Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 178 (2008) 2433–2447

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A novel secret image sharing scheme in color images using small shadow images

Chin-Chen Chang [a,b], Chia-Chen Lin [c,*], Chia-Hsuan Lin [a], Yi-Hui Chen [b]

[a] *Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan, ROC*
[b] *Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 621, Taiwan, ROC*
[c] *Department of Computer Science and Information Management, Providence University, Taichung 43301, Taiwan, ROC*

Received 18 April 2007; received in revised form 23 December 2007; accepted 29 December 2007

Abstract

Over the past several years, secret image sharing techniques have become another branch of the effort to prevent secret images from being eavesdropped on, in addition to traditional cryptography. Because smaller shadows can speed up the transmission of a secret color image, in this paper we combine Chang and Wu's gradual search algorithm for a single bitmap BTC (GSBTC) and Shamir's (k,n) threshold concept to propose a novel secret color image sharing scheme that generates smaller shadows. Experimental results confirm that the proposed scheme successfully reduces shadow size and that each shadow behaves as a random-like image that prevents leakage of information about the secret color image. Furthermore, the correlation between two vertically or horizontally adjacent pixels in each shadow is significantly less than those in a color secret image, and the presented scheme also achieves, on average, an NPCR = 0.414% and AUCI = 32.78%. Thus, with our scheme one-pixel difference could cause a significant difference in the corresponding shadows. Therefore, the security of the presented scheme is also confirmed.
© 2008 Elsevier Inc. All rights reserved.

Keywords: Secret sharing; Small shadows; Secret color image sharing; (k,n) Threshold

1. Introduction

With the growth in computer technologies, multimedia products such as digital cameras have become more and more popular, and more digital images are being widely shared and transmitted over the Internet as a result. However, transmitting secret or important images, such as those used by the military or by commercial businesses, over the Internet poses certain dangers. Persons with malicious intent can monitor the Internet and try to detect and grab these valuable images. To protect transmitted images from being grabbed or tampered with, secret sharing techniques [1–3,13,17,19] have been proposed as another branch alongside traditional

* Corresponding author. Tel.: +886 4 26328001x18108; fax: +886 4 26324045.
E-mail addresses: ccc@cs.ccu.edu.tw (C.-C. Chang), mhlin3@pu.edu.tw (C.-C. Lin), m9405188@fcu.edu.tw (C.-H. Lin), chenyh@cs.ccu.edu.tw (Y.-H. Chen).

0020-0255/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2007.12.016

cryptographic techniques [10,11,15,16] and steganography [7,12,14] to protect vital images from predatory behaviors. The concept behind this secret sharing technique, also called a (k,n) threshold scheme, was proposed independently by Blakley [3] and Shamir [17]. These (k,n) threshold schemes have three properties, where $k \leqslant n$:

(1) The secret data can be divided into n parts.
(2) Any k or more than k parts can recover the secret data.
(3) Any $k-1$ or fewer than k parts cannot compute the secret data.

Using the (k,n) threshold scheme, in 1995 Naor and Shamir first introduced a secret image sharing technique that focused on image data [15]. In essence, the secret image sharing technique uses several random-like images, called shadows, instead of the original image to transmit data over the Internet. Shadows can thwart malicious attackers and prevent secret images from being accessed directly. Secret image sharing schemes also inherit the three properties of the (k,n) threshold scheme.

Many secret image sharing schemes have been proposed over the past decade [9,15,18,20,24]. They share one characteristic, that secret images can be recovered by stacking their shadows. In other words, these secret images can easily be recovered without complex computation. However, in these schemes the size of the shadows is usually larger than or equal to the size of the secret image. Many schemes [4,5,8,21–23] have been proposed to overcome this problem. In 2002, Thien and Lin [21] proposed a (k,n) secret image sharing scheme that generates smaller shadows. Their scheme generates the $k-1$ degree polynomial by letting the k coefficients be the gray values of k pixels and moduloing 251. By moduloing 251, they truncate all gray values greater than 250, which results in confining all secret image gray values to the 0–250 range. A key is then used to generate a permutation sequence to shuffle the pixels of the secret image so that any correlation between neighboring pixels is hidden. Later, the k not-shared pixels of the shuffled secret image are collected and become the k coefficients of the $k-1$ degree polynomial. These coefficients and the $k-1$ degree polynomial generate n pixels for the n shadow images until all pixels in the secret image are processed. Using this process, Thien and Lin successfully reduced the size of each shadow to $1/k$ size of the secret image.

Inspired by Thien and Lin, Wang and Su [23] later designed a secret image sharing scheme that improves on Thien and Lin's scheme. Their scheme first generates the difference image for a t-bit secret image. Next, the difference image is encoded by using Huffman coding, and the resulting t bits form a sharing coefficient. Then, k sharing coefficients are used as the coefficients of the polynomial sharing function of degree $k-1$ until all the Huffman coding results are processed. Experiments with Wang and Su's scheme showed that the size of the generated shadows is smaller than that of Thien and Lin's scheme. However, neither scheme can be directly applied to color images because the size of a generated color shadow is triple the size of a gray-level shadow.

Because a secret image maybe a color image, we present a lossy secret color image sharing scheme in this paper that expands the application of secret image sharing. In our scheme, the size of the shadows depends on the number of k, and larger k's generate smaller shadows. Experimental results confirm that the presented scheme can efficiently reduce the size of the shadows.

The rest of this paper is organized as follows. Section 2 contains a brief review of the (k,n) threshold scheme and GSBTC. Section 3 presents our scheme in detail. Section 4 shows the experimental results and gives a statistical analysis and security analysis of the presented scheme. Future work and some conclusions are described in Section 5.

2. Related work

Two important techniques were applied to design our presented scheme: the (k,n) threshold scheme and GSBTC. The former allows one secret image to be decomposed into n shadow images, with only k shadows required to construct the original secret image. The latter can condense one secret color image into a smaller size with acceptable distortion. In other words, by adopting the (k,n) threshold scheme and GSBTC, potential loss during data transmission can be overcome and shadow sizes can be reduced successfully. To give sufficient background knowledge of the presented scheme, these techniques are illustrated in the following sections.

2.1. Shamir's (k,n) threshold scheme

In 1979, Shamir proposed a (k,n) threshold scheme based on polynomial interpolation [17]. His scheme assumes that data D is divided into n pieces $D_1, D_2, \ldots, D_n$, and only any k pieces of D_i's, where $k \leqslant n$, are required to reconstruct the original secret data D. The kth piece cannot be derived by using the received $(k-1)$ pieces; therefore, if only $(k-1)$ or fewer than $(k-1)$ pieces are received, then secret data D still cannot be revealed.

Assume that the secret data D is a number. In Shamir's scheme, a prime number g is randomly chosen and the polynomial sharing function of degree $k-1$ is defined as in Eq. (1) to divide D into n shadows.

$$f(x) = (a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}) \bmod g, \tag{1}$$

where $a_1, a_2, \ldots, a_{k-1}$ are random numbers and $a_0 = D$. Each D_i for $i = 1$–n can be derived by Eq. (2)

$$D_i = f(i), \tag{2}$$

where $i = 1$–n, and each one of $D_1, D_2, \ldots, D_n$ is treated as a shadow. To obtain the secret data D, any k or more shadows of D_i's can reconstruct $f(x)$. All coefficients of the polynomial function $f(x)$ can be derived from Lagrange's interpolation formula, and the secret data $D = a_0 = f(0)$ can finally be calculated.

The secret color image sharing scheme presented here is based on Shamir's (k,n) threshold scheme, so that the secret image D is divided into n shadows $(D_1, D_2, \ldots, D_n)$. However, in the proposed scheme, the size of each share decreases as the number of shares increases.

2.2. Gradual search algorithm for one single bitmap BTC

Before adopting Shamir's (k,n) threshold scheme to divide a secret image into several shadows in order to overcome any potential loss during data transmission, we needed to deal with the other issue; that is, how to reduce the original color image size so that the shadow size also can be decreased. The gradual search algorithm for one single bitmap BTC (GSBTC) designed by Chang and Wu was our solution. GSBTC, a block truncation coding (BTC) variant, is an efficient and simple algorithm for color image compression [6].

In traditional BTC, an image is first divided into several non-overlapping square blocks, and each block can be 4×4, 8×8, and so on. The mean pixel value $\bar{x}$ for each block is then computed. Generally, $\bar{x}$ is a quantization threshold. In a block, each pixel value X_i is compared with $\bar{x}$ so that a bitmap composed of two groups is generated. If $\bar{x} < X_i$, the corresponding bit in the bitmap belongs to *group-1* and is denoted as '1'; otherwise, the bit belongs to *group-0* and is denoted as '0'. Although BTC is designed to compress gray rather than color images, the same procedure can be applied to color images. Unfortunately, the compression ratio is not the same when BTC is used to compress a color image because a color image is composed of three planes, R, G and B, and these three planes form three bitmaps when BTC is used.

To improve the BTC compression ratio in color images, a general approach is to find a common bitmap that causes the least distortion to represent the three individual bitmaps. However, it is difficult to design an algorithm to generate a common bitmap. Chang and Wu's GSBTC algorithm generates an effective common bitmap that provides acceptable image quality of the decompressed image while maintaining low computation complexity. In this paper, we use GSBTC to compress a secret color image and then present a lossy secret color image sharing scheme. To better explain our presented scheme, we first introduce Chang and Wu's GSBTC algorithm in detail in the following paragraphs.

Suppose a color image is divided into several non-overlapping blocks having $m \times m$ pixels. At first, each block is processed with traditional BTC. In other words, each block consists of three bitmaps, which map to the three planes R, G and B. Each bitmap contains two values: '0', which denotes that the corresponding pixel value is less than the mean pixel value, and '1', which denotes that the corresponding pixel value is equal to or larger than the mean pixel value. To generate a common bitmap B_C for each block, bits that are the same in the R, G and B bitmaps are called determined elements and are reserved in the common bitmap B_C first. Bits having different values in the three bitmaps, called non-determined elements, are not reserved in the common bitmap B_C. That is, the elements of a common bitmap B_C are composed of two groups, D_C and ND_C, as follows:

$$D_C = \{c_i | c_i \in B_C \text{ and } c_i \neq \text{non-determined}\}, \tag{3}$$

$$ND_C = \{c_i | c_i \in B_C \text{ and } c_i = \text{non-determined}\}, \tag{4}$$

where D_C is the determined set and ND_C is the non-determined set. The bitmaps of the *R*, *G* and *B* planes and a common bitmap are defined as

$$\begin{aligned} &B_R = \{r_1, r_2, \ldots, r_{m\times m} | r_i = L \text{ or } H\}, \\ &B_G = \{g_1, g_2, \ldots, g_{m\times m} | g_i = L \text{ or } H\}, \\ &B_B = \{b_1, b_2, \ldots, b_{m\times m} | b_i = L \text{ or } H\}, \\ &\text{and } c_i = \begin{cases} r_i & \text{if } r_i = g_i = b_i, \\ \text{non-determined} & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

The initial mean square error (MSE) of B_C is generated by ignoring any non-determined elements, as follows:

$$\text{MSE} = \sum_{c_i = L} (x_i - \overline{X_L})^2 + \sum_{c_i = H} (x_i - \overline{X_H})^2, \tag{6}$$

$$\overline{X}_L = \frac{\sum_{c_i=L} x_i}{q}, \tag{7}$$

$$\overline{X}_H = \frac{\sum_{c_i=H} x_i}{p},$$

where *L* is ‘0’, *H* is ‘1’, *q* is the number of *L*’s in a block, *p* is the number of *H*’s in a block, c_i is the *i*th element in the common bitmap B_C, $i = 1,2,\ldots,m \times m$, and $x_i = (x_{Ri}, x_{Gi}, x_{Bi})$ is the *i*th pixel value for one block.

The GSBTC algorithm is summarized in the following steps:

Step 1. Generate the initial common bitmap B_c for an $m \times m$ block.
Step 2. Compute the initial MSE by using Eq. (6).
Step 3. Remove an element c_j from ND_C, which is calculated for the *newMSE* for $c_j = L$ and $c_j = H$ as follows:

$$newMSE_j = \sum_{c_i = L} (x_i - \overline{X_L})^2 + \sum_{c_j = H} (x_i - \overline{X_H})^2, \tag{8}$$

where $c_i \in D_C \cup \{c_j\}$, *q* is the number of *L*’s in $D_C \cup \{c_j\}$, *p* is the number of *H*’s in $D_C \cup \{c_j\}$.
Step 4. Select the minimum *newMSE* of all computed *newMSE* as follows:

$$newMSE = \min\{newMSE_j | c_j \in ND_C\}. \tag{9}$$

Step 5. Add c_j to D_C so that $D_C = D_C \cup \{c_j\}$, and $ND_C = ND_C - \{c_j\}$.
Step 6. Repeat Steps 3–5 until ND_C is empty.

When GSBTC processing is completed, the best common bitmap $B_C = D_C$ is generated and three quantization pairs $\{R_{X_L}, R_{X_H}\}$, $\{G_{X_L}, G_{X_H}\}$, and $\{B_{X_L}, B_{X_H}\}$ for the *R*, *G* and *B* planes are obtained from B_C, as follows:

$$R_{X_L} = \left(\sum_{r_i \in B_R \text{ and } r_i = L} r_i \right) \Bigg/ q,$$

$$R_{X_H} = \left(\sum_{r_i \in B_R \text{ and } r_i = H} r_i \right) \Bigg/ ((m \times m) - q),$$

$$G_{X_L} = \left(\sum_{g_i \in B_G \text{ and } g_i = L} g_i \right) \Bigg/ q,$$

$$G_{X_H} = \left(\sum_{g_i \in B_G \text{ and } g_i = H} g_i \right) \Bigg/ ((m \times m) - q),$$

$$B_{X_L} = \left(\sum_{b_i \in B_B \text{ and } b_i = L} b_i \right) \Big/ q,$$

$$B_{X_H} = \left(\sum_{b_i \in B_B \text{ and } b_i = H} b_i \right) \Big/ ((m \times m) - q), \tag{10}$$

where q is the number of L's in B_C.

For each block of the secret color image, three quantization pairs $\{R_{X_L}, R_{X_H}\}$, $\{G_{X_L}, G_{X_H}\}$, and $\{B_{X_L}, B_{X_H}\}$ and a common bitmap $B_C m \times m$ bits in size can be generated. That is, the total size of a block is $\left(6 + \frac{m \times m}{8}\right)$ bytes. Each block can be recovered by using its three quantization pairs and a common bitmap.

3. Proposed scheme

The concept behind secret image sharing is that a secret image is divided into n shadows by the polynomial sharing functions, and the generated shadows are individually transmitted to authorized recipients. For the original secret to be restored, these shared small-sized shadows must be collected in at least k parts to reconstruct the polynomial sharing functions by Lagrange's interpolation formula. That is, a secret image needs at least k or more shadows to be reconstructed.

To expand secret image sharing to encompass secret color images without resulting in large-sized shadows, we applied Chang and Wu's GSBTC scheme and Shamir's (k,n) threshold scheme to design our presented scheme. The presented secret image sharing scheme is divided into two procedures: sharing procedure and revealing procedure. In essence, the major feature of our scheme is that as more shadows are generated, the size of each shadow decreases.

3.1. Sharing procedure

The sharing procedure requires two phases to divide a secret color image into n shadows, as shown in Fig. 1. The *Phase 1* GSBTC algorithm has been described in detail in Section 2.2. Therefore, the following paragraphs give more detail about *Phase 2*.

3.1.1. Phase 1: GSBTC

A secret color image I is first divided into several non-overlapping blocks and inputted into *Phase 1*. The output of *Phase 1* for each block of secret color image I is three quantization pairs $\{R_{X_L}, R_{X_H}\}$, $\{G_{X_L}, G_{X_H}\}$,and $\{B_{X_L}, B_{X_H}\}$ for the R, G, and B planes, plus a common bitmap B_C. Assume that each block size is $m \times m$ and that the size of the compression codes for each block is $(6 + m \times m/8)$ bytes. In the following discussion, the three quantization pairs for the R, G, and B planes $\{R_{X_L}, R_{X_H}\}$, $\{G_{X_L}, G_{X_H}\}$, and $\{B_{X_L}, B_{X_H}\}$ and the common bitmap B_C for each block of secret color image I are called the compression codes. When all blocks have been processed, all compression codes are inputted into *Phase 2*. Note that m is the power of 2, and is usually set as 4 to obtain a good compression result. If the size of secret color image I cannot be divided by $m \times m$, a random pixel stream can be generated and appended to the end of secret color image, and the size of the random pixel stream becomes part of the secret used later to reveal the secret image.

3.1.2. Phase 2: constructing the sharing functions and generating n shadows

In this phase, the (k,n) threshold and the $k - 1$ degree of polynomial sharing functions are decided first. The $k - 1$ degree polynomial sharing function can be constructed by using Eq. (11):

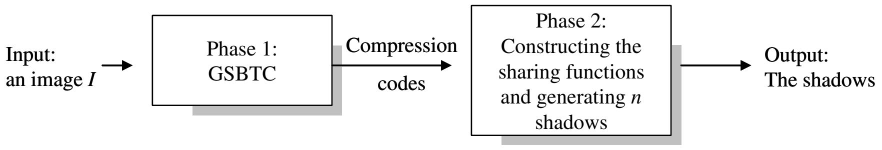

Fig. 1. Diagram of the presented two-phased sharing procedure.

$$f_j(x) = (a_0 + a_1x + \cdots + a_{k-1}x^{k-1})\text{mod}g, \tag{11}$$

where $a_0, a_1, \ldots, a_{k-1}$ are k coefficients, $1 \leqslant j \leqslant s$ and s is defined below:

$$s = b \times \left(\frac{6 + (m \times m/8)}{k}\right). \tag{12}$$

In Eq. (12), b is the number of non-overlapping blocks for secret color image I and k is the number of shadows required for reconstructing the secret color image.

Then, the compression codes $(6 + m \times m/8)$ bytes in size that belong to one of the blocks of secret color image I are transformed into $(6 + m \times m/8)$ binary bit streams, and these bit streams are converted into new compression codes $(6 + m \times m/8)$ bytes in size. The first byte of the new $(6 + m \times m/8)$ bytes is derived by collecting the first bit of each of the eight binary bit streams. Similarly, the second byte of the new $(6 + m \times m/8)$ bytes is derived by collecting the second bit of each of the eight binary bit streams. The whole procedure is repeated until the last byte of the new $(6 + m \times m/8)$ bytes is derived. All these new compression codes are embedded into the coefficients of the sharing functions in sequence.

Assume that k and n are determined so that secret color image I is shared by n shadows $I_1, I_2, \ldots, I_n$ and that any k or more shadows of $I_1, I_2, \ldots, I_n$ are required to reveal secret color image I by using Lagrange's interpolation formula. The pixel values $v_{j1}, v_{j2}, \ldots, v_{jn}$ for each shadow are obtained from $f_j(x)$, where $1 \leqslant j \leqslant s$ by using Eq. (13):

$$v_{j1} = f_j(1),\ v_{j2} = f_j(2),\ \ldots,\ v_{jn} = f_j(n). \tag{13}$$

In essence, the pixel values $v_{j1}, v_{j2}, \ldots, v_{jn}$ are separately assigned to n shadows $I_1, I_2, \ldots, I_n$. The detailed steps of *Phase 2* are described below.

Step 1. Determine a (k,n) threshold, where n is the number of shadows and k is the number of shadows required to reveal the secret image.
Step 2. Convert the $(6 + m \times m/8)$ byte compression codes generated in *Phase 1* that belong to one block of secret color image I into $(6 + m \times m/8)$ binary bit streams.
Step 3. For $(6 + m \times m/8)$ binary bit streams, collect the first bit of each of the eight binary bit streams to generate the first byte of the new compression codes. Similarly, collect the second bit of each of the eight binary bit streams to generate the second byte of the new compression codes. Repeat this procedure until the last byte of the new compression codes is derived, as shown in Fig. 2.
Step 4. Convert the new $(6 + m \times m/8)_2$ binary bit streams into $(6 + m \times m/8)_2$ decimal digits so that new compression codes for one block of secret color image I can be generated, as shown in Fig. 3.
Step 5. Repeat Steps 2–4 until all blocks in secret color image I have been processed. Compute the sum and variance of all digits of the new compression codes of the secret color image. Use the sum or variance

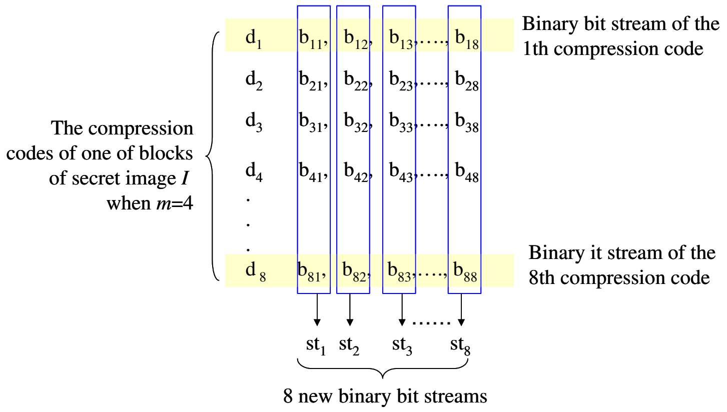
The compression codes of one of blocks of secret image I when m=4
d1 b11, b12, b13,....., b18
d2 b21, b22, b23,....., b28
d3 b31, b32, b33,....., b38
d4 b41, b42, b43,....., b48
d8 b81, b82, b83,....., b88
Binary bit stream of the 1th compression code
Binary it stream of the 8th compression code
st1 st2 st3 st8
8 new binary bit streams

Fig. 2. Generating new binary bit streams by collect bits having the same order in each binary bit stream when $m = 4$.

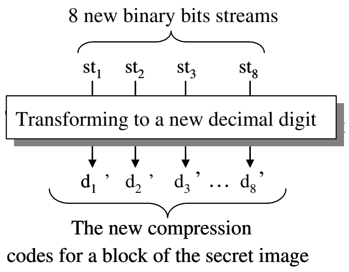
8 new binary bits streams
st1 st2 st3 st8
Transforming to a new decimal digit
d1' d2' d3' … d8'
The new compression codes for a block of the secret image

Fig. 3. Transformation from new binary bit streams into new compression codes when $m = 4$.

as a key, depending on whether the sum is even or odd. If the sum is even, use sum as the key; otherwise, use variance as the key. Finally, use the key to permute all digits of the new compression codes of the secret color image, as shown in Fig. 4.

Step 6. Select k digits from $(6 + m \times m/8) \times b$ digits from the permuted compression codes of the secret image as coefficients $a_0, a_1, \ldots, a_{k-1}$, in sequence, where b is the number of non-overlapping blocks of secret color image I, until all digits are assigned, as shown in Fig. 5. Then, m sharing functions $f_1(x), f_2(x), \ldots, f_m(x)$ can be generated.

Step 7. Use n shadow numbers $1, 2, \ldots, n$ to obtain pixel values $v_{j1}, v_{j2}, \ldots, v_{jn}$ by using Eq. (12) and assign pixel values $v_{j1}, v_{j2}, \ldots, v_{jn}$ to $I_1, I_2, \ldots, I_n$ shadows, individually, where $1 \leqslant j \leqslant s$ and m is the number of sharing functions.

3.2. Revealing procedure

To extract a compressed secret color image, at least any k of n shadows must be collected in advance. A diagram of the revealing procedure is shown in Fig. 6.

First, *Phase 1* uses k collected shadows to reconstruct the sharing functions. Then, *Phase 2* decodes the coefficients of the sharing functions to reveal the compression codes and reconstruct the secret color image.

3.2.1. Phase 1: reconstructing the sharing functions

Suppose k shadows $(I_1, I_2, \ldots, I_k)$ are collected and pixel values $v_{j1}, v_{j2}, \ldots, v_{jn}$ are derived from the collected shadows so that sharing functions $f_1(x), f_2(x), \ldots, f_s(x)$ can be reconstructed by Lagrange's interpolation formula, where $1 \leqslant j \leqslant s$ and s is the number of sharing functions.

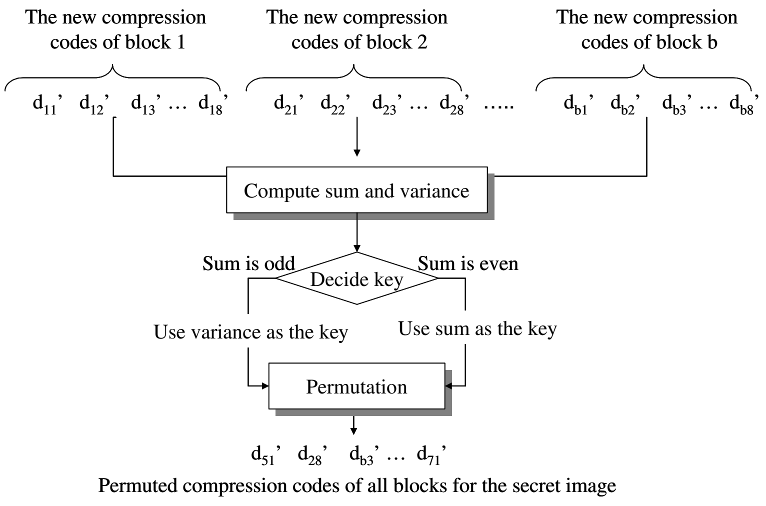
The new compression codes of block 1
The new compression codes of block 2
The new compression codes of block b
d11' d12' d13' … d18'
d21' d22' d23' … d28' …..
db1' db2' db3' … db8'
Compute sum and variance
Sum is odd
Decide key
Sum is even
Use variance as the key
Use sum as the key
Permutation
d51' d28' db3' … d71'
Permuted compression codes of all blocks for the secret image

Fig. 4. Permuted compression codes for all blocks of the secret image when $m = 4$.

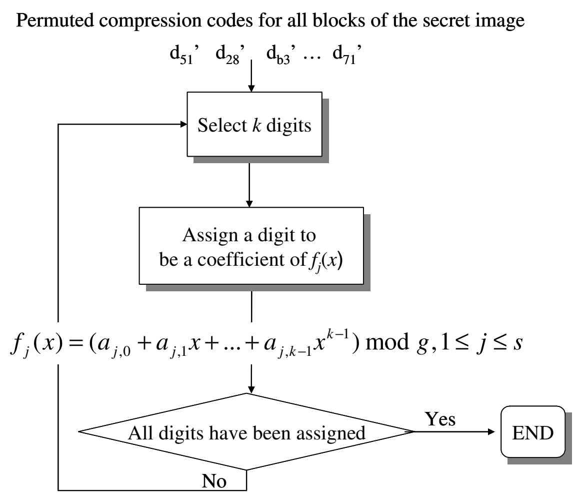
Permuted compression codes for all blocks of the secret image
d51' d28' db3' … d71'
Select k digits
Assign a digit to be a coefficient of fj(x)
$f_j(x) = (a_{j,0} + a_{j,1}x + \ldots + a_{j,k-1}x^{k-1}) \bmod g, 1 \le j \le s$
All digits have been assigned
Yes
END
No

Fig. 5. Assigning each digit as a coefficient of sharing function $f_j(x)$ when $m = 4$.

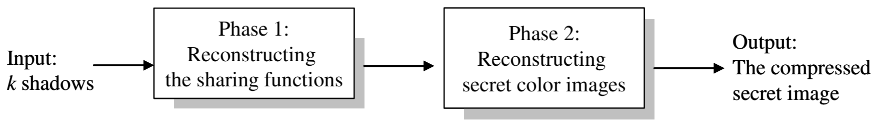
Input:
k shadows
Phase 1: Reconstructing the sharing functions
Phase 2: Reconstructing secret color images
Output:
The compressed secret image

Fig. 6. Diagram of the presented two-phased revealing procedure.

3.2.2. Phase 2: reconstructing the secret color image

All coefficients of the sharing functions are reorganized from the original compression codes after all digits of the permuted compression codes have been permuted by using a key. Therefore, *Phase 2* must first compute the sum and variance of all digits of the permuted compression codes of the secret image. If the sum is even, sum is used as the key; otherwise, variance becomes the key. The key is then used to obtain the restored compression codes, as shown in Fig. 7.

Later, *Phase 2* is used to reconstruct the secret image by simply reversing the rest of the *Phase 2* of the sharing procedure (in Section 3.1) to reconstruct the original compression codes for each block in the secret color

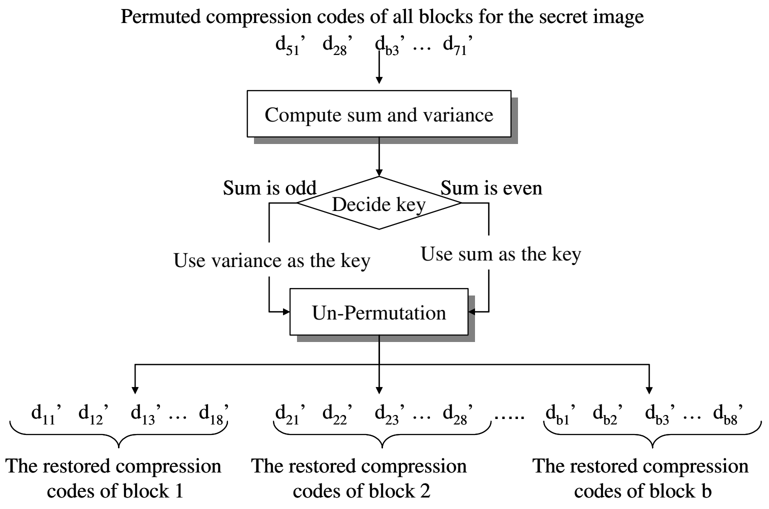

Fig. 7. Unpermute the permuted compression codes of all blocks.

image. When all compression codes have been obtained, the secret color image can be restored by using the GSBTC decoding procedure. That is, each block of the secret color image can be restored by using the three quantization pairs and the common bitmap, and the GSBTC decoding procedure is the same as that used for traditional BTC.

4. Experimental results

In this section, the four 512×512 pixel test color images shown in Fig. 8, “Baboon”, “Scenes”, “Tiffany”, and “Peppers”, are used. The experimental platforms were implemented on a PC with a 1.73 GHz Pentium® CPU and a 1 GB RAM. The operating system is Windows XP Professional, and our algorithms were programmed by Matlab 7.0. To evaluate the image quality of the reconstructed secret color image with the presented secret sharing scheme, a modified peak signal-to-noise ratio ($\mathrm{PSNR}_{\mathrm{color}}$) was used in the following experiments. $\mathrm{PSNR}_{\mathrm{color}}$ is defined as

$$\mathrm{PSNR}_{\mathrm{color}} = 10\log_{10}\frac{255^2}{\mathrm{MSE}_{\mathrm{color}}}, \tag{14}$$

where $\mathrm{MSE}_{\mathrm{color}}$ is the mean square error between the original color image and the compressed color image. For an original color image $m \times n$ in size, the formula for $\mathrm{MSE}_{\mathrm{color}}$ is

$$\mathrm{MSE}_{\mathrm{color}} = \frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{(Rx_{ij} - Ry_{ij})^2 + (Gx_{ij} - Gy_{ij})^2 + (Bx_{ij} - By_{ij})^2}{3}, \tag{15}$$

Fig. 8. Four test images with 512×512 pixels.

where Rx_{ij} and Ry_{ij} are the pixel values of the R plane in the original color image and the compressed color image, Gx_{ij} and Gy_{ij} are the pixel values of the G plane in the original color image and the compressed color image, and Bx_{ij} and By_{ij} are the pixel values of the G plane in the original color image and the compressed

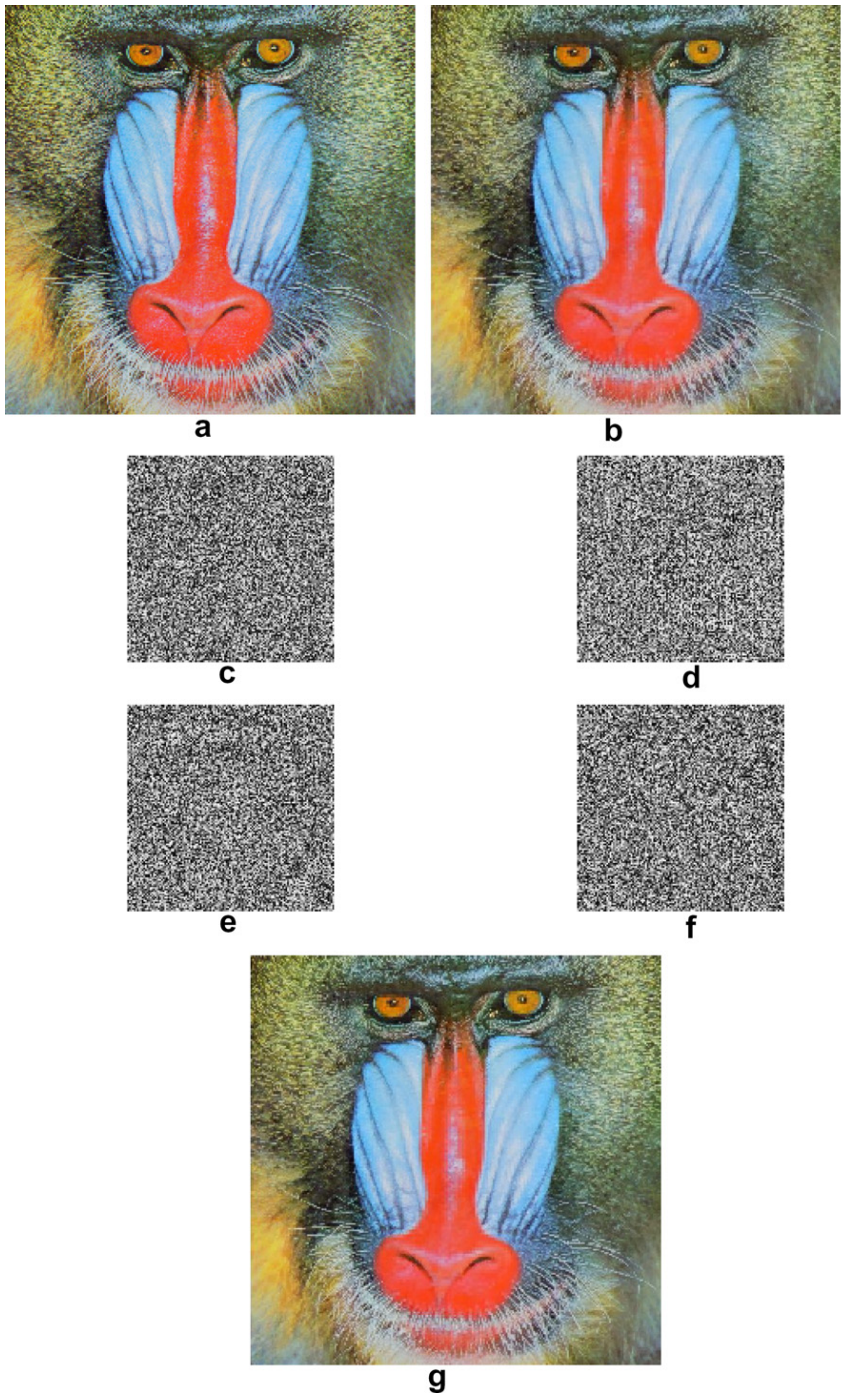

Fig. 9. Example of “Baboon”. (a) 512 × 512 “Baboon”. (b) Reconstructed “Baboon” with GSBTC ($\text{PSNR}_{\text{color}} = 24$ dB). (c) Shadow image-1. (d) Shadow image-2. (e) Shadow image-3. (f) Shadow image-4. (g) Revealed result by any two of four shadow images ($\text{PSNR}_{\text{color}} = 24$ dB).

color image, respectively. A higher $\mathrm{PSNR}_{\mathrm{color}}$ means that the quality of the reconstructed secret color image is similar to that of the original image. The typical $\mathrm{PSNR}_{\mathrm{color}}$ value in image compression is between 30 dB and 40 dB.

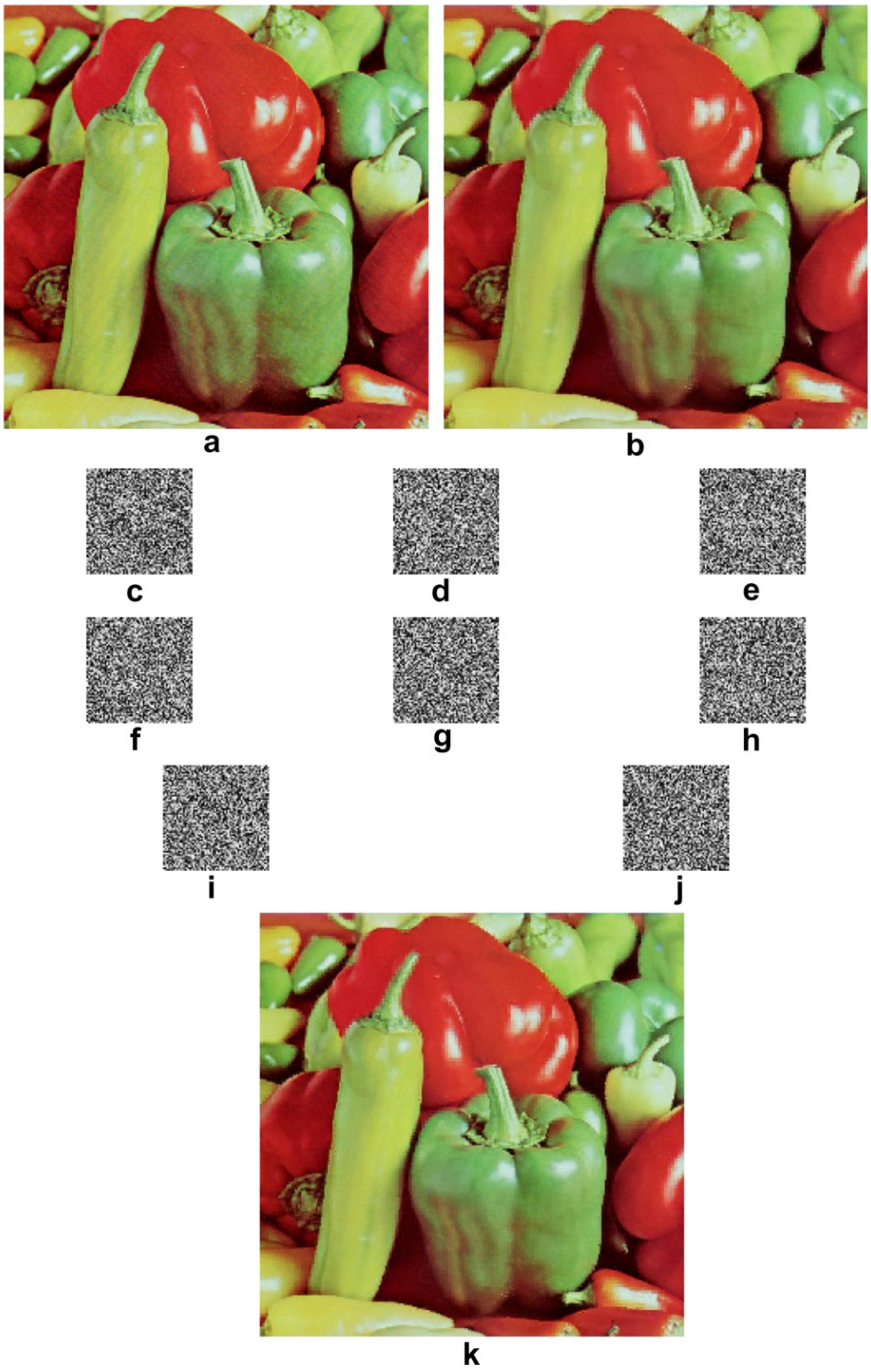

Fig. 10. Example of "Peppers". (a) 512 × 512 "Peppers". (b) Reconstructed "Peppers" with GSBTC ($\mathrm{PSNR}_{\mathrm{color}} = 29$ dB). (c) Shadow image-1. (d) Shadow image-2. (e) Shadow image-3. (f) Shadow image-4. (g) Shadow image-5. (h) Shadow image-6. (i) Shadow image-7. (j) Shadow image-8. (k) Revealed result by any two of four shadow images ($\mathrm{PSNR}_{\mathrm{color}} = 29$ dB).

To demonstrate the performance of our scheme, in the first experiment, a $(2,4)$ threshold scheme was used to generate four shadows, with each shadow one-quarter the size of the original color image. Fig. 9 shows the experimental results. Fig. 9a is the original secret color image "Baboon"; Fig. 9b is the reconstructed result from (a) with GSBTC; Fig. 9c–f are four shadows with 256×256 pixels when $k = 2$ and $n = 4$; and Fig. 9g is the reconstructed result by using the presented scheme.

Note that the presented scheme can reduce the size of each shadow in Fig. 9, and that each shadow is one-quarter the size of its original image when $k = 2$. Furthermore, each shadow is a random-like image.

In the second experiment, an $(8,8)$ threshold scheme was used to generate eight shadows. Fig. 10 shows the experimental results. Fig. 10a is the original secret color image "Peppers"; Fig. 10b is the reconstructed result from (a) with GSBTC; Fig. 10c–j are eight shadows with 128×128 pixels when $k = 8$ and $n = 8$; and Fig. 10g is the reconstructed result by using the presented scheme.

Fig. 10 shows that each shadow is 1/16 the size of the original when $k = 3$. The two experimental results above support the concept that the presented scheme can efficiently reduce shadow size. The relationship between the number of shadows and the size of each shadow is listed in Table 1.

Because a sharing function with degree $k - 1$ has k coefficients, a larger degree $k - 1$ has more coefficients in which to hide a secret. Therefore, when the number of shadows is increased the size of each shadow decreases gradually with our secret image sharing scheme. Table 1 shows the relationship between the size in bytes of generated shadows and (k,n) threshold's k when the secret image size is 512×512 pixels and 256×256 pixels, respectively. The experimental results suggest that users can consider their requirements and the available bandwidth in selecting the number of the (k,n) threshold and the secret image size.

4.1. Statistical analysis

To test the correlation between two horizontally adjacent pixels and two vertically adjacent pixels, respectively, in the original secret image and a shadow, two formulas were used to calculate the correlation coefficient of each pair.

$$\begin{aligned} &\operatorname{cov}(x,y) = E(x - E(x)(y - E(y))), \\ &r_{xy} = \frac{\operatorname{cov}(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}, \end{aligned} \tag{16}$$

where x and y are the gray-scale values of two adjacent pixels in a share. The following discrete formulas were used in numerical computation:

$$\begin{aligned} &E(x) = \frac{1}{w \times h}\sum_{i=1}^{w\times h} x_i, \\ &D(x) = \frac{1}{w \times h}\sum_{i=1}^{w\times h} (x_i - E(x))^2, \\ &\operatorname{cov}(x,y) = \frac{1}{w \times h}\sum_{i=1}^{w\times h} (x_i - E(x))(y_i - E(y)). \end{aligned} \tag{17}$$

Table 1
Relationship between the sizes in bytes of generated shadow images and (k,n) threshold's k using two sizes of secret images in our scheme

k	Secret image (256×256)	Secret image (512×512)
2	16,384	65,536
3	10,923	43,691
4	8192	32,768
8	4096	16,384
16	2048	8192

The correlations of two horizontally adjacent pixels and two vertically adjacent pixels in the original secret image and in each shadow are listed in Tables 2 and 3, respectively. From Tables 2 and 3, we can see that the two horizontally or vertically adjacent pixels exhibit only slight positive or negative correlation. Moreover, the correlation coefficients of the two horizontally adjacent pixels and two vertically adjacent pixels of each shadow are significantly less than those of the original secret images.

4.2. Security analysis

In essence, if one minor change in an original secret image can cause a significant change in each generated shadow with respect to diffusion and confusion, then any differential attack becomes practically useless. To protect our scheme from the differential attack, the reorganized compression codes for all blocks of the secret image must be permuted by a key, which is the sum or variance of all digits of the reorganized compression codes, depending on whether the sum is even or odd. When one pixel of the original secret image is changed, the sum and variance of the reorganized compression codes become different. In other words, the key used to permute digits of the reorganized compression codes also will be different, so that the relationship between each shadow and its original secret image is reduced. To test the influence of a one-pixel change on each shadow with our scheme, two common measures were used: number of pixels change rate (NPCR) and unified average changing intensity (UACI), which are defined as follows:

$$\mathrm{NPCR} = \frac{\sum_{i,j} D(i,j)}{w \times h} \times 100\%, \tag{18}$$

where w and h are the width and height of C_1, and C_2. C_1 and C_2 are two shadows whose corresponding secret images have only a one-pixel difference. D is a bipolar array the same size as image C_1 or C_2. If $C_1(i,j) = -C_2(i,j)$, then $D_1(i,j) = 1$; otherwise, $D_1(i,j) = 0$.

$$\mathrm{UACI} = \frac{1}{w \times h}\left[\sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{255}\right] \times 100\%, \tag{19}$$

NPCR measures the percentage of different pixel numbers between the two images C_1 and C_2, whereas UACI measures the average intensity of differences between the two images.

Table 2
Correlation coefficients of two horizontally adjacent pixels in original secret images and in each shadow of the secret images

Shadows	Horizontal coefficients	Secret images			
		Baboon	Scenes	Tiffany	Peppers
		0.878	0.972	0.950	0.970
Shadow 1	R_{xy}	−0.015	−0.011	−0.032	0.002
Shadow 2	R_{xy}	0.006	0.011	−0.023	−0.004
Shadow 3	R_{xy}	−0.006	0.008	−0.027	−0.001
Shadow 4	R_{xy}	−0.008	−0.001	−0.002	0.012

Table 3
Correlation coefficients of two vertically adjacent pixels in original secret images and in each shadow of the secret images

Shadows	Vertical coefficients	Secret images			
		Baboon	Scenes	Tiffany	Peppers
		0.784	0.970	0.912	0.977
Shadow 1	R_{xy}	0.022	−0.007	0.082	0.001
Shadow 2	R_{xy}	0.013	0.008	0.045	0.007
Shadow 3	R_{xy}	0.065	0.005	0.017	−0.008
Shadow 4	R_{xy}	0.003	−0.001	0.036	−0.005

Table 4
NPCRs vs. UACIs for each shadow of four test images when $n = 4$

Images	Shadow 1		Shadow 2		Shadow 3		Shadow 4	
	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI
Baboon	0.40	33.60	0.43	32.83	0.39	32.49	0.39	32.57
Scenes	0.39	32.61	0.39	32.57	0.39	32.72	0.43	32.85
Tiffany	0.46	30.02	0.46	34.37	0.44	33.07	0.46	33.45
Peppers	0.34	32.93	0.41	32.82	0.42	32.62	0.43	32.89

From Table 4, we can see that when one pixel in an original secret image is changed, not only is the percentage of changed pixels in its corresponding shadow significant, but the UACI also becomes significant. From the results listed in Table 4, we can see that our scheme has a good ability to resist differential attack.

Furthermore, because our scheme uses the Lagrange interpolation formula to reconstruct sharing functions by using the pixel values derived from collected shadows, the complexity of the presented scheme arises in determining how to reconstruct the polynomial sharing functions. A polynomial function of degree $k - 1$ requires at least k values to be reconstructed. In other words, at least any k shadow images are required to construct all sharing functions with the Lagrange interpolation formula. If a malicious attacker has $k - 1$ shadows and wants to construct all sharing functions, there will be a $(1/p)^s$ probability of reconstructing all functions if there is $1/p$ probability of reconstructing one sharing function. To reconstruct a sharing function, an attacker must correctly guess all coefficients for a sharing function. In our scheme, each sharing function has k coefficients and each coefficient can be converted into a binary representation having 8 bits. That means only a $\left(\frac{1}{2}\right)^{8k}$ probability of reconstructing a sharing function with our scheme. For example, if the secret image is 512×512 pixels, each block is 4×4 pixels, and $k = 2$, there are 16,384 blocks and 65,536 polynomial functions. In this case, the probability of reconstructing the secret image is $\left(\frac{1}{2}\right)^{65536 \times 8k}$.

5. Conclusions and future work

To protect a secret color image, prevent it from being eavesdropped by malicious attackers and ensure generated shadows can be transmitted smoothly without concern about bandwidth limitations, this paper presents a novel secret image sharing scheme for color images having smaller shadow images. Our scheme replaces a secret color image with many smaller random-like images called “shadows”. Based on the major concept of a (k, n) threshold, any k or more than k authorized recipients of a secret image can present their own collected shadows to recover the secret color image by using the revealing phase of the presented scheme.

To make sure the size of a generated shadow is as small as possible; we adopted GSBTC to reduce the transmission bits of a secret color image, while still maintaining satisfactory image quality in the secret color image. Then, our sharing procedure transforms the compressed image into several smaller random-like shadows according to a predetermined k. Experiments confirm that our scheme successfully generates smaller random-like shadows without affecting the image quality of the reconstructed secret color image. In addition, the correlation coefficients of the two horizontally and vertically adjacent pixels in each shadow are significantly less than those in the secret color image. The presented scheme is also very sensitive to small changes in the original secret color image because the average NPCR and average AUCI obtained with our scheme are 0.414% and 32.78%, respectively. That means a one-pixel difference could cause a significant difference in each shadow. Therefore, the security of our scheme is also confirmed. Considering that the reconstructed secret color image is a lossy secret color image in the proposed scheme, even compression distortion is slight. Extending the presented scheme to design a secret color image sharing scheme that can reconstruct a lossless secret color will be our future work.

References

[1] R. Ahlswede, I. Csiszar, Common randomness in information theory and cryptography – Part I: secret sharing, IEEE Transactions on Information Theory 39 (August) (1993) 1121–1132.

[2] A. Beimel, B. Chor, Universally ideal secret-sharing schemes, IEEE Transaction on Information Theory 40 (May) (1994) 786–794.

[3] G.R. Blakley, Safeguarding cryptographic keys, in: Proceedings of the National Computer Conference, American Federation of Information Processing Societies Proceedings, New York, USA, 1979, pp. 313–317.
[4] C.C. Chang, R.J. Hwang, Sharing secret images using shadow codebooks, Information Sciences 111 (1–4) (1998) 335–345.
[5] C.C. Chang, W.L. Tai, C.C. Lin, Hiding a secret colour image in two colour images, Imaging Science Journal 53 (4) (2005) 229–240.
[6] C.C. Chang, M.N. Wu, An algorithm for color image compression based on common bit map block truncation coding, Proceedings of Joint Conference on Information Sciences (2002) 964–967.
[7] T.S. Chen, C.C. Chang, M.S. Hwang, A virtual image cryptosystem based on vector quantization, IEEE Transactions on Image Processing 7 (10) (1997) 1485–1488.
[8] C.C. Chen, W.Y. Fu, C.C. Chen, A geometry-based secret image sharing approach, in: Proceedings of Image and Vision Computing New Zealand, Dunedin, Otago, New Zealand, 2005, pp. 428–431.
[9] Y.F. Chen, Y.K. Chan, C.C. Huang, C.C. Tsai, Y.P. Chu, A multiple-level visual secret-sharing scheme without image size expansion, Information Sciences 177 (21) (2007) 4696–4710.
[10] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory IT-22 (6) (1976) 644–654.
[11] W. Diffie, M.E. Hellman, Privacy and authentication: an introduction to cryptography, Proceedings of the IEEE 67 (3) (1979) 397–427.
[12] N.F. Johnson, S. Jajodia, Exploring steganography: seeing the unseen, IEEE Computer 31 (2) (1998) 26–34.
[13] K. Kaya, A.A. Selcuk, Threshold cryptography based on Asmuth–Bloom secret sharing, Information Sciences 177 (19) (2007) 4148–4160.
[14] L.M. Marvel, C.G. Boncelet Jr., C.T. Retter, Spread spectrum image steganography, IEEE Transactions on Image Processing 8 (8) (1999) 1075–1083.
[15] M. Naor, A. Shamir, Visual Cryptography, Advances in Cryptology: Eurocrypt'94, Spring-Verlag, Berlin, 1995, pp. 1–12.
[16] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (2) (1978) 120–126.
[17] A. Shamir, How to share a secret, Communications of ACM 22 (11) (1979) 612–613.
[18] S.J. Shyu, Efficient visual secret sharing scheme for color images, Pattern Recognition 39 (5) (2006) 866–880.
[19] D.R. Stinson, Decomposition constructions for secret-sharing schemes, IEEE Transactions Information Theory 40 (Jan) (1994) 118–124.
[20] D.R. Stinson, Visual cryptography and threshold schemes, IEEE Potentials 18 (Jan) (1999) 13–16.
[21] C.C. Thien, J.C. Lin, Secret image sharing, Computers & Graphics 26 (5) (2002) 765–770.
[22] C.C. Thien, J.C. Lin, An image-sharing method with user-friendly shadow images, IEEE Transactions on Circuits and Systems for Video Technology 12 (12) (2003) 1161–1169.
[23] R.Z. Wang, C.H. Su, Secret image sharing with smaller shadow images, Pattern Recognition Letters 27 (6) (2006) 551–555.
[24] C.N. Yang, T.S. Chen, Aspect ratio invariant visual secret sharing schemes with minimum pixel expansion, Pattern Recognition 26 (2) (2005) 193–206.