Multithreshold progressive image sharing with compact shadows

Lee Shu-Teng Chen
Ja-Chen Lin
National Chiao Tung University
Department of Computer Science and Information Engineering
Hsinchu, Taiwan, 300
E-mail: stlee@cs.nctu.edu.tw

Abstract. *We propose a multithreshold progressive reconstruction method. The image is encoded three times using Joint Photographic Experts Group (JPEG): first with a low-quality factor, then with a medium-quality factor, and last with a high-quality factor. Huffman coding is employed to encode the difference between the important image and the high-quality JPEG decompressed image. The three JPEG codes and the Huffman code are shared, respectively, according to four prespecified thresholds. The n-generated equally important shadows can be stored or transmitted using n channels in parallel. Cooperation among these generated shadows can progressively reconstruct the important image. The reconstructed image is loss-free when the number of collected shadows reaches the largest threshold. Each shadow is very compact and so can be hidden successfully in the JPEG codes of cover images to reduce the probability of being attacked when transmitted in an unfriendly environment. Comparisons with other image sharing methods are made. The contributions, such as easiness to apply to scalable Moving Picture Experts Group (MPEG) video transmission or resistance to differential attack, are also included.* © 2010 SPIE and IS&T.
[DOI: 10.1117/1.3295710]

1 Introduction

Secret sharing is an approach for protecting data.[1–9] Blakley[1] and Shamir[2] were the first to propose the idea of a (t,n) threshold sharing scheme. Polynomials were used to share a secret among n participants. Any t of the n participants $(t \leqslant n)$ could reconstruct the secret, but $t-1$ participants could not. Thien and Lin[4] extended the work of Shamir by sharing a secret image and generated n shadows. The size of each shadow in their work is only $1/t$ of that of the original secret image, and so the storage space and transmission time are kept low. To reduce the size of each shadow further, Tso[9] elegantly quantized the secret image and then shared it. Lin and Tsai[7] also transformed the secret image into the frequency domain and then shared the first discrete cosine transform (DCT) coefficient, which was used as a seed in a random number generator to yield a sequence of numbers that were then used to rearrange the values of the second to tenth DCT coefficients in each transformed block. In almost all sharing methods, since each generated shadow looks like noise, hiding methods[10–16] may be employed to hide each noise-like shadow in a cover image.

The missing-allowable feature makes image sharing methods useful to the distributed storage of a secret image. Specifically, the n shadows of an image can be stored in n places. Later, to reconstruct the image, the n shadows are grabbed over n distinct channels. Some of the n communication channels or the n storage disks may be out of service temporarily or deliberately if the owner of a shadow refuses to cooperate, but neither case will affect the reconstruction, as long as the number of missing shadows is not more than $n-t$. The potential problem of losing an image forever is thus erased using image sharing. Additionally, collecting fewer than t shadows yields nothing but noise, and this feature increases security.

Conventional image sharing methods reveal either the entire secret image (when any t of the n shadows are collected) or nothing (when fewer than t shadows are collected). This all-or-nothing property is useful when the image being shared is top secret. However, not all images in daily life are top secret. In many circumstances, the shared image might be sensitive in some way and yet not top secret. Restated, although an image must not be viewed by only a minority of the participants, the reconstruction of the image can still involve certain quality levels, such as low quality, middle quality, and high quality, based on whether the number of collected shadows reaches the corresponding thresholds. Such sharing is called progressive sharing: the sensitive image is reconstructed with improving quality, as determined by the number of the collected shadows in the decoding meeting.

Progressive sharing has a range of applications. Consider, for example, image searching in an antiterrorism intelligence office or a witness-protection program, when an authorized officer searches for a sensitive image from a missing-allowable database system with n distributed storage places. If the shadows of each image have been formed earlier by a traditional all-or-nothing sharing scheme, a user must wait for the entire image to be downloaded (by collecting t out of the n shadows) and then check whether the reconstructed image is useful. In contrast, using shadows in a progressive manner can reduce searching time: in the earlier stage of the reconstruction, people can obtain a rough version of the image by collecting a smaller number (t_1) of

Paper 09029RR received Mar. 9, 2009; revised manuscript received Dec. 1, 2009; accepted for publication Dec. 4, 2009; published online Feb. 9, 2010.

1017-9909/2010/19(1)/013003/12/$25.00 © 2010 SPIE and IS&T.

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

shadows ($t_1 < t \leq n$); they can then abort further transmission as soon as the rough version indicates that the candidate image is absolutely not the sought image. Meanwhile, fewer than t_1 shadows reveal nothing but noise, providing a certain degree of protection of a sensitive image.

Another example involves the increasing use of mobile devices or computers to browse the web. Providing progressive versions of an image allows each authorized customer or team member to have more choices. Moreover, the number of downloadable shadows, which control the quality of the reconstructed versions, can be determined by the level of the paid membership (or authorized rank) of the downloader. In particular, if the image is too offensive or violent or allowed to be inspected only by a particular police team or intelligence squad, then controlling the number of shadows in a progressive manner can yield flexible design benefits or facilitate management of the system (as members of a single team but with different ranks are authorized to download different numbers of the n created shadows). An example of the third application has been presented elsewhere:[17] assume that the leader of a research team wants to prohibit any employee from selling high-quality sensitive pictures or blueprints on the black market, but the leader still wants the employees to cooperate every day—for example, to improve a design in blueprints, preparations for surgery, or a body-guarding program, which may be directed at people shown in the images. The leader keeps some of the n generated shadows, and each of the employees has one of the remaining shadows. If the employees want to take a closer look, they have to ask the leader for permission. The leader can lend them one or more shadows to increase the clarity of the image. If an employee takes the shadows to an enemy, the remaining employees can still seek permission from the leader when they want to view the images with great clarity.

Recently, various progressive image sharing schemes have been proposed.[17–20] However, in Fang's scheme,[18] the size of each shadow was expanded to four times larger than the input image. To avoid expansion, people may use approaches[17,19,20] that were based on the sharing scheme of Thien and Lin.[4] Chen and Lin[19] adopted a bit-plane scanning procedure to rearrange the input image pixels; the rearranged data were then shared. Wang and Shyu[20] elegantly decomposed the input image using spatial and depth information simultaneously and then shared the decomposed image. Hung *et al.*[17] shared the quantized DCT coefficients of the input image. Although the shadows in their work were much smaller than those in the preceding three works,[18–20] the reconstruction of their image was not lossless when all n shadows were collected.

This work offers a new design with all of the advantages of lossless reconstruction (when most of the shadows are collected), compact size, and progressive sharing. All n products (or n shadows) of the input image are compact and so can be hidden in stego media easily without excessively affecting the image quality of the cover media. These compact shadows are equally significant, meaning that the reconstructed version of the image depends only on the number of collected shadows. (Any of the shadows could be missing, so the sender or the receiver need not worry about which shadows are sent or collected first. This increases the probability of success of the decoding meeting.) The proposed scheme is easier than the progressive image sharing schemes[17–20] to apply to scalable Moving Picture Experts Group (MPEG) video transmission,[21,22] and the shadows herein can resist differential attack.[23–25]

The rest of this work is organized as follows: Section 2 reviews related works. Section 3 presents the proposed scheme. Section 4 presents experimental results and makes comparisons with other methods. Section 5 discusses security. Last, Sec. 6 summarizes the contributions of this work.

2 Related Works

2.1 *JPEG*

Joint Photographic Exports Group (JPEG)[26] is an international image compression standard that is used commonly on the Internet. A given image is divided into several blocks of 8×8 pixels each. The 8×8 pixels of each block are transformed using DCT, and the transformed 8×8 coefficients are quantized using a quantization table. The quantized 8×8 coefficients are then scanned in zigzag order for entropy coding. After all of the blocks have been sequentially processed, the JPEG code is generated. A parameter called the quality factor QF (between 0 and 100) controls the quality of the JPEG decompressed image. A higher QF corresponds to higher quality of the JPEG decompressed image (and a larger created JPEG file).

2.2 *Thien and Lin's Image Sharing Method*

Thien and Lin[4] propose a (t,n) threshold method for splitting a grayscale secret image into n shadows. First, all of the gray values between 251 and 255 in the secret image must be truncated to 250 because the arithmetic operations in Eq. (1) are modulo 251. They then use a key to permute the pixels of the secret image; the permuted image is then partitioned into several sectors of t pixels each. For each not-yet-processed sector, define a polynomial:

$$f(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_{t-1} z^{t-1} \pmod{251}, \tag{1}$$

where a_0 to a_{t-1} are the t pixel values. The n values $f(1)$, $f(2),\ldots,$ and $f(n)$ are calculated and then attached to the n shadows. After all sectors have been processed, the n shadows are created. Since every t pixels in the secret image contribute a single pixel to each of the n generated shadows, each shadow size is $1/t$ of the secret image size.

In collecting at least t shadows, Thien and Lin take the first not-yet-used pixel from each of the t shadows and use these t pixel values $f(z_1)$, $f(z_2),\ldots,$ and $f(z_t)$ to evaluate the t coefficients a_0 to a_{t-1} in Eq. (1) for the first sector by reconstructing the $(t-1)$–deg polynomial $f(z)$ as

$$\begin{aligned} f(z) = {} & f(z_1)\frac{(z-z_2)(z-z_3)\ldots(z-z_t)}{(z_1-z_2)(z_1-z_3)\ldots(z_1-z_t)} \\ & + f(z_2)\frac{(z-z_1)(z-z_3)\ldots(z-z_t)}{(z_2-z_1)(z_2-z_3)\ldots(z_2-z_t)} + \ldots \\ & + f(z_t)\frac{(z-z_1)(z-z_2)\ldots(z-z_{t-1})}{(z_t-z_1)(z_t-z_2)\ldots(z_t-z_{t-1})} \pmod{251}. \end{aligned} \tag{2}$$

By processing all pixels of the obtained t shadows in order, they obtain the permuted image, which is then depermuted to reveal the secret image.

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

An example is presented in the following. To divide $t=2$ pixel values 100 and 200 into $n=3$ shadows, Eq. (1) is used to compute the three shadows: $f(1)=(100+200\times 1)\bmod 251=49$; $f(2)=(100+200\times 2)\bmod 251=249$; and $f(3)=(100+200\times 3)\bmod 251=198$. Later, if two shadows $f(1)=49$ and $f(3)=198$ are received, Eq. (2) is used to reconstruct the polynomial as

$$\begin{aligned} f(z) &\equiv f(1)\times(z-3)/(1-3)+f(3)\times(z-1)/(3-1) \\ &\equiv 49\times(z+248)/249+198\times(z+250)/2 \\ &\equiv 49\times(z+248)\times 125+198\times(z+250)\times 126 \\ &\equiv 100+200z \pmod{251}. \end{aligned}$$

The original pixel values, 100 and 200, are thus obtained.

2.3 *Chen et al.'s JPEG Data Hiding Method*

Chen *et al.*[27] propose a reversible JPEG steganography method for hiding secret data in a JPEG compression code. First, the JPEG compression code of the cover image is entropy decoded to obtain all quantized 8×8 blocks $\{F\}$ and an 8×8 quantization table Q. Next, to embed the secret in the quantized coefficient $F(i,j)$ of each block F, the scaling factor $Q(i,j)$ of the original quantization table Q is modified to a smaller factor $Q'(i,j)$ that satisfied $\lfloor Q(i,j)/Q'(i,j)\rfloor \geqslant 2$, enabling the two nearby noninteger points $F(i,j)-0.5$ and $F(i,j)+0.5$ of $F(i,j)$ to be mapped to two integer points $M(i,j)$ and $N(i,j)$, respectively, by

$$M(i,j)=\lfloor[F(i,j)-0.5]\times Q(i,j)/Q'(i,j)\rfloor, \tag{3}$$

$$N(i,j)=\lfloor[F(i,j)+0.5]\times Q(i,j)/Q'(i,j)\rfloor, \tag{4}$$

where $0\leqslant i,j<8$. After the values of $M(i,j)$ and $N(i,j)$ are determined, based on the to-be-hidden secret digit, an integer point in the half-open interval $[M(i,j),N(i,j))$ is indentified as the stego quantized coefficient $F'(i,j)$. For example, let $F'(i,j)$ be $M(i,j)$ if the to-be-hidden secret is 0; let $F'(i,j)$ be $M(i,j)+1$ if the to-be-hidden secret is 1; let $F'(i,j)$ be $M(i,j)+2$ if the to-be-hidden secret is 2; and so on. Last, each embedded quantized block F' is entropy encoded, and the modified quantization table Q' is included in the JPEG file header to yield the JPEG stego code. The original quantization table Q and generated JPEG stego code are transmitted to the receiver.

When the original quantization table Q and the generated JPEG stego code are received, the secret can be completely extracted and the original JPEG code can also be reconstructed. First, the JPEG stego code is entropy decoded to obtain all stego quantized blocks $\{F'\}$ and the 8×8 modified quantization table Q'. Next, based on the stego quantized coefficient $F'(i,j)$ of each block F', the $Q'(i,j)$–to–$Q(i,j)$ inverse scaling is employed to reconstruct the original quantized coefficient $F(i,j)$ using

$$F(i,j)=Round[F'(i,j)\times Q'(i,j)/Q(i,j)], \tag{5}$$

where $0\leqslant i,j<8$. After the value of $F(i,j)$ is reconstructed, use Eq. (3) to compute $M(i,j)$, which is then used to extract the decimal equivalent of the hidden data $Z(i,j)$ as

$$Z(i,j)=F'(i,j)-M(i,j). \tag{6}$$

An example of the embedding and extraction processes is as follows. Assume that the quantized coefficient is $F(0,0)=12$, and $Q(0,0)=16$ is the original quantizer that was used by JPEG. Let $Q'(0,0)=4$ be the modified quantizer. According to Eqs. (3) and (4), the values $M(0,0)$ and $N(0,0)$ are computed using

$$M(0,0)=\lfloor(12-0.5)\times 16/4\rfloor=46,$$

$$N(0,0)=\lceil(12+0.5)\times 16/4\rceil=50.$$

Accordingly, in the half-open interval $[M(0,0)=46, N(0,0)=50)$, the value $F'(0,0)=46$ is the stego quantized coefficient after 0 is embedded. [$F'(0,0)=47$ after 1 is embedded; $F'(0,0)=48$ after 2 is embedded; $F'(0,0)=49$ after 3 is embedded.] Later, assume that the obtained stego quantized coefficient is $F'(0,0)=47$. To reconstruct the quantized coefficient $F(0,0)$ from $F'(0,0)=47$, use Eq. (5) to evaluate

$$F(0,0)=Round(47\times 4/16)=12.$$

(Notably, 12 is the original quantized coefficient. Hence, the JPEG hiding method[27] is called reversible.) Then use Eq. (3) to compute

$$M(0,0)=\lfloor(12-0.5)\times 16/4\rfloor=46.$$

Last, from Eq. (6), the decimal equivalent of the hidden data is extracted as

$$Z(0,0)=F'(0,0)-M(0,0)=47-46=1.$$

2.4 *Galois Field*

A Galois field (GF) is a finite field of p^k elements with addition $(+)$ and multiplication $(\times)$ operations that satisfy commutative, associative, and distributive laws where p is a prime number and k is a positive integer. In general, the arithmetic over GF(p) is the same as modulo p, and thus Thien and Lin[4] use a Galois field with $p^k=p^1=p=251$ elements. The proposed method employs a Galois field with 2^k elements, and the arithmetic over GF(2^k) is based on the representation of each element in GF(2^k). An element in GF(2^k) is generally represented using a polynomial-basis representation, as a binary polynomial of degree less than k. The k-tuple of coefficients of the binary polynomial corresponds to the binary representation of an integer between 0 and 2^k-1.

Let $A=(a_{k-1}\ldots a_1a_0)_2$ and $B=(b_{k-1}\ldots b_1b_0)_2$ be two k-bits binary elements in GF(2^k). In the polynomial-basis representation, A and B are $A(X)=a_{k-1}X^{k-1}+\ldots+a_1X+a_0$ and $B(X)=b_{k-1}X^{k-1}+\ldots+b_1X+b_0$, respectively. Define the addition of A and B as

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

$$A+B=\sum_{i=0}^{k-1} a_i \oplus b_i, \tag{7}$$

where $\oplus$ is the exclusive-or (XOR) operator. For the subtraction, because each element in GF(2^k) is its own additive inverse, the subtraction of B from A is thus defined as

$$A-B=A+(-B)=A+B=\sum_{i=0}^{k-1} a_i \oplus b_i. \tag{8}$$

The multiplication and division involve a primitive polynomial $H(X)$, where $H(X)$ is a k-deg irreducible polynomial (i.e., it has no nontrivial factors). To multiply A by B, the remainder $C(X)=c_{k-1}X^{k-1}+\ldots+c_1X+c_0$ is computed in a long division, defined by

$$C(X)=A(X)B(X) \bmod H(X), \tag{9}$$

where the operations of the binary coefficients in the polynomial multiplication and in the mod $H(X)$ operations are all modulo 2 such that all the resulting coefficients are still in {0, 1} and therefore binary. After the binary polynomial $C(X)=c_{k-1}X^{k-1}+\ldots+c_1X+c_0$ has been determined using Eq. (9), the multiplication of the two k-bits binary elements A and B is defined as

$$A \times B=C=(c_{k-1}\ldots c_1c_0)_2. \tag{10}$$

Last, to divide A by B, Eqs. (9) and (10) are used to multiply A by B^{-1}, where B^{-1} is the unique element E in GF(2^k) such that $[B(X)E(X)] \bmod H(X)=1$.

3 Proposed Method

3.1 *Encoding*

The quality factor $QF \in \{0,1\ldots,100\}$ in JPEG is used to control image quality. To reconstruct an important image s with various quality levels based on the number of received JPEG stego codes, a low-quality factor $QF_L \in \{0,1,\ldots,5\}$, a medium-quality factor $QF_M \in \{10,11,\ldots,25\}$, and a high-quality factor $QF_H \in \{55,56,\ldots,85\}$ can be used to generate, respectively, a low-quality JPEG code c_1, a medium-quality JPEG code c_2, and a high-quality JPEG code c_3. The quality levels of the JPEG images r_1, r_2, and r_3 decompressed from codes c_1, c_2, and c_3, respectively, are around 18 to 28 dB, 30 to 34 dB, and 36 to 40 dB. To reconstruct the image s error-freely, a difference image d is created by subtracting from the image s the high-quality JPEG image r_3 that is decompressed from the high-quality JPEG code c_3. The difference image d is compressed using Huffman coding to generate the Huffman code c_4. Last, based on the five user-defined integer parameters $1<t_1<t_2<t_3<t_4\leq n$, the generated codes c_1, c_2, c_3, and c_4 are shared according to Eq. (11). As shown in Fig. 1, the proposed $[(t_1,t_2,t_3,t_4),n]$ threshold scheme comprises four phases: (1) codes generation, (2) sharing, (3) shares combining, and (4) data hiding. The encoding algorithm is given here:

Input: An important image s; five positive integer parameters t_1, t_2, t_3, t_4, and n, where $1<t_1<t_2<t_3<t_4\leq n$; and n cover images.

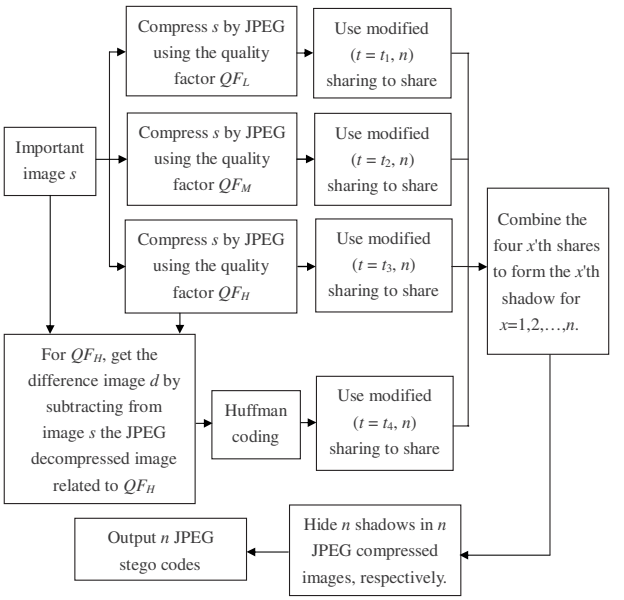


Fig. 1 Flowchart for encoding.

Output: The n JPEG stego codes

Step 1a: Compress the important image s using JPEG three times (with quality factors of QF_L, QF_M, and QF_H, respectively), yielding a low-quality JPEG code c_1 of s, a medium-quality JPEG code c_2 of s, and a high-quality JPEG code c_3 of s.

Step 1b: Compute the difference image d by subtracting from the image s its high-quality JPEG image r_3 that is decompressed from the JPEG code c_3. Compress the difference image d using Huffman coding to generate the Huffman code c_4 of the image d.

Step 2: For each code c_i $(i=1,2,3,4)$, use Eq. (11) to split the code c_i into n shares.

Step 3: For each $x=1,2,\ldots,n$, the x'th shadow is formed by binding together the x'th share of c_1, the x'th share of c_2, the x'th share of c_3, and the x'th share of c_4.

Step 4: Use the JPEG hiding method[27] to hide the n shadows in the n JPEG codes of the n cover images, respectively. (This generates n desired JPEG stego codes, and the cooperation of several stego codes can view the important image s at certain quality levels.)

Note: In step 2, the code c_i is divided into sectors of t_i bytes each. Each byte is treated as a number between 0 and 255. Our share-generating polynomial is

$$g(x)=b_0+b_1x+b_2x^2+\cdots+b_{t_i-1}x^{t_i-1} \text{ [over GF(256)]}, \tag{11}$$

where b_0 to b_{t_i-1} are the t_i numbers of the sector, and the computations in Eq. (11) are over GF(256). Then, $g(1)$ to $g(n)$ are sequentially assigned to n shares. Since each sector of t_i bytes contributes only a single byte [a value in the range 0 to 255 and determined by Eq. (11)] to each generated share, the size of each share of the code c_i is t_i times smaller than that of the code c_i. In step 3, the size of each

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

shadow is $\Sigma_{i=1}^{4}|c_i|/t_i$, where $|c_i|$ denotes the length of the code c_i. In step 4, to avoid attracting the attention of attackers, the n shadows are hidden in n JPEG codes.

3.2 *Decoding*

When any t_1 of the n JPEG stego codes are received, the t_1 shadows can be extracted from the t_1 JPEG stego codes by inverse hiding. For each $x=1,2,\ldots,t_1$, the x'th shadow is partitioned to yield the x'th share of each code $c_i(1\leqslant i\leqslant 4)$. The t_1 shares of the code c_1 are then used to reconstruct the low-quality JPEG code c_1 in inverse sharing, which can be done either by the matrix multiplication method (detailed in the following) or by the Lagrange interpolation method used in Thien and Lin[4] (The two methods are with similar computation loads.) The reconstructed JPEG code c_1 is decompressed to yield the low-quality JPEG image r_1, which is an approximate version of the original important image s. When t_2 (or t_3) JPEG stego codes are available, the reconstruction process is similar to that described earlier, and the reconstructed image r_2 (or r_3) will be of medium (or high) quality.

Last, if at least t_4 JPEG stego codes are received, the t_4 shadows can also be extracted from the t_4 JPEG stego codes by inverse hiding. Then, for each $x=1,2,\ldots,t_4$, the x'th shadow is divided to generate the x'th share of each code $c_i(1\leqslant i\leqslant 4)$. The obtained t_4 shares of the code c_4 are used in inverse sharing to reconstruct the Huffman code c_4 by matrix multiplication or Lagrange interpolation, and the reconstructed code c_4 is then decompressed to generate the difference image d. Adding the image d to the image r_3 yields the error-free image s.

The use of matrix multiplication to reconstruct the code c_i from any t_i out of the n shares $(1\leqslant i\leqslant 4)$ is described in the following. Recall that the sharing process uses Eq. (11) to generate the n pixel values $g(1)$ to $g(n)$. These n values can also be computed using

$$[b_0\ b_1\ \ldots\ b_{t_i-1}]\begin{bmatrix}1 & 1 & \ldots & 1\\ 1 & 2 & \ldots & n\\ \ldots & \ldots & \ldots & \ldots\\ 1 & 2^{t_i-1} & \ldots & n^{t_i-1}\end{bmatrix} = [g(1)\ g(2)\ \ldots\ g(n)]\ [\text{over GF}(256)]. \tag{12}$$

Accordingly, in the inverse-sharing process, when any t_i of the n shares are obtained (and without loss of generality, suppose that the first t_i shares are obtained), the first t_i not-yet-used pixels $g(1)$, $g(2)$,..., and $g(t_i)$ are taken from the t_i shares, and the t_i coefficients b_0 to b_{t_i-1} of the first sector are reconstructed using

$$[g(1)\ g(2)\ \ldots\ g(t_i)]\begin{bmatrix}1 & 1 & \ldots & 1\\ 1 & 2 & \ldots & t_i\\ \ldots & \ldots & \ldots & \ldots\\ 1 & 2^{t_i-1} & \ldots & t_i^{t_i-1}\end{bmatrix}^{-1} = [b_0\ b_1\ \ldots\ b_{t_i-1}]\ [\text{over GF}(256)]. \tag{13}$$

The arithmetic computations in Eq. (13) are still over GF(256). Code c_i is reconstructed by sequentially processing all pixels of the obtained t_i shares.

3.3 *Example of Sharing and Inverse-Sharing Processes Based on GF(256)*

An example of the sharing and inverse-sharing processes based on GF(2^8=256) and $H(X)=X^8+X^4+X^3+X+1$ is presented in the following. To partition $t_i=2$ numbers {100 and 200} of 8 bits each into $n=3$ shares, Eq. (11) is used to compute: $g(1)=100+200\times 1$[over GF(256)]$=172$; $g(2)=100+200\times 2$[over GF(256)]$=239$; and $g(3)=100+200\times 3$[over GF(256)]$=39$. In obtaining the two shares $g(1)=172$ and $g(3)=39$, the inverse matrix of $\left[\begin{smallmatrix}1 & 1\\ 1 & 3\end{smallmatrix}\right]$ is computed over GF(256) as

$$\begin{bmatrix}1 & 1\\ 1 & 3\end{bmatrix}^{-1} = \begin{bmatrix}140 & 141\\ 141 & 141\end{bmatrix}\ [\text{over GF}(256)].$$

Then Eq. (13) yields

$$[b_0\ b_1] = [172\ 39]\begin{bmatrix}140 & 141\\ 141 & 141\end{bmatrix}\ [\text{over GF}(256)],$$

where $b_0=172\times 140+39\times 141$[over GF(256)]$=100$, and $b_1=172\times 141+39\times 141$[over GF(256)]$=200$. Notably, the two numbers 100 and 200 can also be revealed by the Lagrange interpolation approach, as

$$\begin{aligned} g(z) &\equiv g(1)\times(z-3)/(1-3)+g(3)\times(z-1)/(3-1)\\ &\equiv 172\times(z+3)/(1+3)+39\times(z+1)/(3+1)\\ &\equiv 172\times(z+3)\times 141+39\times(z+1)\times 141\\ &\equiv 100+200z\ [\text{over GF}(256)]. \end{aligned}$$

4 Experiments and Comparisons

4.1 *Experimental Results*

The inequalities $(t_1=3)<(t_2=4)<(t_3=5)<(t_4=6)$ and the irreducible polynomial $H(X)=X^8+X^4+X^3+X+1$ are used to generate $n=6$ shadows of the important image. The JPEG source code that is used in the experiments is taken from the fourth public release of the Independent JPEG Group's free software.[28] The quality of an image is measured by the peak signal-to-noise ratio (PSNR).

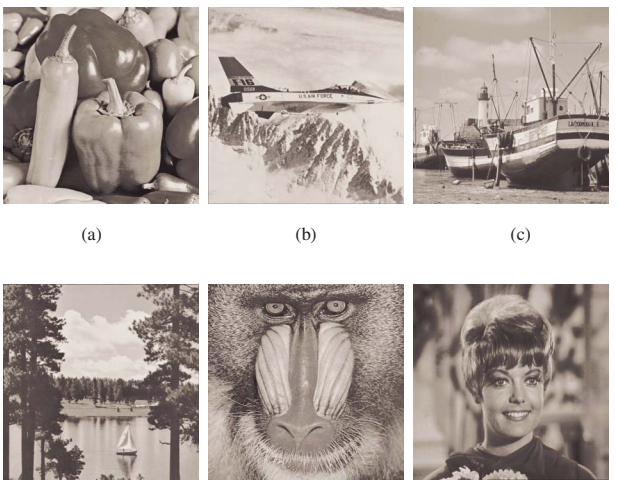
In the first experiment, the 512×512 grayscale important image s Lena, displayed in Fig. 2, is encoded by JPEG with three quality factors $QF_L=5$, $QF_M=25$, and $QF_H=85$. The four codes c_1, c_2, c_3, and c_4 have lengths 5750, 13,787, 45,972, and 115,458 bytes, respectively. The six cover images Peppers, Jet, Boat, Lake, Baboon, and Zelda, shown in Fig. 3, are all encoded using JPEG with $QF=75$ to hide the six shadows and thus obtain the six JPEG stego codes. Figure 4 displays the $n=6$ images that are decompressed from our six JPEG stego codes without any extraction of the hidden shadows, and the PSNRs of them are 37.42, 37.41, 36.40, 34.39, 32.73, and 38.67 dB, respectively. When different numbers of JPEG stego codes are received, the reconstructed versions r_1, r_2, r_3, and r_4 of Lena are as plotted in Fig. 5, and the respective PSNRs are 27.32, 33.67, 39.35, and ∞ dB.

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

Fig. 2 Original 512×512 important image Lena used in the first experiment.

In the second experiment, the important image is the 512×512 grayscale image Tiffany. The six shadows are generated and then remain hidden in the six JPEG codes that are generated in the first experiment. The PSNRs of the decompressed images from these six JPEG stego codes are 37.75, 37.70, 36.65, 34.61, 32.97, and 38.96 dB, respectively. (These values are a little better than the 37.42, 37.41, 36.40, 34.39, 32.73, and 38.67 dB values obtained in the first experiment.) For Tiffany, the PSNRs of the versions reconstructed using any 3, 4, and 5 JPEG stego codes are 28.37, 34.12, and 39.79 dB, respectively. (These values are a little better than those, 27.32, 33.67, and 39.35 dB, for Lena.) When six JPEG stego codes are obtained, the reconstructed Tiffany is identical to the original Tiffany.

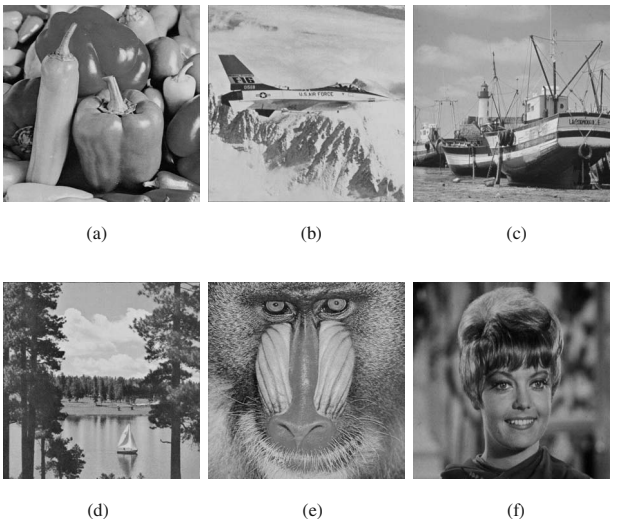
Last, Table 1 shows the bit rates [bits per pixel (bpp)] of the JPEG-Q75 codes (created using JPEG with $QF=75$) before and after hiding our shadows. The bit rate will increase significantly after hiding a large-size secret. However, the bit rate of the JPEG stego code herein still falls in the reasonable range of JPEG, i.e., the bit rate of the stego code herein is smaller than that of the JPEG code generated using $QF=95$, as shown in Table 1. This alleviates the problem of code length. If the shadows of other image-sharing methods[4,8,9,18–20] had been used, then the problem would have been worse. [The reason for using $QF=95$ as the upper bound to examine the bit rate of the JPEG stego codes is that, as stated in Kim *et al.*'s work,[29] the general quality factors (QFs) that are used in digital cameras are between 90 and 95.]



(a) (b) (c)

(d) (e) (f)

Fig. 3 Six 512×512 cover images Peppers, Jet, Boat, Lake, Baboon, and Zelda, which are utilized to cover the important image Lena.



(a) (b) (c)

(d) (e) (f)

Fig. 4 Six 512×512 images decompressed from six output JPEG stego codes. (The six decompressions are done independently, without any extraction of the hidden important image Lena.) PSNRs of (a) to (f) are 37.42, 37.41, 36.40, 34.39, 32.73, and 38.67 dB, respectively.

(a) (b)

(c) (d)

Fig. 5 Four versions of important image Lena reconstructed from various numbers of received JPEG stego codes: (a) from any three JPEG stego codes (PSNR=27.32 dB); (b) from any four JPEG stego codes (PSNR=33.67 dB); (c) from any five JPEG stego codes (PSNR=39.35 dB); and (d) from the six JPEG stego codes (and identical to the original important image Lena).

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

Table 1 Bit rates (bpp) of the JPEG codes of cover images.

Cover image	bpp of the (no-hiding) JPEG-Q75 code	bpp after hiding a shadow of size 14,559 bytes in the JPEG-Q75 code	bpp after hiding a shadow of size 33,802 bytes in the JPEG-Q75 code	*bpp of the (no-hiding) JPEG-Q95 code*
Peppers	0.94	1.62	2.33	*2.52*
Jet	0.94	1.64	2.37	*2.47*
Boat	1.02	1.73	2.46	*2.74*
Lake	1.25	1.95	2.64	*3.44*
Baboon	1.89	2.63	3.25	*4.09*
Zelda	0.81	1.48	2.21	*2.54*

4.2 *Comparisons*

Table 2 compares other sharing schemes[4,8,9,17–20] with ours in terms of shadow-size expansion, progressive ability, and lossless reconstruction ability. Each shadow in two of the related works[8,18] is four times larger than the original important image, indicating that size expansions had occurred. In contrast, each shadow in all of the associated works[4,9,17,19,20] and ours is smaller than the original important image. Although Thien and Lin,[4] Tso,[9] and Hung *et al.*[17] all shared the image without size expansion, Thien and Lin[4] and Tso[9] could not reconstruct the image progressively, whereas Hung *et al.*[17] could not reconstruct the image in an error-free manner. Only Chen and Lin,[19] Wang and Shyu,[20] and ourselves have achieved reconstruction with all three desired characteristics. Among these three methods, as presented in Table 3, each shadow size herein (12.89% of 512×512 bytes) is smaller than those in Chen and Lin's method[19] (22.22%) and Wang and Shyu's[20] (50%). Hence, the transmission time in this work is less, and the survival rate in an unfriendly environment, in which the network connection time is unstable among the n channels used to store the n shadows, is increased. Equivalently, in this work, the storage space in a distributed storage system is most reduced. The smaller size of the shadows also facilitates the hiding of shadows in stego media.

The construction of Table 3, which compares the shadow sizes among nonexpanded schemes, is explained in the following. All data (except those obtained herein) are directly taken from the aforementioned works.[4,9,17,19,20] For fairness of comparison, the shadow sizes in Table 3 are all measured *before hiding*: all are shadow sizes, and none is a stego media size. This action eliminates the size-altering effects of particular post-processing (hiding) approaches. Assume that the given important image is the 512×512 grayscale image Lena, and the (largest) threshold value is set to 6 for all schemes, except that Hung *et al.*'s scheme[17] uses 5 as the largest threshold value because their scheme did not provide a version with a threshold value being 6. For each $x=1,2,\ldots,n$, the four x'th shares are combined to form the x'th shadow, and thus each shadow herein has size $\Sigma_{i=1}^{4}|c_i|/t_i=(5750/3)+(13,787/4)+(45,972/5)+(115,458/6)=33,802$ bytes (which is 12.89% of the size of the 512×512 grayscale image Lena).

According to Table 3, the shadow sizes in the proposed method and Hung *et al.*'s[17] are more economic than those in the related works.[4,9,19,20] However, Hung *et al.*'s method[17] is not lossless when all shadows are collected. In fact, if the original important image can be satisfactorily reconstructed with some loss, then our step 1b can be omitted, such that no Huffman code c_4 is generated. Then, each of our shadows can be reduced to $\Sigma_{i=1}^{3}|c_i|/t_i=(5750/3)+(13,787/4)+(45,972/5)=14,559$ bytes (which is 5.55% of the size of the 512×512 grayscale important image Lena). Restated, the size of each shadow in this lossy version is about half of that in Hung *et al.*'s scheme.[17] Moreover, in this lossy version, the total shadow size herein is $14,559\times6=87,354$ bytes, which is still smaller than $30,723\times5=153,615$ obtained by Hung *et al.*[17] When the five shadows are collected, the 39.35-dB Lena [identical to that in Fig. 5(c)] is reconstructed, better than the 37.04-dB Lena revealed by Hung *et al.*[17] Notably, Tso[9] reconstructed

Table 2 Comparisons among image sharing methods (Refs. 4, 8, 9, and 17–20).

Scheme	Nonexpanded size of each shadow	Progressive ability	Lossless reconstruction using *all* shadows
Lin and Lin[8]			✓
Fang[18]		✓	✓
Thien and Lin[4]	✓		✓
Tso[9]	✓		✓
Chen and Lin[19]	✓	✓	✓
Wang and Shyu[20]	✓	✓	✓
Hung *et al.*[17]	✓	✓	
Our scheme	✓	✓	✓

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

Table 3 Comparison of shadow sizes in nonexpanded methods.[4,9,17,19,20] The (largest) threshold in all works is set to 6. But Hung *et al.*'s method[17] uses 5 as the largest threshold value because their method did not provide a version with a threshold value of 6.

Scheme	Each shadow size (bytes), and the quality of the reconstructed image	Each shadow size over 512×512 (given image size)
Thien and Lin[4] (nonprogressive)	43,691; lossless	16.67%
Tso[9] (nonprogressive)	43,691 if lossless (or 27,307 in lossy version with 45.1-dB image quality)	16.67% (10.42% in lossy version)
Chen and Lin[19]	58,254; lossless	22.22%
Wang and Shyu[20]	131,072; lossless	50%
Hung *et al.*[17]	30,720 in lossy version with 37.04-dB image quality	11.72% in lossy version
Our scheme	33,802 if lossless (or 14,559 in lossy version with 39.35-dB image quality)	12.89% (5.55% in lossy version)

the image with very high PSNR (45.1 dB) when collecting shadows of size 27,307 bytes. Although only the 39.35-dB image can be reconstructed herein with the collection of shadows of size 14,559 bytes, the image can be reconstructed without any loss when shadows of size 33,802 bytes (which are smaller than those of size 43,691 bytes obtained by Tso's nonprogressive lossless approach) are collected.

Last, the size of each shadow in the proposed $[(t_1,t_2,t_3,t_4),n]$ threshold scheme is $\Sigma_{i=1}^{4}|c_i|/t_i$. Therefore, it is suggested that the readers set the largest threshold t_4 to n, in order to save space. However, if the readers want to have more freedom, they may use their own choice of a threshold $t_4<n$, at the price of wasting space for the shadows. When $t_4<n$, a simulation is done in the following. Assume that n, the number of cover images, is at least 6. In general, the smallest threshold t_1 cannot be 1 because the purpose of sharing is that no participant alone can be trusted. Hence, $\{1<t_1=2<t_2=3<t_3=4<t_4=5<n=6\}$ is used to generate $n=6$ shadows, the size of which is then compared to those in the image-sharing schemes[4,9,17,19,20] when the (largest) threshold value is set to 5 for all these works. The comparison results are given in Table 4. It is observed that each shadow herein is still smaller than those

Table 4 Same as Table 3 except that the (largest) threshold value in all works (Refs. 4, 9, 17, 19, and 20) is set to 5 instead of 6.

Scheme	Each shadow size (bytes), and the quality of the reconstructed image	Each shadow size over 512×512 (given image size)
Thien and Lin[4] (nonprogressive)	52,429; lossless	20%
Tso[9] (nonprogressive)	52,429 if lossless (or 32,768 in lossy version with 45.1-dB image quality)	20% (12.5% in lossy version)
Chen and Lin[19]	74,898; lossless	28.57%
Wang and Shyu[20]	131,072; lossless	50%
Hung *et al.*[17]	30,720 in lossy version with 37.04-dB image quality	11.72% in lossy version
Our scheme	42,056 if lossless (or 18,964 in lossy version with 39.35-dB image quality)	16.04% (7.23% in lossy version)

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

in the related works,[4,9,19,20] and each shadow in our lossy approach is also smaller than that in Hung *et al.*'s lossy approach.[17]

5 Security Discussion

The code c_i $(1 \leqslant i \leqslant 4)$ cannot easily be revealed if fewer than t_i shadows are intercepted. To determine the coefficients b_0 to b_{t_i-1} in Eq. (11), t_i equations are required. If only t_i-1 equations are available [and without loss of generality, suppose that $g(1)$, $g(2),\ldots$, and $g(t_i-1)$ are intercepted], then the following t_i-1 equations can be constructed:

$$\begin{cases} g(1) = (b_0 + b_1 + \ldots + b_{t_i-1}) \text{ [over GF(256)]}, \\ g(2) = (b_0 + 2b_1 \ldots + 2^{t_i-1} b_{t_i-1}) \text{ [over GF(256)]}, \\ \ldots \\ g(t_i-1) = [b_0 + (t_i-1)b_i \\ \qquad + \ldots + (t_i-1)^{t_i-1} b_{t_i-1}] \text{ [over GF(256)]} \end{cases}.$$

The preceding t_i-1 equations are solved for the t_i unknown coefficients. The set of possible solutions has 256 members, so the probability of guessing the right solution is $1/256$. Since $|c_i|/t_i$ polynomials exist for the code c_i, the probability of obtaining the right code c_i is $(1/256)^{|c_i|/t_i}$. For example, for a low-quality JPEG code c_i of size 5000 bytes, the number of sectors is 2500 if t_i is 2. The probability of obtaining the correct JPEG code c_1 is only $(1/256)^{2500}=10^{-6020}$.

If the security of the shadows is to be increased, a seed may be used in a random number generator to generate a random sequence for each shadow, and then XOR operations can be applied between the random sequence and the shadow. Each row of the XOR-encrypted shadows of the code c_i can then be circularly shifted by a certain number of bytes. Similar operations are applied to each column. This will transform the shadows to their safer versions. In this process, each seed that is used to generate a random sequence is based on the creation time and shadow index. The seed can then be kept by all n participants or held by the company leader if the leader insists on attending the decoding meeting.

As stated in three works,[23–25] attackers may slightly change the plaintext and then observe the change in the ciphertext. This is so-called differential attack, which the related progressive image-sharing methods[18–20] cannot handle. The attackers may try to find a relationship between the plaintext and its ciphertext. Therefore, to ensure high security, the change in the ciphertext should cover a very large area if change occurs over a small area in the plaintext. To check this, the number of pixels change rate (NPCR) is used to measure the number of pixels that are changed in the ciphertext when only one pixel is changed in the plaintext. To define NPCR, let X and Y be two ciphertexts of size $W \times H$, where the plaintexts of X and Y differ by only one pixel. Let $X(i,j)$ and $Y(i,j)$ be the pixel values at position (i,j) in X and Y, respectively. Define

$$NPCR = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\%, \tag{14}$$

where $D(i,j)$ is defined as

$$D(i,j) = \begin{cases} 0, & X(i,j) = Y(i,j) \\ 1, & X(i,j) \neq Y(i,j) \end{cases}. \tag{15}$$

The unified average changing intensity (UACI) is used to measure the average intensity of the differences between two ciphertexts X and Y. It is defined as

$$UACI = \frac{1}{W \times H}\left[\sum_{i,j} \frac{|X(i,j) - Y(i,j)|}{255}\right] \times 100\%. \tag{16}$$

For random images, the expected values of NPCR and UACI are 99.609375% and 33.46354%, respectively, according to Kwok and Tang's work,[25] which is an image encryption method rather than an image-sharing method. The NPCR values herein are between 99.54% and 99.60% (very close to 99.609375%), indicating that each XOR-enhanced shadow is very sensitive to a change in a single byte in the code c_1; the UACI values are between 33.36% and 33.45% (very close to 33.46354%), suggesting that the change of each XOR-enhanced shadow that is associated with a single-byte change in the code c_1 is very large. Similar observations are made when the code c_1 is replaced by the code c_2, c_3, or c_4. Hence, the enhanced version can resist differential attack. Notably, to achieve this ability to resist differential attack, the design is based on simple XOR operations, unlike other designs.[23–25] Also, this XOR-enhanced version does not increase the shadow size.

Last, since the shadows are hidden using Chen *et al.*'s JPEG hiding method,[27] the security after hiding is discussed in the following. Possible attack due to visual inspection is avoided. As presented at the end of Sec. 4.2, in the $\{1<t_1=2<t_2=3<t_3=4<t_4=5\}$ and $\{1<t_1=3<t_2=4<t_3=5<t_4=6\}$ experiments, each shadow has size $\Sigma_{i=1}^{4}|c_i|/t_i=(5750/2)+(13,787/3)+(45,972/4)+(115,458/5)=42,056$ bytes and $\Sigma_{i=1}^{4}|c_i|/t_i=(5750/3)+(13,787/4)+(45,972/5)+(115,458/6)=33,802$ bytes, respectively. If other sets of $\{1<t_1<t_2<t_3<t_4\}$ are used to generate n shadows, then each still has size smaller than 42,056 bytes. Hence, 42,056 bytes is the largest possible shadow size for all possible combinations of $\{1<t_1<t_2<t_3<t_4\}$. Now, each shadow of size 42,056 bytes can be hidden in a JPEG-Q65 code of a cover image after a JPEG compression with $QF=65$. Figure 6 shows the six images decompressed from the six created JPEG stego codes. (The hidden secret is left untouched when the JPEG decompression is performed.) Visual quality of these decompressed images is acceptable, reducing the probability that the codes get attacked when the attackers use visual inspection to find suspicious images.

Suspicious JPEG code length is avoided. Besides the evidence shown in Table 1, Table 5 lists the bit rates of the JPEG-Q65 codes before and after hiding the largest shadow of size 42,056 bytes. The bit-rates of the JPEG codes corresponding to $QF=95$ are also listed to observe whether the bit rates of our JPEG stego codes are reasonable. From

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

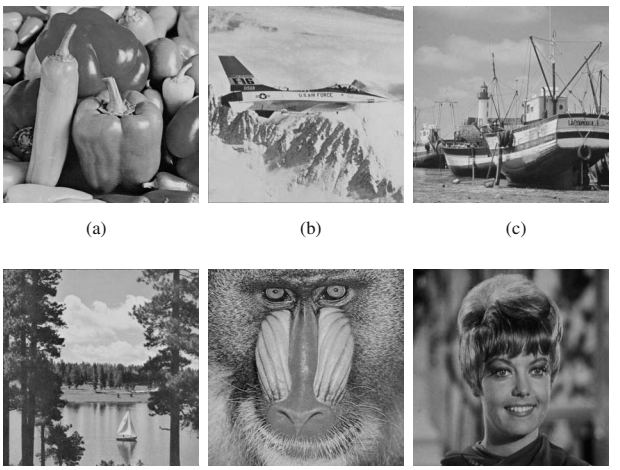


Fig. 6 Same as Fig. 4 except that the to-be-hidden shadows are generated by the proposed $[(t_1=2, t_2=3, t_3=4, t_4=5), n=6]$ threshold scheme. PSNRs of (a) to (f) are 35.73, 35.57, 34.65, 33.14, 30.94, and 35.94 dB, respectively.

Table 5, it is observed that even after hiding the largest shadow of size 42,056 bytes, the bit rate herein is still below that of the plain JPEG-Q95 code. Hence, the attackers will not be suspicious about the length of our stego codes.

The proposed method can pass the Chi-square[30] and StegSpy[31] analyses, which are tools to determine whether a secret is hidden in an image or a JPEG code. For the images Peppers or Jet, Figs. 7(a)–7(d) display the results after Guillermito's Chi-square analysis tool is utilized to examine each pixel of the JPEG images mentioned there. The red curve indicates the probability that pairs of values follow a random distribution, and the green one represents the average value of all least significant bits (LSBs) in one block of pixels. The green curves in Figs. 7(a)–7(d) suggest to the attackers that there is nothing strange in these JPEG images because all four green curves are around $(0+1)/2=0.5$. Also, after a sort of latency, all red curves are flat at zero, indicating that the possibility of the existence of the hidden

Table 5 Same as Table 1 except that the JPEG codes used in hiding are created using $QF=65$.

Cover image	bpp of the (no-hiding) JPEG-Q65 code	bpp after hiding a shadowof size 42,056 bytes in the JPEG-Q65 code	*bpp of the (no-hiding) JPEG-Q95 code*
Peppers	0.76	2.40	*2.52*
Jet	0.77	2.45	*2.47*
Boat	0.83	2.52	*2.74*
Lake	1.02	2.65	*3.44*
Baboon	1.57	3.16	*4.09*
Zelda	0.65	2.30	*2.54*

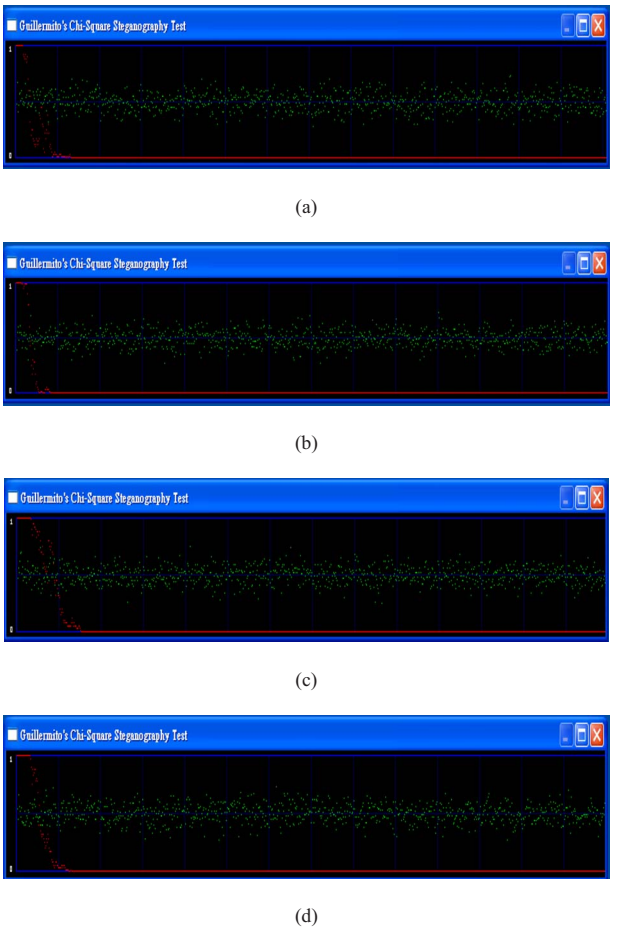


Fig. 7 Results of Chi-square analysis: (a) is for the original JPEG cover image Peppers (no-hiding); (b) is for our JPEG stego image Peppers in Fig. 6(a); (c) is for the original JPEG cover image Jet (no-hiding); and (d) is for our JPEG stego image Jet in Fig. 6(b).

secret is very low. (color online only.) A similar phenomenon also holds when the image Peppers or Jet is replaced by other images Boat, Lake, etc. As for StegSpy analysis, let OriginalPeppers.jpg be the JPEG-Q65 code of the gray-scale image Peppers. A secret (either our shadow or a randomly secret of size 3000 bytes) is then embedded in OriginalPeppers.jpg by the Hiderman[31] and JPegX[32] hiding tools to generate two JPEG stego codes: Stego-of-Hiderman.jpg and Stego-of-JPegX.jpg. Let our six created JPEG stego codes, the decompressed images of which are in Fig. 6, be ourstegoPeppers.jpg, ourstegoJet.jpg, etc. Figure 8 displays the results after the StegSpy analysis tool inspects these nine JPEG codes. The tool found that some data are hidden in Stego-of-Hiderman.jpg and Stego-of-JPegX.jpg, but not in our stego codes, and hence the hidden data will be ignored by the attackers.

6 Summary

This work proposes a $[(t_1,t_2,t_3,t_4),n]$ threshold progressive image-sharing scheme. The contributions of this work are as follows.

- The proposed scheme has progressive ability. (Those in other works[4,9] do not have progressive ability,

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

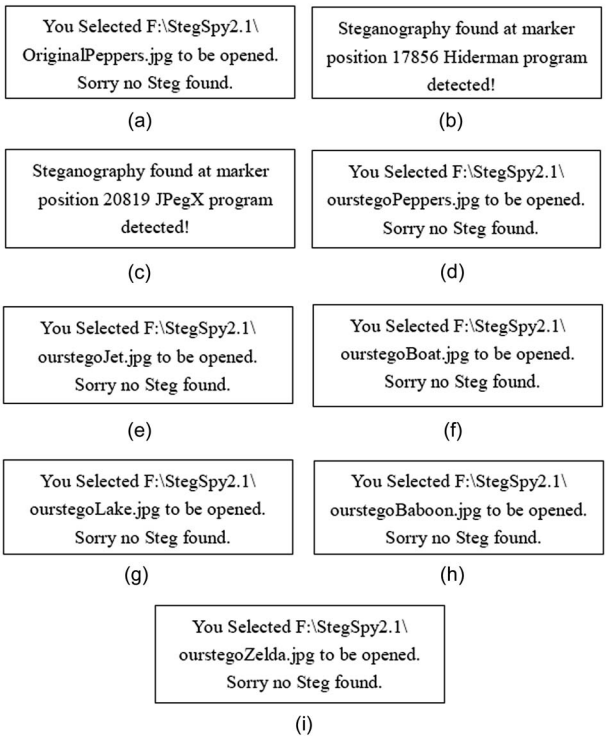


Fig. 8 Results of StegSpy analysis: (a) is for the original JPEG code of Peppers. (b) and (c) are for the JPEG stego codes that are created by Hiderman and JPegX hiding tools, respectively. (d) to (i) are for our six JPEG stego codes, the decompressed images of which are in Fig. 6.

whereas the progressive methods,[17,19,20] which also use the sharing algorithm of Thien and Lin[4] to generate shadows, either have shadows that are larger than ours or cannot achieve lossless reconstruction.) Traditional image sharing schemes are suitable for sharing top-secret images because of their all-or-nothing property. Progressive schemes provide a flexible means of viewing sensitive images progressively at certain quality levels. As indicated in Table 3, each of the shadows herein is smaller than those in the related works[4,9,19,20] (and about half the size of that in Hung *et al.*,[17] as determined by comparing their lossy approach with our lossy approach). This improvement reduces storage space and transmission time and facilitates the hiding of shadows. Therefore, the proposed method is better suited to transmit an image though limited communication channels.

- Easier to apply to scalable MPEG video transmission. The proposed scheme can also be adopted in the transmission of scalable MPEG video. Scalable MPEG video transmission depends on the adapting of video compression bit streams with a range of quality levels to meet various network environments or different end-user requirements. Since an MPEG video encoder also uses quality factors to control the quality of decoded video, MPEG video codes of various quality levels can be generated with various quality factors, and then these MPEG codes can be shared. Hence, the proposed scheme conveniently provides a scalable video transmission system (and still with much smaller shadows than those of progressive schemes[17–20]; the reasoning for being much smaller is skipped in order to reduce paper length).
- Each of the shadows in the proposed scheme can resist differential attack, whereas the image-sharing methods[4,9,19,20] in Table 3 cannot. Simple XOR and circular-shift operations are adopted to enhance the security of noiselike shadows, to enable them to resist differential attack.
- Unlike in several works,[4,5,8,9,12,14,18–20] the to-be-shared data herein are the compression code rather than the raw file. The use of the compression code has at least the following two advantages: (1) since compression disturbs the correlation between adjacent pixels of an image, the permutation process that is employed elsewhere[4,9,19,20] before the image is shared can be omitted; (2) after inverse-sharing reconstruction, the compression code requires less storage space than the raw file used in several works;[4,5,8,9,12,14,18–20] yet the benefit of lossless reconstruction when most of the shadows are collected is retained.
- Arithmetic operations are performed over GF(2^8) which can be replaced by GF(2^k) for any positive integer k, rather than GF(p), which is used in the image sharing schemes[4,5,7,12,14,17,19] for a prime number p. This increases the convenience of sharing digital data, which are often in binary form, regardless of whether they are pixel values or bit streams.

Acknowledgments

The work was supported by National Science Council, Taiwan, under NSC Grant No. 97-2221-E-009-120-MY3. The authors thank the reviewers for valuable suggestions.

References

1. G. R. Blakley, "Safeguarding cryptographic keys," *Proc. AFIPS National Computer Conf.* 48, 313–317 (1979).
2. A. Shamir, "How to share a secret," *Commun. ACM* **22**(11), 612–613 (1979).
3. A. Beimel and B. Chor, "Secret sharing with public reconstruction," *IEEE Trans. Inf. Theory* **44**(5), 1887–1896 (1998).
4. C. C. Thien and J. C. Lin, "Secret image sharing," *Comput. Graphics* **26**(5), 765–770 (2002).
5. C. C. Thien and J. C. Lin, "An image-sharing method with user-friendly shadow images," *IEEE Trans. Circuits Syst. Video Technol.* **12**(13), 1161–1169 (2003).
6. C. C. Lin and W. H. Tsai, "Visual cryptography for gray-level images by dithering techniques," *Pattern Recogn. Lett.* **24**(1–3), 349–358 (2003).
7. C. C. Lin and W. H. Tsai, "Secret image sharing with capability of share data reduction," *Opt. Eng.* **42**(8), 2340–2345 (2005).
8. 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**(12), 3652–3666 (2007).
9. H. K. Tso, "Sharing secret images using Blakley's concept," *Opt. Eng.* **47**(7), 077001 (2008).
10. D. C. Wu and W. H. Tsai, "Spatial-domain image hiding using image differencing," *IEE Proc. Vision, Image, Signal Process.* 147(1), 29–37 (2000).
11. D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recogn. Lett.* **24**(9–10), 1613–1626 (2003).
12. C. C. Lin and W. H. Tsai, "Secret image sharing with steganography and authentication," *J. Syst. Softw.* **73**(3), 405–414 (2004).
13. C. C. Thien and J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function," *Pattern Recogn.* **36**(12), 2875–2881 (2003).
14. Y. S. Wu, C. C. Thien, and J. C. Lin, "Sharing and hiding secret images with size constraint," *Pattern Recogn.* **37**(7), 1377–1385

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms

(2004).
15. S. J. Wang, "Steganography of capacity required using modulo operator for embedding secret image," *Appl. Math. Comput.* **164**(1), 99–116 (2005).
16. C. Y. Yang and J. C. Lin, "Image hiding by base-oriented algorithm," *Opt. Eng.* **45**(11), 117001 (2006).
17. K. H. Hung, Y. J. Chang, and J. C. Lin, "Progressive sharing of an image," *Opt. Eng.* **47**(4), 047006 (2008).
18. W. P. Fang, "Friendly progressive visual secret sharing," *Pattern Recogn.* **41**(4), 1410–1414 (2008).
19. S. K. Chen and J. C. Lin, "Fault-tolerant and progressive transmission of images," *Pattern Recogn.* **38**(12), 2466–2471 (2005).
20. R. Z. Wang and S. J. Shyu, "Scalable secret image sharing," *Signal Process. Image Commun.* **22**(4), 363–373 (2007).
21. U. Horn and B. Girod, "Scalable video transmission for the Internet," *Computer Networks ISDN Syst.* **29**(15), 1518–1529 (1997).
22. T. Schierl, T. Stockhammer, and T. Wiegand, "MPEG video transmission using scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.* **17**(9), 1204–1217 (2007).
23. N. K. Pareek, V. Patidar, and K. K. Sud, "Image encryption using chaotic logistic map," *Image Vis. Comput.* **24**(9), 926–934 (2006).
24. H. E. H. Ahmed, H. M. Kalash, and O. S. F. Allah, "An efficient chaos-based feedback stream cipher (ECBFSC) for image encryption and decryption," *Informatica* **4**(31), 121–129 (2007).
25. H. S. Kwok and W. K. S. Tang, "A fast image encryption system based on chaotic maps with finite precision representation," *Chaos, Solitons Fractals* **32**(4), 1518–1529 (2007).
26. W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York (1993).
27. L. S. T. Chen, S. J. Lin, and J. C. Lin, "Reversible JPEG-based hiding method with high hiding-ratio," accepted by *Int. J. Pattern Recognit. Artif. Intell.* (in press).
28. Fourth public release of Independent JPEG Group's free JPEG software, available at http://packetstormsecurity.nl/crypt/stego/jpeg-steg/jpeg-v4.tar.gz.
29. J. Kim, Y. Byun, and J. Choi, "Image forensics technology for digital camera," in *Pacific-Rim Conf. Multimedia 2004*, K. Alzawa, Y. Nakamura, and S. Satoh, Eds., *Lect. Notes Comput. Sci.* **3333**, 331–339 (2005).
30. S. Guillermito, Chi-square steganography test program, available at http://www.guillermito2.net/stegano/tools/index.html (2004).
31. M. T. Raggo, StegSpy steganography test program, http://www.spyhunter.com/stegspydownload.htm (2002).
32. JPegX hiding tool, available at http://www.globalfreeware.com/Jpegx-211.html.

Lee Shu-Teng Chen received his BS degree in computer science from National Chiao Tung University (NCTU), Taiwan, in 1999, and his MS degree in computer science and information engineering from National Taiwan University, Taiwan, in 2001. He has been in the PhD program since 2004 and is currently a PhD candidate in the Department of Computer Science and Information Engineering at NCTU. His current research interests include data hiding and image sharing.

Ja-Chen Lin received his BS degree and MS degree from NCTU, Taiwan. In 1988, he received his PhD degree in mathematics from Purdue University, West Lafayette, Indiana. He joined the Department of Computer and Information Science at NCTU and became a professor there. His research interests include pattern recognition and image processing. Dr. Lin is a member of the Phi-Tau-Phi Scholastic Honor Society.

Downloaded From: http://electronicimaging.spiedigitallibrary.org/ on 04/25/2014 Terms of Use: http://spiedl.org/terms