable shadows. These are also called user-friendly shadows (first mentioned in Ref. 6 and then in Ref. 7), because their visually identifiable features (each shadow looks like a visual-quality-reduced version of a given image) make the job of managing shadows easier for the database manager.

Although Thien and Lin[6] first introduced the idea of using user friendly (visually recognizable) shadows, their method is not progressive, and the reconstruction by all shadows is not lossless. These two weaknesses will be avoided by our method. So far, only Fang's method[7] (which is lossless when all shadows are collected) simultaneously provides two application-convenient features: user-friendly shadows and progressive decoding. Unfortunately, its shadows are four times larger than the input image and thus are not economic in memory. To improve this, we propose here a novel progressive and user-friendly approach based on modulus operations. Better than Fang's method,[7] our method possesses extra advantages: nonexpansion of the shadow size and controllable quality of shadow images. Meanwhile, like Fang's method, our method has lossless recovery, when all n shadows are used, and the decoding complexity is $O(k)$ for the reconstruction using k shadows ($k \leq n$).

The remaining portion of this paper is organized as follows. Section 2 briefly describes Fang's user-friendly progressive sharing method.[7] Section 3 presents the proposed method. Experimental results and some comparisons are shown in Sec. 4. Last, conclusions are given in Sec. 5.

2 Brief Review of Fang's Method

This section reviews briefly Fang's progressive and user-friendly method.[7]

Sharing phase. See the flowchart in Fig. 1 for Fang's sharing phase:

Paper 09003R received Jan. 4, 2009; revised manuscript received Jun. 2, 2009; accepted for publication Jul. 1, 2009; published online Aug. 21, 2009.

1017-9909/2009/18(3)/033008/9/$25.00 © 2009 SPIE and IS&T.

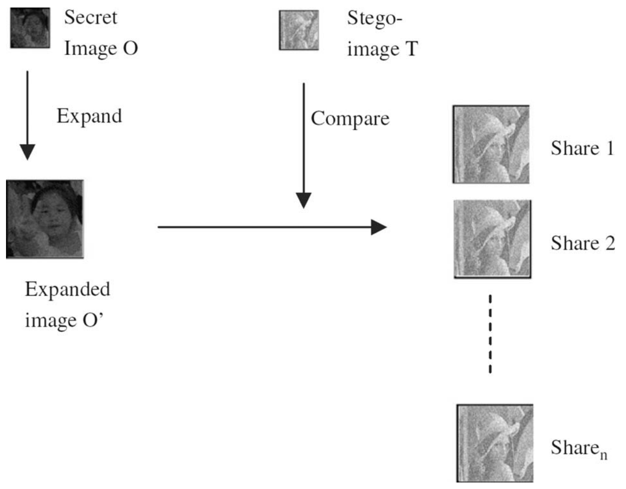

Fig. 1 The sharing flowchart of Fang's method.

Step 1. According to the two leftmost columns in Table 1, expand every pixel $O(x,y)$ of the input binary image O to a 2×2 block at the corresponding position of the expanded image O'. [If $O(x,y)$ is white, then the corresponding 2×2 block is randomly selected from the six possibilities listed in the lower part of column O'.]
Step 2. For each 2×2 block of the expanded image O', by checking the pixel value at the corresponding position of a given stego-image T, Fang randomly selected one of the corresponding patterns listed in the rightmost column of Table 1 to create the 2×2 sharing block at the corresponding position of the first shadow S_1. Similar arguments created each of the remaining $n-1$ shadows.

Recovering phase. Assume that k shadows are collected. Then, each pixel j of the black-or-white image is reconstructed using the k sharing pixels at the same position j of the k shadows. The reconstruction rule is an OR-like operation: The reconstructed pixel is black iff at least one of the k sharing pixels is black.

Fang's method has two disadvantages: (1) The size of each shadow S_i is four times larger than the input image O; and (2) the image quality [such as peak signal-to-noise ratio (PSNR)] of shadows is not easy to control. We will improve these aspects.

3 Proposed Method

This section presents our user-friendly progressive sharing method based on modulus operations. The method generates n user-friendly shadows whose image quality (such as PSNR) is lower than the input image's quality; later, the input image can be reconstructed with progressively improved image quality after gathering k $(2\leqslant k\leqslant n)$ shadows. The description of the method is divided into three subsections. First, a fundamental (n,n) sharing version based on modulus operations is introduced in Sec. 3.1. This simple version is neither user-friendly nor progressive. Then, the fundamental version is extended in Sec. 3.2 to an intermediate version with friendly shadows, although the intermediate version is still nonprogressive. Last, Sec. 3.3 presents the final version by extending the intermediate (user-friendly) version further to include both progressive decoding and user-friendly features. A comparison between our progressive and user-friendly method (Sec. 3.3) and Fang's (Sec. 2) is given in Sec. 3.4.1, while a stego version of our method is given in Sec 3.4.2.

3.1 *An (n,n) Fundamental Sharing Version Based on Modulus Operations*

This section illustrates a fundamental (n,n) sharing version for grayscale images based on modulus operations. This version splits a grayscale image A among n extremely noisy shadows $B_1,B_2,\ldots,B_n$ whose sizes are all the same as A. The n noisy shadows together can reconstruct each pixel of A by using one modulus operation and $n-1$ addition. (In this paper, + and **Mod** denote addition and modulus operations, respectively.) The sharing and recovering phases of the fundamental version are listed in the following.

Sharing phase.

Step 1. Input a grayscale secret image A.
Step 2. Generate $n-1$ random images $B_1,B_2,\ldots,B_{n-1}$ as shadows. Each is as large as A.
Step 3. Create the n'th shadow B_n by

$$B_n=(A+\{256-[(B_1+B_2+\ \ldots+B_{n-1})_{\mathbf{Mod}\ 256}]\})_{\mathbf{Mod}\ 256}. \tag{1}$$

Step 4. Output the n noisy (nonfriendly) shadows $B_1,B_2,\ldots,B_n$.

Recovering phase. Retrieve A using the formula

$$A=(B_1+B_2+\ \ldots+B_n)_{\mathbf{Mod}\ 256}. \tag{2}$$

Notably, both + and **Mod** are pixel-by-pixel operations. This sharing scheme can also work for binary or color images by using $2(=2^1)$ and $16777216(=2^{24})$, respectively, to replace the constant 256 in the two preceding formulas. An experiment using the grayscale image Lena as image A is shown in Fig. 2, with $(n,n)=(4,4)$.

3.2 *A User-Friendly but Nonprogressive (n,n) Version*

This section describes how to extend the (n,n) fundamental version in Sec. 3.1 to an intermediate version whose n shadows are all user-friendly. What we do is to use a smaller value m to replace the value 256 in the modulus operations in Sec. 3.1.

Sharing phase.

Step 1. Input an integer parameter m $(2\leqslant m\leqslant 256)$ and an 8-bit grayscale image A. Generate a smaller range image

$$A'=(A)_{\mathbf{Mod}\ m}, \tag{3}$$

whose size is identical to A, but with pixel value less than m (rather than 256).

Table 1 Fang's selection of sharing patterns (Ref. 7).

Secret pixel $O(x,y)$	Expanded secret O'	Cover pixel $T(x,y)$	Possible choices for the related 2×2 block of a share $S_i(1\leq i\leq n)$
B[a]	(B,B,B,B)	B	(B,B,W,W),(B,W,B,W), (B,W,W,B), (W,B,B,W), (W,B,W,B), (W,W,B,B) (W,W,W,W),(B,W,W,W),(W,B,W,W),
		W	(W,W,B,W), (W,W,W,B)
W	(B,B,W,W)[b]	B	(B,B,W,W)
		W	(W,W,W,W), (B,W,W,W), (W,B,W,W)
	(B,W,B,W)	B	(B,W,B,W)
		W	(W,W,W,W), (B,W,W,W), (W,W,B,W)
	(B,W,W,B)	B	(B,W,W,B)
		W	(W,W,W,W), (B,W,W,W), (W,W,W,B)
	(W,B,B,W)	B	(W,B,B,W)
		W	(W,W,W,W), (W,B,W,W), (W,W,B,W)
	(W,B,W,B)	B	(W,B,W,B)
		W	(W,W,W,W), (W,B,W,W), (W,W,W,B)
	(W,W,B,B)	B	(W,W,B,B)
		W	(W,W,W,W), (W,W,B,W), (W,W,W,B)

Note: (See Fig. 1 for definitions of O, O', and T).

[a]B represents a black pixel; W represents a white pixel.

[b]Each 2×2 block in the expanded image O' (or in each shadow S_i) is represented as (left-top pixel, right-top pixel, left-bottom pixel, right-bottom pixel).

Step 2. Generate $n-1$ "random" images $B'_1,\ldots,B'_{n-1}$ whose sizes are all as large as A, but each pixel is a random value chosen from $[0,1,\ldots,(m-1)]$. Then create

$$B'_n=(A'+\{m-[(B'_1+B'_2+\ \ldots+B'_{n-1})_{\mathbf{Mod}\ m}]\})_{\mathbf{Mod}\ m}, \tag{4}$$

which implies that $(B'_1+B'_2+\ldots+B'_n)_{\mathbf{Mod}\ m}=A'$.

Step 3. Output the n friendly shadows $\{B_1,\ldots,B_n\}$ defined by

$$B_i=(A-A')+B'_i \quad \text{for } i=1,\ldots,n. \tag{5}$$

Recovering phase. Retrieve A by

$$A=[B_i-(B_i)_{\mathbf{Mod}\ m}]+[(B_1+B_2+\ \ldots+B_n)_{\mathbf{Mod}\ m}]. \tag{6}$$

In Eq. (6), it does not matter which one of $B_1,B_2,\ldots,B_n$ is used as B_i; the result is the same. Also, if $m=256$ is used in Eqs. (3)–(6), then this intermediate version is identical to the (n,n) sharing one in Sec. 3.1.

3.3 *The User-Friendly and Progressive Version*

The intermediate version (Sec 3.2) is still nonprogressive, although user-friendly. Section 3.3 extends the intermediate

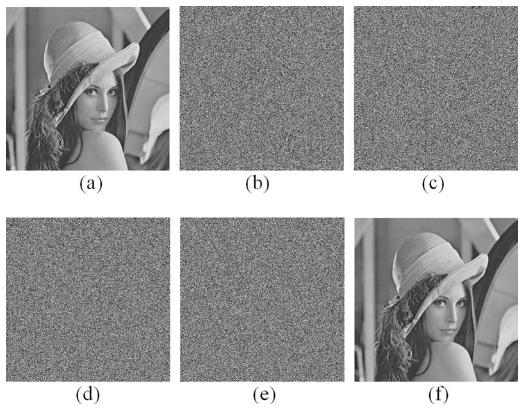

Fig. 2 An example of the (n,n) fundamental sharing version introduced in Sec 3.1. Here, $(n,n)=(4,4)$; (a) is the given grayscale image Lena A; (b) to (e) are the four generated "nonfriendly" shadows B_1, B_2, B_3, B_4; and (f) is the recovered error-free Lena using the formula $A=(B_1+B_2+B_3+B_4)_{\mathbf{Mod}\ 256}$.

version to a progressive one. Because it is an extension of Sec. 3.2, the modulus-base notation m $(2\leq m\leq 256)$ is still used in this section.

Sharing phase.

Step 1. Input an integer parameter m $(2\leq m\leq 256)$ and an 8-bit grayscale secret image A. (A can also be one of the three 8-bit color-components of a 24-bit color image.)

Step 2. In a pixel-by-pixel manner, generate a smaller-range image

$$A' = (A)_{\mathbf{Mod}\ m}, \tag{7}$$

whose size is identical to A, but pixel value is at most $m-1$, rather than 255.

Step 3. Generate $n-1$ random images $R_1,R_2,\ldots,R_{n-1}$. (Each image R_i is as large as A, and each pixel of R_i is 8-bit.)

Step 4. Create n images $B'_1,B'_2,\ldots,B'_n$ in a pixel-by-pixel manner:

If $A'=0$, then $B'_1=0$; else $B'_1=(R_1)_{\mathbf{Mod}(A'+1)}$. Anyway, define $A'_1=A'-B'_1$.

If $A'_1=0$, then $B'_2=0$; else $B'_2=(R_2)_{\mathbf{Mod}(A'_1+1)}$. Anyway, define $A'_2=A'_1-B'_2$.

…

If $A'_{n-2}=0$, then $B'_{n-1}=0$; else $B'_{n-1}=(R_{n-1})_{\mathbf{Mod}(A'_{n-2}+1)}$. Anyway, define $A'_{n-1}=A'_{n-2}-B'_{n-1}$. Last, let $B'_n=A'_{n-1}$.

Here, $B'_1=(R_1)_{\mathbf{Mod}(A'+1)}$ means that $B'_1(t)=[R_1(t)]_{\mathbf{Mod}[A'(t)+1]}$ at pixel t. Then, after creating $B'_1(t)$, we create $A'_1(t)$ by the formula $A'_1(t)=A'(t)-B'_1(t)$. The explanation for the remaining operations in step 4 is the same. Also, as t changes, for random effect, we randomly switch the order of assigning these values to $[B'_1(t),B'_2(t),\ldots,B'_n(t)]$. For example, when $t=0$, assign the computed values to $B'_1(t),B'_2(t),\ldots,B'_n(t)$ as earlier, respectively; then, when $t=1$, assign the computed values to $B'_n(t),B'_{n-1}(t),\ldots,B'_1(t)$, respectively; then, when $t=2,\ldots$. Here, we may use a random number generator to create the permutation order for this.

Step 5. Output n final shadows $B_1,B_2,\ldots,B_n$ defined by

$$B_i = (A - A') + B'_i \quad \text{for } i = 1,\ldots,n. \tag{8}$$

Recovering phase. After gathering any $k(2\leq k\leq n)$ shadows

$$B_{i(1)},B_{i(2)},\ldots,B_{i(k)} \quad (1\leq i(j)\leq n \text{ for } 1\leq j\leq k),$$

which are a subset of the n shadows $\{B_1,B_2,\ldots,B_n\}$, retrieve A using the formula

$$\widetilde{A} = [B_{i(j)} - (B_{i(j)})_{\mathbf{Mod}\ m}] + [(B_{i(1)} + B_{i(2)} + \ldots + B_{i(k)})_{\mathbf{Mod}\ m}]. \tag{9}$$

Here, $B_{i(j)}$ can be any one of $B_{i(1)},B_{i(2)},\ldots,B_{i(k)}$.

Lemma 1. In Eq. (9), any one of $B_{i(1)},B_{i(2)},\ldots,B_{i(k)}$ can be used as $B_{i(j)}$.

Proof. Equation (8) implies that

$$B_i - B'_i = A - A' \quad \text{for all } i=1,\ldots,n, \tag{10}$$

so we have $(B_{i(j)}-B'_{i(j)})_{\mathbf{Mod}\ m}=(A-A')_{\mathbf{Mod}\ m}$ for all $1\leq i(j)\leq n$ for $1\leq j\leq k$. However, $(A-A')_{\mathbf{Mod}\ m}=0$ because $A'=(A)_{\mathbf{Mod}\ m}$ by Eq. (7). Hence, $(B_{i(j)}-B'_{i(j)})_{\mathbf{Mod}\ m}=0$. So

$$(B_{i(j)})_{\mathbf{Mod}\ m} = (B'_{i(j)})_{\mathbf{Mod}\ m} = B'_{i(j)}, \tag{11}$$

where the last identity is due to the fact that $B'_{i(j)}<(A'+1)$ by step 4 earlier, and the range of A' is $\{0,\ldots,m-1\}$ by Eq. (7). We may thus say that

$$B_{i(j)} - (B_{i(j)})_{\mathbf{Mod}\ m} = B_{i(j)} - B'_{i(j)} = A - A' \quad (\text{here, } 1\leq i(j) \leq n \text{ for } 1\leq j\leq k). \tag{12}$$

End of proof

Lemma 2. In step 4 of the preceding sharing phase,

$$A' = B'_1 + B'_2 + \ldots + B'_{n-1} + B'_n. \tag{13}$$

Proof. Because $A'_1=A'-B'_1$, we have $A'=B'_1+A'_1$.
Because $A'_2=A'_1-B'_2$, we have $A'=B'_1+A'_1=B'_1+B'_2+A'_2$.
Because $A'_3=A'_2-B'_3$, we have $A'=B'_1+B'_2+B'_3+A'_3$.
…
Because $A'_{n-1}=A'_{n-2}-B'_{n-1}$, we have $A'=B'_1+B'_2+\ldots+B'_{n-1}+A'_{n-1}$.
Last, because $B'_n=A'_{n-1}$, we have $A'=B'_1+B'_2+\ldots+B'_{n-1}+B'_n$.

End of proof

Lemma 3. When all n shadows are received—i.e., when $k=n$—then A can be recovered losslessly by Eq. (9). In other words,

$$A = [B_i - (B_i)_{\mathbf{Mod}\ m}] + [(B_1 + B_2 + \ldots + B_n)_{\mathbf{Mod}\ m}]. \tag{14}$$

(Again, it does not matter which one of $\{B_1,B_2,\ldots,B_n\}$ is used as B_i.)

Proof. Here, we show why the recovery image $\widetilde{A}$ becomes the original image A when $k=n$. Since $k=n$, Eqs. (9), (12), and (13) imply that

$$\widetilde{A} = [B_{i(j)} - (B_{i(j)})_{\mathbf{Mod}\ m}] + [(B_{i(1)} + B_{i(2)} + \ldots + B_{i(n)})_{\mathbf{Mod}\ m}],$$

$$= [B_i - (B_i)_{\mathbf{Mod}\ m}] + [(B_1 + B_2 + \ldots + B_n)_{\mathbf{Mod}\ m}],$$

$$= [A - A'] + [(B_1)_{\mathbf{Mod}\ m} + (B_2)_{\mathbf{Mod}\ m} + \ldots + (B_n)_{\mathbf{Mod}\ m}]_{\mathbf{Mod}\ m},$$

$$= [A - A'] + [B'_1 + B'_2 + \ldots + B'_n]_{\mathbf{Mod}\ m},$$

$$= [A - A'] + [A'],$$

$$= A.$$

End of proof

Table 2 The PSNR of shadows when n=4 shadows were generated for each image.

m's value	$PSNR(B_i)$ in Eq. (19)	PSNR in Lena's shadows	PSNR in Jet's shadows	PSNR in Monkey's shadows	PSNR in Pepper's shadows	PSNR in Boat's shadows
m=256	7.26	7.37	7.40	7.16	7.45	7.34
m=128	13.31	13.41	13.74	13.12	13.09	13.50
m=64	19.40	19.02	20.09	18.46	18.75	19.89
m=32	25.56	25.45	25.13	25.36	25.46	25.85
m=16	31.87	31.21	31.21	31.15	31.11	31.97

Step 4 implies that each pixel of $B'_1, B'_2, \ldots, B'_n$ is nonnegative because each pixel is created by a modulus function. Moreover, in step 4, the pixel values of A' are distributed randomly among $B'_1, B'_2, \ldots, B'_n$, and Eq. (13) reads

$$B'_1 + B'_2 + \ldots + B'_{n-1} + B'_n = A',$$

in whichall pixel values are nonnegative. So to estimate the image quality (PSNR) of shadows $B_1, B_2, \ldots, B_n$, we may start from the rough estimation

$$B'_i \approx \frac{A'}{n}. \tag{15}$$

Now, the root-mean-square error (RMSE) for each B_i ($1 \leqslant i \leqslant n$), as compared with the input image A, is defined as

$$RMSE(B_i) = \left\{ \frac{\sum_{all\ t} [A(t) - B_i(t)]^2}{Count(t)} \right\}^{1/2}. \tag{16}$$

Here, $A(t)$ is a pixel value in A, and $B_i(t)$ is in B_i. By Eq. (8), $RMSE(B_i)$ is evaluated as

$$\left(\frac{\sum_{all\ t} \{A(t) - [A(t) - A'(t) + B'_i(t)]\}^2}{Count(t)} \right)^{1/2},$$

which can be reduced as

$$\left\{ \frac{\sum_{all\ t} \left[\frac{A'(t) \times (n-1)}{n} \right]^2}{Count(t)} \right\}^{1/2}$$

by Eq. (15). Because $(n-1)/n$ is a given constant due to the known value of n, the preceding rough estimation of $RMSE(B_i)$ can be rewritten as

$$\left[\frac{\sum_{all\ t} A'(t)^2}{Count(t)} \right]^{1/2} \times \frac{(n-1)}{n}.$$

Although the actual value of

$$\left[\frac{\sum_{all\ t} A'(t)^2}{Count(t)} \right]^{1/2}$$

depends on the histogram of the image A', we may roughly estimate

$$\left[\frac{\sum_{all\ t} A'(t)^2}{Count(t)} \right]^{1/2}$$

as

$$\left[\left(\int_0^{m-1} t^2 \mathrm{d}t \right) / (m-1) \right]^{1/2} = (m-1)/1.73, \tag{17}$$

which is the probabilistic average value considering the fact that $A'(t) \in \{0, 1, \ldots, (m-1)\}$. Therefore, we have

$$RMSE(B_i) \approx \frac{(m-1) \times (n-1)}{1.73 \times n}. \tag{18}$$

Then, we can get the rough estimation

$$PSNR(B_i) = 10 \times \log_{10} \frac{255^2}{[RMSE(B_i)]^2} \approx 10 \times \log_{10} \frac{255^2}{\left[\frac{(m-1) \times (n-1)}{1.73 \times n} \right]^2}. \tag{19}$$

Some experimental results of $PSNR(B_i)$ are shown in Table 2, which uses the five images in Fig. 2(a) and Fig. 3. From

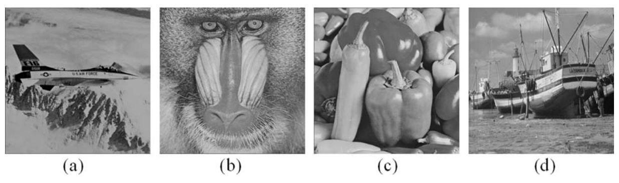

Fig. 3 The other four images {Jet, Monkey, Pepper, Boat} used in Table 2.

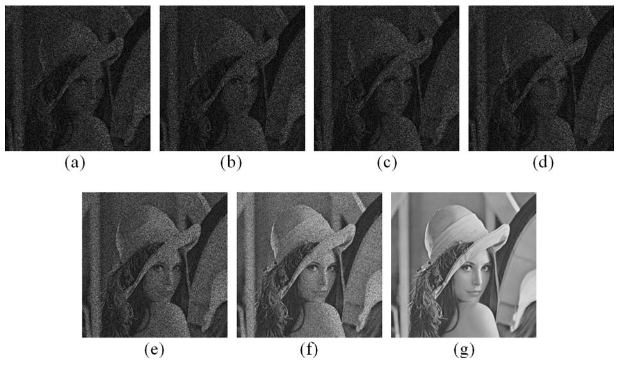

Fig. 4 An example of the $(n=4)$ case using $m=256$ in the non-stego version (Sec. 3.3). Here, (a) to (d) are the final shadows B_1, B_2, B_3, B_4 [RMSE=109.13 and PSNR=7.37 for (a) to (d))]; (e) to (g) are the recovered Lena images [RMSE=80.22 and PSNR=10.04 for (e); RMSE=49.75 and PSNR=14.20 for (f); lossless for (g)] using (respectively) any two, any three, and all four final shadows.

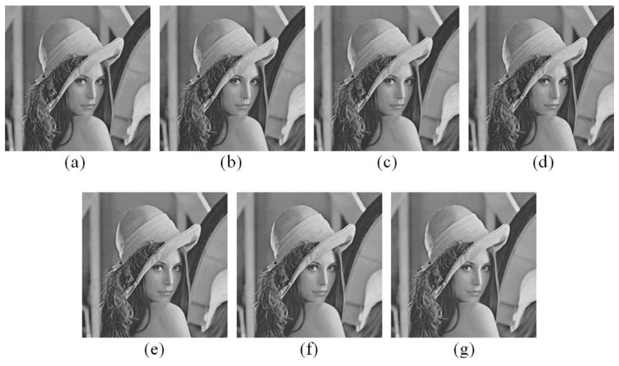

Fig. 6 An example of the $(n=4)$ case using $m=16$ in the non-stego version (Sec. 3.3). Here, (a) to (d) are the final shadows B_1, B_2, B_3, B_4 [RMSE=7.01 and PSNR=31.21 for (a) to (d)]; (e) to (g) are the recovered Lena images [RMSE=5.21 and PSNR=33.79 for (e); RMSE=3.28 and PSNR=37.81 for (f); lossless for (g)] using (respectively) any two, any three, and all four final shadows.

this table, we can see that the experimental value of PSNR is close to the estimation given by Eq. (19).

In our experiments, for the same value k, all reconstructed images have similar PSNR values. For example, in each of the three experimental results of Figs. 4–6, the four images respectively reconstructed by shadows $\{B_1,B_2,B_3\}$ (or by $\{B_1,B_2,B_4\}$, or by $\{B_1,B_3,B_4\}$, or by $\{B_2,B_3,B_4\}$) all have very similar PSNR values. Likewise, the six images reconstructed by any two shadows of $\{B_1,B_2,B_3,B_4\}$ also have similar PSNR values. In the recovering phase, when more shadows are gathered (k becomes larger), the reconstructed image then has higher image quality. In particular, when all n shadows are gathered, then $k=n$, and the reconstructed image A is error-free due to Lemma 3. In summary, the proposed version has a progressive decoding feature, and it uses only one subtraction, two modulus operations, and k additions to reconstruct a gray value from pixels of k available shadows.

The $+$, $-$, and **Mod** in this section are all byte-by-byte operations among gray values. Hence, if the input image is color (24 bits per pixel), then A must be first decomposed into three components (A^R, A^G, and A^B) of 8 bits each. Then the preceding sharing process is implemented for each component to generate n shadows. Then, for each index $i=1,\ldots,n$, the three corresponding shadows B_i^R, B_i^G, and B_i^B are combined to get final shadow B_i.

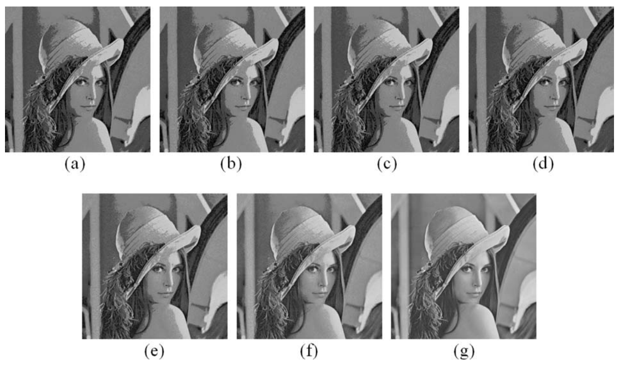

Fig. 5 An example of the $(n=4)$ case using $m=64$ in the non-stego version (Sec. 3.3). Here, (a) to (d) are the final shadows B_1, B_2, B_3, B_4 [RMSE=28.54 and PSNR=19.02 for (a) to (d)]; (e) to (g) are the recovered Lena images [RMSE=21.07 and PSNR=21.66 for (e); RMSE=13.10 and PSNR=25.79 for (f); lossless for (g)] using (respectively) any two, any three, and all four final shadows.

3.4 *Comparison with Fang's Method and a Stego Version of Our Method*

3.4.1 *Comparison with Fang's method*

Comparing to Fang's method[7] reviewed in Sec. 2, which is also user-friendly and progressive, our method in Sec. 3.3 has two more advantages:

- The size of each of our shadows in $B_1,B_2,\ldots,B_n$ is the same as A (not expanded).
- Our shadows' image quality $PSNR(B_i)$ can be roughly controlled by the base parameter m of modulus operations ($2 \leqslant m \leqslant 256$ is an integer). Just estimate m by

$$m \approx \frac{441 \times n}{(10^{PSNR(B_i)/10})^{1/2} \times (n-1)} + 1. \tag{20}$$

Equation (20) is derived from Eq. (19), an estimation tool whose validity is checked in Table 2.

3.4.2 *Stego version of our method*

In Fang's method,[7] each shadow is hidden using a cover image T (also known as a *host image*) so that all shadows (called *stego-shadows*) look like T. Our method in Sec. 3.3 can also be modified to have a stego version by using stego-shadows smaller in size than Fang's. Our stego version is as follows.

Sharing phase.

Step 1. Input an integer parameter m ($2 \leqslant m \leqslant 64$ in stego version, but $16 \leqslant m \leqslant 64$ is suggested to avoid large *per*); input an 8-bit grayscale cover image T whose size $(w \times h)$ is also the size of the 8-bit grayscale secret image A.

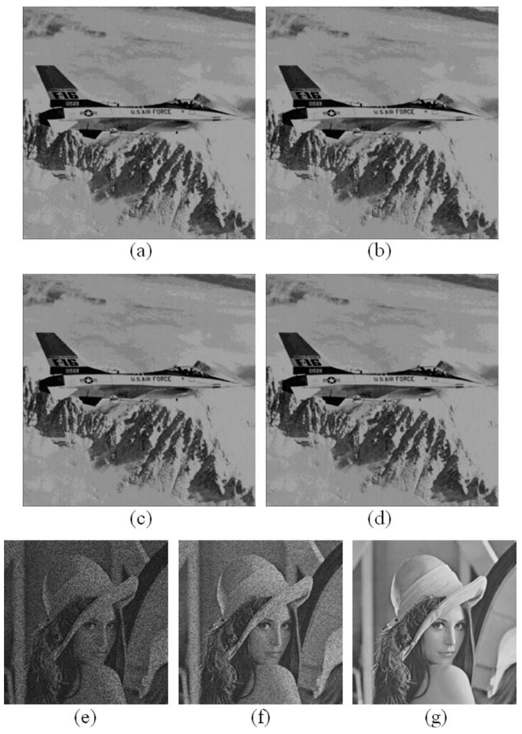

Fig. 7 An example of the (n=4) case using m=32 in the stego version (Sec. 3.4.2). Here, (a) to (d) are the final stego-shadows B_1, B_2, B_3, and B_4; (e) to (g) are the progressively recovered Lena images using, respectively, "any" two, "any" three, and all four final shadows. PSNR=26.66 for (a) to (d); PSNR=10.04 for (e); PSNR =14.21 for (f); and (g) is lossless.

Step 2. Let $sz=8/\lceil \log_2 m \rceil$. Use pixels duplication to expand T to a larger image T' whose size is

$$(\sqrt{sz}\times w)\times(\sqrt{sz}\times h). \tag{21}$$

Step 3. Generate $n-1$ random images $R_1,R_2,\ldots,R_{n-1}$ (Each image R_i is as large as A, and each pixel of R_i is 8-bit.)
Step 4. Create n images $B'_1,B'_2,\ldots,B'_n$ according to step 4 of the sharing phase in Sec. 3.3, except that here we use A to replace the role of A' in all formulas there.
Step 5. Use a random key r to create an order to permute all pixels in B'_1. Each of the remaining $n-1$ images $B'_2,\ldots,B'_n$ is also permuted using the random key r. Then use Shamir's $(2,n)$)-threshold sharing method[1] to share the key r among n created numbers $r_1,r_2,\ldots,r_n$. Then, store r_i in B'_i for each $i=1,\ldots,n$.
Step 6. Treat each grayscale image B'_i $(1\leq i\leq n)$ as a bit stream (i.e., a very big binary integer), then partition each B'_i to $(\sqrt{sz}\times w)\times(\sqrt{sz}\times h)$ smaller range numbers $B'_i(t)$ [Here, $0\leq B'_i(t)<m$ and $0\leq t<(sz\times w\times h)$.] Then hide each number $B'_i(t)$ in $T'(t)$ to get a pixel value $B_i(t)$ by the formula

$$B_i(t)=round\left[\frac{T'(t)-B'_i(t)}{m}\right]\times m+B'_i(t), \tag{22}$$

where the *round* operator rounds its argument to the nearest integer. Add (subtract) m to (from) the result of Eq. (22) if $B_i(t)<0$ or >255.
Step 7. Output n stego-shadows $B_1,B_2,\ldots,B_n$ whose sizes are all identical to T'.

Recovering phase.

Step 1. After gathering any k $(2\leq k\leq n)$ shadows $\{B_{i(1)},B_{i(2)},\ldots,B_{i(k)}\}\subset\{B_1,B_2,\ldots,B_n\}$, where $1\leq i(j)\leq n$ for each $j=1,\ldots,k$, retrieve all $(sz\times w\times h)$ smaller range numbers $B'_{i(j)}(t)$ in each stego-image $B_{i(j)}$ by the dehiding formula:

$$B'_{i(j)}(t)=[B_{i(j)}(t)]_{\mathbf{Mod}\ m} \quad \text{for } t=0,\ldots,(sz\times w\times h)-1. \tag{23}$$

Step 2. Combine the $(sz\times w\times h)$ smaller range numbers $B'_{i(j)}(t)$ to retrieve each $B'_{i(j)}$ as an 8-bit grayscale image of $w\times h$ pixels.
Step 3. Recover the random key r by inverse sharing. Then use the key r to restore the original pixels' order in image $B'_{i(1)},B'_{i(2)},\ldots,B'_{i(k)}$.
Step 4. Last, retrieve A in pixel-by-pixel manner by the formula

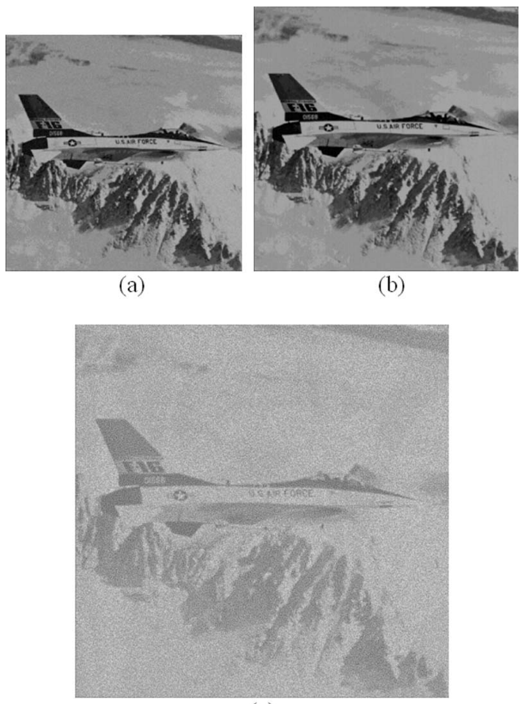

Fig. 8 Comparing the stego-shadows in two stego methods for the (n=4) case. The hidden image is Lena [Fig. 2(a)], and the host image is Jet [Fig. 3(a)]. Here, (a) is one of the four stego-shadows with PSNR=26.66 dB in our stego version (when m=32); (b) is one of the four stego-shadows with PSNR=31.26 dB in our stego version (when m=16); (c) is one of the four stego-shadows with PSNR=10.02 in Fang's method (Sec. 2). Note that our stego size is only 1.6 times [in (a)] or 2 times [in (b)] larger than the original Jet image's size, whereas Fang's stego size is 4 times larger than original Jet.

Table 3 Comparisons with reported image sharing methods (Refs. 3–8).

Methods	Computational complexity[a]	Memory size[b] for each shadow	Recovered quality[c]
Wang and Su (Ref. 3)	$O(\log^2 k)$ (math operations)[d]	$per \approx (1/k) \times 60\% \geqslant (1/n) \times 60\%$	Lossless
Wang *et al.* (Ref. 4)	$k-1$ (XOR operations)	$per=1$	Lossless
Lin and Tsai (Ref. 5)	$O(k \times per)$ (OR-like operations)	$per \geqslant 2$	Lossless
Thien and Lin (Ref. 6) (visually recognizable shadows)	$O(\log^2 k)$ (math operations)	$per \approx 1/k \geqslant 1/n$	Lena's PSNR=37.98 Jet's PSNR=39.93 Monkey's PSNR=35.33
Fang (Ref. 7) (visually recognizable shadows and progressive)	$4 \times (k-1)$ (OR-like operations)	$per=4$	Lossless
Jin *et al.* (Ref. 8) (progressive)	$4 \times (k-1)$ (XOR operations)	$per=4$	Lossless
Section 3.3 (visually recognizable shadows and progressive)	k additions; 2 Mod operations; 1 subtraction	$per=1$	Lossless
Our stego version, Sec. 3.4.2 (visually recognizable shadows and progressive)	$(k-1)$ additions; 2 Mod operations; 1 attaching of a short binary number to the other to get an 8-bit number	$1.33 \leqslant per = 8/\lceil \log_2 m \rceil \leqslant 2$ when $64 \geqslant m \geqslant 16$	Lossless

[a]Operations needed to recover one secret pixel by k shadows in (k, n) system.
[b]The pixel expansion rate (*per*) of each shadow as compared to the input secret image.
[c]The secret image recovered by all shadows.
[d]Math operations: $+,-,\times,\div$.

$$\widetilde{A} = B'_{i(1)} + B'_{i(2)} + \ldots + B'_{i(k)}. \quad (24)$$

In our preceding stego version, the final stego-shadows $B_1, B_2, \ldots, B_n$ are $sz = 8/\lceil \log_2 m \rceil$ times larger than the input secret image A. So the pixel expansion rate is $per = 8/\lceil \log_2 m \rceil \leqslant 8/4 = 2$ if we set the parameter $m \geqslant 16$. An example using $m=32$ is shown in Fig. 7, where the Jet images are stego-shadows utilized to cover (and progressively recover) the important image Lena. In this example ($m=32$), our stego version's pixel expansion rate is $per = 8/5 = 1.6$, better than Fang's $per=4$ [shown in Fig. 8(c)]. Moreover, our shadows' image quality is also better than Fang's. For example, as shown in Figs. 7(a)–7(d) or Fig. 8(a), our Jet shadows have image quality of PSNR =26.66 dB. [PSNR would be 31.26 dB, as shown in Fig. 8(b), if we used $m=16$ to get the shadows whose size are all two times larger than the original Jet image.] On the contrary, after implementing Fang's method in each bit-plane of the same grayscale important image A (Lena) and the same cover image T (Jet), each of Fang's $n=4$ quadruple-size stego-shadows has PSNR=10.02 dB only [see Fig. 8(c)]. Our stego version is still progressive in decoding, lossless when all n shadows are collected, and has small decoding complexity $O(k)$ when k of the n shadows are used in decoding.

4 Experimental Results and Some Comparisons

4.1 *Experimental Results*

In the proposed method in Sec. 3.3, the input image A is the grayscale image Lena in Fig. 2(a). Figure 4 shows the experimental result for the $(n=4)$ case when $m=256$. The image A can be roughly seen in any of the four generated user-friendly shadows shown in Figs. 4(a)–4(d). In Figs. 4(e)–4(g), when more shadows are available in retrieval, the recovered image has better quality.

Other experiments using $m=64$ and $m=16$ for $(n=4)$ case are shown in Figs. 5 and 6, respectively. The shadows in Fig. 6 have higher PSNR than those in Figs. 4 and 5 due to the use of a smaller m value. This is according to Eq. (19), where we have

$$PSNR(B_i) \approx 10 \times \log_{10} \frac{(441 \times n)^2}{[(m-1) \times (n-1)]^2} = 10 \times \log_{10} \frac{(441 \times 4)^2}{[(m-1) \times 3]^2} \text{ because } n = 4.$$

Notably, when $m=256$, 64, and 16, respectively, the $PSNR(B_i)$ values estimated by Eq. (19) are 7.26 db, 19.40 dB, and 31.87 dB. These are all very close to the actual PSNR values of the shadows (7.37 dB, 19.02 dB, and 31.21 dB, respectively) shown in Figs. 4–6.

4.2 *Comparisons*

Our method provides at least two convenient features: it is user-friendly and progressive. We can compare our method with other image-sharing researches.[3–8] Table 3 compares in three aspects: computational complexity to reconstruct a pixel; memory space of a shadow (represented by pixel expansion rate [*per*], as compared to the size of input image); and image quality of the image recovered by all n shadows. Table 3 shows that in our method: (1) each pixel can be reconstructed by k shadows using about k operations; (2) the size of each shadow is not expanded ($per=1$) for the nonstego version; and (3) the recovery by all n shadows is lossless. Although our *per* or computational complexity is in the middle rank rather than the best, note that in Table 3, only Refs. 6 and 7 and ours are user-friendly (provide visually recognizable shadows). In these three user-friendly approaches, Ref. 7 is four times expanded in shadow size, whereas Ref. 6 is neither progressive nor lossless in recovery. As for Refs. 3–5, they are neither progressive nor user-friendly.

To compare with Fang's further, we provide the stego version in Sec 3.4.2, in which the pixel expansion rate (*per*) is $1.33 \leqslant per = 8/\lceil \log_2 m \rceil \leqslant 2$ when $64 \geqslant m \geqslant 16$. For example, $per=1.6$ when $m=32$. These *per* values are still better than Fang's $per=4$. (Hence, regardless of whether the stego version is used, our *per* is better than Fang's.) Moreover, our stego-shadow's image quality is also better than Fang's. (See Fig. 8; our Jet stego-shadows are with PSNR $=26.66$ dB for $m=32$ and 31.26 dB for $m=16$, both better than Fang's 10.02 dB.)

5 Conclusion

In this paper, based on modulus operations, we successfully designed a novel image sharing method with user- friendly shadows and progressive decoding. According to the experimental results and comparisons in Sec. 4, in addition to being user-friendly, and progressive, each pixel is reconstructed by k shadows quickly with about k operations, and the recovery is lossless after collecting all n shadows. The proposed method also provides following features: the non-stego-shadows' image quality can be controlled by the parameter value m using Eq. (20); each shadow is not expanded in the non-stego version (Sec. 3.3) and is only $1.33 \leqslant per = 8/\lceil \log_2 m \rceil \leqslant 1.6$ times larger than the original secret image if we restrict $64 \geqslant m \geqslant 32$ in the stego version (Sec. 3.4.2); and the stego-shadows have quality much better than Fang's shadows (Fig. 8).

Acknowledgment

This work is supported by the National Science Council, Taiwan, R.O.C., under Grant NSC 97-2221-E-009-120-MY3.

References

1. A. Shamir, "How to share a secret," *J. Assoc. Comput. Mach.* **22**(11), 612–613 (1979).
2. M. Naor and A. Shamir, "Visual cryptography," in *Advances in Cryptology-Eurocript'94*, A. D. Santis, Ed., *Lect. Notes Comput. Sci.* **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. D. Wang, L. Zhang, N. Ma, and X. Li, "Two secret sharing schemes based on Boolean operations," *Pattern Recogn.* **40**, 2776–2785 (2007).
5. C. C. Lin and W. H. Tsai, "Visual cryptography for gray-level images by dithering techniques," *Pattern Recogn. Lett.* **24**, 349–358 (2003).
6. C. C. Thien and J. C. Lin, "An image-sharing method with user-friendly shadow images," *IEEE Trans. Circuits Syst. Video Technol.* **13**(12), 1161–1169 (2003).
7. W. P. Fang, "Friendly progressive visual secret sharing," *Pattern Recogn.* **41**, 1410–1414 (2008).
8. D. Jin, W. Q. Yan, and M. S. Kankanhalli, "Progressive color visual cryptography," *J. Electron. Imaging* **14**(3), 033019 (2005).

Kun-Yuan Chao received his BS degree in computer and information science in 1996 from National Chiao Tung University, Taiwan. 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 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 his MS degree in applied mathematics, both from National Chiao Tung University, Taiwan. He received his PhD degree in mathematics from Purdue University, West 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.