Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 846967, 17 pages
doi:10.1155/2008/846967

Research Article

A Sharing-Based Fragile Watermarking Method for Authentication and Self-Recovery of Image Tampering

Yu-Jie Chang,[1] Ran-Zan Wang,[2] and Ja-Chen Lin[1]

[1] *Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan*
[2] *Department of Computer Science and Engineering, Yuan Ze University, Chung-Li 320, Taiwan*

Correspondence should be addressed to Yu-Jie Chang, yjchang@cis.nctu.edu.tw

Received 15 August 2008; Revised 24 November 2008; Accepted 24 December 2008

Recommended by C.-C. Kuo

This study presents an authentication method with self-recovery ability. The hidden watermark in the proposed method not only detects and locates the tampered portion of an image, but also self-recovers the tampered portion. To increase the ability to recover from tampering, the recovery data are shared by using an (r, n) threshold scheme and then scattered all over the image. Experimental results indicate that the visual quality of the watermarked image is acceptable, and that the positions of the tampered portions can be identified. Most importantly, the *self*-recovery result is very competitive.

1. INTRODUCTION

Transmission of digital images across networks has become very popular due to the rapid development of Internet and computer technologies. However, tampering with digital images is easy to get, since modern pervasive and powerful image manipulation tools have made imperceptible modification of images very easy. Therefore, protecting the ownership and integrity of images is an important issue. Digital watermarking is a technique for inserting information (the watermark) into an image, which can be later extracted or detected for a variety of purposes, including identification and/or authentication. For copyright protection, robustness is a major concern; that is, even if the watermarked image is processed by some common image processing operations, the extracted watermark (e.g., a copyright logo) should be free from big distortion, so that it is still recognizable. A watermark embedded for this purpose is called a robust watermark [1]. Conversely, in contrast to copyright protection, a watermark embedded for content authentication should be fragile; that is, the extracted watermark might be severely deformed even if the watermarked image is tampered only slightly. This is because the basic requirement for authentication is just one simple thing; that is, if the already watermarked image (or image area) is modified later, then the mismatch of the extracted watermark can reflect this simple fact (that the already watermarked image (or image area) has been altered further) [2]. Hence, a watermark for authentication is classified as a fragile watermark. This study designs a method for fragile digital watermarking in order to verify the integrity of the contents of a digital image.

Image authentication is divided into two common approaches, digital-signature-based and watermarking-based [3]. A digital-signature-based scheme [2, 4–6] stores a so-called digital signature as the second file. The digital signature is a set of important features extracted from the original image and can be used for authentication. The digital signature approach can tolerate some slight manipulations of the original image unless the important features of the original image are changed. However, the separation of the authentication data from the images may increase management overhead for transmission and storage. Additionally, the digital signature usually does not locate where the image is tampered and thus is not directly applicable to image recovery. Unlike digital-signature-based authentication schemes [2, 4–6] which store the authentication data separately from the image, watermarking-based authentication schemes [7–14] embed the authentication data (watermark) directly into the original image. The direct embedding reduces the overhead caused by storing a separate file (digital signature), thus facilitating transmission. However, the embedded watermark is usually very sensitive to any manipulation to the watermarked image.

Many watermarking-based schemes can identify the image areas that have been tampered or manipulated by checking the presence and integrity of the local fragile watermarks. For instance, Wong [11] proposed a block-based authentication method. A hash function and XOR operations are applied to each block to create a corresponding watermarked block. Later in verification procedure, the embedding watermark is extracted from the least significant bits (LSBs) of the query image, decrypted using a public key, and finally applied an XOR operation with the hash value calculated from the query image block. Due to the property of the hash function, any tampering with a block generates an undesired binary output for that block. This scheme elegantly integrates cryptography with watermarking and works well in detecting cropping attacks. However, because Wong's approach [11] is block-wise independent, it is vulnerable to the collage attack [15] (also known as the "vector quantization" (VQ) attack [16]). The attacker constructs a vector quantization codebook with codewords taken directly from the blocks of a set of watermarked images. The image to be counterfeited is then approximated by this codebook. Since each block is already authenticated by itself, the counterfeit image appears authentic to the watermarking algorithm. To solve this problem, Wong and Memon [12] proposed two more input parameters to the hash function of Wong's previous method [11]. These new input parameters are the block index k and the image identification number ID, both of which increase the difficulty of performing a collage attack. Their scheme not only makes the collage attack infeasible, but also maintains the tampered area locating ability of the original version. However, the issue of the recovery of the tampered area is not yet provided in their scheme.

Several detecting-locating (authentication) methods with tamper recovery ability have recently been presented. Lin et al. [9] proposed an attractive block-based digital watermarking scheme for images' tampering detection, location, and recovery. They used parity check and comparison to generate the authentication watermark of each block, and then added the recovery information, which recorded the six most significant bits (MSBs) of the mean value of another block, to form the embedding watermark. Finally, the embedding watermark was embedded using the simple LSB substitution method. Their verification procedure uses a four-layer hierarchical inspection system to increase the accuracy of locating the tampered area. Although their method can recover a tampered area by storing the mean of each block in the LSBs of another block (the backup block), it cannot recover a block if both this block and its single backup block are tampered at the same time (e.g., in a cropping attack of an extensive area). Hence, the tampered-area recovery ability is not strong enough in some situations. Luo et al. [10] proposed a pixel-wise and block-wise complementary watermarking scheme based on digital halftoning and inverse halftoning techniques. Their method transforms the original image into its halftone version, which is then treated as the watermark and embedded in the original image using the simple LSB substitution method. In tampering recovery, inverse halftoning is performed to repair the tampered areas. Wu and Chang [13] developed a method based on the JPEG compression technique. Their method uses an edge detection technique to identify the edges of the image before the JPEG compression, then embeds the resulting edge characteristic into some AC coefficients of the frequency domain after the JPEG compression. If the image is tampered, then the embedded edge characteristic can be used to detect the tampered areas and cooperate the interpolation method to reconstruct it. Wang and Tsai [14] proposed a block-based digital watermarking scheme for image authentication and recovery. Their method selects the region of importance (ROI) in the image and applies the fractal image coding for the blocks of ROI to generate the recovery data. If the tampered area is not the ROI, then damaged region is restored by image inpainting.

This study proposes a competitive image authentication and tampered-area recovery method. The authentication data for each block are calculated using a cryptographic hash function, in which the input includes the MSBs information within the block, the user's secret key, the block address information, and some private information about the image. The recovery data are VQ index value of the block, which can be generated by any vector quantization (VQ) technique (e.g., [17]). To increase the recovery ability, the VQ index value is shared by Thien and Lin's secret image sharing method [18], which is reviewed briefly in the next section. Finally, the recovery information is combined with the authentication data to form the watermark. To improve security, when a part of the watermark is embedded into a block's LSBs, the embedding locations of the bits are arranged according to a pseudorandom permutation, based on the Mersenne Twister (MT) pseudorandom number generator [19]. Experimental results show that the quality of the watermarked image is acceptable, and that the positions of the tampered parts are located accurately. The recovery of large-area corruption is also good.

In conclusion, the proposed method has the following novelty compared with previously published schemes (particularly, image authentication or recovery methods [7–14] and image sharing methods [18, 20–26]).

(i) The recovery data are generated by using VQ compression technique (an index file). A VQ index file has at least three advantages. (a) *The matched codeword of an image block is more suitable for showing the texture of the block than the mean value or the halftone result, as used in some other publications.* (b) VQ compression technique is block based, and a block-based approach is sufficiently easy to apply for tampering recovery. (c) VQ decompression is simple and has a very short decoding time, thus reducing the reconstruction cost.

(ii) To increase the survival rate of the recovery data (VQ index file), a modified version of the (r, n) threshold sharing [18] is used to generate n index shares. Some remarks about this are given below.

(a) In earlier applications (reviewed in Section 2) of secret sharing techniques to images, people shared a secret image among n shadows and then transmitted them to n participants. The secret image can be revealed when any r of the n participants gather together. However, the proposed method shares the VQ index file of the host image to generate n index shares and then embeds these n index shares into the image itself, rather than transmitting the shares to n users. In the recovery phase, the tampered area is reconstructed by extracting any r authenticated VQ index shares embedded earlier in the watermarked image itself.

(b) When applying secret sharing techniques to images, people often generate n shares for each pixel (or for each block of the pixels) of the secret image. Hence, the share value has a wide range, and each share is represented by m bits, where m is the number of bits per pixel. For example, if each pixel is represented by 8 bits, then each share needs 8 bits. However, the proposed method assumes that the codebook has L codewords and divides each VQ index value into r sections before generating n index shares for each index value. Thus, the size of each index share is reduced to $\lceil(\log_2 L)/r\rceil$, which is usually smaller than 8 (e.g., $\lceil(\log_2 L)/r\rceil = 2$ bits if ($L = 1024$, $r = 5$), or 3 bits if ($L = 4096$, $r = 4$)). The smaller size of each share helps maintain the quality of the watermarked image in the embedding process.

(c) All arithmetic calculations in the proposed method are performed in the power-of-two Galois Field GF $2^{\lceil(\log_2 L)/r\rceil}$), rather than the Mod_{251} used in many previous image sharing. This modification not only reduces the number of bits of each index share from $\lceil\log_2 251\rceil = 8$ bits to $\lceil(\log_2 L)/r\rceil$ bits, but also makes the proposed method more suitable for the various codebook sizes L, thus increasing flexibility.

(iii) Many published image recovery techniques embed the recovery data of a unit (i.e., a block or a pixel) for backuping into another block or pixel according to a permutation function. (Notably, having many copies of the recovery data might increase survival rate, while decreasing the quality of the watermarked image.) The proposed method is sharing based, and each share is small in size. Hence, unlike published recovery methods, the proposed method allows many backup shares (n shares) without significantly increasing the total size of recovery data. Since the proposed method has more backup pieces (n shares rather than 1 or 2 copies), it can use a hybrid two-layer strategy to scatter the backup (the n index shares created to backup a VQ index value). More specifically, the host image is divided into at least $(n + 1)$ nonoverlapping regions, and the n index shares of each block are, respectively, embedded into the blocks of n other regions. After this layer of n-to- n random mapping, the position of the corresponding block in each region can be randomized again in the second layer and be distinct among regions by an MT pseudorandom number generator [19]. With this strategy, the recovery data can be uniformly scattered (*in a distributed and missing-allowable manner*) in the whole host image to resist a cropping attack of an extensive area.

The rest of this paper is organized as follows. Section 2 briefly reviews the secret sharing method. The encoding (watermarking) and decoding (detection, locating, and recovery) of the proposed method are described in Sections 3 and 4, respectively. Experimental results are presented in Section 5. The discussion and comparison are in Section 6, and the summary is in Section 7.

2. A REVIEW OF SECRET SHARING METHODS

The concept of secret sharing was introduced independently by Shamir [20] and Blakley [21]. Their (r, n) threshold scheme divides the secret data into n shares. If any r of these shares are available (r is a predefined number and $2 \le r \le n$), then the secret data can be reconstructed, but fewer than r shares cannot. One major advantage of secret sharing is fault tolerant, because it allows n-r shares to be absent due to network delay problem or storage damage. Consequently, the survival rate of secret data increases. Several secret sharing methods based on an (r, n) threshold scheme have been proposed [18, 22–26]. Chang and Hwang [22] applied the VQ technique [17] to encode the secret image. Their method shares the generated codebook for reconstructing the secret image among the n participants by applying the (r, n) threshold scheme. Thien and Lin [18] proposed a secret image sharing scheme. Their (r, n) threshold scheme shares a secret image among n participants, and each participant holds a generated shadow image whose size is only $1/r$ of that of the secret image. The secret image can be reconstructed if at least r of the n shadow images are received. Thus, their method is also fault tolerant, because n-r shadow images can be lost during the reconstruction. The smaller size of their shadow images ($1/r$ times the size of the shadow images of ordinary sharing methods) is an advantage in the transmission and storage. They further developed a scheme [23] in which the shadow images look like portraits of the original secret image and thus provide a user-friendly interface to facilitate the management of the shadow images. Lin and Tsai [24] integrated the (r, n) threshold scheme with watermarking, thus ensuring that each share has authentication capability to verify its integrity before reconstructing the secret image. Wang and Shyu [26] presented a scalable secret image sharing scheme with three sharing modes, (1) multisecrets, (2) priority, and (3) progressive, to increase the potential application for secret image sharing. Unlike the aforementioned spatial-domain methods, Lin and Tsai [25] proposed a method that transforms the secret image into the frequency domain then applies 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 the proposed method utilizes the sharing polynomial of Thien and Lin's method [18], their work is reviewed below. In [18], a secret image O containing m pixels is shared by n participants based on Shamir's polynomial (r, n)-threshold scheme [20] 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 sections so that each section contains r pixels. Let $q(x)$ be the xth shadow image, and let $q_j(x)$ be the jth pixel in $q(x)$, where $1 \le x \le n$ and $1 \le j \le m/r$. For each section j, the r coefficients $a_0, a_1, \dots, a_{r-1}$ of the corresponding polynomial

$$q_j(x) = a_0 + a_1 \times x + \cdots + a_{r-1} \times x^{r-1} (\bmod p) \quad (1)$$

are used as the gray values of the r pixels of the corresponding section j in Q. The xth shadow image $q(x)$ is the collection $\{q_j(x) \mid j = 1, 2, \dots, m/r\}$. Since each section j, which has r pixels, contributes only one pixel $q_j(x)$ to the xth shadow image, the size of each generated shadow image is only $1/r$ of that of the secret image O. This property holds for every shadow image, that is, for every $x \in \{1, 2, 3, \dots, n\}$. Any r of the n shadow images can be utilized to reconstruct Q because the inverse finding of the r coefficients $a_0, a_1, \dots, a_{r-1}$ in (1) only needs r of the n values $\{q_j(1), q_j(2), \dots, q_j(n)\}$. The detail is omitted.

To enhance the recovery ability of tampered image, the proposed method employs the polynomial sharing equation (1) to share the information needed for recovery. This procedure is described in detail in Section 3.1. Notably, Thien and Lin's method [18] uses $p = 251$, but this study uses $p = 2^{\lceil (\log_2 L)/r \rceil}$, where L is the VQ codebook size. Additionally, all arithmetic calculations in the proposed method are performed in the power-of-two Galois Field $\mathrm{GF}(2^{\lceil (\log_2 L)/r \rceil})$, thus reducing the number of bits needed to express a share value (the left hand side of (1)) from $\lceil \log_2 251 \rceil = 8$ bits to $\lceil (\log_2 L)/r \rceil$ bits.

3. THE PROPOSED METHOD (ENCODING)

Figure 1(a) shows the flowchart of the encoding (watermarking) procedure, while Figure 1(b) illustrates the decoding (verification and recovery) procedure. Section 3 explains Figure 1(a) only.

The encoding procedure consists of the following three parts. (i) The input 8-bit grayscale image is first encoded by VQ compression technique [17] to generate a VQ index file of the image. The VQ index file of the image is then shared by Thien and Lin's method [18] (see (1)) to generate n shares. These index shares are treated as the recovery data. (ii) Next, the authentication data for each image block are generated using a cryptographic hash function, in which the input includes the local information of the image block, and the interrelated information (about the image) to resist the collage attack. (iii) Finally, the recovery data of other blocks are combined with the authentication data to form the watermark for the block. The watermark is embedded in the LSBs of the block in the image. The position of the so-called "other block" may be determined by a permutation method based on the Mersenne Twister (MT) pseudorandom number generator [19].

The details of the encoding are described in the following three subsections accordingly.

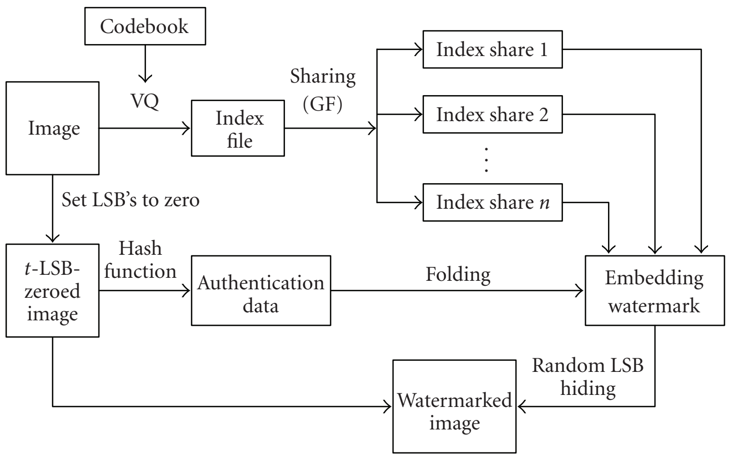
3.1. Generation of the recovery data

The image recovery data are generated first. To do this, the image is partitioned into nonoverlapping blocks of $h \times w$ pixels each, where h is the block height, and w is the block width. Each block of the image is then represented by an index value according to a given codebook of size L (i.e., the number of codewords in the codebook is L). Therefore, the compression result of the image is an index file. The index value of each image block can be used as the recovery information when the block is tampered.

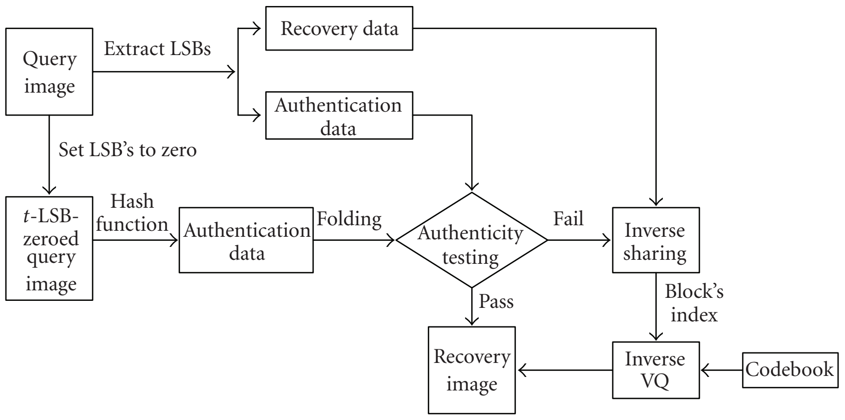
To increase the survival rate in the recovery process, Thien and Lin's sharing method [18] is utilized to share the index file of the VQ compressed image, as described below. The index file in a sequence of *index value by index value* (or equivalently, *block by block*) is shared according to the raster-scan order of the image, which is from left to right and top to bottom. Each index value is transformed into a binary stream of $\log_2 L$ bits, and this binary stream is then divided into r sections. Therefore, each section has $\lceil \log_2 L/r \rceil$ bits, where L is the codebook size, and $\lceil \cdot \rceil$ denotes the *ceiling* operation. Each section of the binary stream is then transformed into a decimal number, and these r decimal numbers of an index value are assigned as the r coefficients in (1). By substituting n distinct values for the variable x in (1), n shares can be generated. In other words, the index value of an image block is shared among n index shares. Notably, while Thien and Lin [18] used $p = 251$, as stated at the end of Section 2, the proposed method uses $p = 2^{\lceil (\log_2 L)/r \rceil}$ and performs all arithmetic calculations in the power-of-two Galois Field $\mathrm{GF}(2^{\lceil (\log_2 L)/r \rceil})$. This change reduces the number of bits needed to express a share value (the left hand side of (1)) from $\lceil \log_2 251 \rceil = 8$ bits to $\lceil (\log_2 L)/r \rceil$ bits. Additionally, the bit length of each share value (which has $\lceil (\log_2 L)/r \rceil$ bits) is only about $1/r$ of that of an index value (which has $\log_2 L$ bits).

Since the (r, n) threshold scheme is used to share an index value among n shares, the index value can still be recovered even if (up to) n-r shares are lost. This increases the survival rate (the recovery ability) of a tampered block if the n shares of the tampered block are stored in n distinct places of the image in advance. Moreover, because the bit length of each share value is only $1/r$ of that of the index value, it is easy to embed such a compact share value in the watermark.

Here we explain more clearly how (r, n) sharing and uniform scattering increase the survival rate (the recovery ability) of a tampered block. Notably, the proposed method uses VQ index as the recovery data. For an image block, the matched codeword is more informative to show the texture of the block than the mean value or the halftone result used in some other publications. Although the size of a VQ index might be larger than that of the mean value or the halftone representation of the block, we will use (r, n) sharing to shorten it. (Each share is $1/r$ times the length of the VQ index being shared.) Hence, unlike published recovery methods, the proposed method allows many backup shares



(a)



(b)

FIGURE 1: Two flowcharts show the proposed method: (a) the encoding (watermarking) procedure, (b) the decoding (verification and recovery) procedure.

(n shares) for each block without significantly increasing the total size of recovery data. Since the proposed method has more backup pieces (n shares rather than 1 or 2 copies), we can use a hybrid two-layer strategy to scatter the backup (the n created index shares) all over the image in a uniformly and not-too-sparse manner. (This two-layer strategy is described in detail in Section 3.3.) With this strategy, the recovery data can be uniformly scattered throughout the whole image to resist a cropping attack of an extensive area. Finally, since the (r, n) sharing being used is a threshold system, if the image is tampered in certain area, recovering a tampered block A only requires that the remaining (nontampered) areas contain at least r of the n scattered shares for A. This missing allowable property (n-r shares can be missed), together with the uniformly scattered distribution of the n shares, increases the survival rate of the recovery data. As for previous publications, they use a backup approach to store the recovery data directly. In such approaches, if a block and its single or double backup blocks are tampered at the same time (e.g., in a cropping attack of an extensive area), then the recovery data of this block is lost completely.

3.2. Generation of the authentication data

After generating the recovery data, the next work is to generate the authentication data to verify the image's integrity. The authentication data of each block consists of some important information about the block, including the MSBs of the pixels within the block and the block's position, as well as some private information, such as the image owner's secret key, and the image identification number, width and length.

The watermark in the proposed scheme is hidden in the t LSB planes of the image, and the remaining (8-t) MSBs of each pixel within the block are used in the generation procedure of authentication data. The parameter $t < 8$ must satisfy

$$\left\lceil \frac{\lceil (\log_2 L)/r \rceil \times n}{h \times w} \right\rceil < t < 8. \qquad (2)$$

This is because if each block has $h \times w$ pixels, then t LSB planes together provide $h \times w \times t$ bits to hide the recovery data of n shares, of which each share has $\lceil (\log_2 L)/r \rceil$ bits. Of course, the authentication data naturally occupies additional bits. The number of additional bits should be added to the numerator of (2) if the "$<t$" in (2) is to be replaced by "$=t$". According to the decided parameter t, the image is transformed into the t-LSB-zeroed image by setting all t LSBs of each pixel in the image to zero. Notably, if the parameter value of t is increased, then the hiding capacity of each block (to embed recovery and authentication data) rises, leading to stronger recovery and authentication ability. However, the quality of the watermarked image worsens if the value of t increases.

The t-LSB-zeroed image is used to generate the authentication data for verification, as shown in Figure 1(a). Consider a cryptographic hash function

$$H(S) = (d_1, d_2, \ldots, d_u), \tag{3}$$

where S is an input bit string of arbitrary length; d_i is the output binary bits of the hash function for $1 \le i \le u$, and u is the length of the output bit string. Wong and Memon [12] concluded that a cryptographic hash function should have the property that once an input bit string S and its corresponding output $(d_1, d_2, \ldots, d_u)$ are given, then finding another input bit string of any length that will be hashed to the same output $(d_1, d_2, \ldots, d_u)$ should be computationally infeasible. Commonly seen hash functions include the MD5 [27] and secure hash algorithm (SHA). For MD5, collision attacks are computationally feasible on a standard desktop computer [28], while SHA-1 attacks require heavier computational power [29] (about 2^{69} hash operations), making attacks less feasible. Hence, the proposed method employs SHA-1 as the hash function and produces a single output 160-bit message (the output hash value) from the input message, that is, $u = 160$. Of course, other cryptographic hash functions can also be used instead of SHA-1.

Let ID be the identification number of an image, and let SK denote a secret key of the image owner. The SK can be either constant or dynamic, where dynamic means changing randomly or according to the image ID. Consider the kth block of an image ($k = 1, 2, \ldots, (\text{Height} \times \text{Width})/(h \times w)$ if the image size is given as Height $\times$ Width). The gray value of the $h \times w$ pixels of the block (computed using the $(8 - t)$ MSBs only) is denoted by $p_1^k, p_2^k, \ldots, p_{h\times w}^k$. The authentication data of the kth block can be computed as

$$\begin{aligned} &H(ID \| SK \| \text{Height} \| \text{Width} \| k \| p_1^k \| p_2^k \| \cdots \| p_{h\times w}^k) \\ &\quad = (d_1^k, d_2^k, \ldots, d_u^k), \end{aligned} \tag{4}$$

where the symbol $\|$ is the concatenation of all input streams. Notably, according to the discussion in [12], the image identification number ID and the block sequence number k are two important parameters for resisting the collage attack [15].

Finally, because each pixel within a block only has t LSBs to be embedded, each block might have insufficient embedding space for embedding the recovery data and the 160-bit authentication data. In this case, the bit stream of the authentication data needs to be folded, that is, converted into another shorter bit stream. In the folding procedure, the length of the embedding authentication data δ is determined by

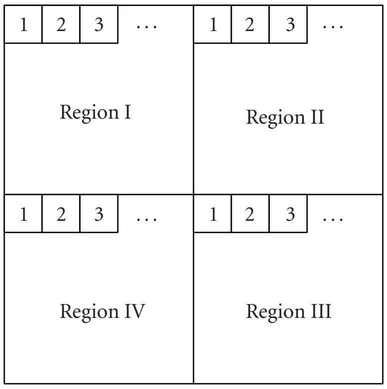


FIGURE 2: An example of region division and number assignment.

$$\delta = h \times w \times t - \left\lceil \frac{\log_2 L}{r} \right\rceil \times n. \tag{5}$$

The original bit stream generated by (4) is then divided equally into several sections of δ bits each and folded into δ-bit embedding authentication data with an exclusive or operation. For example, a bit stream "10100111" can be folded into "1101", since $1101 = 1010 \oplus 0111$. In this case, $\delta = 4$; "1010" is the first section of the original bit stream, and "0111" is the last section of the original bit stream. If the last section does not have enough bits, then it is padded with "0" to make it convenient to fold.

3.3. The embedding procedure

Finally, as shown in Figure 1(a), the recovery and the authentication data of a given image are combined to form the embedding watermark. This section describes the procedure that embeds the watermark. First, the image is divided equally into $(n + 1)$ nonoverlapping big regions, where n is a number not less than the threshold value r. The block addresses $\{1, 2, 3, \ldots\}$ are assigned to the blocks of each region according to the raster scan order, that is, from left to right and top to bottom. The n index shares of each block are then embedded, respectively, into the blocks, which are in the corresponding positions of the other n regions. For example, assuming that $r = 2$ and $n = 3$, then, as shown in Figure 2, the image is divided equally into $n + 1 = 3 + 1$ nonoverlapping regions, and numbers are assigned sequentially to the blocks of each region by raster scan order. According to Section 3.1, the VQ index value of the first block in Region I is shared among $n = 3$ index shares, which are embedded into the first block in the remaining $n = 3$ regions (Regions II, III, and IV), respectively. Likewise, the three index shares of the first block in Region II are embedded into the first block in the remaining $n = 3$ regions (Regions III, IV, and I), respectively. Similar action is performed for each block in each region.

On the other hand, each block has its own authentication data for verification purpose, according to Section 3.2. The authentication data of a block are embedded into the block itself. Therefore, the data to be embedded into each block include its own authentication data and the n index shares obtained from the corresponding block of the remaining n regions (one share per region). To simplify the embedding, the concatenation operation is used to combine these n index shares and the authentication data to form a watermark for the block. The watermark is then transformed into a binary stream and is embedded into the t LSBs of the pixels in the block. To increase security level, the MT pseudorandom number generator [19] may also be integrated into the embedding procedure to confuse the embedding positions of the binary stream, as illustrated below using the aforementioned example ($r = 2$ and $n = 3$). Assume that the embedding watermark has 27 bits, which consist of $n = 3$ index shares (b_0–b_4, b_5–b_9 and b_{10}–b_{14}) and a set of local authentication data (b_{15}–b_{26}). This watermark is embedded into the $t = 3$ LSBs of the block with a size of 3×3. Figure 3(a) shows the positions of the $3 \times 3 = 9$ pixels in the block. The 3 LSBs of the nine pixels in the block provide a hiding space of 27 bits, which are assigned position numbers 0–26, as shown in Figure 3(b). Inputting a randomly chosen 32-bit natural number, called the "seed", into the MT pseudorandom number generator [19], generates a series of pseudorandom real numbers R_j ($j = 0, 1, 2, \ldots, 2^{19937} - 1$), which are uniformly distributed on the interval $[0, 1]$. After a simple mapping (the detail is omitted), we get a pseudorandom sequence whose elements are integers from 0 to 26. The embedding binary stream is then embedded into the corresponding locations in the block according to the pseudorandom integer sequence. For example, if the generated pseudorandom embedding sequence for 0–26 is {21, 4, 26, 13, 6, 9, 0, 25, 13, 14, 22, 3, 16, 2, 17, 18, 5, 19, 7, 20, 8, 11, 1, 23, 15, 24, 10}, then the bit b_0 of the binary stream watermark is embedded in the bit-position marked as number 21 in Figure 3(b), and the bit b_1 is embedded in the bit-position marked as number 4 in Figure 3(b), and similarly along the entire sequence. Figure 3(c) shows the result of embedding the 27-bit watermark b in this example.

The watermarked image is obtained by performing the embedding procedure block by block until the entire image is processed. The seed used in the MT pseudorandom number generator [19] can be the secret key (SK) in (4), and only the legal users who own the same secret key can obtain the same pseudorandom embedding sequence to verify correctly the authenticity and integrity of the image.

4. THE PROPOSED METHOD (DECODING)

4.1. Verification

In the verification phase, the query image is first divided into nonoverlapping blocks of $h \times w$ pixels, and the verification is performed block by block until all blocks are processed. The watermark of each query block is extracted from the t LSBs of each of its $h \times w$ pixels. Notably, only legal users know (the codebook and) the parameters used earlier in the watermark embedding procedure. Legal users can obtain the previously used pseudorandom embedding sequence from the secret key SK and thus obtain the authentication data and recovery data from the extracted watermark, according to the locations corresponding to the obtained sequence. The authentication data "extracted" this way using the t LSBs of the query block's pixels should be the same as the authentication data "recomputed" directly from the (8-t) MSBs of the block's pixels, with the recomputation as described in the two paragraphs containing (4) and (5). A block in which the extracted authentication data coincide with the recomputed authentication data is called an authentic block; otherwise, it is regarded as a tampered block. An authentic block is one that almost certainly has no tampering occurs, and the recovery data embedded in this block are trustworthy for reconstructing other blocks in the recovery phase, if other blocks are tampered. In contrast, a block that fails the authentication test is regarded as a tampered block, and the backup content stored in it should never be trusted, that is, should not be used to reconstruct other blocks in the later recovery phase.

Because the hiding capacity is not enough to hide the whole hashed sequence, a folding procedure is performed to shorten the hashed sequence. This procedure is a many-to-one mapping (2^{160} to 2^{δ}), leading to the possibility that two hashed sequences have the same folded sequence. (A smaller δ value leads to a higher collision probability.) However, experiments results show that the folded sequence can locate almost all tampered places when $\delta = 11$ bits (see Figures 6(d)–6(f) and 7(d)–7(f)).

Moreover, to reduce the probability of misdetection resulting from collision, a hierarchical check system can be used to locate the tampered area. In other words, each authentic block can be checked again after performing the verification procedure described in Section 4.1. If the eight neighboring blocks of an authentic block are marked as tampered blocks, then the block status can be changed from authentic to tampered because a collision might have occurred in the block. This block is then recovered by recovery data, which can still yield an acceptable look for this block, even if no collision occurred.

4.2. Recovery of tampered area

The recovery phase starts if any tampered block is found by the block authentication step. For a tampered block, we collect its backup data, which are the n VQ index value shares embedded in the n corresponding blocks of other n regions (one share per region, see Section 3.3), to reconstruct the VQ index value of the block. Of course, some of these n backup blocks might also be marked as "tampered" after the authentication test in Section 4.1. These tampered, and hence useless, backup blocks are simply skipped. The VQ index value of the tampered block can be traced back by inverse sharing if at least r (out of the n) VQ index value shares survive. The tampered block can then be rebuilt by this VQ index value and the codebook.

5. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section presents the experimental results and comparison to demonstrate the performance and feasibility of the

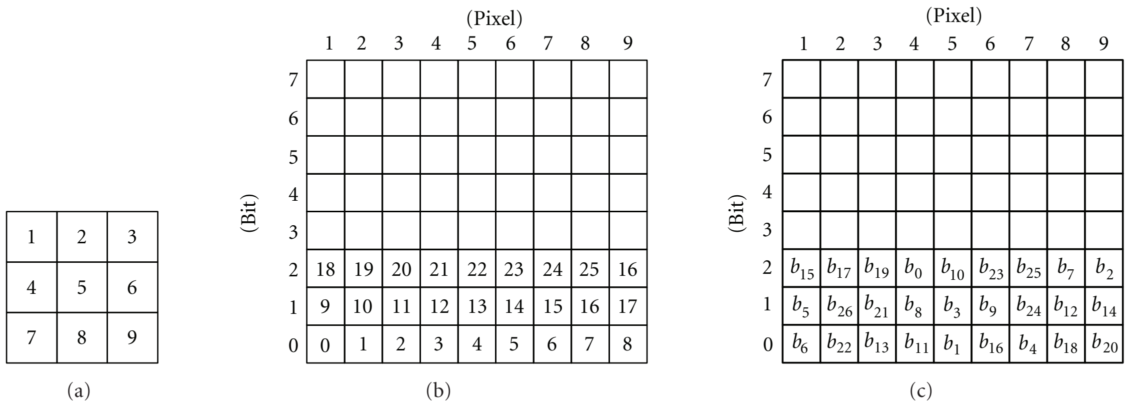

1 2 3
4 5 6
7 8 9
(a)
(Pixel)
1 2 3 4 5 6 7 8 9
(Bit)
7
6
5
4
3
2 18 19 20 21 22 23 24 25 16
1 9 10 11 12 13 14 15 16 17
0 0 1 2 3 4 5 6 7 8
(b)
(Pixel)
1 2 3 4 5 6 7 8 9
(Bit)
7
6
5
4
3
2 b_{15} b_{17} b_{19} b_0 b_{10} b_{23} b_{25} b_7 b_2
1 b_5 b_{26} b_{21} b_8 b_3 b_9 b_{24} b_{12} b_{14}
0 b_6 b_{22} b_{13} b_{11} b_1 b_{16} b_4 b_{18} b_{20}
(c)

FIGURE 3: An example of embedding a 27-bit watermark b in a 3×3 block: (a) the arrangement of the 9 pixels, (b) the arrangement for the 3-LSBs of the 9 pixels, (c) the result of embedding.

proposed method. All images were 8-bit gray valued, and the VQ codebook was generated by the LBG algorithm [30]. Each block had 4×4 pixels, and a set of 512×512 standard gray-value images was used as the training set to generate the VQ codebook. The ($r = 2$, $n = 3$) threshold sharing scheme was used to share the VQ index values. The image was divided equally into $n + 1 = 4$ nonoverlapping regions according to Section 3.3, and the parameter t became $t = 2$ by (2) when $L = 16{,}384$ and $h \times w = 4 \times 4$, which means that the watermark was embedded in the $t = 2$ least significant bits of each pixel of the image.

Figure 4 shows the four original images "Lena," "Jet," "Baboon," and "Barbara". Each image was 512×512. Figure 5 shows their watermarked version, after executing the proposed watermarking method described in Section 3. The qualities of all watermarked images and recovery images were measured by the peak signal-to-noise ratio (PSNR), defined as

$$\text{PSNR} = 10 \times \log_{10}\frac{255^2}{\text{MSE}}, \tag{6}$$

in which MSE denotes the mean square error between the pixel values of the original and of the watermarked images. From Figure 5, we can see that the qualities of the watermarked images are acceptable. Their PSNR values are all greater than 44.13 dB. The images in Figures 4(a)–4(d) and 5(a)– 5(d) are visually indistinguishable using naked eyes. In other words, the proposed watermarking method has transparency property (the watermark is perceptually invisible).

Experiments were performed to tamper with the watermarked images shown in Figure 5 to measure the performance of the proposed method in tampering detection, location, and recovery.

5.1. *Cropping attacks*

A part of each watermarked image was cropped. Figure 6(a) shows the watermarked image Lena$^{(W)}$ with 25% of the image cropped. Figures 6(b) and 6(c) show the same image with 50% of the image cropped. The cropped parts were detected by the verification procedure described in Section 4.1, as shown in Figure 6(d)–6(f), where the black blocks are the detected tampered blocks, and the white blocks are the detected authentic blocks. Finally, Figures 6(g)–6(i) show the recovery results obtained by applying the recovery techniques in Section 4.2 on the images Figures 6(a)–6(c), respectively (using the white blocks in Figures 6(d)–6(f) to recover the black blocks). The PSNR values between the recovery images in Figures 6(g)–6(i) and the original images in Figures 6(a)–6(c) were 40.04 dB, 34.77 dB, and 36.54 dB, respectively. In summary, the cropped portions were correctly detected and located, and the cropped portions were recovered although the cropped portions occupied a big area of the watermarked image in both Figures 6(b) and 6(c).

5.2. *Collage attacks*

As reviewed in Section 1, a block-based watermarking scheme may be vulnerable to a collage attack [15]. Therefore, an experiment was performed to verify the effectiveness of the proposed scheme against a collage attack. To perform collage attack, several authentic blocks were collected from a set of watermarked images to forma "forgery" codebook as used in collage attack. Figure 7(a) shows a tampered watermarked image "Lena$^{(W)}$" (Figure 5(a)), in which a flower (formed of blocks chosen from forgery codebook, i.e., blocks chosen from another watermarked image) is inserted into her hat. Similarly, Figure 7(b) shows a tampered watermarked image "Jet$^{(W)}$" (Figure 5(b)), in which the country name "U.S." and the emblem printed on the plane are replaced, respectively, by the country name "R.O.C." and another emblem. Finally, Figure 7(c) shows a tampered watermarked image "Barbara$^{(W)}$" (Figure 5(d)), in which a part of the bookshelf in the top-left corner is removed by copying some blocks of the same watermarked image. Figures 7(d), 7(e), and 7(f), respectively, show the detection results of the tampered images, and Figures 7(g), 7(h), and 7(i), respectively, show the recovery results, of which the PSNR values were 41.58 dB, 42.45 dB, and 43.01 dB, respectively. The experiment results indicate that the proposed method accurately detected and located the replaced

Figure 4: Four original images: (a) Lena , (b) Jet, (c) Baboon, (d) Barbara.

parts and recovered back the replaced parts with acceptable quality. Additionally, inserting the block index k and the image identification ID in the authentication data helped resist collage attack [15], as suggested by Wong and Memon [12].

6. DISCUSSIONS

6.1. Modified version

A security-enhanced approach can be used to increase the security level. In this new model, the image is divided equally into at least $n + 1$ nonoverlapping regions. Then, for each block, the block's n VQ index value shares are embedded into n blocks chosen from n of the remaining regions (one corresponding block per mentioned region). However, the positions of the corresponding blocks in each region can be randomized and made distinct among regions. Therefore, the MT pseudorandom number generator [19] could be applied to perform this rearrangement.

6.2. Comparison

Because the proposed scheme is a watermarking-based approach with detecting locating and recovery abilities of tampered areas, Table 1 compares it with other watermarking-based methods that have the same functions.

The data in Table 1 indicate that the proposed scheme is competitive. The qualities of our watermarked images are competitive to the counterparts obtained in those elegant methods reported in literature. Because the PSNR values of the watermarked images in [10] and the recovered images in [10, 13] are not shown in the published papers [10, 13], they are denoted by the symbol "N/A (not available)". Although the PSNR value of the watermarked images of [10] is not shown in their published paper, the value must be very high, since the halftone information is embedded by 1-LSB (in the worst case, even if the LSB bit of each pixel is altered, the PSNR is still higher than 48 dB). Even so, their recovered images were still slightly worse than those obtained by the proposed method, as shown in Figure 8 (where Figures 8(b) and 8(e) are from [10, page 168] and [13, page 157], resp.). Additionally, Figures 8(g)–8(h) show the experimental result of [14]. Clearly, the lost upper 25% of the image recovered by [14] (Figure 8(h)) contains more distortion than that of the proposed method (Figure 8(i)).

To measure the performance of tampering recovery, the qualities of the recovered images were compared with those published methods when the same area was cropped from the watermarked images of both sides. From Table 1 and Figure 8, we can see that the quality of the recovered image in the proposed method is better than that of the published methods. This is because 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. Therefore, a block and its backup block may both be tampered at the same time if a watermarked image is tampered extensively in a large area and randomly. In this case, the recovery ability of the tampered block in many approaches is gone. In contrast to the aforementioned approach (which uses a backup block to store the recover data), the recovery data of a block in the proposed method are the n "shares" of the block's VQ index value, which are

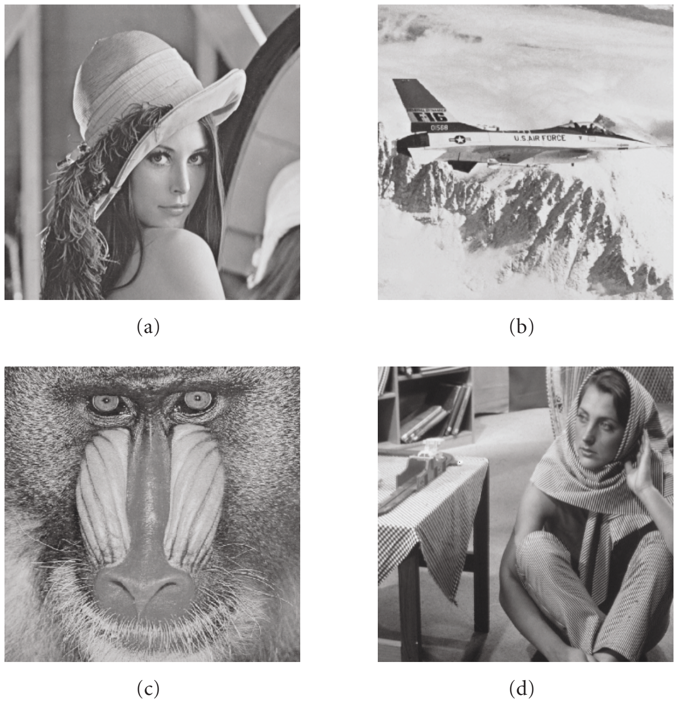


Figure 5: The watermarked images of Figure 4: (a) Lena$^{(W)}$ (PSNR = 44.14 dB), (b) Jet$^{(W)}$ (PSNR = 44.13 dB), (c) Baboon$^{(W)}$ (PSNR = 44.15 dB), and (d) Barbara$^{(W)}$ (PSNR = 44.15 dB).

Table 1: A comparison between the reported recovery methods and the proposed method. ("$*$" means "quoted directly from the reported paper", and "N/A" means not mentioned in the reported paper). The unit of PSNR is the dB.

Method	Abilities to detect and locate tampering	PSNR of the watermarked image	PSNR of the recovered image
[9]	Yes	44.37* (Beach 256 × 256)	30.85* (cropping 7.1%)
[14]	Yes	42.11* (Lena 256× 256)	30.14* (cropping 12.5%) 25.39* (Figure 8(h) where cropping is 25%)
Ours	Yes	44.15 (Beach 256 × 256)	42.44 (cropping 7.1%)
		44.15 (Lena 256 × 256)	41.28 (cropping 12.5%) 38.81 (Figure 8(i) where cropping is 25%)
		44.14 (Lena 512 × 512)	43.20 (cropping 6%) 34.77 or 36.54 (Figure 6(h) or Figure 6(i), resp.; each cropping is 50%)
		44.13 (Jet 512 × 512)	43.27 (cropping 6%) 34.34 or 36.07 (if crop 50% like Figure 6(b) or Figure 6(c) does)
[10]	Yes	N/A	N/A (but see Figure 8(b))
[13]	Yes	34.34* (Lena 512 × 512)	N/A (but see Figure 8(e))

embedded into blocks located in n other nonoverlapping regions of the image. (Notably, instead of directly embedding the recovery data, the proposed method shares the data and spreads the n shares throughout the whole image. The scattering of the n shares in the whole image reduces the chance that all VQ index value shares are lost when a connected portion of the image is damaged.) With sharing (and the scattering in the whole image), the tampered query block can still be recovered even if only r out of the n backup blocks of a tampered query block survive in an attack. This increases the recovery rate of the proposed method, as compared with those methods in which each query block only has one backup block. Thus, the proposed scheme can handle the case in which a large area is tampered even if 50%

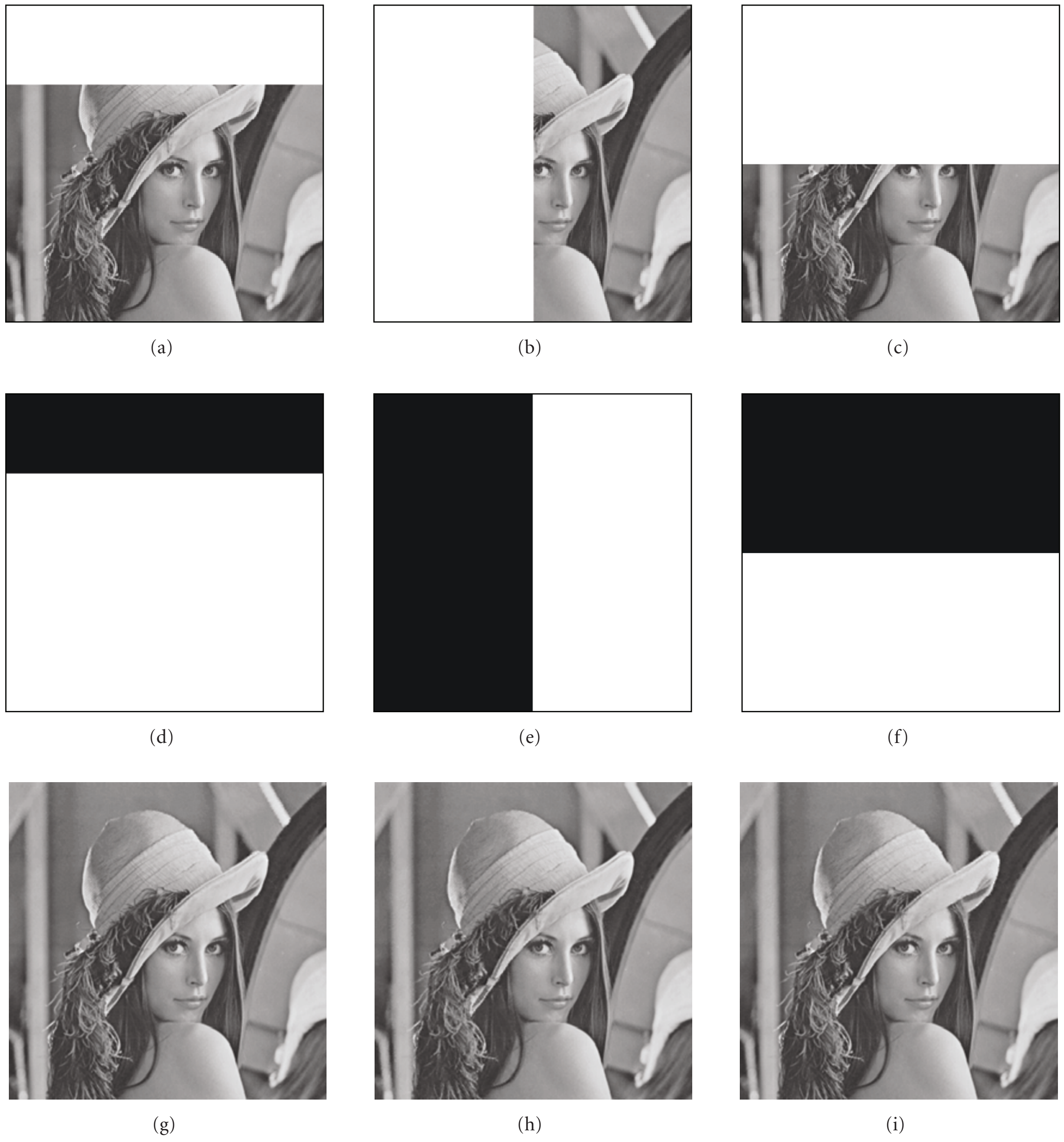


Figure 6: The cropping attack experiments: (a) 25% of the watermarked image Lena$^{(W)}$ is cropped; (b) a vertical cropping of 50%; (c) a horizontal cropping of 50%; (d)–(f) the detected tampered blocks of (a)–(c), respectively; (g)–(i) the images recovered from (a)–(c), respectively. The PSNR values are 40.04 dB for (g); 34.77 dB for (h); and 36.54 dB for (i).

of the watermarked image is cropped, as long as r out of the n backup blocks are authentic.

To make a fair comparison with published methods, Table 2 shows the compression ratio, amount of the recovery data, and preservation system of each method. The compression ratio is the ratio of the recovery data size to the original data size. For a 256 × 256 host image, because [9] stores the 6-bit mean value of each 2 × 2 block, its compression ratio is $(2 \times 2 \times 8)/6 = 5.33$. Moreover, [9] uses only one backup to preserve the recovery data and thus has $((256 \times 256)/(2 \times 2)) \times 6 = 98{,}304$ bits of recovery data, which are embedded in the 2-LSB of the watermarked image. (The recovery data of a block is preserved in another block whose address is determined by a permutation function.) Consequently, the block and its backup block could be tampered simultaneously if the watermarked image is tampered extensively in a large area and randomly. In this case, the recovery of the block fails. Lin et al. [9] found that the recovery rate was not higher than 94% when the cropping area occupies 50% of the watermarked image. In contrast, the proposed method had a recovery rate of 100%. In [14], Wang and Tsai preserved the recovery data of ROI (regions of importance) and embedded them in the host image. In their experiments, because the size of the so-called range block was 4 × 4, and the codes of each range block were 31 bits, the compression ratio is (4 × 4 ×

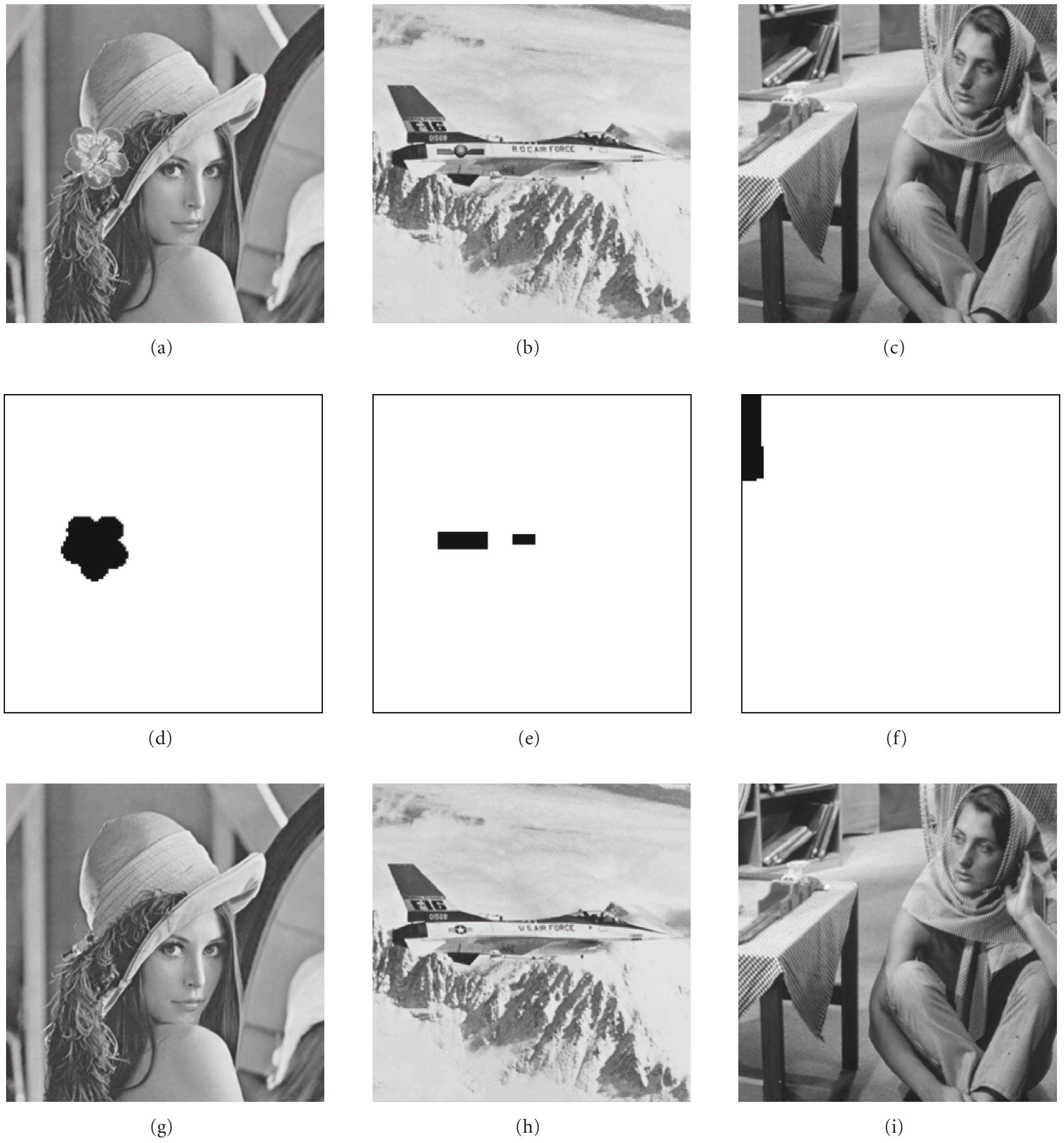


Figure 7: The collage attack experiments: (a) a tampered version of Lena$^{(W)}$; (b) a tampered version of Jet$^{(W)}$; (c) a tampered version of Barbara$^{(W)}$, respectively; (g)–(i) the images recovered from (a)–(c), respectively. The PSNR values are 41.58 dB for (g); 42.45 dB for (h); and 43.01 dB for (i).

8)/31 = 4.13. Their scheme also uses a backup approach to preserve the recovery data. Hence, if the range of ROI is about 60% of the host image, then $((256 \times 256)/(4 \times 4)) \times 31 \times 0.6 = 76{,}186$ bits are embedded in the 256×256 watermarked image. The recovery ability is good in the ROI, but slightly worse in the non-ROI area, because the tampered non-ROI area is recovered using image inpainting technique. Luo et al. [10] used a binary halftone image with the same size (512×512) as the gray-valued host image as the recovery data, thus yielding a compression ratio of 8. Therefore, their recovery data have $512 \times 512 \times 1 = 262{,}144$ bits. Because their method has a small amount of embedding data, its watermarked image quality is good, but the recovery ability is not good enough to resist cropping attack of an extensive area. Wu and Chang's method [13] backs up three copies of the recovery data, which are the results of edge detection (one bit per group of 4×4 pixels). The compression ratio is $(4 \times 4 \times 8)/1 = 128$, and recovery data size is $((512 \times 512)/(4 \times 4)) \times 3 = 49{,}152$ bits. The advantage of [13] is that the resulting image is a JPEG-compressed image and can be used directly in transmission. However, the recovered image is slightly worse than the proposed method because the recovery data are the result of edge detection.

Our experiments used a codebook with 16 384 codewords to encode each 4×4 block, and hence the index value of each block had $(\log_2 16384) = 14$ bits. The compression

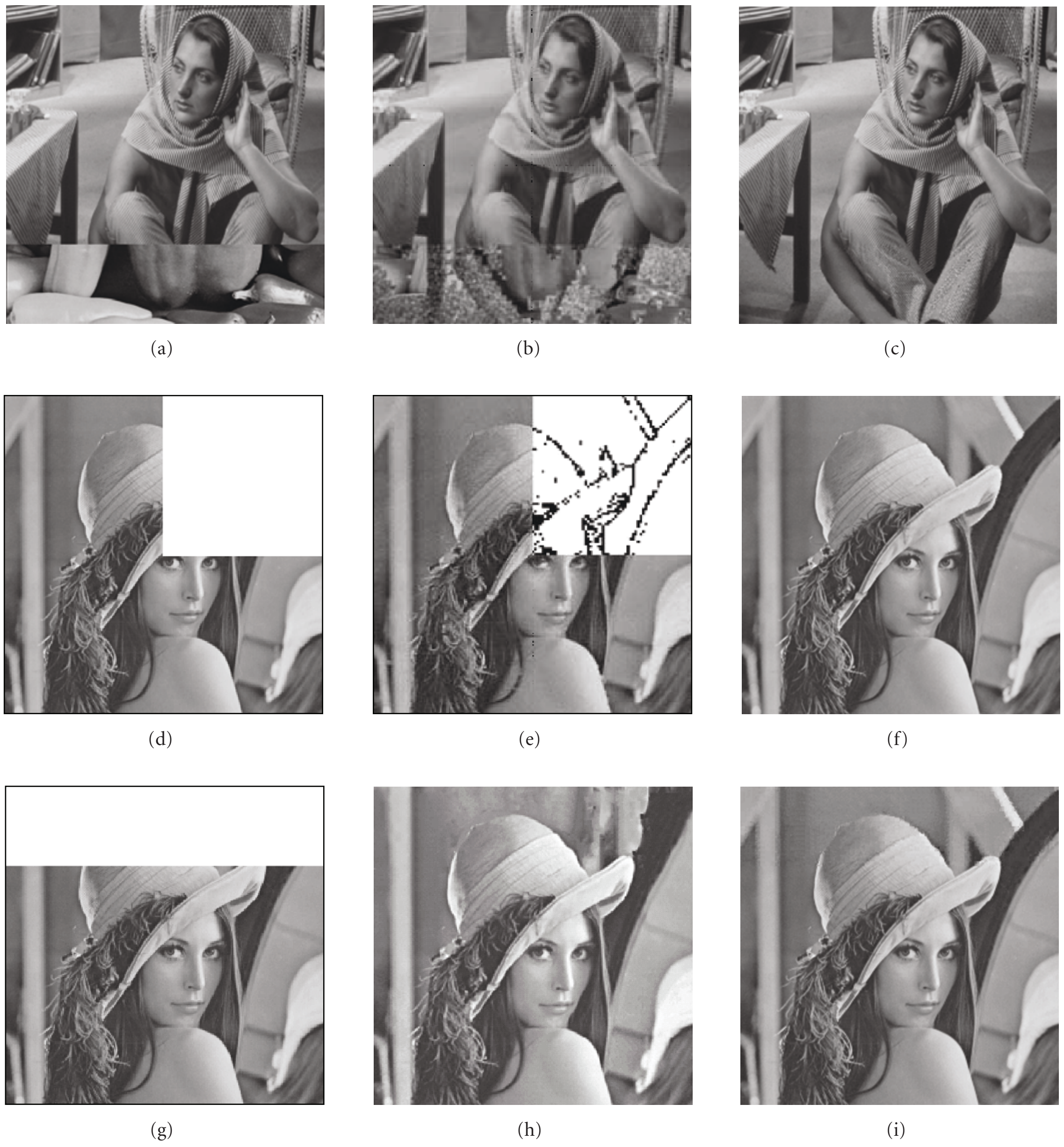


Figure 8: A comparison of the recovered images of [10, 13, 14] and the proposed method, with tampering or cropping in the same region: (a) 25% of a 512 × 512 watermarked image Barbara$^{(W)}$ is replaced; (b) the recovered image of [10]; (c) the recovered image of the proposed method; (d) 25% of a 512 × 512 watermarked image Lena$^{(W)}$ is cropped; (e) the recovered image of [13]; (f) the recovered image of the proposed method; (g) 25% of a 256 × 256 watermarked image Lena$^{(W)}$ is cropped; (h) the recovered image of [14]; (i) the recovered image of the proposed method.

rate was $(4 \times 4 \times 8)/\log_2 16384 = 9.14$. After $(r = 2, n = 3)$ threshold sharing, each block's VQ index had three index shares, so each share had $\lceil(\log_2 L)/r\rceil = \lceil(\log_2 16384)/2\rceil = 7$ bits. The total amount of recovery data was $((512 \times 512)/(4 \times 4)) \times 7 \times 3 = 344\,064$ bits for a 512×512 image (or $((256 \times 256)/(4 \times 4)) \times 7 \times 3 = 86\,016$ bits for a 256×256 image).

In general, the recovery ability is related to the amount of the recovery data embedded in the protected image. In case of 256 × 256 host images, the amount of recovery data in our method was 86 016 bits after (2, 3) threshold sharing (57 344 bits before sharing) and was 98 304 and 76 186 bits in [9, 14], respectively. Although our method did not have the largest data size, its recovery ability was most competitive. This is because the (r, n) threshold sharing (Section 3.1) and the wide scatter manner (Section 3.3) were used to scatter the recovery data in a distributed and missing allowable manner.

In case of 512 × 512 host images, our recovery data had 344 064 bits after sharing (229 376 bits before sharing), while [10, 13] had 262 144 and 49 152 bits, respectively. The proposed method had a larger data size than [10, 13], so it is not surprising that our recovery quality is better

Table 2: Comparison of the size of recovery data.

Method	Compression ratio (C.R.) of the recovery data	Total amount of the recovery data (counting the copies)	Manner of preserving the recovery data
[9]	Represent each 2×2 block by 6 bits C.R. = $(2 \times 2 \times 8)/6 = 5.33$	$((256 \times 256)/(2 \times 2)) \times 6 = 98,304$ bits	Backup in another block
[10]	Represent each pixel by one bit C.R. = 8	$512 \times 512 = 262{,}144$ bits	Backup in another pixel
[13]	Represent each 4×4 block by 1 bit C.R. = $(4 \times 4 \times 8)/1 = 128$	$((512 \times 512)/(4 \times 4)) \times 3 = 49{,}152$ bits	Three backup copies of the recovery data are embedded in another 3 blocks
[14]	Represent each 4×4 block by 31 bits C.R. = $(4 \times 4 \times 8)/31 = 4.13$	$((256 \times 256)/(4 \times 4)) \times 31 \times 0.6 = 76{,}186$ bits	Backup in another block
Ours	Represent each $\times$ 4 block by 14 bits C.R. = $(4 \times 4 \times 8/14) = 9.14$	For 256×256 image: Before sharing: $((256 \times 256)/(4 \times 4)) \times 14 = 57{,}344$ bits After (2, 3) sharing: $((256 \times 256)/(4 \times 4)) \times (14/2) \times 3 = 86{,}016$ bits For 512×512 image: Before sharing: $((512 \times 512)/(4 \times 4)) \times 14 = 229{,}376$ bits After (2, 3) sharing: $((512 \times 512)/(4 \times 4)) \times (14/2) \times 3 = 344{,}064$ bits	n shares are embedded in blocks of n distinct regions

than [10, 13] (see Figure 8). Nonetheless, because the (r, n) threshold sharing was used to distribute the more detailed recovery data, the total size of all n shares for a VQ index was only $n \times \lceil(\log_2 L)/r\rceil$ bits ($3 \times 7 = 21$ bits), rather than $n \times L$ ($= 3 \times 14 = 42$) bits as used in the traditional approach, in which $n = 3$ copies of the recovery data are made directly. This is why the watermarked image of our method still has a good PSNR (44.1 dB) quality, after embedding the detailed recovery data.

Some negative properties of the proposed scheme should be mentioned as follows.

(a) The codebook used in the VQ compression procedure must be available during tamper detection and recovery. (This codebook can be public.)

(b) The legal receiver must know the parameters used in the encoding procedure, that is, the values of r, n, SK, image size (Height and Width), and block size.

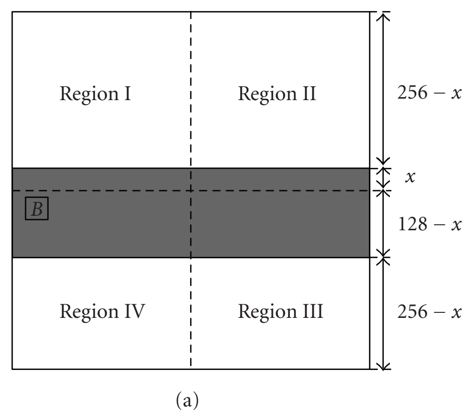
(c) Due to the collision of the hashed sequence, a nonsecure pass (in which a tampered block is treated as authentic block) might occur. A hierarchical check system (see the ending paragraph of Section 4.1) can avoid this problem, but it creates the chance of false alarm (in which a nontampered block is considered as a tampered block). Nonetheless, the misalarmed block is replaced by its recovery data, which can still yield an acceptable look for this block, since our recovery data cover enough information about the block.

6.3. Security analysis of the recovery data

If an image block is marked as tampered, then the recovery data embedded in it can no longer be used for recovery. Let us discuss the situation when an attack damages an extensive area of the watermarked image. It suffices to use Figure 8(a) or Figure 8(g), where the damaged area is one horizontal quarter of the image, as an example to show how to analyze the recovery rate of the proposed method. The (r, n) threshold is (2, 3) in Figure 8; hence, to recover a block B in the damaged area needs any $r = 2$ of its $n = 3$ index shares distributed in other three regions (i.e., Regions II, III, and IV, if block B is in Region I).

The recovery rate of the proposed method is 100% for Figure 8(a) or Figure 8(g), because two of the four regions {I–IV} are never touched (see the four regions shown in Figure 2), which means that two authentic shares required to rebuild block B can always be obtained from these two untouched regions. Therefore, some attackers might try to scratch an area across all Regions I–IV, but they can only achieve this when they know how the image was partitioned into $(n + 1)$ regions. (An image can be partitioned into four regions in many ways besides the one shown in Figure 2; e.g., Region I could be a long vertical bar in the leftmost quarter, with Regions II, III, and IV obtained by dividing the remaining area into three horizontal bars. Notably, the number of possible partitions increases significantly if each region itself is allowed to be an unconnected set.) To continue the analysis, still assume the partition of regions is as in Figure 2.

Then, even if the scratched horizontal quarter belt (the dark area in Figure 9(a)) happens to touch all four regions, a tampered block B can still be recovered as long as $r = 2$ of the



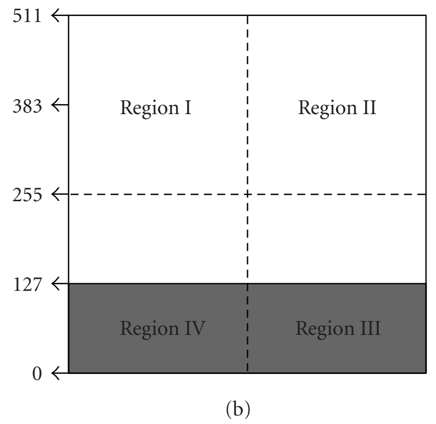


FIGURE 9: Analysis of 25% horizontal cropping for a 512 × 12 image. Dark area is tampered.

$n = 3$ support (recovery) shares are in the remaining 75% of the image (i.e., the white part in Figure 9(a)). The recovery rates are distinct between the lower pixels (intersection of dark area with Regions III and IV) and the upper pixels (intersection of dark area with Regions I and II) of the quarter belt. More specifically, the recovery rate is

$$P_L = \frac{1}{(256)^3}\left[(256)^3 - (256)^2x + 512x^2 - 2x^3\right] \tag{7}$$

for each lower pixel and

$$P_U = \frac{1}{256^3}\left[\frac{3}{4}(256)^3 + \frac{1}{2}(256)^2x - 256x^2 + 2x^3\right], \tag{8}$$

for each upper pixel. For the dark quarter belt shown in Figure 9(a), there are 512/4 = 128 rows of pixels. Among them, x rows of the dark pixels are in the upper plane, and $(128-x)$ rows are in the lower plane. Therefore, in Figure 9(a), the recovery rate for a dark-area pixel is

$$\begin{aligned}&\frac{128-x}{128}P_L + \frac{x}{128}P_U\\ &\quad = \frac{1}{(256)^3}\left[(256)^3 - \frac{3}{2}(256)^2x + 1280x^2 - 8x^3 + \frac{1}{32}x^4\right],\end{aligned} \tag{9}$$

after certain evaluations. Now, as the value of x varies from 0 through 128, the average probability to recover a dark area pixel is the integration of the above equation on the range from $x = 0$ to $x = 128$, followed by a division over the range length 128–0 = 128, which is

$$\begin{aligned}&\frac{1}{128} \times \frac{1}{(256)^3}\left\{(256)^3x - 49152x^2 + \frac{1280}{3}x^3\right.\\ &\qquad\left. - 2x^4 + \frac{1}{160}x^5\right\}\Bigg|_{x=0}^{x=128} = 89.\overline{16}\%.\end{aligned} \tag{10}$$

Finally, consider that there are 1/3 = 33.3% (or 2/3 = 66.6%) chances that the quarter belt will (will not) cross all four regions {I–IV}; the recovery rate of a tampered pixel for the quarter-belt tamper is

$$\left(\frac{2}{3} \times 100\%\right) + \left(\frac{1}{3} \times 89.\overline{16}\%\right) = 96.39\%. \tag{11}$$

To see why there exists 1/3 = 33.3% chance that the quarter belt will cross all four regions {I–IV}, we can use Figure 9(b) to explain. It is obvious that if the dark horizontal belt gradually goes through the whole 512 × 512 image from bottom to top, with the constraint that the whole belt must stay in the image, then the upper boundary of the 128-row dark belt goes from the y-coordinate value 127 through 511 (assuming that the lowest line of the image has coordinate $y = 0$). Obviously, the quarter belt will touch all four regions {I–IV} when the dark belt's top row hits the 512 × 512 image's $y = 256$ line ($y = 257$, $y = 258, \ldots, y = 382$, or $y = 256 + 128 - 1 = 383$). Therefore, the probability that all four regions {I–IV} are touched by the belt is $(383 - 256 + 1)/(511 - 127 + 1) \approx 1/3$.

Below we discuss how we got the formulas for P_L and P_U above. Without the loss of generality, we only prove the formula for P_L. Because of the symmetry (left versus right), we may assume that the damaged block B is in Region IV, as shown in Figure 9(a). (The recovery rate for the case that block B is in Region III is identical to the recovery rate for the case that B is in Region IV.) Now, as stated earlier, block B can be recovered as long as any $r = 2$ of its $n = 3$ supporting shares (stored in {Regions I, II, III}, resp.) are in the white part of Figure 9(a). Now, according to sequences I, II, III, there are four subcases; this requirement is satisfied. They are (Dark, White, White), (White, Dark, White), (White, White, Dark), and (White, White, White). For example, (Dark, White, White) means that the supporting share in Region I is in dark area, but the supporting shares in Region II and III are both in white area. The probabilities for these four cases are

$$\begin{gathered}\left(\frac{x}{256} \times \frac{256-x}{256} \times \frac{128+x}{256}\right),\\ \left(\frac{256-x}{256} \times \frac{x}{256} \times \frac{128+x}{256}\right),\\ \left(\frac{256-x}{256} \times \frac{256-x}{256} \times \frac{128-x}{256}\right),\\ \left(\frac{256-x}{256} \times \frac{256-x}{256} \times \frac{128+x}{256}\right),\end{gathered} \tag{12}$$

respectively.

Summing up these four terms, we get

$$\begin{aligned}P_L &= 2\left(\frac{x}{256} \times \frac{256-x}{256} \times \frac{128+x}{256}\right)\\ &\quad + \left[\left(\frac{256-x}{256} \times \frac{256-x}{256}\right) \times \left(\frac{128-x}{256} + \frac{128+x}{256}\right)\right]\\ &= \left(\frac{256x+2x^2}{256^2} + \frac{256^2-256x}{256^2}\right) \times \frac{256-x}{256}\\ &= \left(\frac{2x^2+256^2}{256^2}\right) \times \frac{256-x}{256}\\ &= \frac{1}{(256)^3}\left[(256)^3 - (256)^2x + 512x^2 - 2x^3\right].\end{aligned} \tag{13}$$

The analysis for P_U is similar. Notably, increasing the value of n increases the recovery rate for a fixed r but reduces the PSNR quality of the watermarked image.

6.4. Recovery in case of multiple users

The method described in Sections 3 and 4 can be regarded as a recovery work for a single user who owns the image. The recovery data of the image is uniformly embedded into the protected image itself. A single user can perform the recovery work alone.

On the other hand, the case of multiple users can also be considered. For example, consider a case where a team of four members have to produce a new product according to its blueprint. First, divide the blueprint into ($n + 1 = 3 + 1 = 4$) nonoverlapping regions as shown in Figure 2, and each member holds one region. Then, for each 4×4 block of Region I, use ($r = 2$, $n = 3$) sharing to generate $n = 3$ index shares (recovery data) and embed, respectively, the three index shares in other regions (i.e., Regions II, III, and IV). Perform the analogous process for each block of each region. By this, each member knows nothing about the complete look of the new product and thus cannot leak the information to a competitor. On the other hand, if one of the four members is absent in the team meeting, then the region held by the absent member can still be reconstructed through the mutual support of other members. The recovery procedure is similar to Section 4.2. In fact, since $r = 2$, any two of the four members (e.g., members I and II) can meet together to handle the team work by roughly reconstructing the other two quarter images owned by the other two team members (III and IV) in order to review the approximated look of the whole product.

The compression ratio of the recovery data in this multiple-users case is $(4 \times 4 \times 8)/\log_2 16384 = 9.14$. If one member transmits the recovery data of the absent region to another member, for instance, in the previous example ($r = 2$, $n = 3$), if member I wants to transmit data to member II in order to help member II to construct Region III), then the amount of transmitted data is $((512 \times 512)/(4 \times 4) \times 7) \times (1/4) = 28{,}672$ bits for a 512×512 blueprint. Of course, this amount is doubled if member I wants to help member II to build up both Regions III and IV. On the other hand, each member holds the recovery data of other three regions, which are embedded in the held region, and the total data size is $[((512 \times 512)/(4 \times 4) \times 7) \times (1/4)] \times 3 = 86{,}016$ bits. Anyway, we can increase the application of the proposed method by the mutual cooperation of multiple users. In the above ($r = 2$, $n = 3$) case, the recovery work is performed by any two team members instead of a single user.

7. CONCLUSIONS

The study proposes a watermarking method for image authentication, and it is with good self-recovery ability. The proposed method has the following functions: (1) detecting whether the watermarked image is tampered, (2) indicating the locations of the tampered area, (3) self-recovering the tampered portion using the nontampered portion of the same watermarked image, and (4) enhancing the recovery ability by utilizing (r, n) threshold sharing [18], followed by scattering the shares all over the image.

Feature (4) above gives the proposed method a good recovery rate. The sharing polynomial of Thien and Lin's method [18], which was devised to share a secret image among several participants, is used to reduce the amount of recovery data without significantly degrading the visual quality of the watermarked image. Experimental results (Figures 5–8) and the comparison Tables 1 and 2 show that the proposed method is competitive.

ACKNOWLEDGMENTS

The work was supported by the National Science Council, Taiwan, under Grant no. NSC972221-E-009-120-MY3. The authors would like to thank the reviewers and editor for their valuable comments to improve this work.

REFERENCES

[1] C.-S. Lu, S.-K. Huang, C.-J. Sze, and H.-Y. M. Liao, "Cocktail watermarking for digital image protection," *IEEE Transactions on Multimedia*, vol. 2, no. 4, pp. 209–224, 2000.

[2] C.-S. Lu and H.-Y. M. Liao, "Structural digital signature for image authentication: an incidental distortion resistant scheme," *IEEE Transactions on Multimedia*, vol. 5, no. 2, pp. 161–173, 2003.

[3] B. Zhu, M. D. Swanson, and A. H. Tewfik, "When seeing isn't believing," *IEEE Signal Processing Magazine*, vol. 21, no. 2, pp. 40–49, 2004.

[4] C.-Y. Lin and S.-F. Chang, "A robust image authentication method distinguishing JPEG compression from malicious manipulation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 2, pp. 153–168, 2001.

[5] P. Tsai, Y.-C. Hu, and C.-C. Chang, "Using set partitioning in hierarchical trees to authenticate digital images," *Signal Processing: Image Communication*, vol. 18, no. 9, pp. 813–822, 2003.

[6] C. W. Wu, "On the design of content-based multimedia authentication systems," *IEEE Transactions on Multimedia*, vol. 4, no. 3, pp. 385–393, 2002.

[7] M. U. Celik, G. Sharma, E. Saber, and A. M. Tekalp, "Hierarchical watermarking for secure image authentication

with localization," *IEEE Transactions on Image Processing*, vol. 11, no. 6, pp. 585–595, 2002.

[8] C.-C. Chang, Y.-S. Hu, and T.-C. Lu, "A watermarking-based image ownership and tampering authentication scheme," *Pattern Recognition Letters*, vol. 27, no. 5, pp. 439–446, 2006.

[9] P. L. Lin, C.-K. Hsieh, and P.-W. Huang, "A hierarchical digital watermarking method for image tamper detection and recovery," *Pattern Recognition*, vol. 38, no. 12, pp. 2519–2529, 2005.

[10] H. Luo, S.-C. Chu, and Z.-M. Lu, "Self embedding watermarking using halftoning technique," *Circuits, Systems, and Signal Processing*, vol. 27, no. 2, pp. 155–170, 2008.

[11] P. W. Wong, "Public key watermark for image verification and authentication," in *Proceedings of IEEE International Conference on Image Processing (ICIP '98)*, vol. 1, pp. 455–459, Chicago, Ill, USA, October 1998.

[12] P. W. Wong and N. Memon, "Secret and public key image watermarking schemes for image authentication and ownership verification," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1593–1601, 2001.

[13] H.-C. Wu and C.-C. Chang, "Detection and restoration of tampered JPEG compressed images," *Journal of Systems and Software*, vol. 64, no. 2, pp. 151–161, 2002.

[14] S.-S. Wang and S.-L. Tsai, "Automatic image authentication and recovery using fractal code embedding and image inpainting," *Pattern Recognition*, vol. 41, no. 2, pp. 701–712, 2008.

[15] J. Fridrich, M. Goljan, and N. D. Memon, "Further attacks on Yeung-Mintzer fragile watermarking scheme," in *Security and Watermarking of Multimedia Contents II*, vol. 3971 of *Proceedings of SPIE*, pp. 428–437, San Jose, Calif, USA, January 2000.

[16] M. Holliman and N. Memon, "Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 432–441, 2000.

[17] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[18] C.-C. Thien and J.-C. Lin, "Secret image sharing," *Computers & Graphics*, vol. 26, no. 5, pp. 765–770, 2002.

[19] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.

[20] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[21] G. R. Blakley, "Safeguarding cryptography keys," in *Proceedings of the AFIPS National Computer Conference*, vol. 48, pp. 313–317, New York, NY, USA, June 1979.

[22] C.-C. Chang and R.-J. Hwang, "Sharing secret images using shadow codebooks," *Information Sciences*, vol. 111, no. 1–4, pp. 335–345, 1998.

[23] C.-C. Thien and J.-C. Lin, "An image-sharing method with user-friendly shadow images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 12, pp. 1161–1169, 2003.

[24] C.-C. Lin and W.-H. Tsai, "Secret image sharing with steganography and authentication," *Journal of Systems and Software*, vol. 73, no. 3, pp. 405–414, 2004.

[25] C.-C. Lin and W.-H. Tsai, "Secret image sharing with capability of share data reduction," *Optical Engineering*, vol. 42, no. 8, pp. 2340–2345, 2003.

[26] R.-Z. Wang and S.-J. Shyu, "Scalable secret image sharing," *Signal Processing: Image Communication*, vol. 22, no. 4, pp. 363–373, 2007.

[27] R. L. Rivest, "The MD5 message digest algorithm," Tech. Rep. RFC 1321, MIT Laboratory for Computer Science and RSA Data Security, Redwood City, Calif, USA, 1992.

[28] V. Klima, "Finding MD5 collisions—a toy for a notebook," Cryptology ePrint Archive, 2005/075, 2005.

[29] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Proceedings of the 25th Annual International Cryptology Conference (CRYPTO '05)*, V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, pp. 17–36, Springer, Santa Barbara, Calif, USA, August 2005.

[30] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.