Authentication and cross-recovery for multiple images

Yu-Jie Chang
Sian-Jheng Lin
Ja-Chen Lin
National Chiao Tung University
Department of Computer Science
Hsinchu, 300, Taiwan
E-mail: yjchang@cis.nctu.edu.tw

Abstract. *We propose a system with image authentication and cross-recovery ability to protect a group of n given digital images. The system is a* (r,n) *threshold scheme (r is a prespecified threshold satisfying* $2 \leqslant r < n$*). Any r of these images can reconstruct the whole group of n images, but less than r images cannot. Therefore, the system has cross-recovery ability because if some [up to* $(n\text{-}r)$*] images in the group are destroyed or lost in a distributed storage scheme or transmission mission, the destroyed or lost can be rebuilt vividly by the mutual support of r survived members. The design is composed of compression, a two-layer sharing, cryptographic hash function, and information hiding.* © *2008 SPIE and IS&T.*
[DOI: 10.1117/1.2991410]

1 Introduction

Due to the convenience of the Internet, transmission of multiple images via networks has become very popular. However, it is possible for the network connection to be unstable or under hacker attacks. A general remedy is to request the sender end to resend again, but then transmission time has been wasted. Unless the sender immediately handles the request and the network is stable for awhile, the waiting time may seriously influence an important business decision when a cooperative company is eager to know what the new product images look like. Another situation is that in a distributed storage system of multiple images, if some images are lost by the hardware failure or the manager's carelessness, a trivial way to recover them is using back-up copies identical to the lost images but stored elsewhere. Although the back-up copies can increase the reliability of a storage system, it also increases the cost of the system.

As a result, an interesting and important issue becomes how to protect the integrity of multiple images during a transmission or in a storage system. Image authentication is a technique that can verify an image's integrity by inserting a digital watermark into the protected image, or storing its digital signature elsewhere. In recent years, several image authentication methods[1–10] have been proposed. Most of them emphasize the detection of whether malicious manipulations have occurred, and some have the position-locating ability for the tampered parts of the test image. Some approaches[7–10] additionally possess the gorgeous capability of automatic recovery for the tampered parts to a certain extent, after detection and locating work. However, these recovery approaches usually only target a single image rather than a group of images.

We review these approaches for recovery of a single image. Wu and Chang[7] proposed an elegant method based on the JPEG compression technique. They used an edge detection technique to identify the edges of the image before JPEG compression, and then embedded the obtained edge characteristic into some ac coefficients of the frequency domain after JPEG compression. If the image was tampered with, then the embedded edge characteristic could be used to detect the tampered areas and cooperate the interpolation method to reconstruct it. Lin, Hsieh, and Huang[8] proposed an attractive block-based watermarking scheme for image tampering detection, locating, and recovery. They used a parity check and a kind of comparison to generate the watermark of each block, and then added to the watermark the recovery information that recorded the six most significant bits (MSBs) of the mean value of another block. In their verification procedure, a four-layer hierarchical inspection system was used to increase the accuracy of locating the tampered area. Yeh and Lee[9] proposed content-based fragile watermarking to recover tampered areas of an image with JPEG-compressed quality. They categorized each block as smooth, mid-textured, or textured type, and the recovery data of the smooth and mid-textured blocks were replicated to raise the recovery ability. The relationship between a block and its backup block was by an automorphism. Chan and Chang[10] also proposed an efficient method consisting of three techniques: Hamming code, Torus automorphism, and bit rotation. The Hamming code was to generate parity check data as the authentication information and recover burst bit errors that happen during transmission of the digital image. Torus automorphism spread the authentication information around the image. Bit rotation was used to improve the security of the authentication information.

Our goal is to design an image authentication scheme that deals with a group of n images simultaneously instead of a single image, and the recovery of any member image in this group can be done through the mutual support of r

Paper 08019R received Feb. 1, 2008; revised manuscript received Aug. 5, 2008; accepted for publication Aug. 7, 2008; published online Oct. 14, 2008.

1017-9909/2008/17(4)/043007/12/$25.00 © 2008 SPIE and IS&T.

of the n-1 remaining member images, as long as these r images pass the authentication tests. The method uses some sharing techniques.[11]

The rest of the work is organized as follows. Section 2 briefly reviews image sharing. The encoding and decoding (the latter includes verification and recovery) of the proposed method are described in Secs. 3 and 4, respectively. Experimental results are in Sec. 5. The comparison is in Sec. 6, and the summary is in Sec. 7.

2 Review of Sharing

In this section, related works about sharing are briefly reviewed to provide some necessary background for the proposed method. The concept of secret sharing was introduced independently by Shamir[12] and Blakley[13] in 1979. Their (r,n) threshold scheme divides a secret numerical value into n shares, and any r of these shares can reconstruct the secret numerical value, but less than r shares cannot ($r \leqslant n$ is a prespecified threshold. Notably, $r=n$ is allowed in their papers, because they might require that the unveiling of the secret is impossible unless all $r=n$ shares arrive. However, we do not use $r=n$ here. The reason we require $r<n$ is because of our goal: "When some images (at least one image) in a group of n images are destroyed or lost, they can be rebuilt vividly by the mutual support of the r survived members." Several secret sharing methods based on the (r,n) threshold scheme have been proposed.[11,14–21] Among them, Chang and Huang[12] applied the vector quantization technique[22] to encode the secret image, then the generated codebook is shared among the n participants by applying the (r,n) threshold scheme. Thien and Lin[11] proposed an (r,n) scheme for sharing images. In their scheme, a secret image was shared among n participants, and each participant held a generated shadow image whose size was only $1/r$ of that of the secret image. The secret image could be reconstructed if at least r of the n shadow images were received. The smaller size of their shadow images (r times smaller than the shadow images generated by ordinary sharing methods) is an advantage in transmission and storage. They further developed a method[15] that made the shadow images looked like portraits of the original secret image, and thus provided a user-friendly interface to facilitate the management of the shadow images.

As for other variations, Lin and Tsai[16] integrated the (r,n) threshold scheme with a fragile watermarking technique to make their shares have authentication capability to verify the shares' integrity before reconstructing the secret image. Chen and Lin[17] applied a secret image sharing scheme[11] to transmit an image in a progressive way. Wang and Shyu[18] also proposed a scalable secret image sharing scheme with three sharing modes: 1. multisecrets, 2. priority, and 3. progressive, according to the spatial and depth information to increase the potential application for secret image sharing. Extensions of Shamir's masterpiece[12] combined with visual cryptography[23] can be found in Ref. 19. Even the visual cryptograph[23] itself has many extensions. For example, Wu and Chang[24] skillfully employed circular shape of shares to improve the amount of the embedded message in traditional visual cryptography,[23] which uses rectangular shapes, without sacrificing the clarity quality of the stacking result.

Besides the aforementioned spatial-domain methods, Lin and Tsai[20] proposed a frequency domain method to transform the secret image into the frequency domain, and then utilized a sequence of random numbers to record the lower frequency coefficients (the ac values), except the dc value. The dc value of each block is regarded as the secret key and is shared among the n participants by applying the (r,n) threshold scheme.

Because we utilize the sharing polynomials of Thien and Lin's method[11] in our two-layer sharing, below we review Ref. 11 particularly. In Ref. 11, a secret image O containing m pixels is shared by n participants based on Shamir's polynomial (r,n)-threshold scheme[12] with a module base $p=251$. The image O is first transformed into a noisy image Q by permuting pixels according to a secret key. Then, Q is further divided into m/r nonoverlapping sectors so that each sector contains r pixels. Let $q(x)$ be the x'th shadow image and $q_j(x)$ be the j'th pixel in $q(x)$, where $1 \leqslant x \leqslant n$ and $1 \leqslant j \leqslant m/r$. For each sector j, the r coefficients $a_0, a_1, \ldots, a_{r-1}$ of the corresponding polynomial

$$q_j(x) = a_0 + a_1 \times x + \cdots + a_{r-1} \times x^{r-1} (\mathrm{mod}\ p), \tag{1}$$

are used as the gray values of the r pixels of the corresponding sector j in Q. The x'th shadow image $q(x)$ is the collection $\{q_j(x) | j=1,2,\ldots,m/r\}$. Any r of the n shadow images can be utilized to reconstruct Q. For inverse finding of the r coefficients, $a_0, a_1, \ldots, a_{r-1}$ in Eq. (1) only needs r of the n values $\{q_j(1), q_j(2), \ldots, q_j(n)\}$. The detail is omitted.

Notably, Thien and Lin's method[11] used $p=251$, but we use $p=256$ here for 8-bit grayscale images that have a brightness range of 0 to 255. Since 256 is not a prime number, all arithmetic calculations in Ref. 11 are now done in the sense of Galois field GF(256). The change from 251 to 256 is more convenient for sharing digital data such as pixel value, binary bit stream, and index value of vector quantization.

3 Proposed Method (Encoding)

Figure 1(a) shows the flowchart of the encoding procedure, while Fig. 1(b) is for the decoding (verification and recovery) procedure. In Sec. 3, we explain Fig. 1(a) only. Assume that there are n grayscale (8-bit) images in the protected group. Our goal is that after watermarking these n images, if at least r (out of the n) watermarked images survive (must pass the authentication test) in a transmission or disk crash, where r is a prespecified threshold and $2 \leqslant r < n$, all nonsurvived images can be recovered to a certain extent. On the other hand, if less than r watermarked images passes the authentication test, then the recovery work for the remaining images fails. In conclusion, our method can tolerate the loss or modification of up to $(n\text{-}r)$ watermarked images.

The encoding procedure consists of three parts. First, create the rough image by setting all t least significant bits (LSBs) of each pixel of the original image to zero. To reduce the amount of recovery data, encode each of the n rough images respectively by the JPEG2000 compression technique to generate n bit streams. These n bit streams are treated as the recovery data. Then, share these n bit streams

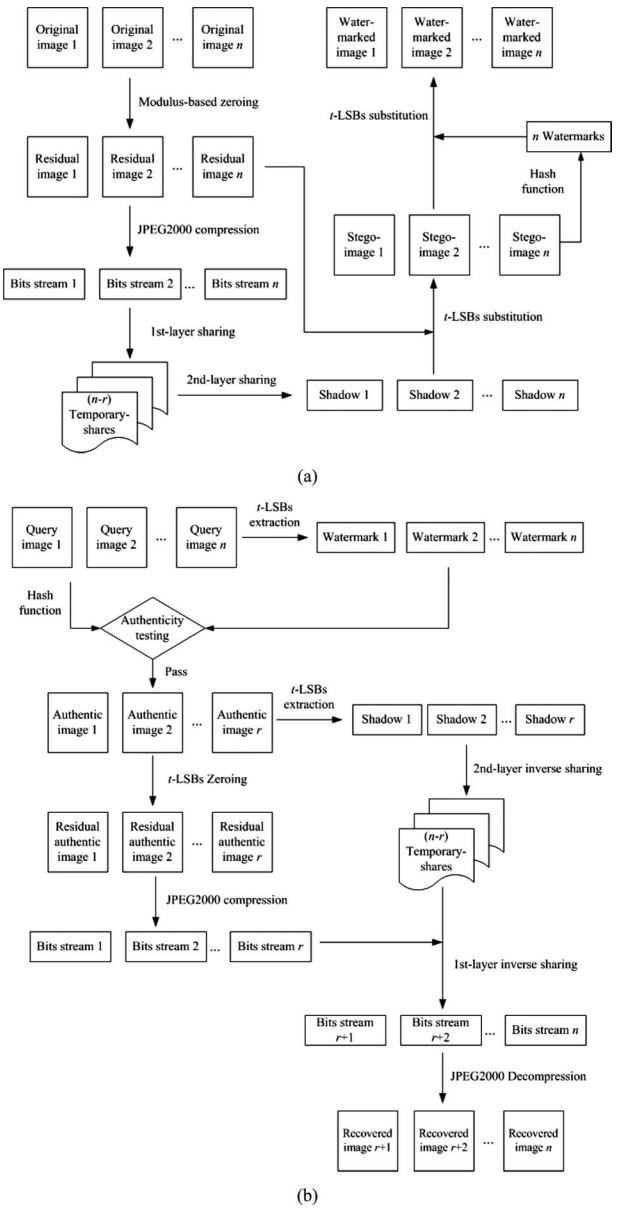

Fig. 1 Two flowcharts: (a) is the encoding procedure and (b) is the authentication and recovery procedure.

by means of a two-layer sharing method to generate n shadows. Use a module method, modified from the simple LSB substitution method, to embed these n shadows in the n original images of the group, respectively, to form n stego-images. Finally, generate the authentication data (a 128-bits watermark) for each stego-image by using a cryptographic hash function, in which the input includes the local and interrelated information of the stego-image. The authentication data are embedded into a certain space of the stego-image to form the watermarked image. The watermarked images are the final results, which not only look like the original images but also contain the authentication and cross-recovery data needed to protect the transmission and storage of the group. Details of the encoding are described in the following three sections accordingly.

3.1 *Generation of the Recovery Data*

In the proposed method, the recovery data of each image is generated first. As a remark, all the bit values of the t LSBs at each pixel of the original image will be completely destroyed (erased) to hide the recovery and authentication data. Therefore, after the hiding procedure, only the remaining $(8\text{-}t)$ most significant bits (MSBs) of the original image are directly related to the local gray values. Hence, the rough image, which is a $2^{(8-t)}$-level image formed of the $(8\text{-}t)$ MSB bit-planes, of each original image is the input data for generating the corresponding recovery data.

Furthermore, a small skill used in our scheme is that the rough image is obtained by using the modulus operation rather than directly setting all t LSBs of each pixel of the original image to zero. (The modulus-based zeroing scheme here is inspired by Thien and Lin's modulus-based hiding method.[25] Reference 25 has proven that when doing hiding, the gray value distortion of the simple LSB substitution method is never smaller than that caused by modulus-based hiding.)

Our modulus-based zeroing scheme is as follows. To set t LSBs of a pixel value y $(0 \leqslant y \leqslant 255)$ of an 8-bit grayscale image to zero, the new value $\hat{y}$ of our rough image can be expressed as

$$\hat{y} = 2^t \times \text{rounding}\left(\frac{y}{2^t}\right), \tag{2}$$

where the rounding$(\cdot)$ operator means rounding its content to the nearest integer. This rounding operator is better than the truncation operator, which directly truncates the noninteger part; and the latter is exactly the one used in a simple LSB method. Of course, we need to check whether or not the $\hat{y}$ falls in the valid gray-value range $0 \leqslant \hat{y} \leqslant 255$. If it is out of the range, then $\hat{y}$ should be corrected further as

$$\hat{y} = \begin{cases} \hat{y} + 2^t & \text{if } \hat{y} < 0 \\ \hat{y} - 2^t & \text{if } \hat{y} > 255 \end{cases}. \tag{3}$$

Now, the rough image is an image whose gray values are $\hat{y}$, of which 2^t is a factor, rather than the old pixel values y of the original image.

For example, assume that a rough image of zeroing 2-LSB (i.e., $t=2$) is desired, and the gray value y of a pixel in the original image is $107=(01101011)_2$. Instead of using $104=(01101\underline{\mathbf{00}})_2$, which is the result of directly erasing the 2 LSBs by zeros, we use the modulus-based zeroing formula in Eq. (2), i.e., the value of the corresponding pixel in our rough image is $2^2 \times \text{rounding}(107/2^2)=108=(01101\underline{\mathbf{1}}\mathbf{00})_2$. Obviously, the distortion $(108-107=1)$ of using Eq. (2) is smaller than the distortion $(107-104=3)$ of using direct truncation. Before going further, we remark here on the influence of the value of t. The bigger the parameter value of t, the larger the hiding capacity of each image (to embed the recovery and authentication data into the space provided by t bit planes) and, hence, the better the quality of the recovered image. However, the quality of the watermarked image will become worse as the value of t increases, for the final t bits have been changed at each pixel. So, there is a tradeoff.

Next we discuss the design of the recovery data. In general, the data amount of a digital image is large, so it is difficult to embed all information of a protected image in quite a limited hiding space. For this reason, we first use a compression technique to reduce the data amount, so that it is easier to perform the embedding procedure later. JPEG is a very common image compression standard on the Internet. In contrast to JPEG, JPEG2000 is a new international standard for image compression that is advantageous over JPEG in terms of compression ratio, freedom of compression mode selection, and adaptation to different kinds of images. Therefore, JPEG2000 is employed as the image compression technique in our method to generate the recovery data. As shown in Fig. 1(a), after modulus-base zeroing, n rough images of the group are compressed by the JPEG2000 compression encoder, and the compression result of each rough image is a bit stream. The n bit streams are the input of the first-layer sharing introduced in Sec. 3.2. As a rule, the bit stream of each rough image is treated as the recovery data for that image, i.e., the decompressed image obtained from the corresponding bit stream will be used as the recovered image to replace a watermarked image, should the watermarked image be tampered with or lost.

3.2 *Two-Layer Sharing*

After generating the recovery data, the next thing to do is to share it. We design next a two-layer sharing scheme to share the information needed in recovery.

3.2.1 *First-layer sharing*

As shown in Fig. 1(a), the recovery data (the n compressed bit streams of the rough images) are shared in the first-layer sharing process to generate the $(n\text{-}r)$ temporary shares. The process is explained next.

Assume that each bit stream has m bits. Divide each stream into $m/8$ nonoverlapping cells so that each cell contains 8 bits. Then, each cell is converted from base 2 to base 10 (a decimal number ranges between 0 and 255); in other words, each cell value is a decimal value in the range of commonly seen gray values.

For each cell j, where $1 \leqslant j \leqslant m/8$, let $\{A_1, A_2, \ldots, A_n\}$ be, respectively, the n decimal values of the j'th cell in the n bit streams. Then, let the first-layer sharing polynomial (for the j'th cell) be

$$p_j(x) = [A_1 \times (c_1)^x + A_2 \times (c_2)^x + \cdots + A_n \times (c_n)^x] \pmod{256}, \quad (4)$$

where the arbitrarily chosen positive integer constants $\{c_1, c_2, \ldots, c_n\}$ satisfy $c_i \neq c_k$, for all $i \neq k$. For example, let $c_1=1$, $c_2=2, \ldots, c_n=n$.

For each integer $x \in \{1, 2, \cdots, (n-r)\}$, let

$$p(x) = [p_1(x), p_2(x), \ldots, p_{m/8}(x)] \quad (5)$$

denote the x'th temporary share, in which each $p_j(x)$ is just the j'th cell value of $p(x)$. Since each cell value $p_j(x)$ is still an 8-bit value by Eq. (4), each temporary share $p(x)$ can be treated as an image; and it has $8 \times (m/8) = m$ bits, just like the length of each bit stream.

After first-layer sharing, there are $(n\text{-}r)$ temporary shares. Because the tampered or lost parts of some bit streams can be reconstructed by the cooperation between the $(n\text{-}r)$ temporary shares and the r survived bit streams (the details are illustrated in Sec. 4.2), the $(n\text{-}r)$ temporary shares are the key data for the recovery purpose. To ensure the survival of these $(n\text{-}r)$ temporary shares, they are shared again among the n images.

Notably, the original recovery data (n bit streams), which keeps the main information about the n original images, are condensed to the $(n\text{-}r)$ temporary shares by the first-layer sharing. In other words, the first-layer sharing not only contributes to the recovery work, but also decreases further the total amount of the recovery data, for there are $(n\text{-}r)$ temporary shares of m bits each, which are more economic (in total size) than the n bit strings of m bits each.

3.2.2 *Second-layer sharing*

In second-layer sharing, we use Thien and Lin's (r,n) threshold sharing method[11] to create n shadows, which share the $(n\text{-}r)$ temporary shares just generated from the first layer. The detail is as follows. For each temporary share $p(x)=[p_1(x), p_2(x), \ldots, p_{m/8}(x)]$, divide it into sectors of r pixels each. [Recall from Eq. (4) that each element $p_j(x)$ has 8 bits and, hence, can be treated as a pixel.] Then, for each sector, use its r pixels as r coefficients in Eq. (1), and then plug in n prespecified integer values for the variable x. This creates n transformed values for a sector. After doing this for all sectors of a temporary share i, and collecting the transformed values for each x, the temporary share i is transformed to n shadows. As i runs from 1 through $(n\text{-}r)$, each temporary share gets its own n shadows. Then, the first shadows of all $(n\text{-}r)$ temporary shares are concatenated, and the result is still called shadow 1. Similarly, second shadows of all $(n\text{-}r)$ temporary shares are concatenated, and the result is still called shadow 2. As the process goes on, the concatenation sequentially yields shadows $1, 2, \ldots, n$.

As shown in Fig. 1(a), after the aforementioned two-layer sharing phase, the recovery data of the n original images in the group are condensed to n shadows. To have the ability to later recover any polluted image by using the remaining images of the group, we may "hide" (see Refs. 26–30 for examples of hiding techniques) the n obtained shadows, respectively, into the n rough images of the protected group. Recall that the t least significant bits of each rough image have been zeros, so we may use these t least significant bits to embed shadow (embed one shadow in one rough image). This hiding technique is the famous t-LSB substitution method. Now, n stego-images are thus generated, which still look like their original images. The reason we use t-LSB substitution for hiding is because it is simple, fast, and with great hiding capacity without downgrading too much the quality of the stego-images. Notably, in the hiding step here, we use t-LSB substitution rather than the modulus-based substitution,[25] because we do not want to change the $(8\text{-}t)$ MSBs of the rough images when creating stego-images. This ensures the $(8\text{-}t)$ MSBs are the same between each rough image and its stego-image. Therefore, in later days, when a lost or tampered stego-

image is to be recovered, we can grab the rough images obtained from the (8-t) MSBs of nontampered stego-images, then do JPG2000 compression, and thus obtain recovery data [see Figs. 1(a) and 1(b)].

3.3 *Generation of the Authentication Data*

After the hiding procedure, the n original images of the protected group are transformed to the n stego-images. Later, to determine whether a stego-image of the protected group is tampered or not, people can use image authentication to verify the integrity of each stego-image. Next we describe the details of the generation of our authentication code (the watermark).

As the final work for the encoding procedure, the watermark to verify the stego-image's integrity is generated, and it is then embedded in stego-images to obtain the watermarked image. In our method, we use a cryptographic hash function to generate the watermark of each stego-image. It consists of some important information of the stego-image, including width, length, the image's identification number, and the pixel values of the stego-image. Consider a cryptographic hash function

$$H(S) = (b_1, b_2, \ldots, b_u), \tag{6}$$

where S is an input bit string of arbitrary length, b_i is the output binary bits of the hash function for $1 \leqslant i \leqslant u$, and u is the length of the output bit string. Wong and Memon[6] suggest that a cryptographic hash function should have the property that once an input bit string S and its corresponding output $(b_1, b_2, \ldots, b_u)$ are given, then it is almost computationally infeasible to find another input bit string that will be hashed to the same output $(b_1, b_2, \ldots, b_u)$. Notably, MD5[31] is a famous collision-resistant one-way hash function, in which any input bit string is hashed into a bit stream of 128 bits, i.e., $u=128$. In our method, MD5 is employed as the hash function. Of course, using other cryptographic hash functions to replace MD5 is also allowable.

Let ID be the image's identification number of the original image in the protected group. Now, for each stego-image whose size is $h \times w$, the authentication data can be computed as

$$H(ID \| h \| w \| p_1 \| p_2 \| \cdots \| p_{h\times w-128} \| \hat{p}_{h\times w-127} \| \cdots \| \hat{p}_{h\times w}) = (b_1, b_2, \ldots, b_u), \tag{7}$$

where the symbol $\|$ is the concatenation of all input streams, and $p_1, p_2, \ldots, p_{h\times w-128}$ are the stego-image pixels' gray values according to the raster-scan order of the stego-image, which is from left to right and top to bottom. Because the least-significant bit of the final 128 pixels $\hat{p}_{h\times w-127}, \hat{p}_{h\times w-126}, \cdots, \hat{p}_{h\times w}$ of each stego-image are particularly reserved for embedding its 128-bits authentication data, their pixel values $\hat{p}_{h\times w-127}, \hat{p}_{h\times w-126}, \cdots, \hat{p}_{h\times w}$ are computed just using the $8-1=7$ most significant bits (MSBs) only. Finally, the generated 128-bits watermark is embedded in the LSB of the final 128 pixels of the corresponding stego-image to form the watermarked image. These watermarked images are used in transmission or storage.

The version in the previous paragraph, which sequentially hides the 128-bits authentication data (watermark) in the LSB of the last 128 pixels of a stego-image, is just the simplest version. For security concerns, we can use a pseudorandom number generator to randomize the embedding locations of the watermark in the stego-image. More specifically, assume that the pixels of the stego-image are numbered sequentially from 0 to $L-1$ according to the raster scan order, which is from left to right and top to bottom. In other words, L is the total number of the pixels in the stego-image (i.e., L is 262, 144 for a 512×512 image). By a pseudorandom number generator, we can generate a sequence of random integer numbers. Then, we can embed the watermark into the stego-image according to the generated sequence one by one. For example, if the generated sequence is {214,257, 133,785, 34,480, 9284,…}, the first bit of the watermark is embedded into the LSB of the pixel number 214,257 in the stego-image, and the second bit is embedded into the LSB of the pixel number 214,257, and so on. Therefore, the pseudorandom sequence is used to increase the difficulty in getting the watermark for an unauthorized user. In our method, the Mersenne Twister (MT) pseudorandom number generator[32] is employed as the permutation function of the watermark, and its seed can be regarded as a secret key. In the image-disclosure site, only the authorized person who owns the same secret key can obtain the same sequence of pseudorandom integer numbers to extract the watermark in the verification phase.

4 Proposed Method (Decoding)

This section introduces the decoding procedure. Figure 1(b) shows the flowchart for decoding. The detail subprocedures for verifying and recovering n query images are as follows.

4.1 *Verification*

When a user receives some of the n query images from the protected group, the first thing to do is to verify the integrity of each query image. This is because the pervasive and powerful image manipulation tools now have made the imperceptible modification of images become very easy, and a user usually cannot determine whether the query image is tampered with or not by using the naked eye. According to the final paragraph of Sec. 3.3, a legal user can have the secret key, which is the seed of the MT pseudorandom number generator,[32] to obtain the embedding locations of the hidden watermark.

For each query image, its extracted watermark should be the same as the authentication data recomputed directly according to the hash function described in the paragraph containing Eq. (7). If the extracted watermark coincides with the recomputed watermark, then the query image is called an authentic image; otherwise, it is regarded as a tampered image. An authentic image means that it is extremely likely that no tampering occurs in this image, and the recovery data embedded in this image are therefore considered trustworthy in joining the reconstruction team to recover some tampered images. Note that if an image fails to pass the authentication test, then the recovery information stored in it should never be used.

4.2 *Cross-Recovery of Tampered Images Through the Cooperation of Authentic Images*

After checking the authentication status of all the query images, the recovery phase starts if there is at least one

tampered or lost image; and simultaneously, there exist at least r authentic images. To recover a tampered or lost image, we may collect its recovery data from any r authentic images. The recovery data were embedded earlier in the encoding phase in the t LSBs of all n images (one shadow per image, as stated in Sec. 3.2). When at least r (out of the n) images are authentic, we can extract r shadows from the t LSBs of the r authentic images. Then, by doing the inverse of the second-layer sharing, we can use these r shadows to extract back the $(n\text{-}r)$ temporary shares, which are the result of the first-layer sharing. (The detail steps of inverse sharing are similar to those in Sec. 3.2 of Ref. 11.)

Meanwhile, the r authentic stego-images can replace all their t LSBs by zeros to obtain r rough images. Then use JPEG2000 compression (as described in Sec. 3.1 and Fig. 1) to obtain r bit streams. Because the r authentic stego-images are verified to have not been tampered with, the r rough images obtained now in the decoding phase, and hence the r bit streams obtained now are identical to those obtained in the encoding phase.

Then, the r bit streams are combined with the $(n\text{-}r)$ temporary shares obtained from the aforementioned second-layer inverse sharing, to rebuild other $(n\text{-}r)$ bit streams by solving n coefficients $(A_1, A_2, \ldots, A_n)$ of Eq. (4).

Notably, these $(n\text{-}r)$ bit streams are the compression result corresponding to those $(n\text{-}r)$ tampered or lost images, so we may decompress these $(n\text{-}r)$ bit streams to recover the $(n\text{-}r)$ tampered or lost images in lower resolution.

In Lemma, we prove that the $(n\text{-}r)$ bit streams could be recovered successfully through the cooperation of the r (out of the n) bit streams and the $(n\text{-}r)$ temporary shares.

Lemma. If any r (out of the n) bit streams and $(n\text{-}r)$ temporary shares are obtained, then the absent $(n\text{-}r)$ bit streams can be recovered successfully.

Proof. In first-layer sharing, the $(n\text{-}r)$ temporary shares $p(1), p(2), \ldots, p(n-r)$ are generated by Eq. (4). For each cell j, the following shows the relation between the coefficients $\{A_1, \ldots, A_n\}$ of the n bit streams and temporary share values $\{p_j(x): x=1,2,\ldots,(n\text{-}r)\}$. For convenience, we dropped the subscript j in this proof.

$$\begin{bmatrix} c_1^1 & c_2^1 & c_3^1 & \cdots & c_n^1 \\ c_1^2 & c_2^2 & c_3^2 & \cdots & c_n^2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ c_1^{n-r} & c_2^{n-r} & c_3^{n-r} & \cdots & c_n^{n-r} \end{bmatrix} \cdot \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(2) \\ \vdots \\ p(n-r) \end{bmatrix}. \tag{8}$$

Without the loss of generality, assume that the obtained r (out of the n) bit streams are $A_1, A_2, \ldots, A_r$, and the tampered or lost $(n\text{-}r)$ bit streams that need to be reconstructed are $A_{r+1}, A_{r+2}, \ldots, A_n$. Then, Eq. (8) can be rewritten as

$$\begin{bmatrix} c_1^1 & c_2^1 & \cdots & c_r^1 \\ c_1^2 & c_2^2 & \cdots & c_2^2 \\ \vdots & \vdots & \cdots & \vdots \\ c_1^{n-r} & c_2^{n-r} & \cdots & c_r^{n-r} \end{bmatrix} \cdot \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_r \end{bmatrix} + \begin{bmatrix} c_{r+1}^1 & c_{r+1}^1 & \cdots & c_n^1 \\ c_{r+1}^2 & c_{r+2}^2 & \cdots & c_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ c_{r+1}^{n-r} & c_{r+2}^{n-r} & \cdots & c_n^{n-r} \end{bmatrix} \cdot \begin{bmatrix} A_{r+1} \\ A_{r+2} \\ \vdots \\ An \end{bmatrix} = \begin{bmatrix} p(1) \\ p(1) \\ \vdots \\ p(n-r) \end{bmatrix}. \tag{9}$$

Equation (9) can be expressed as the abbreviation

$$\mathbf{C}_k \cdot \vec{\mathbf{A}}_k + \mathbf{C}_u \cdot \vec{\mathbf{A}}_u = \vec{\mathbf{P}}. \tag{10}$$

Equation (10) therefore becomes

$$\mathbf{C}_u \cdot \vec{\mathbf{A}}_u = \vec{\mathbf{P}} - \mathbf{C}_k \cdot \vec{\mathbf{A}}_k. \tag{11}$$

From Eq. (9), because $c_i \neq c_k \neq 0$, if $i \neq k$, we can see that

$$\mathbf{C}_u^T = \begin{bmatrix} c_{r+1}^1 & c_{r+1}^2 & \cdots & c_{r+1}^{n-r} \\ c_{r+2}^1 & c_{r+2}^2 & \cdots & c_{r+2}^{n-r} \\ \vdots & \vdots & \cdots & \vdots \\ c_n^1 & c_n^2 & \cdots & c_n^{n-r} \end{bmatrix},$$

and obviously, it is a Vander Monde matrix.

Note that for any Vander Monde matrix

$$\mathbf{V} = \begin{bmatrix} 1 & e_1 & e_1^2 & \cdots & e_1^{h-1} \\ 1 & e_2 & e_2^2 & \cdots & e_2^{h-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & e_h & e_h^2 & \cdots & e_h^{h-1} \end{bmatrix}, \tag{12}$$

the determinant will be

$$\det(\mathbf{V}) = \prod_{l<m} (e_m - e_l), \tag{13}$$

and hence nonzero as long as all e_i are distinct. Because matrix $\mathbf{C}_u^T$ is the form of the Vander Monde matrix, det $(\mathbf{C}_u^T)$ is nonzero, that is, $\det(\mathbf{C}_u)$ is nonzero. It means that $\mathbf{C}_u$ has the inverse matrix $\mathbf{C}_u^{-1}$. Therefore, Eq. (11) can be rewritten as

$$\vec{\mathbf{A}}_u = \mathbf{C}_u^{-1}(\vec{\mathbf{P}} - \mathbf{C}_k \cdot \vec{\mathbf{A}}_k). \tag{14}$$

Because the r (out of the n) bit streams $\vec{\mathbf{A}}_k = (A_1, \ldots, A_r)$ are known, and the $(n\text{-}r)$ temporary shares $\vec{\mathbf{P}} = \{p(1), \ldots, p(n\text{-}r)\}$ are also known, $(\vec{\mathbf{P}} - \mathbf{C}_k \cdot \vec{\mathbf{A}}_k)$ can be calculated. By the inverse operation of matrix $\mathbf{C}_u$, the absent $(n\text{-}r)$ bit streams $\vec{\mathbf{A}}_u = (A_{r+1}, \ldots, A_n)$ can be recovered easily by the multiplication operation between $\mathbf{C}_u^{-1}$ and $(\vec{\mathbf{P}} - \mathbf{C}_k \cdot \vec{\mathbf{A}}_k)$.

Fig. 2 A group of test images: (a) Baboon, (b) Lena, (c) Jet, and (d) Scene.

Fig. 3 The watermarked image of Fig. 2: (a) Baboon (PSNR =44.17 dB), (b) Lena (PSNR=44.14 dB), (c) Jet (PSNR =44.16 dB), and (d) Scene (PSNR=44.15 dB).

5 Experimental Results and Robustness-Related Issues

5.1 *Experimental Results*

Experimental results are presented to demonstrate the performance and feasibility of the proposed method. A group of 512×512 standard 8-bit gray-scaled images (Baboon, Lena, Jet, Scene) shown in Fig. 2 are used as the test images in the experiments.

In the experiments, we used a $(r=2, n=4)$ threshold scheme in the two-layer sharing procedure to share the recovery data, i.e., the group of four images could survive even if $(n\text{-}r=4-2)$ of the watermarked images was lost or tampered. The parameter setting for the t-LSBs is $t=2$, i.e., the watermark is embedded into the 2 LSBs of the images' pixels. Figure 3 shows the watermarked images of Fig. 2 using the proposed method described in Sec. 3. The qualities of all watermarked images and recovered images are measured by the peak signal-to-noise ratio (PSNR) defined as

$$\mathrm{PSNR} = 10 \times \log_{10} \frac{255^2}{\mathrm{MSE}}, \tag{15}$$

in which the MSE denotes the mean square error between the pixel values of the original and watermarked/recovered images. From Fig. 3, we can see that the qualities of the watermarked images are acceptable (their PSNR values range between 44.14 and 44.17 dB). It is visually indistinguishable between Figs. 2 and 3 using the naked eye. In other words, our watermarked images have the transparency property for the hidden data, i.e., the recovery data and watermark are perceptually invisible. To inspect our scheme's recovery ability, some of the watermarked images shown in Fig. 3 are lost or tampered with in the following experiments.

5.1.1 *Case 1: one watermarked image is lost*

When the watermarked image "Lena" [Fig. 3(b)] is lost due to transmission error or hardware storage failure, the remaining images are Baboon, Jet, and Scene, as shown in Figs. 4(a)–4(c). After the verification procedure described

Fig. 4 The experiment result when the watermarked image Lena is lost. (a), (b), and (c) are the survived 44.1 dB images that passed the authentication test, and (d) is the recovered image Lena' (PSNR=41.57 dB) saved from using (a), (b), and (c).

Fig. 5 Some alternations of Fig. 3: (a) the watermarked image Baboon is rotated through 90 deg; (b) the watermarked image Lena filtered by a low-pass filter; (c) the watermarked image Jet is cropped; (d) noise is uniformly added to the watermarked image Scene; (e) the watermarked image Jet is shrunk to a quarter of its size; and (f) the watermarked image Scene is completely replaced by another image Boat.

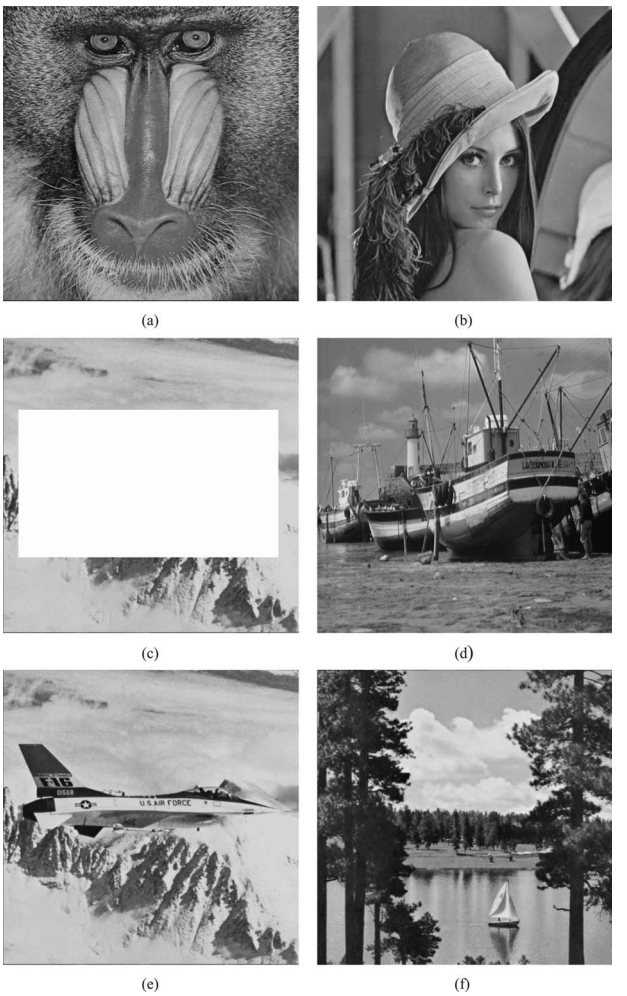

Fig. 6 An experiment when two images are tampered with. (a) and (b) are the watermarked images [Figs. 3(a) and 3(b)] that pass the authentication test; (c) and (d) fail the test, where (c) is the cropped watermarked image Jet, and (d) is when the watermarked image Scene is completely replaced by another image Boat. Finally, using (a) and (b), the two recovered images are the image Jet' (PSNR =42.54 dB) shown in (e), and the image Scene' (PSNR =38.32 dB) shown in (f).

in Sec. 4.1, these three images are authentic, and thus participate in the recovery team to save the lost member Lena. Using the support from the three authentic images Baboon, Jet, and Scene, we recover the Lena′ shown in Fig. 4(d), whose PSNR value is 41.57 dB.

5.1.2 *Case 2: two watermarked images are tampered with*

In the experiment, we consider the situation when (any) two of the four watermarked images shown in Fig. 3 are tampered with by manipulations or processing such as rotation, filtering, cropping, noise addition, resizing, and replacement (see Fig. 5). Because the authentication code was generated by a hash function using the pixel values of the stego-image, the image's width and length, and the image's identification number as the input information (see Sec. 3.3), any manipulation of pixel values, image size, or image *ID* would cause the input image to fail our verification test. So, these images shown in Figs. 5(a)–5(f) are judged as tampered images by the authentication test. Therefore, the recovery phase begins, and the two tampered members of the protected group are successfully reconstructed using the remaining two authentic images (all images in Fig. 3 can pass the authentic test, so any two from Fig. 3 can achieve this).

One such example is shown in Fig. 6, in which Figs. 3(c) and 3(d) are tampered with and become Figs. 6(c) and 6(d). Their recovery versions become Jet' and Scene' in Figs. 6(e) and 6(f). As for the other examples of (r=2, n =4), each time two of the four images in Fig. 3 are replaced by two of the six images shown in Fig. 5. The two images identical to the images shown in Fig. 3 can pass the authentication test and thus need no recovery. Moreover, after passing the test, they will back-up the recovery of the two images missing from Fig. 3. No matter which two are tampered or missing from Fig. 3, the recovered versions are always identical to two of the four images shown in Fig. 7.

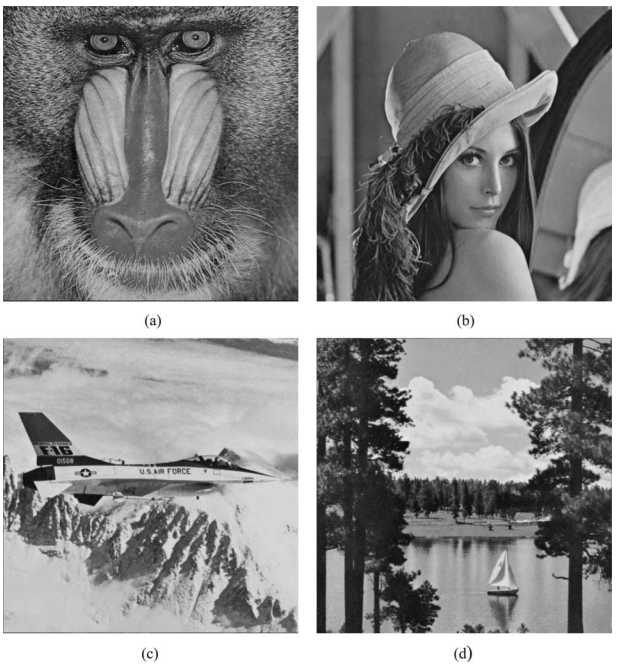

Fig. 7 The recovered versions (33.82-dB Baboon', and/or 41.57-dB Lena', and/or 42.54-dB Jet', and/or 38.32-dB Scene') of the two lost or tampered images in a sequence of (r=2, n=4) experiments. In each experiment, only two (=r) of the received images are exactly the watermarked images shown in Fig. 3 (and hence can pass the authentic test). The remaining n-r=4−2=2 images are either lost or tampered with (for example, they can be any two images in Fig. 5).

5.2 *Robustness-Related Issues*

Now we discuss the robustness of the watermarked images. There are two parts of information carried in each watermarked image: 1. watermark (i.e., the authentication code), and 2. recovery data. Their robustness are discussed respectively next.

1. The authentication code is powerful enough so that if a received image R is not exactly (100%) a product of our method (a watermarked image of ours), then the received image R is rejected immediately, no matter if the difference between R and our product is small or big, and also no matter if the difference is due to noise addition, filtering, resizing, rotation, cropping, replacement, hacker attack, etc. This is shown in the experiment (related to Figs. 6 and 7). The reason we require the authentication code to be so sensitive is that we want to make sure the recovery data that we grab from the received image R is 100% exact, avoiding confusion in the later step of helping the recovery of other broken images $\{R_2, R_3, \ldots\}$. (For example, if the recovery of R_2 from r=3 images $\{A, B, C\}$ are distinct from $\{A, B, R\}$, or $\{B, C, R\}$, or $\{A, C, R\}$, then there are some other troubles to judge which recovery of R_2 is to be believed.

2. As a result of this, after the authentication test, if a received image R cannot pass the authentication test, then we discard all information hidden in or carried by image R, because we do not think this information is trustworthy. Therefore, in some sense, we might say that the recovery data are not robust against any change of the watermarked image.

3. The discussions in 1. and 2. are from logic sense. As for the technique sense, notably, the recovery data are usually much larger in size than the authentication code. Hiding much larger sized data and making it robust is more difficult than hiding much smaller sized data and making it robust. This is because the robustness process against every kind of manipulation or attack often enlarges the hidden data. If the original data to be hidden are small, such as the authentication code, then robust hiding is less difficult. But if the original data to be hidden is already large, such as the cross-recovery data of multiple images, then this is difficult.

6 Discussion

Table 1 compares our method with Refs. 4–10 and 33. Among them, Refs. 4–6 are for authentication only. References 7–10 can have both authentication and self-recovery ability of a "single" image. Reference 33 is the only one (other than ours) dealing with multiple images. So we introduce and compare with Ref. 33 in more detail in the following paragraphs. Before that, let us take a look at single-image recovery methods. Usually, for single-image methods, their schemes can recover the tampered parts of the protected image by using the recovery data, which is often embedded in blocks of other areas of the same image. However, when a watermarked image is tampered extensively in a large area and randomly, it is not rare that a block and its back-up block are tampered with at the same time. In this case, the recovery ability of the tampered block in their approaches is gone, but our scheme can handle this case, even if the whole image is lost, as long as the remaining r members are authentic.

Next we compare our method with Ref. 33 in which Tsai, Chang, and Chen proposed a method whose goal is somehow related to ours. Their goal is to share multiple secret images among a group of cover images, so that each pair of stego-images can recover a unique secret image for that pair [see Fig. 8(a)]. In Fig. 8(a), the six secret images {Jet, Goldhill, Girl, Toys, Boat, and Scene} with size of 200×200 pixels are shared in the four cover images {Lena, Baboon, Tiffany, and Zelda} with size of 600×600 pixels. The secret image Jet is shared between the pair of stego-images Lena and Baboon; the secret image Boat is shared between the pair of stego-images Lena and Tiffany, and so on. The PSNR values of their stego-images range between 42.41 and 42.61 dB.

Reference 33 has two main advantages. First, it is effective. By an ingenious design of the sharing order and an adequate utilization of bit planes of the cover image, one can share multiple secret images with multiple cover images, and the quality of all stego-images is visually acceptable to cover communication. Second, it is efficient, because the stego-images are generated using simple operators such as addition and exclusive-or.

However, according to the experiment done in Ref. 33 [shown in our Fig. 8(a)], when one of the stego-images is tampered with or lost, three of the multiple secret images cannot be reconstructed and then lost forever. For example, if the stego-image Lena is tampered with or lost, the secret images Jet, Boat, and Toys cannot be recovered by the remaining three stego-images. Therefore, the method of Ref. 33 is not fault tolerant when some stego-images are

Table 1 A comparison between some published methods and our scheme. (* means "quoted directly from the reported paper," and N/A means not mentioned in the reported paper.) The unit of PSNR is decibels.

Schemes	PSNR of the watermarked image (images, if Ref. 33 or ours)	Authentication check ability	Recovery ability	PSNR of the recovered image (images, if Ref. 33 or ours)
Ref. 4	49.19* (Jet 512×512)	Yes	No	No
Ref. 5	N/A	Yes	No	No
Ref. 6	N/A	Yes	No	No
Ref. 7	34.34* (Lena 512×512)	Yes	Self (single image)	N/A
Ref. 8	44.37* (Beach 256×256)	Yes	Self (single image)	30.85* to 48.48*
Ref. 9	44* (Jet 512×512)	Yes	Self (single image)	32.82* to 44*
Ref. 10	N/A	Yes	Self (single image)	N/A
Ref. 33	42.41 to 42.61 (four 600×600 images)	No (but it can be modified and become "yes")	Cross (Multiple images)	27.92 to 39.63 for Fig. 8(b)
Ours	44.14 to 44.17 (four 512×512 images)	Yes	Cross (Multiple images)	33.82 to 42.54

polluted. As a contrast, our recovery still works even if $(n\text{-}r)$ of the n watermarked images are tampered with or lost. All member images of the protected group can be recovered by our method.

In summary, each method has its own advantage. Reference 33 is effective and efficient, and ours is fault tolerant. Reference 33 is for sharing multiple secrets, and ours is for easy recovery of distributed storage systems in unstable environments.

To compare with Ref. 33 further, we use a structure shown in Fig. 8(b) so that Ref. 33 can also be used for cross-recovery of multiple images. Here, the hidden secret images are in fact the JPEG2000-compressed images of the four cover images. Then we generate the authentication code of each stego-image using the skill aforementioned in Sec. 3.3, and finally embed the code in stego-images to form four watermarked images in Fig. 8(b).

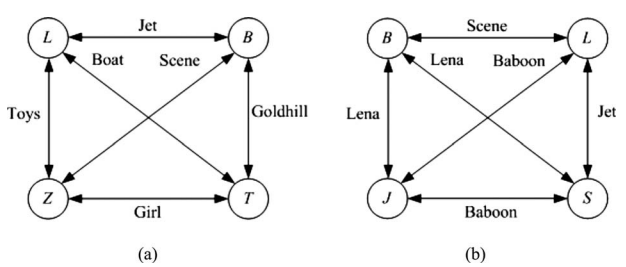

Fig. 8 (a) The relationships among four cover images {L, Z, B, T} in the original experiment of Ref. 33. (b) The relationships among four cover images {B, L, J, S} in a new version slightly modified from Ref. 33 (it is an application version of Ref. 33, the hidden images being the JPEG2000 version of the cover images. *L* means Lena, *B* means Baboon, etc.).

In Fig. 8(b), if one of the four watermarked images is lost or tampered with, then the remaining three members can reconstruct the lost image. For example, if Baboon is lost, we can reconstruct it by the cooperation between Lena and Jet or Jet and Scene. Of course, the recovered version of the lost image is its JPEG2000-compressed version. However, when two of the four watermarked images are lost, only one of the lost images can be reconstructed. For example, if Jet and Scene are lost, then the JPEG2000-compressed Scene can be reconstructed by the cooperation between Baboon and Lena, but the image Jet is gone. To the contrary, our (2, 4) scheme can tolerate two tampered or lost images, for they can be recovered by the cooperation of the remaining two authentic images (see Case 2 of Sec. 5).

Table 2 lists the comparison between our $(r=2, n=4)$ scheme and Ref. 33. From Table 2, we can see that the qualities of the watermarked and recovered images in our scheme are not worse than that of Ref. 33. Moreover, in the recovery procedure, the fault-tolerant ability of our scheme is better than the method of Ref. 33. As for processing speed, Ref. 33 is faster than ours, because Ref. 33 uses logic operations rather than arithmetical computations for their sharing process.

7 Conclusions

We propose an authentication and cross-recovery method for a group of n images. The goal of this work is that if some images in the group are tampered with or lost, these

Table 2 Comparison between two recovery methods for multi-images: 1. Ref. 33 using the structure shown in Fig. 8(b); and 2. our scheme with (r=2, n=4) threshold. (The unit of PSNR is the decibel.)

Scheme	PSNR of the watermarked images	PSNR of the recovered images	Number of images allowed to be lost in the recovery procedure	Number of cover or watermarked images	Purpose of the design in the original paper	Features
Ref. 33	42.41 to 42.61	27.92 to 39.63 for Fig. 8(b).	None, if cover and hidden images are irrelevant [Fig. 8(a)]. One image if cover and hidden images are related [Fig. 8(b)]	4	To cover multiple secret images [So, its application to cross-recovery will use some structures such as Fig. 8(b)].	Faster processing speed, but less tolerable about missing images
Ours	44.14 to 44.17	33.82 to 42.54	$n-r$= 4−2=2 images	n=4 in this case	To mutually support images of the same group	Slower processing speed, but more tolerable about missing images.

images can be identified and reconstructed by the mutual support of the r survived members. In the method, two-layer sharing is designed to create n shadows that share the recovery data. The sharing design reduces the recovery data amount, and also makes the recovery fault tolerant for up to n-r shadows can be lost.

The experimental results show that: 1. the quality of our watermarked images is acceptable, i.e., the proposed method keeps the transparency of the hidden data, including the recovery data and watermark; 2. authentication and cross-recovery can both be done when some watermarked images are altered, tampered with, or lost; 3. the visual quality of the recovered damaged images is maintained; and 4. it can be applied to distributed image storage system or multiimage transmission.

Acknowledgments

The work was supported by the National Science Council, Taiwan, under grant number NSC962221-E-009-039. The authors thank the reviewers and editor for their valuable comments to improve this work.

References

1. C. S. Lu and H. Y. M. Liao, "Structural digital signature for image authentication: An incidental distortion resistant scheme," *IEEE Trans. Multimedia* **5**(2), 161–173 (2003).
2. C. Y. Lin and S. F. Chang, "A robust image authentication method distinguishing JPEG compression from malicious manipulation," *IEEE Trans. Circuits Syst. Video Technol.* **11**(2), 153–168 (2001).
3. P. Tsai, Y. C. Hu, and C. C. Chang, "Using set partitioning in hierarchical tree to authenticate digital images," *Signal Process. Image Commun.* **18**(9), 813–822 (2003).
4. C. C. Chang, Y. S. Hu, and T. C. Lu, "A watermarking-based image ownership and tampering authentication scheme," *Pattern Recogn. Lett.* **27**(5), 439–446 (2006).
5. M. U. Ceilk, G. Sharma, E. Saber, and A. M. Tekalp, "Hierarchical watermarking for secure image authentication with localization," *IEEE Trans. Image Process.* **11**(6), 585–594 (2002).
6. P. W. Wong and N. Memon, "Secret and public key image watermarking schemes for image authentication and ownership verification," *IEEE Trans. Image Process.* **10**(10), 1593–1601 (2001).
7. H. C. Wu and C. C. Chang, "Detection and restoration of tampered JPEG compressed images," *J. Syst. Softw.* **64**(2), 151–161 (2002).
8. P. L. Lin, C. K. Hsieh, and P. W. Huang, "A hierarchical digital watermarking method for image tamper detection and recovery," *Pattern Recogn.* **38**(12), 2519–2529 (2005).
9. F. H. Yeh and G. C. Lee, "Content-based watermarking in image authentication allowing remedying of tampered images," *Opt. Eng.* **45**(7), 077004 (2006).
10. C. S. Chan and C. C. Chang, "An efficient image authentication method based on Hamming code," *Pattern Recogn.* **40**(2), 681–690 (2007).
11. C. C. Thien and J. C. Lin, "Secret image sharing," *Comput. Graphics* **26**(5), 765–770 (2002).
12. A. Shamir, "How to share a secret," *Commun. ACM* **22**(4), 612–613 (1979).
13. G. R. Blakley, "Safeguarding cryptography keys," in *1979 National Computing Conf.*, *AFIPS Conf. Proc.* **48**, 313–317 (1979).
14. C. C. Chang and R. J. Huang, "Sharing secret images using shadow codebooks," *Inf. Sci. (N.Y.)* **111**(1–4), 335–345 (1998).
15. C. C. Thien and J. C. Lin, "An image-sharing method with user-friendly shadow images," *IEEE Trans. Circuits Syst. Video Technol.* **13**(12), 1161–1169 (2003).
16. C. C. Lin and W. H. Tsai, "Secret image sharing with steganography and authentication," *J. Syst. Softw.* **73**(3), 405–414 (2004).
17. S. K. Chen and J. C. Lin, "Fault-tolerant and progressive transmission of images," *Pattern Recogn.* **38**(12), 2466–2471 (2005).
18. R. Z. Wang and S. J. Shyu, "Scalable secret image sharing," *Signal Process. Image Commun.* **22**(4), 363–373 (2007).
19. 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).
20. C. C. Lin and W. H. Tsai, "Secret image sharing with capability of

share data reduction," *Opt. Eng.* **42**(8), 2340–2345 (2003).
21. J. B. Feng, H. C. Wu, and C. S. Tsai, "A new multi-secret images sharing scheme using Largrange's interpolation," *J. Syst. Softw.* **76**(3), 327–339 (2005).
22. R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory* **44**(6), 2325–2383 (1998).
23. M. Naor and A. Shamir, "Visual cryptography," *Lect. Notes Comput. Sci.* **950**, 1–12 (1995).
24. H. C. Wu and C. C. Chang, "Sharing visual multi-secrets using circle shares," *Computer Standards Interfaces* **28**(1), 123–135 (2005).
25. 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).
26. R. Z. Wang, C. F. Lin, and J. C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm," *Pattern Recogn.* **34**(3), 671–683 (2001).
27. Y. S. Wu, C. C. Thien, and J. C. Lin, "Sharing and hiding secret images with size constraint," *Pattern Recogn.* **37**(7), 1377–1385 (2004).
28. D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recogn. Lett.* **24**(10), 1613–1626 (2003).
29. Y. C. Hu, "High capacity image hiding scheme based on vector quantization," *Pattern Recogn.* **39**(9), 1715–1724 (2006).
30. C. Y. Yang and J. C. Lin, "Image hiding by base-oriented algorithm," *Opt. Eng.* **45**(11), 117001-(1–10) (2006).
31. R. L. Rivest, "The MD5 message digest algorithm," Technique Report of MIT Laboratory for Computer Science and RSA Data Security (1992).
32. M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM Trans. Model. Comput. Simul.* **8**(1) 3–30 (1998).
33. C. S. Tsai, C. C. Chang, and T. S. Chen, "Sharing multiple secrets in digital images," *J. Syst. Softw.* **64**(2), 163–170 (2002).

Yu-Jie Chang received his BS in computer science and information engineering in 1999 from National Central University, Taiwan. In 2001, he received his MS in computer and information science from National Chiao Tung University. He is now a PhD candidate in the computer science department of National Chiao Tung University. His research interests include digital watermarking, image processing, and pattern recognition.

Sian-Jheng Lin received his BS and MS in computer science from National Chiao Tung University in 2004 and 2006, respectively. He is currently a PhD candidate in the computer science department of National Chiao Tung University. His recent research interests include pattern recognition and image processing.

Ja-Chen Lin received his BS in computer science in 1977 and MS in applied mathematics in 1979, both from National Chiao Tung University (NCTU), Taiwan. In 1988, he received his PhD in mathematics from Purdue University, Indiana. From 1981 to 1982, he was an instructor at NCTU. From 1984 to 1988, he was a graduate instructor at Purdue University. He joined the Department of Computer and Information Science at NCTU in August 1988, and became a professor there. His research interests include pattern recognition and image processing. He is a member of the Phi-Tau-Phi Scholastic Honor Society.