Reversible fragile watermarking for locating tampered blocks in JPEG images

Xinpeng Zhang *, Shuozhong Wang, Zhenxing Qian, Guorui Feng

School of Communication and Information Engineering, Shanghai University, Shanghai 200072, P. R. China

Abstract

This paper proposes a novel fragile watermarking scheme for JPEG image authentication. The watermark is generated by folding the hash results of quantized coefficients, and each block is used to carry two watermark bits using a reversible data-hiding method. Because modification to the cover is small, the visual quality of watermarked image is satisfactory. On the receiver side, one may attempt to extract the watermark and recover the original content. By measuring mismatch between the watermark data extracted from the received image and derived from the recovered content, the blocks containing fake content can be located accurately, while the original information in the other blocks is retrieved without any error as long as the tampered area is not extensive.

ARTICLE INFO

Article history:
Received 23 December 2009
Received in revised form
29 April 2010
Accepted 30 April 2010
Available online 7 May 2010

Keywords:
Fragile watermarking
Reversible data-hiding
JPEG image
Image authentication

1. Introduction

The purpose of fragile watermarking is to check integrity and authenticity of digital products, to locate the tampered areas and to recover the original contents [1,2]. Various fragile watermarking schemes have been developed for still images in uncompressed formats. In block-wise fragile watermarking, the host image is always divided into small blocks and the mark, e.g., a hash of the principal content of each block, is embedded into the block itself [3,4]. If the image has been changed, the image content and the watermark extracted from the tampered blocks do not match with each other, therefore the tampered blocks can be identified. In general, block-wise fragile watermarking methods are capable of detecting replacement of an extensive area. However, this type of techniques can only identify tampered blocks, but not the tampered pixels. Some pixel-wise fragile watermarking schemes have been proposed to resolve this problem, in which the watermark information derived from gray values of host pixels is embedded into the host pixels themselves [5,6]. So, tampered pixels can be identified from the absence of watermark information they carry. However, since some information derived from tampered pixel values may coincide with the watermark, localization of the tampered pixels is not complete, i.e., detection of the tampering pattern is inaccurate. In [7], a statistical mechanism is introduced into fragile watermarking, and two different distributions corresponding to tampered and original pixels are used to precisely locate the tampered pixels. In [8], the embedded watermark data are derived both from pixels and blocks, and a receiver can first identify the tampered blocks and then use the watermark hidden in the rest of the blocks to find the specific pattern of modification. Since it possesses advantages of both block-wise and pixel-wise techniques, the performance in locating tampered pixels is better than that of [7]. Furthermore, watermarking approaches that allow reconstruction of the original content in the tampered areas have been proposed [9,10]. In these methods, the main content in a region is embedded into another region of the image. After detecting the malicious
modification, the data extracted from reserved regions can be exploited to recover the principal content of tampered areas.

Fragile watermarking schemes can be integrated with reversible data hiding techniques. In reversible data hiding, some additional data are embedded into the cover signal in an invertible manner so that the original content can be perfectly restored after the hidden data are extracted. For example, the least significant digits of pixel values in an $L$-ary system can be losslessly compressed to provide a space for accommodating additional data [11]. Using the difference expansion (DE) algorithm [12], differences between two adjacent pixels are doubled so that a new LSB plane without carrying any information of the original is generated. The hidden message together with a compressed location map indicating the properties of pixel pairs, but not the host information itself, is embedded into the generated LSB plane. Since the compression rate of the location map is high, and almost every pixel pair can carry one bit, the DE algorithm can embed a fairly large amount of secret data into a host image. Moreover, various techniques have been introduced into the DE algorithm to improve the payload-distortion performance, including generalized integer transform [13], histogram shift [14], prediction of location map [15], and simplification of location map [16]. When a digital signature of the host content is embedded as a fragile watermark using a reversible data hiding technique, a receiver can detect any modification to the marked medium if the embedded watermark has been tampered, otherwise the original host data can be retrieved without error. Using a framework of reversible fragile watermarking [17], one can either locate the modified area from a tampered image or perfectly recover the original content from an authentic image. In [18], a tailor-made watermark is embedded into the host image using the DE technique. Although a malicious modification may destroy part of the embedded watermark, the tampered areas can be located and the watermark data extracted from the remaining regions can be used to restore the host image without any error.

JPEG is a widely used compression standard for transmitting and storing digital images. A number of fragile watermarking schemes have been developed for JPEG image authentication. Since the host data in the JPEG format does not provide sufficient space to accommodate the watermark, it is a challenge to exactly locate the tampered area when keeping a low distortion due to watermark embedding. For example, [19] embeds only one bit into each block of a host JPEG image. In an authentication procedure, the extracted watermark is compared with the original watermark, and a mismatch indicates the tampered blocks. Because some watermark bits extracted from the tampered blocks may coincide with the original ones, these tampered blocks cannot be detected. Using the method in [20], four watermark bits are embedded into each host block, and the embedded bits are dependent on the content of a block and its 8 neighboring blocks. Although coincidence in the tampered blocks is avoided, false alarm will occur in the neighborhood of the modified areas. In [21], all LSB of the quantized DCT coefficients in each block are replaced with the watermark data derived from a chaotic system. This way, detection of the tampered block is accurate, but distortion due to watermark embedding is high. In the above-mentioned fragile JPEG watermarking methods, the embedded watermark is not removable. That means, even though the tampered area is correctly located, the recipient can only obtain the watermarked content in reserved area, but not the original content. However, distortion introduced by watermark embedding, no matter how small it is, is unacceptable to some applications, e.g., military or medical images. So, a JPEG authentication scheme capable of exactly locating tampered blocks and recovering the original authentic content is desirable.

Reversible data-hiding techniques in JPEG images have been studied. In [22], LSB of quantized DCT coefficients corresponding to medium frequencies are losslessly compressed to provide a spare space to carry additional data. The method proposed in [23] expands the histogram of quantized DCT coefficients to produce some new histogram pairs, each of which containing an original position and an expansion position. Then, the original and expansion positions are used to represent the additional 0 and 1 respectively. In another approach, the data-hider exploits the zero-value DCT coefficients to carry data. In [24], a triplet consisting of one non-zero coefficient and two zero-value coefficients is used to accommodate one additional bit, and the first zero-value coefficient is changed according to the embedded bit. Alternatively, a high-frequency coefficient with an original zero value in each block is modified to embed several bits [25]. On the receiver side, after the embedded data are extracted, these modified coefficients are forced to zero to recover the original image. The method proposed in [26] attempts to embed the secret message in zero-value quantized DCT coefficients of medium-frequencies. Moreover, the number of nonzero coefficients that participate in the embedding process is limited so that distortion caused by data hiding is low.

This paper proposes a novel JPEG image authentication scheme, which combines fragile watermarking and reversible data-hiding technique. Having obtained the watermarked JPEG image, one can perfectly recover the original content. If some area in the watermarked image has been altered, the blocks containing fake information can be accurately located and the original content in the other blocks that are not damaged by the tampering can be retrieved as long as the tampered area is not too extensive. In this scheme, the watermark is generated by folding the hash results of quantized coefficients in image blocks as a short bit-sequence, and each block is employed to carry two watermark bits using a reversible data-hiding method. Because modification to the cover is small, the quality of watermarked image is satisfactory. On the receiver side, after extracting the watermark and recovering the original content, mismatch between the extracted watermark data and the recovered content is used to identify the blocks with incorrect restorations so that the tampered blocks are located. It can be assured that the recovered content in the rest of the image is exactly the same as the original unmarked host data.
2. Watermark embedding procedure

In the proposed watermarking scheme, a series of hash bits are obtained from the quantized DCT coefficients of each block. We fold the hash bits to reduce the amount of watermark data to be embedded. Then, the coefficients with the original zero values in each block are exploited to create a spare space to accommodate additional data. For most host blocks, the watermark data are embedded into LSB of selected coefficients and the original LSB are stored in the created space. For other blocks with simple contents, the watermark data are directly inserted by modifying only one zero-value coefficient in each block.

2.1. Watermark data generation

In JPEG compression, an image is first divided into non-overlapping $8 \times 8$ pixel blocks, and DCT is performed for each block. The DCT coefficients are then divided by the quantization steps in a quantization matrix and rounded to integers. Thus, each block becomes a two-dimensional array with $8 \times 8$ integers. Assume the host JPEG image has $N$ blocks, $B_1, B_2, \ldots, B_N$. For each block, feed its 64 integer coefficients into a hash function to calculate $M$ hash-bits, $h_{n,1}, h_{n,2}, \ldots, h_{n,M}$ ($1 \leq n \leq N$). Here, the hash function must have the property that any change on an input results in a quite different output. We may use an existing hash method or design a new one for hash-bit generation. As a simple implementation, after converting the 64 integer coefficients into a vector with $L$ bits, we generate a pseudo-random binary matrix sized $L \times M$ according to a secret key, and then use the modulo-2 product of the vector and the binary matrix as the hash result.

According to a secret key, pseudo-randomly divide the $M \cdot N$ hash-bits into a series of subsets, $S_1, S_2, \ldots, S_N$, each containing $M/2$ bits. Calculate modulo-2 sum of the $M/2$ hash-bits in each subset, and call the $2 \cdot N$ results the watermark-bits. In this way, the hash-bits are folded to the watermark-bits so that the amount of data is reduced. Fig. 1 illustrates a simple example of watermark data generation, in which the number of blocks is $N=4$, and the number of hash-bits of each blocks is $M=6$. After pseudo-random division and sum calculation, 8 watermark-bits are produced.

Note that any modification on some blocks may result in a serious change of the corresponding hash-bits. In other word, if there is any change on input, the new hash result can be viewed as a bit string irrelative to the original hash result. That means the probability of alteration on each hash-bits due to input change is $1/2$, and the collision probability of entire hash result is $2^{-M}$. In this paper, the individual hash bits, rather than the entire hash results, will be used. Since the hash-bits are distributed into different subsets, the watermark-bits calculated from the subsets would be affected. That means the pattern of affected watermark-bits can be used to identify the tampered blocks. In the following, we will embed the watermark-bits, not the hash-bits, into the cover image. Because the amount of watermark-bits is significantly less than that of hash-bits, the distortion caused by data-embedding is lowered.

2.2. Reversible data-embedding

Since energy in a block is concentrated in the low/middle frequency band, most quantized DCT coefficients corresponding to high frequencies are zeros. Therefore zigzag-scanning the $8 \times 8$ coefficient array results in a vector that often contains consecutive zeros in the tail. Fig. 2 shows the order of zigzag scanning. A coefficient block as shown in Fig. 3 is converted to [38, 5, $-3, 2, 0, -1, 2, 0, 1, -1, 0, 1, 0, 0, \ldots$]. Denote the coefficient vector as $[c_1, c_2, \ldots, c_{64}]$, and the index of the last non-zero coefficient in the vector as $i_{\text{FNC}}$. In this example, $i_{\text{FNC}}=12$. If all 64 coefficients in a block are zero, $i_{\text{FNC}}=0$.

We divide the $2 \cdot N$ watermark-bits into $N$ pairs and map the pairs to the $N$ blocks in a one-to-one manner. Embed a pair of watermark-bits into each host block. For a block with $i_{\text{FNC}} \geq 3$, pseudo-randomly select two different indices $i_1$ and $i_2$ ($1 \leq i_1, i_2 \leq i_{\text{FNC}}-1$) according to a secret key, and replace the original least significant bits of the two coefficients with two corresponding watermark-bits.

$$c_i = c_i - \text{mod}(c_i, 2) + w_t, \quad t = 1, 2 \tag{1}$$
If the original LSB of $c_i$ is zero, the value of $c'_{\text{lsbc}+1}$ is increased or decreased by 1 according to the original LSB of $c_i$, and $c'_{\text{lsbc}+2}$ is still zero. If the original LSB of $c_i$ is one, the value of $c_{\text{lsbc}+2}$ is increased or decreased by 1 according to the original LSB of $c_i$, and $c'_{\text{lsbc}+1}$ is still zero. That means the original LSB of $c_i$ and $c_i'$ are carried by $c'_{\text{lsbc}+1}$ or $c'_{\text{lsbc}+2}$, and the absolute value of $c'_{\text{lsbc}+1}$ or $c'_{\text{lsbc}+2}$ must be 1. In this way, we embed two bits by modifying only one coefficient. If $i_{\text{FNC}}$ is 63 or 64, we modify the value of $c_{\text{lsbc}}$ according to the original LSB of $c_i$ and $c_i'$.

\[
c'_{\text{lsbc}} = c_{\text{lsbc}} \cdot 4 + \text{mod}(c_i, 2) \cdot 2 + \text{mod}(c_i, 2) - 2
\]  

In other words, the original LSB of $c_i$ and $c_i'$ are carried by $c'_{\text{lsbc}}$, and the absolute value of $c'_{\text{lsbc}}$ must be more than 1. After these operations, the new value of $i_{\text{FNC}}$ must be more than 4, or equal to 4 when the original $i_{\text{FNC}}$ is 3 and $c_3$ with non-zero value is not modified.

If the original value of $i_{\text{FNC}} \leq 2$, we directly embed the two watermark-bits into $c'_{\text{lsbc}+1}$ or $c'_{\text{lsbc}+2}$ and keep the other coefficients unchanged,

\[
c'_{\text{lsbc}+1} = 1, \quad \text{if } w_1 = 0 \quad \text{and } w_2 = 0
\]

\[
c'_{\text{lsbc}+1} = -1, \quad \text{if } w_1 = 0 \quad \text{and } w_2 = 1
\]

\[
c'_{\text{lsbc}+2} = 1, \quad \text{if } w_1 = 1 \quad \text{and } w_2 = 0
\]

\[
c'_{\text{lsbc}+2} = -1, \quad \text{if } w_1 = 1 \quad \text{and } w_2 = 1
\]

Note that only one coefficient is modified. In other words, if $w_1$ is zero, the value of $c'_{\text{lsbc}+1}$ is increased or decreased by 1 according to $w_2$, and $c'_{\text{lsbc}+2}$ is still zero. If $w_1$ is one, the value of $c'_{\text{lsbc}+2}$ is increased or decreased by 1 according to $w_2$, and $c'_{\text{lsbc}+1}$ is still zero. In this case, the new value of $i_{\text{FNC}}$ must be less than 4, or equal to 4 when $c_3$ with the original value 0 kept unchanged.

For a host block as shown in Fig. 3, if the two watermark-bits to be embedded are 0 and 1, and $i_1=3, i_2=8$, the watermarked coefficient vector is \([38, 5, -4, 2, 0, -1, 2, 1, 1, -1, 0, 0, 1, 0, 0, \ldots]\), and the watermarked block is shown in Fig. 4(a). If both the watermark-bits to be embedded are 1 and $i_1=11, i_2=2$, the corresponding watermarked vector is \([38, 5, -3, 2, 0, -1, 2, 0, 1, -1, 1, 1, -1, 0, 0, \ldots]\), and the watermarked block is shown in Fig. 4(b). If the original block is shown in Fig. 5(a) with $i_{\text{FNC}}=63$ and the two watermark-bits are 10, when $i_1=25$ and $i_2=37$, the original coefficient vector \([44, 11, 1, 4, 0, 0, 0, 2, 0, 1, 0, 1, 0, 0, \ldots]\) is modified into \([44, 11, 1, 4, 2, 0, 0, 2, 0, 1, 0, 1, 0, 0, \ldots]\), and the corresponding watermarked block is given in Fig. 5(b). For another host block with $i_{\text{FNC}}=1$ in Fig. 6(a) and the two watermark-bits being 11, we can modify the original coefficient vector.
3. Authentication procedure

Suppose an adversary alters some blocks in a watermarked JPEG image to create a counterfeit. We will complete the authentication process in two phases. First, an attempt is made to extract the watermark data and recover the original block content. Since extraction and recovery from tampered blocks may be incorrect, the purpose of the second phase is to identify the correctly and incorrectly recovered blocks based on the status of match/mismatch between the extracted watermark data and the recovered blocks. We will show that severe mismatch indicates incorrect recovery, while slight mismatch means authentic original content. The criterion will be discussed in the following. Thus the receiver can accurately locate the tampered blocks and obtain the original content of the other area.

3.1. Initial attempt of watermark extraction and block restoration

In the first phase, we attempt to extract the watermark data and recover the original block content. For each block in a suspicious image, the receiver scans its 64 coefficients to form a vector. Denote the coefficient vector as \( c_0 \), \( c_1 \), \( c_2 \), \( \ldots \), \( c_{64} \), and the index of the final non-zero coefficient in the vector as \( i_{FNC} \). Attempts can be made to extract the embedded watermark data and recover the original image content according to the following four cases.

1) The first case is \( i_{FNC} = 4 \) or \( i_{FNC} = 4 \) and \( c_{i_{FNC}-1} \neq 0 \). If the absolute value of \( c_{i_{FNC}-1} \) is 1, we get two bits \( b_1 \) and \( b_2 \) according to \( c_{i_{FNC}-1} \):

\[
 b_1 = \begin{cases} 
 0, & \text{if } c_{i_{FNC}-1} \neq 0 \\
 1, & \text{if } c_{i_{FNC}-1} = 0 
\end{cases} 
\]

(5)

\[
 b_2 = \begin{cases} 
 0, & \text{if } c_{i_{FNC}-1} > 0 \\
 1, & \text{if } c_{i_{FNC}-1} < 0 
\end{cases} 
\]

(6)

and calculate

\[
 i'_{FNC} = \begin{cases} 
 i_{FNC} - 1, & \text{if } c_{i_{FNC}-1} \neq 0 \\
 i_{FNC} - 2, & \text{if } c_{i_{FNC}-1} = 0 
\end{cases} 
\]

(7)

If the absolute value of \( c_{i'_{FNC}} \) is greater than 1, we get two bits \( b_1 \) and \( b_2 \) according to \( c_{i'_{FNC}} \):

\[
 b_1 = \left\lfloor u/2 \right\rfloor 
\]

(8)

\[
 b_2 = \text{mod}(u,2) 
\]

(9)

where

\[
 u = \text{mod}(c_{i'_{FNC}} + 2, 4) 
\]

(10)

and let

\[
 i'_{FNC} = i'_{FNC} 
\]

(11)

Then, using the same key, pseudo-randomly select two different indices \( i_1 \) and \( i_2 \) (\( 1 \leq i_1, i_2 \leq i_{FNC} \)), and extract two
Furthermore, the receiver may attempt to recover the original DCT coefficients of the block in the following way,
\[
\hat{c}_i = \begin{cases} 
  c_i & \text{if } i = i_1 \\
  c_i \mod (c_i, 2) + b_1 & \text{if } i = i_2 \\
  c_i \mod (c_i, 2) + b_2 & \text{if } i = i_3 \\
  0 & \text{if } i > i_\text{FNC}
\end{cases}
\] 
(13)
and
\[
\hat{c}_\text{FNC} = \begin{cases} 
  c_\text{FNC} & \text{if } |c_\text{FNC}| = 1 \\
  \frac{c_\text{FNC} + 2}{4} & \text{if } |c_\text{FNC}| > 1
\end{cases}
\] 
(14)

For example, with a received block as shown in Fig. 4(a) or (b), the value of \(i_\text{FNC}\) is 14 or 13, respectively. Using Eqs. (5)–(7) and (11)–(14), one can extract two watermark-bits 01 and 11, and recover the original DCT coefficients as in Fig. 3. For the block in Fig. 5(b), the value of \(i_\text{FNC}\) is 63. Using Eqs. (8)–(14), one can extract two watermark-bits 10 and recover the block content as in Fig. 4(a) or (b), the value of \(i_\text{FNC}\) is 14 or 13, respectively.

2) If \(i_\text{FNC} \in \{2, 3\}\), or \(i_\text{FNC} = 4\) and \(c_3 = 0\), first calculate \(i_\text{FNC}\) using (7). Then, extract two watermark-bits
\[
\hat{w}_1 = \begin{cases} 
  0 & \text{if } c_{i_\text{FNC} - 1} \neq 0 \\
  1 & \text{if } c_{i_\text{FNC} - 1} = 0
\end{cases}
\] 
(15)
\[
\hat{w}_2 = \begin{cases} 
  0 & \text{if } c_{i_\text{FNC}} < 0 \\
  1 & \text{if } c_{i_\text{FNC}} < 0
\end{cases}
\] 
(16)
and recover the original content of the block,
\[
\hat{c}_i = \begin{cases} 
  c_i & \text{if } i \leq i_\text{FNC} \\
  0 & \text{if } i > i_\text{FNC}
\end{cases}
\] 
(17)

For the block in Fig. 6(b), the value of \(i_\text{FNC}\) is 3. We can directly extract two watermark-bits 11 using Eqs. (15) and (16), and recover the coefficients as in Fig. 6(a) using Eq. (17).

3) If \(i_\text{FNC} = 1\), extract two watermark-bits
\[
\hat{w}_1 = 0
\] 
(18)
\[
\hat{w}_2 = \begin{cases} 
  0 & \text{if } c_i > 0 \\
  1 & \text{if } c_i < 0
\end{cases}
\] 
(19)
and recover all the 64 DCT coefficients as zeros.

4) If all coefficients in a block are zero, in other words, \(i_\text{FNC} = 0\), this block must be tampered and the embedded watermark-bits are destroyed. In this case, we assign two random bits as extracted watermark-bits and regard 64 zeros as the recovered coefficients.

Table 1 lists the possible original states of a block, the new states after watermark embedding, and the corresponding cases when attempting to extract the watermark and recover the content. Obviously, the attempt of watermark extraction and block restoration is an inverse operation of data embedding. If a block in a watermarked image has not been tampered, the watermark-bit extraction and the content restoration must be correct. On the other hand, if the content of a block has been tampered, the extracted watermark-bits and the recovered content may be incorrect. In the next phase, we will locate the blocks that are not correctly recovered.

### 3.2. Tampered-block localization

As mentioned above, the malicious content replacement may result in incorrect watermark extraction and content restoration. We now identify the incorrectly recovered blocks, for which match between their hash and the extracted watermark-bits is destroyed. Although the match relation of the other blocks that are correctly recovered may also be affected by the tampering, one can still distinguish the incorrectly recovered blocks from the others based on the degree of mismatch as long as the tampering rate is small. The detailed method for identifying the incorrectly recovered blocks is described in the following.

Feed the 64 recovered coefficients of each block into the same hash function to calculate \(M\) hash-bits, called the “calculated hash-bits.” Then, divide the \(M \cdot N\) calculated hash-bits into \(2 \cdot N\) subsets according to the same secret key, and calculate the modulo-2 sum of the \(M/2\) hash-bits in each subset. Call the modulo-2 sum “calculated watermark-bits.” For a certain block, there are \(M\) calculated hash-bits that are distributed over \(M\) subsets, and the \(M\) subsets correspond to \(M\) calculated watermark-bits. That means each block maps \(M\) calculated watermark-bits. On the other hand, each watermark-bit is mapped by \(M/2\) blocks. For each block, let \(t_n\) \((1 \leq n \leq N)\) be the number of mapped watermark-bits that are not the same as the corresponding extracted watermark-bits. Actually, the parameter \(t_n\) measures the mismatch between the recovered content of a block and the corresponding extracted watermark data.

### Table 1

<table>
<thead>
<tr>
<th>Original state</th>
<th>New state after data-hiding</th>
<th>Attempt of watermark extraction and block restoration</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i_\text{FNC} = 64)</td>
<td>(i_\text{FNC} = 64,</td>
<td>(c_{64}</td>
</tr>
<tr>
<td>(i_\text{FNC} = 63)</td>
<td>(i_\text{FNC} = 63,</td>
<td>(c_{63}</td>
</tr>
<tr>
<td>(4 \leq i_\text{FNC} \leq 62)</td>
<td>(i_\text{FNC} = i_\text{FNC} + 1) or (i_\text{FNC} + 2),</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>c</td>
<td>&lt; AC: sup &lt;</td>
</tr>
<tr>
<td>(i_\text{FNC} = 3)</td>
<td>(i_\text{FNC} = 5, c_4 = 1, c_4 = 0)</td>
<td>Case 2</td>
</tr>
<tr>
<td>(i_\text{FNC} = 2)</td>
<td>(i_\text{FNC} = 4, c_4 = 1, c_4 = 0)</td>
<td></td>
</tr>
<tr>
<td>(i_\text{FNC} = 1)</td>
<td>(i_\text{FNC} = 3, c_1 = 1, c_1 = 0)</td>
<td></td>
</tr>
<tr>
<td>(i_\text{FNC} = 0)</td>
<td>(i_\text{FNC} = 2, c_2 = 1, c_2 = 0)</td>
<td></td>
</tr>
<tr>
<td>(i_\text{FNC} = 1)</td>
<td>(i_\text{FNC} = 1, c_1 = 1)</td>
<td></td>
</tr>
<tr>
<td>(i_\text{FNC} = 0)</td>
<td>(i_\text{FNC} = 0)</td>
<td>Case 4</td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Note: The parameters \(i_\text{FNC}\), \(c_1\), \(c_2\), and \(c_3\) are used to denote the states of the block, where \(i_\text{FNC}\) represents the number of watermark-bits, and \(c_1\), \(c_2\), and \(c_3\) are used to denote the content of the block.
Fig. 7 gives a simple example of mismatch measure based on the example in Fig. 1. Assume the content of the second block was tampered and the recovery of the block was incorrect. As a result, the hash-bits of the recovered block were 011000, but not the original 001011 as shown in Fig. 1. In the same way of subset division, the calculated watermark-bits were obtained. We also assume the third watermark-bit was embedded into the second block, and, because of the tampering, the watermark-bit was extracted as 0, but not the original 1. By comparing the calculated and extracted watermark-bits, the match and mismatch are indicated by 'o' and 'x', respectively. Then, we can count \( t_n \) for each block. For example, the first block maps the 4th, 2nd, 3rd, 1st, 2nd, and 7th calculated watermark-bits, and there are 2 mismatches among them. So, \( t_1 = 2 \). The second block maps the 3rd, 1st, 7th, 3rd, 6th, and 8th calculated watermark-bits, and there are totally 5 mismatches since the mismatch of the 3rd watermark-bit is counted twice. Then, \( t_2 = 5 \). The mapping relationship of the second block is highlighted with thick arrows in Fig. 7. Similarly, we got \( t_3 = 2 \) and \( t_4 = 2 \). It can be seen that \( t_n \) of the tampered block is significantly larger than those of the other blocks. Actually, we will identify the incorrectly recovered blocks from the value of \( t_n \).

Let \( \alpha \) be the tampering rate, which is the ratio between the number of tampered blocks and the number of all blocks. If the substituted fake content is independent of the original content, probability of the extracted watermark-bits differing from the original watermark-bits is \( \alpha/2 \). Also, since the recovered result of a tampered block seldom coincides with the original content, probability of the calculated hash-bits differing from the original hash-bits is also \( \alpha/2 \).

Consider a block in which restoration of the 64 DCT coefficients is incorrect. The calculated hash-bits of the block are the same as the corresponding original hash-bits with a probability 1/2 so that the calculated watermark-bits are also the same as the corresponding extracted watermark-bits with a probability 1/2. Then, the value of \( t_n \) satisfies a binomial distribution with the probability distribution function

\[
P_1(t_n = k) = \binom{M}{k} \cdot 2^{-M}, \quad k = 0, 1, \ldots, M \tag{20}
\]

Consider another case in which all 64 DCT coefficients of a block are correctly recovered. The \( M \) calculated hash-bits of the block must be the same as the original hash-bits. The \( M \) calculated hash-bits are distributed over the \( M \) subsets, in other words, each subset contains one of the \( M \) calculated hash-bit and \( M/2 \) – 1 other hash-bits. Since the other hash-bits are flipped with a probability \( \alpha/2 \), the \( M \) calculated watermark-bits will be different from the original watermark-bits with a probability

\[
e = \sum_{m \text{ is odd}} \left[ \binom{M/2-1}{m} \cdot \left( \frac{\alpha}{2} \right)^m \cdot \left( 1 - \frac{\alpha}{2} \right)^{M/2-1-m} \right] \tag{21}
\]

Furthermore, the extracted watermark-bits are different from the original watermark-bits with probability \( \alpha/2 \) due to the malicious content replacement. So, the calculated
watermark-bits of the block differ from the corresponding extracted watermark-bits with the probability

\[
E = (1 - e^{-\frac{\alpha}{2}}) + e^{-\frac{1}{2}}
\]

(22)

Then, the value of \( t_n \) follows another binomial distribution,

\[
P_O(t_n = k) = \binom{M}{k} \cdot (1 - E)^{k-1} \cdot E^{M-k}, \quad k = 0, 1, \ldots, M
\]

(23)

If the tampering rate \( \alpha \) is small, the distribution of \( t_n \) concentrates on an area near \( t_n = 0 \). Fig. 8 compares \( P_T(t_n) \) and \( P_O(t_n) \) with different \( \alpha \) for \( M = 100 \). It is shown that a large value of \( t_n \) indicates incorrect restoration, while a small value of \( t_n \) means correct restoration.

In order to identify the blocks with incorrect content restoration, for a given parameter \( M \), we find an integer \( T \) satisfying

\[
\sum_{k=0}^{T} P_T(t_n = k) < 10^{-6}
\]

(24)

and

\[
\sum_{k=0}^{T+1} P_T(t_n = k) \geq 10^{-6}
\]

(25)

Table 2 lists the values of \( T \) with different \( M \). In the scheme, if \( t_n \) is greater than the threshold \( T \), we judge the recovered content of the block being incorrect, that is, make a decision “tampered.” Otherwise, the recovered DCT coefficients are judged as the original content of the block. This way, probability of considering an incorrect restoration as original block content is less than a negligible \( 10^{-6} \), while probability of an intact block being falsely judged as tampered is

\[
P_F = \sum_{k=T+1}^{M} P_O(t_n = k)
\]

(26)

Fig. 9 shows the value of \( P_F \) with different \( \alpha \) and \( M \). The narrower the tampered area is, the smaller the value of \( P_F \). It can also be seen that the curves with different values of \( M \) mutually intersect. To achieve good performance with different tampering rates, we recommend to choose \( M = 100 \). The threshold \( T \) is set to 26 according to Table 2. In this case, it can be assured that probability of
Fig. 11. Watermarked versions with PSNR 42.1 and 37.4 dB.

Fig. 12. Tampered images.

Fig. 13. Authentication results.
falsely judging an intact block as being tampered is less than $10^{-5}$ when the tampering rate is 0.5%.

4. Experimental results

In the experiment, 100 images with different JPEG quality factors were used as the covers. Table 3 lists the average values of PSNR due to watermarking. Since a lower quality factor corresponds to larger quantization steps, watermark embedding in JPEG image with lower quality factor results in more distortion. Nonetheless, for a quality factor as low as 50, the level of distortion due to watermark embedding is still acceptable since the amount of embedded data is small.

Two host images Crowd and LAX sized 512×512 with JPEG quality factors 80 and 60 are shown in Fig. 10. Fig. 11 gives their watermarked versions. PSNR values are respectively 42.1 and 37.4 dB, and the distortion is imperceptible. We add a person and a plane into the watermarked images to create two forged images as in Fig. 12. Here, numbers of blocks containing fake information are 26 and 25 respectively. In other words, the tampering rate $x$ are 0.63% and 0.61%. Using the proposed scheme, all tampered blocks were correctly located and the original content in all the other blocks were perfectly recovered. Fig. 13 shows the authentication results, in which the recovered content is presented and the tampered areas are highlighted with extreme white.

When replacing more authentic contents with fake information, probability of false "tampered" judgment increases. That means a number of correctly recovered blocks were judged as "tampered." We recorded the number of such blocks, and call the ratio between it and the number of intact blocks as the "actual false-judgment rate." Table 4 compares the actual false-judgment rate and the theoretical value of false-judgment probability as given in Eq. (26) at different tampering rates. The experimental results are in agreement with the theoretical values. It can be seen that false alarm was avoided when the tampering rate was less than 0.63% in this experiment. On the other hand, since the threshold $T$ satisfies Eqs. (24) and (25), probability of regarding incorrect restoration as the original is less than $10^{-6}$. In fact, all blocks containing fake contents were judged as "tampered" in the experiment.

Table 5 gives a performance comparison between different fragile JPEG watermarking schemes. Values of PSNR due to watermark embedding in host images with a JPEG quality factor 80, and probabilities of falsely judging tampered block as authentic and judging authentic block as tampered are listed. In [19], only one bit is embedded into each block for image authentication. Although distortion caused by watermarking is low, only a half of tampered blocks can be detected. When increasing the amount of watermark data to avoid erroneous judgment, [21] replace the least significant bits of all quantized DCT coefficients in each block with watermark data. However, this leads to conspicuous distortion. In [20], four watermark bits are embedded into coefficients with middle frequencies of each block, and the detection results of 8 neighboring blocks are also exploited to make a "tampered" or "authentic" decision for a block. Although tampered blocks can be correctly identified, false alarm may occur for some authentic blocks in the neighborhood of the tampered areas. By using the proposed scheme, the quality of watermarked image is satisfactory, and probabilities of the two types of false judgments are very low when the tampered area is not extensive. Furthermore, the proposed scheme is capable of recovering original contents of the authentic blocks without error.

5. Conclusions

This paper proposes a novel fragile watermarking scheme for JPEG images, in which two watermark bits are embedded into each block using a reversible data-hiding scheme. On the receiver side, after attempting to extract the watermark data and to recover the original content, the number of mismatches between the watermark data extracted from the received image and derived from the recovered contents is used to judge whether a block has been tampered. If the content replacement is not serious, we can always identify the blocks containing fake contents and perfectly recover the original information of the remaining blocks. In the future, approaches capable of locating more extensive tampered area will be studied.
Acknowledgment

This work was supported by the Natural Science Foundation of China (60872116, 60832010, and 60773079), the Key Scientific Research Project of Shanghai Education Committee (10ZZ59), and the Shanghai Rising-Star Program (10QH14011).

References