Authentication and recovery of an image by sharing and lattice-embedding

Sian-Jheng Lin
Ja-Chen Lin
National Chiao Tung University
Department of Computer Science
1001 Ta Hsueh Road
Hsinchu, 300, Taiwan
E-mail: sjhenglin@gmail.com

Abstract. *Based on sharing and lattice-embedding techniques, we present an authentication-recovery method for an image. The recovery data are shared among many shadows, then lattice-embedding is utilized to embed each shadow in the discrete cosine transform domain of an 8×8 block. The proposed method can resist certain content-preserving operations such as JPEG compression, Gaussian noise, and brightness adjustment, up to a tolerance level controlled by a quantization parameter value. The method can also resist certain security attacks such as a cut-and-paste attack, a collage attack, and a vector quantization attack. Compared with previous works, the proposed method has following major advantages and novelty: (1) the method does not require prediction of the tampering trace, and the tampered blocks are always recovered as long as the number of valid blocks reaches a threshold and (2) lattice-embedding yields smaller distortion than does parity-check quantization, and the latter was often used in reported works. © 2010 SPIE and IS&T.*
[DOI: 10.1117/1.3500799]

1 Introduction

It is easy for attackers to modify public digital media, and many authentication techniques,[1–8] called watermarking, have been reported recently to protect digital media. These techniques can check the correctness of the media content. Recently, some fragile watermarking schemes for tampered-region detection and recovery have also been introduced.[6–8] Lin et al.[6] proposed a block-based watermarking scheme to detect and recover the tampering of an image. They used a four-level detection structure to detect the tampered blocks. For each block that is detected as having been tampered with, its recovery data is extracted from another block, which is located by a block mapping sequence. Chan and Chang[7] proposed an image authentication method based on the Hamming code, and this method had three components, namely, the Hamming code, the Torus automorphism, and bit rotation. The parity check bits for each pixel were generated by the Hamming code. The embedding locations for the parity check bits were decided by Torus automorphism, and the bit rotation was used to improve the security. Chang et al.[8] also proposed a method for authentication and recovery. However, just like all of the other mentioned methods, their watermarked image cannot be compressed by JPEG, since the hidden information cannot be extracted after JPEG compression. Zhang and Wang[9] proposed an elegant watermarking method, which can restore the tampered region without error. The method is based on reversible data hiding, which can extract the whole host image from the stego image without error (for more information about reversible data hiding, see the work[10] proposed by Alattar who used color images as host images). However, as stated in paper caption of Ref. 9, the method[9] is still a fragile watermarking method, rather than semifragile. Based on the Pascal transform, Varsaki et al.[11] proposed a semifragile watermarking with the recovery ability to deal with color images. The host is a color image, but the recovery data is a 1/16-size gray-level version of the host. The data is embedded in the Pascal domain of R, G, and B color components, respectively. As a result, the recovered region of the tampered color image is gray-leveled, rather than colored. However, as demonstrated in Fig. 8 of Sec. 4.3, the recovery ability is not so good after cropping in the central area of the image.

In general, if a watermarked image is processed by some content-preserving operations (for example, JPEG compression or Gaussian noise or brightness adjustment), the verification ability should still exist within a certain level of the operation. This type of watermarking method is said to be semifragile, and in the semifragile watermarking method proposed by Ho and Li,[12] users choose the lowest JPEG quality factor they can tolerate, and the verification data is generated and embedded in the quantized DCT domain. Their experiments demonstrated that their method could resist JPEG compression [up to a level of quality factor (QF)]. In the method of Lin et al.,[13] users also choose the lowest JPEG quality factor they can tolerate, and then the verification data is generated from the low/middle frequency of the discrete cosine transform (DCT) domain, followed by an embedding in the high-frequency domain. Their experiments showed that their method can also resist JPEG compression (up to a QF level). However, these two methods[12,13] embed only verification data, and there is no recovery data.

Some semifragile watermarking techniques also embed recovery data in the watermarked image, and the image

Paper 09243RR received Dec. 21, 2009; revised manuscript received Jul. 26, 2010; accepted for publication Aug. 25, 2010; published online Dec. 6, 2010

1017-9909/2010/19(4)/043008/14/$25.00 © 2010 SPIE and IS&T.

itself can recover the tampered regions. Lin and Chang[14] proposed two approaches to semifragile watermarking, one of which has verification ability only, while the other has both verification and recovery ability. The verification data is generated from the DCT coefficients, and the recovery data is generated from a quarter-size shrunken subimage of the host image (if recovery ability is required). Then all of the generated data is embedded in the DCT domain. Their method can resist both JPEG compression and brightness adjustment within a reasonable range. Hsieh et al.[15] also proposed a watermarking scheme with damage-recovery ability. The recovery data is calculated from the host image, and then three copies of the recovery data are embedded in the DCT domain of the host image. Their experiments showed that their method could resist JPEG compression, brightness adjustment, and contrast adjustment. Jiang and Liu[16] proposed an authentication-recovery scheme. Their verification data is a random number sequence generated by a key, and their recovery data is generated from DCT coefficients of the host image. The two sets of data are embedded in the 2 LSBs (the two least significant bits) of the image. Their experiments showed that their method could resist JPEG compression and small-area replacement of the watermarked image. Tsai and Chien[17,18] proposed a method based on discrete wavelet transform. The verification data and the recovery data is generated from low-frequency bands and then embedded in high-frequency bands. This method can resist JPEG compression and Gaussian noise.

Based on (t,n) sharing and lattice embedding, a novel semifragile watermarking method with recovery ability is proposed. The motivation of the proposed method is based on the three observations as follows. First, the (t,n) two-layer sharing, which we introduce in Sec. 2.2, can recover the embedded data, as long as no more than $\lfloor (n-t)/2 \rfloor$ of the n created shadows are damaged shadows. This is an important property for our design to recover the image. Since each of our generated shadows is embedded in a block, the watermarked image can still extract useful recovery data after tampering, as long as the percentage of validity blocks reaches a predefined threshold α. If sharing-related techniques were not used, and if, for example, traditional block-mapping sequence techniques were used instead, the recovery data of some tampered blocks would have been lost because it was hard to predict in advance which blocks would be tampered (more details are addressed in Sec. 5.1). Second, the generated shadows E_i have the ability to both verify and recover. Because both of these abilities are tied together in E_i, we need to embed only one shadow E_i (rather than two data sets) in an 8×8 host block. This simplifies the design. Finally, we use lattice embedding to replace the so-called parity-check embedding used by many methods[14,15,17,18] when data is to be embedded. The major reason for this is that lattice embedding has a smaller impact on the host image (more details are addressed in Sec. 5.2).

The remainder is organized as follows. Section 2 introduces the (t,n) two-layer sharing and lattice embedding, which is utilized in our design. Section 3 describes our method, including a mathematical property. Section 4 shows the experimental results. Section 5 discusses the difference between the proposed method and other works, and Sec. 6 is the conclusion.

We use the following notation:

n = number of created shadows in a (t,n) secret sharing
t = the threshold of (t,n) secret sharing
P_i = the ith shadow of (t,n) two-layer sharing
c = the size of P_i
E_i = the generated shadow, which is attached to P_i by (t,n) two-layer sharing
M = the step size of the lattice embedding
d_i = the ith DCT value of an 8×8 image block
ID = the image's identification number
Key = a secret key
α = a specified threshold for the percentage of valid blocks in a tampered image

2 Background Knowledge

Secret image sharing[19] and Reed-Solomon (RS) code technique[20] are briefly reviewed in Sec. 2.1; the two-layer sharing technique is introduced in Sec. 2.2, and lattice embedding is reviewed in Sec. 2.3. These techniques are used by the proposed method in Sec. 3.

2.1 *Secret Image Sharing[19] and RS Code Technique[20]*

In the sharing phase of Thien and Lin's (t,n) threshold method,[19] for each nonoverlapping t pixel values of the secret image (secret message), a related polynomial is defined as

$$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1} (\mathrm{mod} p), \quad (1)$$

where $a_0, a_1, \ldots, a_{t-1}$ are the gray values of the t pixels, and p is a prime constant. Then $f(1), f(2), \ldots, f(n)$ is evaluated and sequentially attached to the n shadows. Having processed all of the pixels in the secret image, the n shadows are generated. Since each t pixels in the secret image only contributes 1 pixel to each generated shadow, the size of each shadow is $1/t$ of the secret image.

As indicated by Preparata,[21] from the mathematical viewpoint, the encoding phase of using the sharing Eq. (1) is isomorphic to the creation of a Reed-Solomon code.[20] Therefore, by the error-correction property[20] of the RS code, if $\lfloor (n-t)/2 \rfloor$ of the n received shadows are contaminated and become malign shadows, people can still utilize the RS code decoder to decode the n received shadows, locate the malign shadows, and then extract the whole secret data correctly. Two commonly used RS code decoders are the Berlekamp-Massey decoder[22] and the Euclidean algorithm.[23] In our method, we use Euclidean algorithm[23] to locate the position of the malign shadows, or the tampered blocks, because all of the shadows are embedded in image blocks.

2.2 *A (t,n) two-layer sharing technique modified from Chang et al.[8]*

This technique can share n given data sets $\{P_i || P_i | = c, i = 1, 2, \ldots, n\}$ to produce n shadows $\{(P_i, E_i) \mid i = 1, 2, \ldots, n\}$, where constant c is the constant size of each P_i. The created file E_i is attached to P_i, where the size of each E_i is $c(n/t - 1)$ (the analysis is addressed later). In the original design by Chang et al.,[8] the n data sets $\{P_i || P_i | = c, i = 1, 2, \ldots, n\}$ can be decoded using any t received shadows. However, in our method, we always get all n shadows (but some of them may have been tampered). Therefore, certain steps of the

algorithm in Chang et al.[8] are modified so that, for all n shadows, if the number of error shadows is not more than $\lfloor (n-t)/2 \rfloor$, then all n data sets $\{P_i || P_i \models c, i = 1, 2, \ldots, n\}$ can be decoded by the modified decoder. Notably, in our method, the n data sets $\{P_i\}$ are the DCT coefficients, which are used to recover the tampered blocks. Moreover, the location of the error shadows are also the location of error blocks in the tampered watermarked image. The encoder (sharing) and the decoder (inverse-sharing) are shown in the following. Notably, in our method, the calculations of Algorithms 1A and 1B are over GF(2^{12}).

Algorithm 1A: (t,n) two-layer sharing encoder

Input: n data sets $\{P_i \,|\, i = 1, 2, \ldots, n\}$ in which each P_i has constant size c.

Output: n shadows $\{(P_i, E_i) \,|\, i = 1, 2, \ldots, n\}$.

1. *First-layer sharing*: For each P_i, we calculate B_i (each P_i and B_i is treated as a vector, and $|P_i| = |B_i| = c$ for each i) by the following equation:

$$B_i = P_1 + 2^i P_2 + 3^i P_3 + \cdots + n^i P_n.$$

Then we collect and store the $n-t$ vectors $\{B_1, B_2, \ldots, B_{n-t}\}$ in file B.

2. *Second-layer sharing*: For each t digits in file B, we encode the t digits by (t,n) secret image sharing[9] (Sec. 2.1) to get an n-digits codeword. The n digits are dispersed, respectively, to n shadows $\{E_i \,|\, i = 1, 2, \ldots, n\}$. The construction of $\{E_i\}$ is thus done when all of the digits of file B are shared.

Algorithm 1B: (t,n) two-layer sharing decoder

Input: n received shadows $\{(\tilde{P}_i, \tilde{E}_i) \,|\, i = 1, 2, \ldots, n\}$, but some shadows within them may have been tampered.

Output: If the number of tampered shadows is no more than $\lfloor \{(n-t)/2\} \rfloor$, then we output the n error-corrected data sets $\{P_i \,|\, i = 1, 2, \ldots, n\}$ and the verification result.

1. Reconstruct the file B from $\{\tilde{E}_i | i = 1, 2, \ldots, n\}$. Here, if the number of tampered $\tilde{E}_i$ is no more than $\lfloor \{(n-t)/2\} \rfloor$, then the file B can be reconstructed by the RS code decoder, and we can also identify locations of the tampered shadows $\{(\tilde{P}_{j[i]}, \tilde{E}_{j[i]}) \,|\, i = 1, 2, \ldots, v\}$. We can mark the location of these tampered shadows as "tampered," then output the verification result. (However, if the number of tampered $\tilde{E}_i$ is more than $\lfloor \{(n-t)/2\} \rfloor$, the file B cannot be reconstructed, so we would stop the procedure in this case.

2. Let the un-tampered shadows be$\{(\tilde{P}_{j[i]}, \tilde{E}_{j[i]}) \,|\, i = 1, 2, \ldots, n-v\}$ where $\{j[i] | i = 1, 2, \ldots, n-v\}$ are indices of the un-tampered shadows. In other words, $\{j[i] \,|\, i = 1, 2, \ldots, n-v\} \cup \{\tilde{j}[i] \,|\, i = 1, 2, \ldots, v\} = \{1, 2, \ldots, n\}$ and $\{j[i] \,|\, i = 1, 2, \ldots, n-v\} \cap \{\tilde{j}[i] \,|\, i = 1, 2, \ldots, v\} = \Phi$, The tampered data $\{\tilde{P}_{\tilde{j}[i]} \,|\, i = 1, 2, \ldots, v\}$ can be recovered by the equation

$$\begin{bmatrix} \tilde{P}_{\tilde{j}[1]} \\ \tilde{P}_{\tilde{j}[2]} \\ \vdots \\ \tilde{P}_{\tilde{j}[v]} \end{bmatrix} = \begin{bmatrix} \tilde{j}[1] & \tilde{j}[2] & \cdots & \tilde{j}[v] \\ \tilde{j}[1]^2 & \tilde{j}[2]^2 & \cdots & \tilde{j}[v]^2 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{j}[1]^v & \tilde{j}[2]^v & \cdots & \tilde{j}[v]^v \end{bmatrix}^{-1}$$

$$\times \left(\begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_v \end{bmatrix} - \begin{bmatrix} j[1] & j[2] & \cdots & j[n-v] \\ j[1]^2 & j[2]^2 & \cdots & j[n-v]^2 \\ \vdots & \vdots & \ddots & \vdots \\ j[1]^v & j[2]^v & \cdots & j[n-v]^v \end{bmatrix} \times \begin{bmatrix} \tilde{P}_{j[1]} \\ \tilde{P}_{j[2]} \\ \vdots \\ \tilde{P}_{j[n-v]} \end{bmatrix} \right).$$

3. Output $\{P_i | i = 1, 2, \ldots, n\} = \{\tilde{P}_{j[i]} \,|\, i = 1, 2, \ldots, v\} \cup \{\tilde{P}_{j[i]} \,|\, i = 1, 2, \ldots, n-v\}$.

Two size and time issues about the generated shadow E_i are discussed next.

2.2.1 *Size of E_i*

We consider the two-layer sharing encoder, and in the first-layer sharing, since the P_i contains c digits, and the matrix in the multiplication has $(n-t)$ rows, the generated matrix $[B_1 B_2 \ldots B_{n-t}]^T$ contains $(n-t)c$ digits, which is the size of file B. In the second-layer sharing, the file B is divided into $\lceil (n-t)c/t \rceil$ sectors of t digits each, and each t digits are encoded into n digits by (n,t) RS code; and then a digit is assigned to each shadow E_i. So the size of each shadow E_i is $\lceil (n-t)c/t \rceil \approx (n/t - 1)c$.

2.2.2 *Running time of generating E_i*

In the first-layer sharing, the size of the two matrices at the equation right are $(n-t)n$ and nc, so it requires $(n-t)nc$ operations to calculate $[B_1 B_2 \ldots B_{n-t}]^T$. In the second-layer sharing, the file B is divided into $\lceil (n-t)c/t \rceil$ sectors of t digits each, and each t digits require nt operations to generate the code, so it requires $\lceil (n-t)c/t \rceil \times nt \approx (n-t)nc$ operations. Therefore, all of the operations necessary are $2(n-t)nc = \theta[(n-t)nc]$.

2.3 *Lattice Embedding*[24]

Lattice embedding[24] is an embedding method in which a secret digit can be embedded into many signals. Figure 1 is an example, in which a ternary value {0,1,2} is embedded in a pair of signals (p_1,p_2). As shown in Fig. 1, the space of host pair values (p_1,p_2) is divided into many hexagonal regions, and each region corresponds to a ternary value 0, 1, or 2 (squares, circles, and triangles are used to represent the three values). As shown in Fig. 1, in each one of the horizontal and vertical directions, the distance of contiguous hexagonal region's centers is set to our step size M. If a ternary value is to be embedded in the pair (p_1,p_2), we find the nearest region's center to (p_1,p_2) so that the nearest region corresponds to the ternary value to be embedded. Then the coordinate of the region's center is output. Later, to decode the embedded value, the region that contains the stego pair (p_1,p_2) is found, and the corresponding region value 0, 1, or 2 of the region is output. In general, if the value of M is larger, then the image quality after embedding is lower, but the hidden data is more

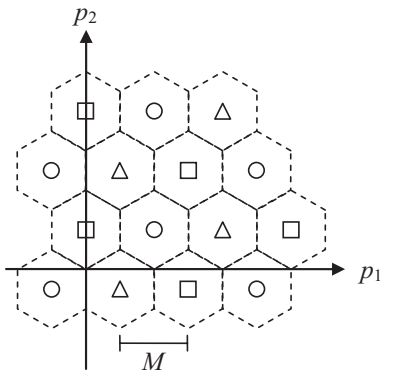


Fig. 1 Diagram of our lattice embedding.

robust. The embedding and extracting algorithms 2A and 2B are shown below.

Algorithm 2A: Embed a ternary value in a pair of signals

Input: a ternary value s, a pair of host signals (p_1,p_2), and step size M.

Output: a pair of stego-signals (p'_1,p'_2).

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} M^{-1} & -(\sqrt{3}M)^{-1} \\ 0 & 2(\sqrt{3}M)^{-1} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 + M/\sqrt{3} \end{bmatrix}$$

$$(lq_1, lq_2) = (\lfloor q_1 \rfloor, \lfloor q_2 \rfloor)$$

$$t = s - (lq_1 + 2lq_2) \,(\text{mod}3)$$

if $t = 1$ or 2 **then**

$$lq_t = lq_t + 1$$

else if $q_1 - lq_1 + q_2 - lq_2 > 1$

$$lq_1 = lq_1 + 1$$

$$lq_2 = lq_2 + 1$$

end if

end if

$$\begin{bmatrix} p'_1 \\ p'_2 \end{bmatrix} = \begin{bmatrix} M & M/2 \\ 0 & \sqrt{3}M/2 \end{bmatrix} \begin{bmatrix} lq_1 \\ lq_2 \end{bmatrix} - \begin{bmatrix} 0 \\ M/\sqrt{3} \end{bmatrix}.$$

Algorithm 2B: Extract a ternary value from a pair of signals

Input: a pair of stego-signals (p'_1, p'_2),and step size M.

Output: the ternary value s hidden in (p'_1,p'_2).

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} M^{-1} & -(\sqrt{3}M)^{-1} \\ 0 & 2(\sqrt{3}M)^{-1} \end{bmatrix} \begin{bmatrix} p'_1 \\ p'_2 + M/\sqrt{3} \end{bmatrix}$$

$$(lq_1, lq_2) = (\lfloor q_1 \rfloor, \lfloor q_2 \rfloor)$$

$$\begin{bmatrix} \Delta p'_1 \\ \Delta p'_2 \end{bmatrix} = \begin{bmatrix} M & M/2 \\ 0 & \sqrt{3}M/2 \end{bmatrix} \begin{bmatrix} lq_1 \\ lq_2 \end{bmatrix} - \begin{bmatrix} p'_1 \\ p'_2 + M/\sqrt{3} \end{bmatrix}$$

$$min = (\Delta p'_1)^2 + (\Delta p'_2)^2$$

$$b = lq_1 + 2lq_2 \,(\text{mod}3)$$

if $min > (\Delta p'_1 + M)^2 + (\Delta p'_2)^2$ **then**

$$min = (\Delta p'_1 + M)^2 + (\Delta p'_2)^2$$

$$s = b + 1 \,(\text{mod}3)$$

end if

if $min > (\Delta p'_1 + M/2)^2 + (\Delta p'_2 + \sqrt{3}M/2)^2$ **then**

$$min = (\Delta p'_1 + M/2)^2 + (\Delta p'_2 + \sqrt{3}M/2)^2$$

$$s = b + 2 \,(\text{mod}3)$$

end if

if $min > (\Delta p'_1 + 3M/2)^2 + (\Delta p'_2 + \sqrt{3}M/2)^2$ **then**

$$min = (\Delta p'_1 + 3M/2)^2 + (\Delta p'_2 + \sqrt{3}M/2)^2$$

$$s = b$$

end if

3 Proposed Method

The proposed method consists of two phases: (1) watermark generation, which generates the shadows $\{E_i\}$, followed by embedding $\{E_i\}$ in the DCT domain of the host image, and (2) tampered block detection, together with the recovery achieved by a two-layer sharing decoder (then the decoded data set $\{P_i\}$ is utilized to recover the tampered blocks).

3.1 *Watermark Generation*

Without the loss of generality, assuming that the host image is 512×512, so there are 4096 blocks of 8×8 pixels each. Then the following four steps are taken: (1) data sets $\{P_i \,|i = 1, 2, \ldots, 4096\}$ are generated for all 4096 blocks in the host image; (2) then these data sets are used to generate 4096 shadows $\{E_i|i = 1, 2, \ldots, 4096\}$; (3) to increase the security and protection, a 12-bit hashing code is generated for each block i, and then the shadow E_i is encrypted by an Exclusive-OR with the 12-bit code; and (4) finally, the (encrypted) shadow E_i is embedded in each block, and the DCT coefficients are converted into spatial domain to obtain the watermarked image. The details of the steps 1 to 1 are explained next.

3.1.1 *Generating date sets* $\{P_i \,|i = 1, 2, \ldots, 4096\}$

Each 8×8 block of the host image is converted to DCT domain, and then four low-frequency DCT coefficients are quantized by step size M (the value of M influences the robustness of the watermarked image). Figure 2 shows the positions of the four DCT coefficients $\{d_0,d_1,d_8,d_9\}$, which are colored dark gray. Then for each dc coefficient the d_0 in the 8×8 block, the difference value Δd_0 is calculated by subtracting the dc coefficient of the previous block from d_0. This step has two advantages. First, the contiguous blocks have similar dc values, so the bit length of d_0 can be reduced

d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7
d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}
d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}	d_{22}	d_{23}
d_{24}	d_{25}	d_{26}	d_{27}	d_{28}	d_{29}	d_{30}	d_{31}
d_{32}	d_{33}	d_{34}	d_{35}	d_{36}	d_{37}	d_{38}	d_{39}
d_{40}	d_{41}	d_{42}	d_{43}	d_{44}	d_{45}	d_{46}	d_{47}
d_{48}	d_{49}	d_{50}	d_{51}	d_{52}	d_{53}	d_{54}	d_{55}
d_{56}	d_{57}	d_{58}	d_{59}	d_{60}	d_{61}	d_{62}	d_{63}

Fig. 2 DCT coefficients that are selected as data P_i (dark gray) and embedded locations of E_i (light gray).

by storing the difference Δd_0. Second, when the brightness of the watermarked image is adjusted, all dc values will increase/decrease a const value, but the difference of two dc values is unchanged, so the integrity of difference Δd_0 is preserved after the brightness is adjusted. Finally, for each 8×8 block, the data P_i is calculated by

$$P_i = \left[\left(\Delta d_0 \times d_1^{\mathrm{MAX}} + d_1\right) d_8^{\mathrm{MAX}} + d_8\right] d_9^{\mathrm{MAX}} + d_9,$$

where d_i^{MAX} is the maximal d_i of all blocks, and the three values $\{d_1^{\mathrm{MAX}}, d_8^{\mathrm{MAX}}, d_9^{\mathrm{MAX}}\}$ are sent to the decoding side for the purpose of recovery. The size of P_i is $c = \log_2(\Delta d_0^{\mathrm{MAX}}\, d_1^{\mathrm{MAX}}\, d_8^{\mathrm{MAX}}\, d_9^{\mathrm{MAX}})$, which is determined by the maximal coefficients in $\{\Delta d_0, d_1, d_8, d_9\}$ among all blocks.

3.1.2 *Generating recovery data* $\{E_i \mid i = 1, 2, \ldots, 4096\}$

Generate 4096 couples $\{(P_i, E_i) \mid |P_i| = c, i = 1, \ldots, 4096\}$ by the $(t,n) = [4096(2\alpha - 1), 4096]$ two-layer sharing encoder (Algorithm 1A). Here, α is a specified threshold for the percentage of valid blocks in a tampered image. Due to embedding capacity limitation, each generated shadow E_i should have 12 bits at most. This will make $\alpha \geq 6/(c + 12)$ a requirement when percentage value α is specified. The proof is given in Sec. 3.3. On the other hand, $100\% = 1 > \alpha$ is a natural requirement. Together, the percentage value α must satisfy $1 > \alpha \geq 6/(c + 12)$.

3.1.3 *Generating the hashing code of a block*

First, a 128-bit MD5 code[25] of the block is calculated, and the formula is

$$\mathrm{MD5}\,(P_i, i, \mathrm{ID}, \mathrm{Key}),$$

where ID is the image's identification number, and key is a secret key. Then the generated 128-bit MD5 code is divided into 10 sectors of 12 bits each (the final 8 bits of the MD5 code are dropped), and the Exclusive-OR on the 10 sectors is used to get a 12-bit hashing code.

3.1.4 *Embedding shadow* E_i *in a block*

For each 8×8 block, the shadow E_i is converted into eight ternary digits, and each digit is embedded in two DCT coefficients with lattice embedding (Algorithm 2A). Figure 2 shows all 2×8 DCT coefficients which are painted light gray. A coefficient in the middle frequency and a coefficient in the low frequency are selected to form a pair of values (p_1,p_2). Here the eight pairs of values being used for (p_1,p_2) are $\{(d_{12},d_2), (d_{40},d_{17}), (d_{19},d_{10}), (d_4,d_{24}), (d_{33},d_{16}), (d_{26},d_3), (d_{11},d_{18}), (d_{32},d_{25})\}$. Then a ternary digit (0 or 1 or 2) grabbed from the shadow E_i is embedded in each (p_1,p_2) pair.

3.2 *Tampered Image Verification and Recovery*

When a tampered watermarked image is received, the following steps can generate the verification result and the recovered image.

1. Data sets $\{P_i' | i = 1, 2, \ldots, 4096\}$ are generated for all blocks in the tampered watermark image (this step is the same as Sec. 3.1.1).
2. All shadows $\{E_i' \mid i = 1, 2, \ldots, 4096\}$ are extracted from all blocks with Algorithm 2B.
3. A 12-bit hashing code is generated from each block (this step is the same as Sec. 3.1.3). Then E_i is decrypted by applying Exclusive-OR to E_i and the 12-bit code.
4. Detect the tampered blocks of the coupled-shadows $\{(P_i', E_i') \mid i = 1, 2, \ldots, 4096\}$ with $(t,n) = [4096 (2\alpha - 1), 4096]$ two-layer sharing decoder (Algorithm 1B). If the percentage of tampered blocks is less than $1 - \alpha$, the data sets $\{P_i \mid i = 1, 2, \ldots, 4096\}$ are decoded. Then output the decoded data sets $\{P_i\}$ and the verification result indicating locations of the tampered blocks. (If the percentage of tampered blocks is more than $1 - \alpha$, the warning-message is output: "Too many blocks are tampered, and hence no recovery can be done," then the procedure should be stopped without going to step 5.)
5. For the data sets $\{P_i \mid i = 1, 2, \ldots, 4096\}$, decode the four values $\{\Delta d_0,d_1,d_8,d_9\}$ by division. The step requires the three values $\{d_1^{\mathrm{MAX}}, d_8^{\mathrm{MAX}}, d_9^{\mathrm{MAX}}\}$. Then the dc value d_0 is obtained by calculating the sum of Δd_0 and the dc coefficient of the previous block.
6. In all tampered blocks, the four DCT coefficients $\{d_0,d_1,d_8,d_9\}$ should be replaced with the dequantized values in P_i.
7. For each block that passes the authentication tests, if its DCT coefficient d_0 is too far away from the dequantized value d_0 in P_i (up to a threshold), then still replace the DCT coefficient d_0 by the dequantized value d_0 extracted from P_i; this is to recover back the whole image after global adjustment of brightness.
8. The DCT coefficients in each block should be converted to the spatial domain.

3.3 *Value of* α

The value of α used in Sec. 3.1 is discussed here. In Sec. 3.1.2, we set $(t,n) = [4096(2\alpha - 1), 4096]$ where 4096 is the number of 8×8 blocks in a 512×512 image. Due to the use of the RS code, which can correct $\lfloor (n - t)/2 \rfloor$ error shadows in all n received shadows in general applications, the tolerable percentage of tampered blocks equals

$$\frac{\left\lfloor \frac{n-t}{2} \right\rfloor}{n} = \frac{\left\lfloor \frac{4096 - 4096(2\alpha-1)}{2} \right\rfloor}{4096} \approx \frac{\frac{4096 - 4096(2\alpha-1)}{2}}{4096} = 1 - \alpha.$$

In other words, if the percentage of tampered blocks exceeds $1 - \alpha$, then our method loses the ability to recover tampered blocks. Analogously, in a tampered image, α is the minimal percentage of valid blocks needed to keep the recovery ability active. In Sec. 3.1, the size of each P_i is c. On the other hand, Sec. 3.1.4 states that shadow E_i is converted into 8 ternary digits. Hence, each E_i has $|E_i| = \log_2(3^8) = 8 \log_2 3 \geq 12$ bits. Thus, by the equation $|E_i| \approx (n/t - 1)c$ of Sec. 2.2, it can be derived that $12 \leq |E_i| \leq c\,[4096/4096(2\alpha - 1)] - 1]$. Hence, $\alpha \geq (6/c + 12)$. Notably, $\alpha < 1$ is required for a recovery system to be meaningful ($\alpha = 1$ would require all blocks to be valid blocks, which means that the recovery system is too weak to tolerate even just one block being altered). Having

combined the two inequalities, a more integrated inequity should be

$$1 > \alpha \geq 6/(c+12).$$

4 Experimental Results

In this section, some experiments are undertaken to check the performance of our watermarked images.

4.1 *Robustness Test*

With the step value M being 20, four watermarked images {"Lena," "Peppers," "Jet," "Scenery"} are generated and shown in Fig. 3(a) to 3(d). From Fig. 3(a) to Fig. 3(d), the PSNR values are 34.75, 34.70, 34.80, and 34.65 dB, respectively. Then a cropping attack is applied to the four watermarked images (we crop the central 192×192 pixels of each 512×512 image). Moreover, some further distortion is made to the cropped images, as illustrated in the following. The cropped "Lena" shown in Fig. 3(e) is compressed by a JPEG with QF = 65 and the compression ratio is 8.57. The brightness of the cropped "Peppers," shown in Fig. 3(f) is increased by 30; Gaussian noises ($\sigma^2 = 6$) are added to the cropped "Jet" shown in Fig. 3(g); an 8-pixels-wide "white" horizontal bar is inserted to the cropped "Scenery" image shown in Fig. 3(h). Figures 3(i) to 3(l) shows the verification result of the four tampered images. White blocks are the places marked as "tampered" blocks. Figures 3(m) to 3(p) are the four recovered images, and Figs. 3(q) to 3(t) are the close-up versions of the recovered images. The Peak SNR (PSNR) values of four recovered images are, from left to right, 30.16, 30.96, 29.83, and 31.75 dB, respectively. More experiments for various values of M are shown in Table 1. In each row, column 1 is the value of the user-specified step value M; and column 2 is the PSNR of the corresponding watermarked image. The remaining columns are for different kinds of attacks. Column 3 is the allowed tampered ratio of watermarked image; i.e., if the tampered ratio is larger than the specified value, our method cannot recover the tampered region. Column 4 is the tolerable bound of the qualify factor (QF) of JPEG compression; i.e., if the compression uses a QF below this bound,our method cannot recover the tampered region. Column 5 is for Gaussian noise, it shows the maximal tolerable variance σ^2 of Gaussian noise. Column 6 is the tolerable range of brightness adjustment. In summary, if the attack uses a parameter value worse than the threshold values specified in Table 1, then our method cannot recover the tampered region. Should this happen, switching to a larger value of M in advance is necessary (In general, using a larger value of M in advance can increase the tolerable range of attack; but the watermarked image's quality degenerates). From Table 1, we see that our method can resist certain levels of area tampering (e.g., cropping or replacement of some areas of image), JPEG compression, Gaussian noise, and brightness adjustment.

4.2 *Security Test*

Three kinds of attack are tested on our watermarked images, which are cut-and-paste attack,[26] collage attack,[27] and vector quantization (VQ) attack.[28]

First, the so-called cut-and-paste attack is tested.[26] Figure 4(a) is our watermarked image and the PSNR value is 48.13 dB. Then a small-size pepper is copied and pasted to the lower-left corner as shown in Fig. 4(b). Figure 4(c) is the verification result, and Fig. 4(d) is the recovered image, the PSNR value of which is 40.54 dB. Second, the so-called collage attack is tested.[27] Figures 5(a) and 5(b) show our two watermarked images, "Boat" and "House," and the PSNR values are 34.63 and 34.76 dB, respectively. Then the car in Fig. 5(b) is copied-and-pasted to the same place in Fig. 5(a). This yields the image shown in Fig. 5(c). Figure 5(d) is the verification result, and Fig. 5(e) is the recovered image, the PSNR value of which is 31.88 dB. Finally, the so-called VQ attack is tested.[28] Twelve aerial images are downloaded from the USC-SIPI Image Database.[29] Each image is 512×512, and is converted to a grayscale image. Then one of them is assigned as the test image [the one shown in Fig. 6(a)]; and the remaining 11 images are watermarked by the proposed method (using $M = 4$). Then, since our method is based on 8×8 blocks, the eleven watermarked images are partitioned to get blocks of 8×8 each. These blocks are collected together and treated as a VQ codebook with many code words. With this VQ codebook, the VQ-compression-decompression version of Fig. 6(a) is shown in Fig. 6(b). Figure 6(c) is the verified result, which shows that the whole Fig. 6(b) is a fake.

4.3 *Image Quality and Our Advantage*

First, as suggested by one of the reviewers, the method of Varsaki et al.[11] was implemented, and the results are shown in Fig. 7. Figure 7(a) is the 512×512 watermarked color image "Lena," which is 40.88 dB. As shown in Fig. 7(b), the embedded recovery data is the size-reduced 128×128 gray-level version of the rotated host, the clockwise rotation is 180 deg, as suggested by Varsaki et al.[11] This 128×128 gray-level rotated version is embedded in the 128×128 blocks of the host image, and each block is 4×4. Thus, the recovery data of the rightmost bottom 4×4 block is embedded in the leftmost top 4×4 block, and the recovery data of the Southwest quadrant is embedded in the Northeast quadrant, and so on. Unfortunately, when the central 192×192 pixels of the watermarked "Lena" are cropped [the tampered image is shown in Fig. 7(c)], the recovery data extracted from the non-cropped area is as shown in Fig. 7(d). We can see that the tampered region still cannot be recovered because the recovery data of the central 192×192-pixel box was embedded earlier in the box itself. In other words, the recovery data of the cropped box is also cropped. This is very different from ours. As shown in Fig. 3, when the central 192 ×192 pixels of our watermarked "Lena" are cropped, the cropped region can still be recovered. This should come as no surprise because, according to Table 1, when $M = 20$, a moderate-size-area's tampering can be tolerated (up to 16.7% of the whole image's blocks can be tampered).

Next, because Tsai and Chien's method[17] used scaled versions of the originals as messages, then embedded the messages in the frequency domain, and also because they provided experimental results about the resistance to JPEG compression and Gaussian noise, we compare their method with ours.

Figure 8 shows the experiment. The results of Tsai and Chien[17] are shown in Figs. 8(a) to 8(d) and ours are shown in Figs. 8(e) to 8(h). Notably, the PSNR value of their

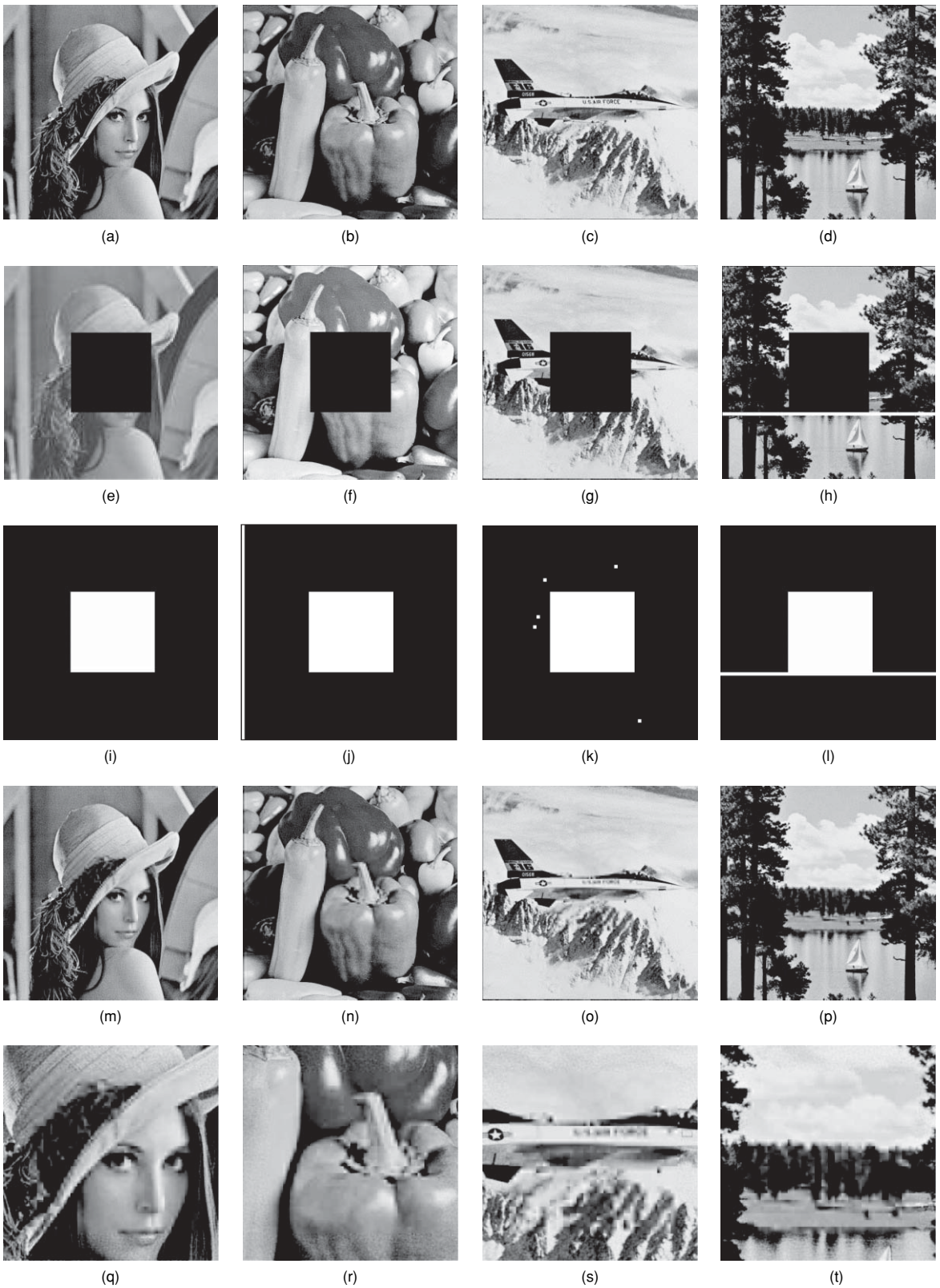


Fig. 3 Robustness test of the proposed method. (a) to (d) our four watermarked images "Lena," "Peppers," "Jet," and "Scenery"; (e) and (f) the four cropped images: (i) to (l) the corresponding verification results, after doing a JPEG compression on (e), adjusting brightness of (f), adding noise to (g), and adding white bar to (h); and (m) to (p) the recovered images; (q) to (t): close-up versions around the recover area of the recovered images (m) to (p).

Table 1 PSNR quality of watermarked image and attack-tolerance (for various quantization step value *M*). The host images are "Lena" (L), "Peppers" (P), "Jet" (J), and "Scenery" (S).

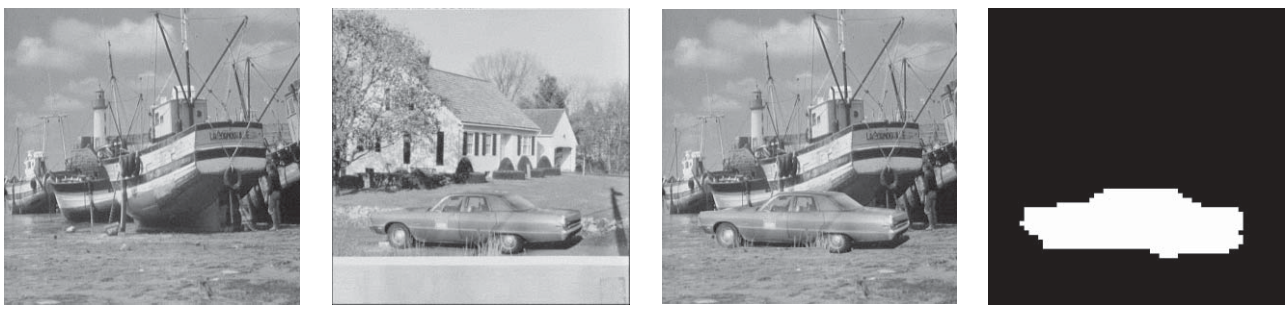
Watermarked Image		Attack Tolerance				
					Range of "Brightness Adjustment"	
Quantiz. Step Value (*M*) Used in our Algorithm	PSNR (dB) of Watermarked Image	$(1-\alpha)$, i.e., Percentage of Area Can Be Cropped or Replaced	JPEG with QF not Less than Thresholds Shown Here	Gaussian Noise (σ^2)	"Lena"; "Pepper"	"Jet"; "Scenery"
2	53.37–53.40	1/8 = 12.5%	100	0	[–35,40]; [–10,40]	[–55,30]; [–20,25]
4	48.13–48.15	1/8	94	0	[–40,50]; [–15,50]	[–70,35]; [–30,30]
6	44.83–44.85	1/8	90	0	[–40,50]; [–15,55]	[–70,35]; [–30,30]
8	42.40–42.48	1/8	86	0 – 1 (0 for J; 1 for L&P&S)	[–40,50]; [–15,55]	[–70,35]; [–30,30]
10	40.52–40.54	1/8	82	2 – 3 (2 for J&P&S; 3 for L)	[–40,50]; [–15,55]	[–70,35]; [–30,30]
12	38.95–39.01	1/8	78	4	[–40,50]; [–20,55]	[–75,35]; [–30,30]
14	37.67–37.72	1/8	75	5 – 6 (5 for L&S; 6 for J&P)	[–40,55]; [–20,55]	[–75,35]; [–30,30]
16	36.56–36.61	1/8	71	7	[–40,55]; [–20,55]	[–75,35]; [–30,30]
18	35.55–35.62	1/8	67	10	[–40,55]; [–25,55]	[–75,35]; [–30,30]
20	34.63–34.80	1/6 = 16.7%	63	13	[–45,60]; [–25,60]	[–85,35]; [–35,35]
22	33.84–33.97	1/6	59	16	[–45,60]; [–25,60]	[–85,35]; [–35,35]
24	33.10–33.24	1/6	55	19	[–45,60]; [–25,60]	[–85,40]; [–40,40]
26	32.43–32.60	1/6	51	22	[–45,60]; [–25,65]	[–85,40]; [–40,40]
28	31.80–32.00	1/6	48	26	[–45,60]; [–25,65]	[–85,40]; [–40,40]
30	31.24–31.42	1/6	45	30	[–45,65]; [–25,65]	[–85,40]; [–40,45]
32	30.70–30.87	1/6	42	34	[–45,65]; [–25,65]	[–85,40]; [–40,45]

watermarked image "Jet" in Fig. 8(a) is 30.8 dB, while ours in Fig. 8(e) is 32.29 dB. The PSNR value of the recovered image is 29.3 dB [Fig. 8(d)] for theirs, and 31.89 dB [Fig. 8(h)] for ours. Notably, Figs. 8(i) and 8(j) show the details of Figs. 8(d) and 8(h), respectively. It is observed that, between the lower-middle and lower-left of the image, there are some artifacts in their recovered snow area below the mountains, as shown in Fig. 8(i). Therefore, our recovered image is better.

Figure 9 shows the second experiment. The watermarked image "Peppers" are under both tampered attack and JPEG compression. The results of Tsai and Chien's[17] method are shown in Figs. 9(a) to 9(d). Ours are shown in Figs. 9(e) to 9(h). Figure 9(a) is their 30.6 dB watermarked image. Figure 9(b) is their tampered image [inserting a subimage compressed-decompressed by JPEG (QF = 80, and the compression ratio is 4.3). Figure 9(e) is our 32.24-dB watermarked image. Figure 9(f) is our tampered image [inserting a sub-image, then use JPEG (QF = 80, and the compression ratio is 5.4] to compress/decompress the mixed image]. Having compared the two recovered images Figs. 9(d) and 9(h), it can be seen that ours [Fig. 9(h)] has better visual quality

(a) (b) (c) (d)

Fig. 4 Cut-and-paste attack: (a) watermarked image, (b) tampered image, (c) verification result, and (d) recovered image.



(a) (b) (c) (d)

(e)

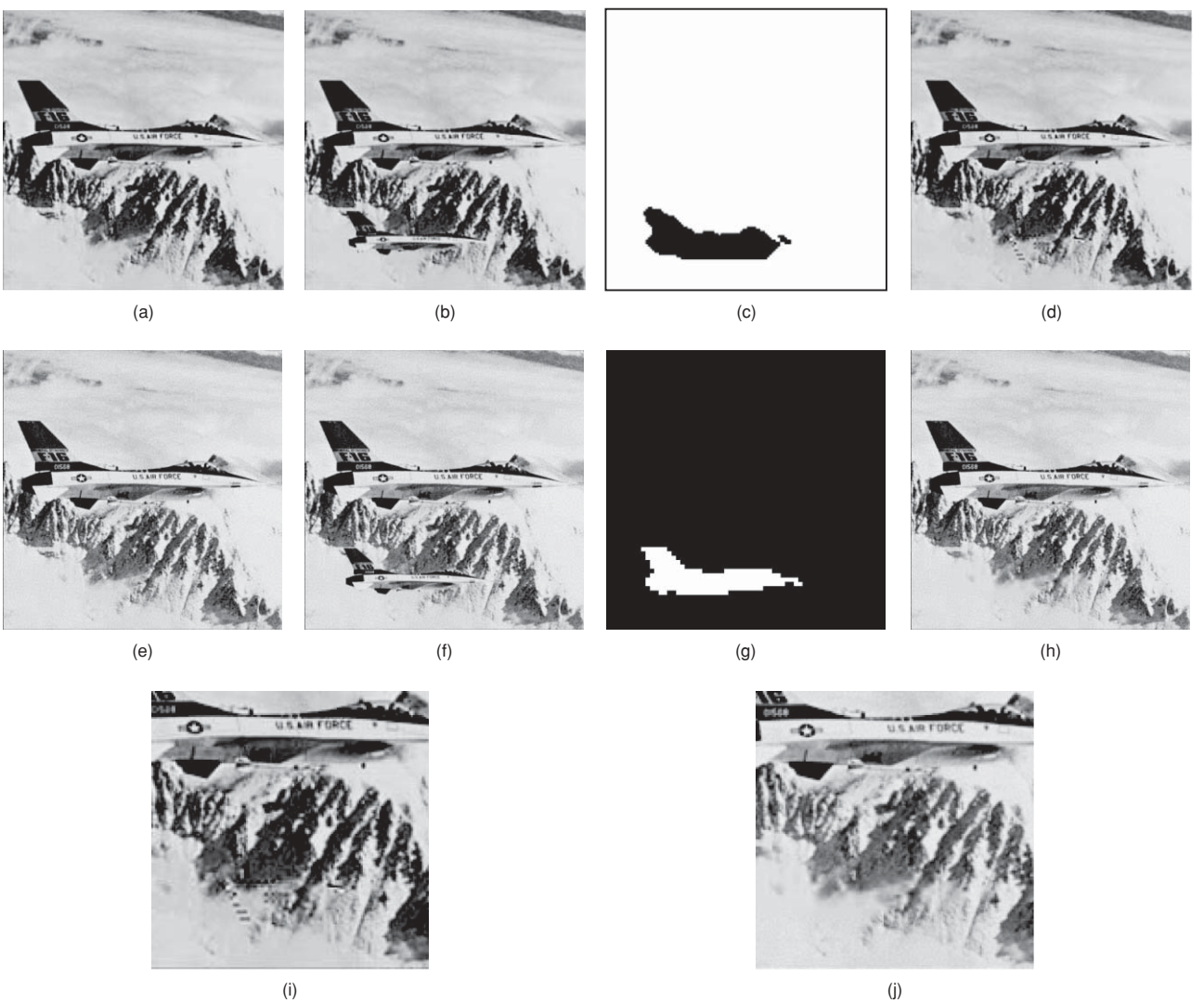
Fig. 5 Collage attack: (a) first watermarked image "Boat;" (b) second watermarked image "House," (c) collaged image in which the car in (b) is copied and pasted to the same place as (a); (d) verification result; and (e) recovered image.

(a) (b) (c)

Fig. 6 Vector quantization (VQ) attack: (a) original image, (b) VQ-attack result of (a) and (c) verification result indicates that the whole image (b) is fake everywhere.

(a) (b) (c) (d)

Fig. 7 Cropping test for Varsaki et al.'s[11] method: (a) watermarked image "Lena," (b) recovery data embedded in (a), (c) when (a) is cropped, and (d) recovery data extracted from the support of the noncropped area.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j)

Fig. 8 Experiment to compare our method with that of Tsai and Chien[17]: (a) their 30.8-dB watermarked image "Jet," (b) their tampered image, (c) their verification result, (d) their 29.3-dB recovered image, (e) our 32.29-dB watermarked image, (f) our tampered image, (g) our verification result, and (h): our 31.89-dB recovered image. Note that (i) and (j) show the details of (d) and (h), respectively. There are some artifacts in (i) on the recovered snow.

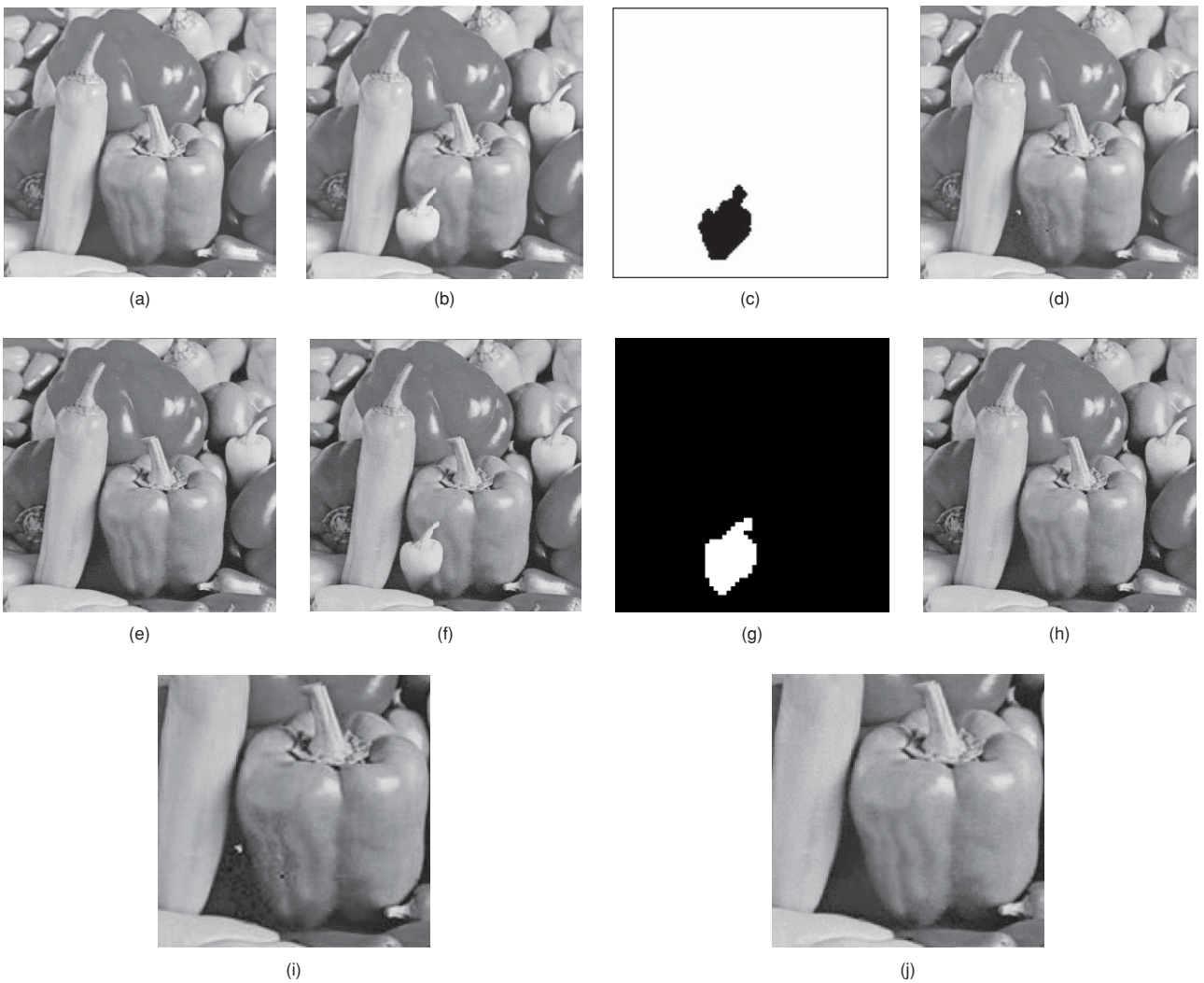


Fig. 9 Second experiment to compare our method with that of Tsai and Chien[17]: (a) their 30.6-dB watermarked image "Peppers," (b) tampering with (a) followed by JPEG compression with QF = 80; (c) their verification result; (d) their recovered image; (e) our 32.24-dB watermarked image "Peppers," (f) tampering with (e), followed by a JPEG compression with QF = 80; (g) our verification result; (h) our recovered image. Note, that (i) and (j) show the details of (d) and (h), respectively.

[because Fig. 9(d) has some noisy dots]. Details are shown in Figs. 9(i) and 9(j).

Figure 10 shows the final experiment. The watermarked image Peppers is tampered with; and then the damaged image is attacked by adding Gaussian noises. The results of Tsai and Chien[17] are shown in Figs. 10(a) to 10(d), and ours are shown in Figs. 10(e) to 10(h). Having compared the two recovered images, Figs. 10(d) and 10(h), again, it can be seen that ours [Fig. 10(h)] still has a better visual quality [because Fig. 10(d) has some noisy dots]. Details are shown in Figs. 10(i) and 10(j).

5 Comparison with Other Studies

In this section, the proposed method is compared with other studies. First, the two studies[12,13] are authentication methods without considering the issue of recovery, but ours is equipped with both authentication and recovery abilities. As for other semifragile watermarking methods[14–18] with both authentication and recovery abilities, to describe the difference between ours and those methods, each watermarking algorithm is divided into major substeps (from the perspective of methodology and system design), and then a comparison is made. The differences are described below.

5.1 *Embedding Location (i.e., Where to Embed) and Our Advantage*

In the methods of Refs. 14 to 16, the recovery data of each block is embedded in another block. For example, the recovery data of block A_0 is embedded in block A_1, the recovery data of block A_1 is embedded in block A_2, and so on. In the verification and recovery phase, if block A_0 is judged as "tampered," then the recovery data in block A_1 is extracted to recover block A_0, and so on. In this example, if blocks A_0 and A_1 are both tampered, then block A_0 cannot be recovered. In Tsai and Chien's method,[17,18] although the processing domain is the discrete wavelet domain, the

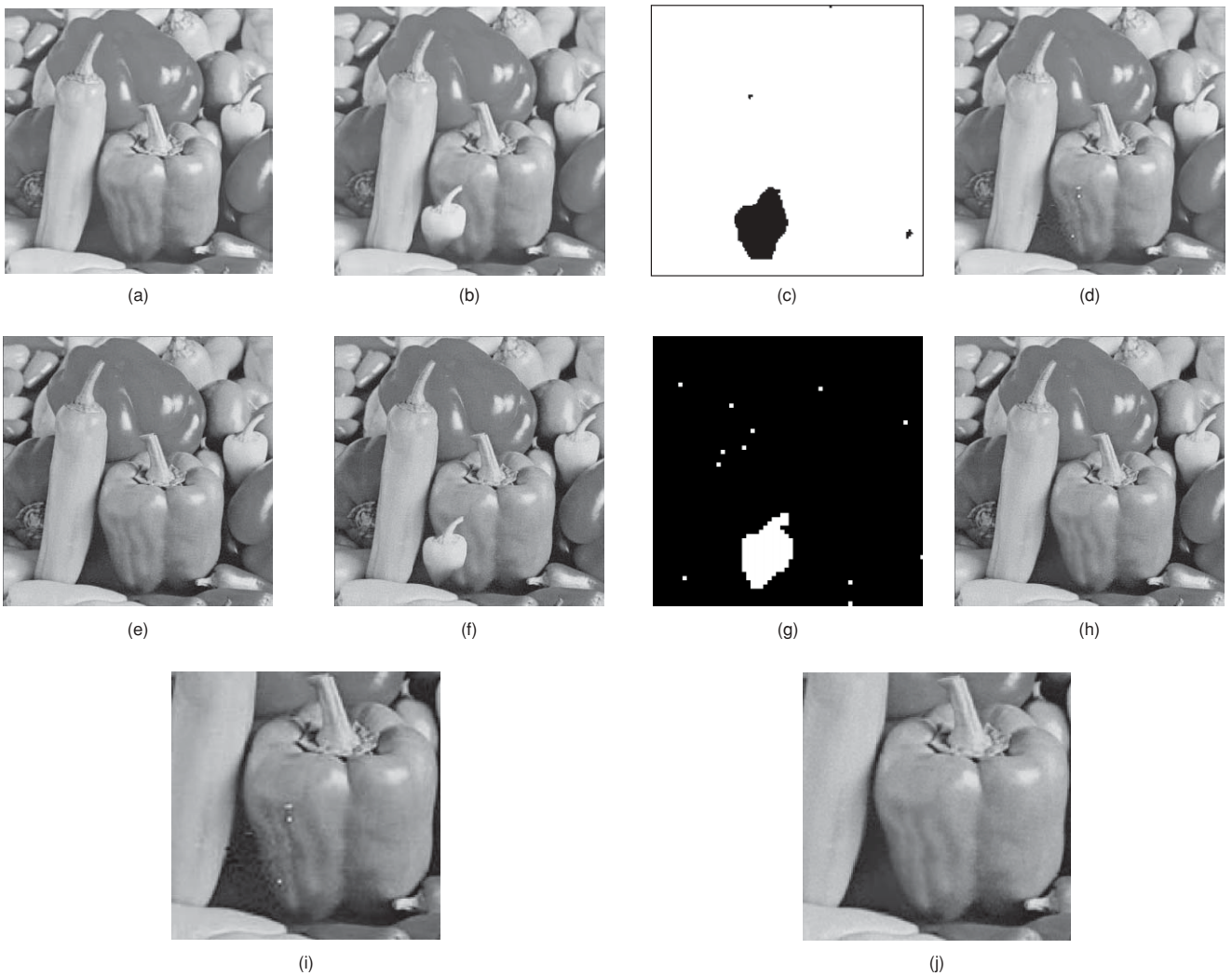


Fig. 10 The third experiment to compare our method with that of Tsai and Chien[17]: (a) their 30.6-dB watermarked image "Peppers;" (b) tampering with (a), followed by adding Gaussian noises with $\sigma^2 = 12$; (c) their verification result; (d) their recovered image; (e) our 32.24-dB watermarked image "Peppers;" (f) tampering with (e), followed by adding Gaussian noises with $\sigma^2 = 12$; (g) our verification result; (h) our recovered image (Note, (i) and (j) show the details of (d) and (h), respectively.

recovery data in low-frequency bands still needs to find some other location in high-frequency domain to undertake embedding. Therefore, if both locations are attacked, a similar recovery-disabled problem exists, although it is less severe.

However, in our method, as long as the number of valid blocks reaches a threshold, our inverse operation of the two-layer sharing can always decode the recovery data, so there is no need to consider the case that a block A_0 and the block A_1 storing its recovery data are simultaneously tampered. We only have to consider the percentage of the damaged area occupied in the whole image. As long as the damaged blocks occupy less than, say, $1/6 = 16.7\%$ of the whole image's blocks, recovery can always be undertaken. In general, it is hard to predict in advance which part would be tampered. There is no way to predict the trace of tampering, and worrying about "the percentage of blocks (in the whole image) being tampered" is simpler than worrying about "how to predict the actual location of the tampered area."

5.2 *Embedding Method (i.e. How to Embed) and Our Advantage*

Parity-check quantization (or similar works) is used in a great many research works.[14–17] As for Ref. [16], it embeds the data in least significant bits (LSBs) of the host image. Notably, 1-bit LSB embeds 1 bit/pixel, so 2 bits are embedded in two pixels, and the largest distortion for a pair of stego pixels is $(1^2 + 1^2)^{1/2} = \sqrt{2} \approx 1.414$, but in our lattice embedding, when $M = 2$ in Fig. 1, our largest distortion for two host pixels is $M/\sqrt{3} = 2/\sqrt{3} \approx 1.155$; hence smaller. As shown in Fig. 11, in 1-D parity-check quantization, the host values are

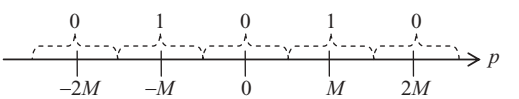


Fig. 11 Diagram of the 1-D parity-check quantization used in many research works.

divided into many regions like (–3*M*/2,–*M*/2), (–*M*/2,*M*/2), (*M*/2,3*M*/2), (3*M*/2,5*M*/2), and so on, with *M* is called the quantization level or step size. Each region corresponds to a binary value 0 or 1. If a bit is to be embedded in a host value *p*, then just find the nearest region of *p* so that the nearest region corresponds to the bit value to be embedded. Then output the central coordinate (0, or $\pm M$, or $\pm 2M$, or . . .) of the picked region as the stego value that replaces *p*. To extract the hidden secret bit, just locate the region which contains the stego value, then output the corresponding bit value 0 or 1. Our method is different, since we use lattice embedding as the embedding method. As shown in Fig. 1, the space of the host pair-values (p_1,p_2) is divided into many hexagonal regions, and the center of each region corresponds to a ternary value 0, 1 or 2. (In Fig. 1, small rectangles, triangles, and circles are used to represent the three values, respectively.) If a ternary value is to be embedded in the host pair (p_1,p_2), the nearest hexagon-center is found (i.e., small rectangle, triangle, or circle), which corresponds to the ternary value to be embedded. Then the 2-D coordinate of the hexagon-center is output. Later, to decode the embedded value, just locate the hexagonal region which contains the stego pair (p_1,p_2), then output the corresponding value 0, 1, or 2 of the region. The major reason for using lattice embedding is that the distortion of the embedding is smaller, because the hexagon-centers are denser in the plane than the square-centers. To see this, first let us inspect Fig. 12, which explains the 2-D case of parity-check quantization, i.e., it explains what happens when two pixels (p_1,p_2) are modified by parity-check quantization in order to embed two bits ($00 = 0$, $01 = 1$, $10 = 2$, or $11 = 3$) of the data. According to the location of (p_1,p_2), the nearest square-center, whose class-reading in {0,1,2,3} must coincide with the given 2-bit data, is picked and the value of (p_1,p_2) is replaced by the coordinate of the square-center. The quantization step size *M* in Fig. 12 is the same as in Fig. 1. Having compared Figs. 1 and 12, it can be seen that the distance between the hexagon-centers is smaller than the distance between the square-centers (each hexagon can be contained by a square box of size *M*-by-*M*). Therefore, when two host pixel values (p_1,p_2) are modified to embed the data, the distortion of the Lattice embedding is smaller. (As shown in Figs. 8 to 10, our watermarked images have a better image quality.)

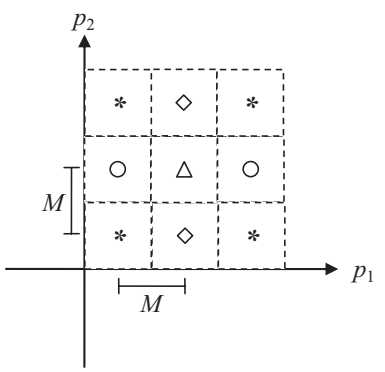


Fig. 12 Diagram to explain a 2-D case of parity-check quantization. Here, two host pixel values (p_1,p_2) are replaced by one of the centers for the purpose of embedding 2-bit data.

Of course, when Fig. 1 is adopted to replace Fig. 12, the price is that the data embedded in a hexagon can only have a data value of 0, 1, 2, but not 3. In other words, we sacrifice the size of the embedding to get a smaller distortion. However, this difficulty is overcome because a sharing technique is used in the paper here to reduce the amount of data to be embedded, along with the second benefit of increasing the recovery ability from scattered large-area tampering. In general, the size of each share is only a small portion of the original size of the data. (As shown in Figs. 8 to 10, although our watermarked images have a better image quality, our recovery ability is still very competitive.)

6 Conclusion

We proposed an authentication-recovery method. The watermarked image can be moderately altered by doing a JPEG compression, adding a Gaussian noise, or adjusting the brightness. Certain security tests, such as a cut-and-paste attack, a collage attack, and a VQ attack, were also tested. In our design, the recovery data is embedded in DCT coefficients using lattice embedding to reduce distortion. The recovery data is dispersed into many blocks by two-layer sharing. Compared with previously reported methods, our specialty is that the tampered region can be recovered as long as the percentage of the tampered blocks does not exceed a pre-defined threshold, say, 16.66%. Notably, as stated in Sec. 5.1, it is hard to predict in advance which part of a watermarked image will be cropped or replaced by attackers. The traditional mapping-sequence strategy for finding locations to hide recovery data is not a suitable strategy. This dilemma is avoided in this paper by using sharing. After all, worrying about "the percentage of blocks being tampered" is simpler than worrying about "how to predict the actual locations of tampered area."

Acknowledgments

This work is supported by the National Science Council of Republic of China, under Grant No. NSC97-2221-E-009-120-MY3. The authors also thank the reviewers and editor for their valuable comments.

References

1. C. Y. Lin and S. F. Chang, "A robust image authentication method distinguishing JPEG compression from malicious manipulation," *IEEE Trans. Circ. Syst. Video Technol.* **11**(2), 153–168 (2001).
2. 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).
3. C. C. Chang, Y. S. Hu, and T. C. Lu, "A watermarking-based image ownership and tampering authentication scheme," *Pattern Recog. Lett.* **27**(5), 439–446 (2006).
4. M. U. Celik, 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).
5. 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).
6. P. L. Lin, C. K. Hsieh, and P. W. Huang, "A hierarchical digital watermarking method for image tamper detection and recovery," *Pattern Recog.* **38**(12), 2519–2529 (2005).
7. C. S. Chan and C. C. Chang, "An efficient image authentication method based on Hamming code," *Pattern Recog.* **40**(2), 681–690 (2007).
8. Y. J. Chang, S. J. Lin, and J. C. Lin, "Authentication and cross-recovery for multiple images," *J. Electron. Imaging* **17**(4), 043007 (2008).
9. X. Zhang and S. Wang, "Fragile watermarking with error-free restoration capability," *IEEE Trans. Multimedia* **10**(8), 1490–1499 (2008).

10. A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process*. **13**(8), 1147–1156 (2004).
11. E. E. Varsaki, V. Fotopoulos, and A. N. Skodras, "Self-authentication of natural color images in Pascal transform domain," in *Proc. 16th Int. Conf. on Digital Signal Processing*, IEEE Circuits and Systems Society; Santorini, Greece, 5–7 July (2009), pp. 1–6.
12. C. K. Ho and C. T. Li, "Semi-fragile watermarking scheme for authentication of JPEG images," in *Proc. IEEE Int. Conf. Information Technology, Coding, and Computing (ITCC'04)*, Las Vegas, USA, Vol. 2, pp. 7–11 (2004).
13. C. H. Lin, T. S. Su and W. S. Hsieh, "Semi-fragile watermarking scheme for authentication of JPEG images," *Tamkang J. Sci. Eng*. **10**(1), 57–66 (2007).
14. C. Y. Lin and S. F. Chang, "Semi-fragile watermarking for authenticating JPEG visual content," *Proc. SPIE* **3971**, 140–151 (2000).
15. S. L. Hsieh, P. D. Wu, I. J. Tsai, and B. Y. Huang, "A recoverable semi-fragile watermarking scheme using cosine transform and adaptive median filter," *Lecture Notes in Comput. Sci*. **5060**, 629–640 (2008).
16. X. Jiang and Q. Liu, "Semi-fragile watermarking algorithm for image tampers localization and recovery," *J. Electron. (China)* **25**(3), 343–351 (2008).
17. M. J. Tsai and C. C. Chien, "Authentication and recovery for wavelet-based semi-fragile watermarking," *Opt. Eng*. **47**(6), 067005 (2008).
18. M. J. Tsai and C. C. Chien, "A wavelet-based semi-fragile watermarking with recovery mechanism," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 3033–3036 (2008).
19. C. C. Thien, and J. C. Lin, "Secret image sharing," *Comput. Graph*. **26**(5), 766–770 (2002).
20. I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Industr. Appl. Math*. **8**(2), 300–304(1960).
21. F. Preparata, "Holographic dispersal and recovery of information," *IEEE Trans. inf. Theory* **35**(5), 1123–1124(1989).
22. J. L. Massey, "Shift-register synthesis and BCH decoding", *IEEE Trans. Inf. Theory* **15**(1), 122–127 (1969).
23. T. K. Truong, W. L. Eastman, I. S. Reed and I. S. Hsu, "Simplified procedure for correcting both errors and erasures of Reed–Solomon code using Euclidean algorithm," *IEE Proc*. **135**(6), 318–324 (1988).
24. P. Moulin and R. Koetter, "Data-Hiding Codes," *Proc. IEEE* **93**(12), 2083–2127(2005).
25. R. Rivest, "RFC1312: the MD5 message-digest algorithm," http://tools.ietf.org/html/rfc1321 (1992).
26. P. S. L. M. Barreto, H. Y. Kim, and V. Rijmen, "Toward secure public-key blockwise fragile authentication watermarking," *IEE Proc. Vis. Image Signal Process*. **149**(2), 57–62(2002).
27. J. Fridrich, M. Goljan, and N. Memon, "Further attacks on Yeung-Mintzer fragile watermarking scheme," *Proc. SPIE* **3971**, 428–437(2000).
28. M. Holliman, and N. Memon, "Counterfeiting attacks on oblivious block-wise independent invisible watemarkhg schemes," *IEEE Trans. Image Proc*. **9**(3), 432–441(2000).
29. The USC-SIPI Image Database, http://sipi.usc.edu/database/.

Sian-Jheng Lin received his BS and MS degrees in computer science in 2004 and 2006, respectively, from National Chiao Tung University, where he is currently a PhD candidate in the Computer Science Department. His recent research interests include image processing and secret sharing.

Ja-Chen Lin received his BS and MS degrees from National Chiao Tung University, Taiwan, and his PhD degree in mathematics from Purdue University, in 1988. He is currently a professor with the Department of Computer and Information Science, National Chiao Tung University. His research interests include pattern recognition and image processing. Dr. Lin is a member of the Phi-Tau-Phi Scholastic Honor Society.