Multimedia Tools and Applications (2024) 83:64441–64465
https://doi.org/10.1007/s11042-024-18165-4

Reversible image authentication using a central folding strategy with two images

Van-At Pham[1] · Ngoc-Hung Nguyen[2] · Quang-Hoa Le[3] · Kim-Sao Nguyen[4] · Thi-Luyen Cao[4] · Minh-Thai Pham[1]

Received: 2 June 2023 / Revised: 20 September 2023 / Accepted: 3 January 2024 /
Published online: 15 January 2024

Abstract
In reversible image authentication (RIA) methods, it is common to use a reversible hiding method to embed the authentication code (AC) bits into each block of an original image to form a watermarking image. Since the reversible hiding methods have low embedding capacity, the authentication methods based on them have limited attack detection ability. In this paper, we use a dual-image reversible embedding method based on the central folding strategy with some slight improvements to construct an RIA method. Due to the large embedding capacity, low image distortion, no location map to be used, and AC bits generated by using hash function MD5 on block's features, the proposed image authentication method can detect any type of attack with higher accuracy compared to existing methods. Its attack detection ability reaches 100% for block sizes of 4 × 4, 3 × 3, and 99.91% for a block size of 2 × 2 while maintaining high image quality.

Keywords Image authentication · Attack detection · Reversible data hiding

1 Introduction

Nowadays, digital documents often are transmitted on the internet network, but this is an unsafe medium, so these documents are easily tampered with by a hacker. So, protecting the integrity and the copyright of the data is an urgent problem in the field of information security. Two main approaches in this field are encryption and data hiding. Encryption approaches generate ciphertexts consisting of a series of meaningless symbols that are curious to the hackers. The data hiding methods overcome this drawback because the image containing data and the original image are difficult to distinguish. But when images containing secret data are sent on the internet, hackers can attack and modify the content of images, and the receivers can extract false information. So protecting the integrity of the image becomes very important. Image authentication is a technique to protect the integrity of images from being illegally

✉ Ngoc-Hung Nguyen
nnhung@ioit.ac.vn

Extended author information available on the last page of the article

modified. The existing image authentication methods can be divided into two categories, that are digital signature-based methods [1, 2] and fragile watermarking-based methods [3, 4]. For the digital signature methods, a signature of an image is obtained using a hash function, and the hashed result is then encrypted and stored in a trusted third party. To authenticate an image, the signature is obtained from the image and compared with the signature stored in the third party. In fragile watermarking methods, the authentication information used as watermarks embedded into the host image to obtain the watermarking image. In general, the authentication information is the feature generated from the host image or a random bit-stream obtained by a pseudo-random number generator. Because the embedded data is fragile if the watermarking image has attacked then the embedded watermarks are likely different from the extracted ones and thus the tampered regions can be defined.

The image authentication methods can be divided into two classes: irreversible methods [5–10] and reversible methods[11–15]. In the irreversible image authentication methods, the lossy data hiding techniques [16–19] often are used for embedding watermarks. Because lossy data hiding techniques often provide a large embedding capacity, the irreversible image authentication methods have a good detection ability while maintaining high watermarking image quality. However, in the lossy data hiding techniques, the original image cannot be restored exactly from a watermarking image. Therefore, the irreversible image authentication methods are not suitable for applications where the original image is demanded such as medical and military images, or for the image carrying secret data mentioned above.

In these cases, we can use the reversible image authentication (RIA) methods, they not only detect the tampered regions but also can restore the original image from the watermarking image. Reversible data hiding (RDH) techniques have been extensively developed in the past decades. The difference expansion (DE) [20] and histogram shifting (HS) [21] are the two most important techniques used in RDH techniques. DE method is the first remarkable RDH method, in which the difference of each pixel pair is expanded to the left enough for embedding one bit on the right. The method has been improved and developed by many researchers to enlarge its embedding capacity such as the methods based on pixel blocks [22–24], the methods for reducing the location map [25–28], and especially the prediction error expansion (PEE) methods [29–32]. In general, DE-based RDH methods provide a high embedding capacity, but stego image quality is still low.

Ni et al. [21] proposed another important RDH method called histogram shifting (HS). First, the histogram of pixels is generated by using a statistical method. Then the bins at which the histogram reaches the largest values are selected to embed the data. In the HS method, each pixel must be modified at most one unit, so its stego image quality is very high, however, the embedding capacity is not very large. To overcome this drawback, prediction error histogram shifting (PEHS) is proposed in [33–37], in which prediction errors of pixels are calculated, and then a histogram of errors is shifted. Recently, Li et al. [38] proposed a new RDH method based on pixel value ordering (PVO), in which firstly the host image is divided into non-overlapped blocks, and the pixels of each block are sorted in ascending order. Then the largest pixel is predicted by the second-largest pixel and the smallest pixel is predicted by the second-smallest pixel. Next, one bit is embedded in the largest pixel (or the smallest pixel) if the prediction error is 1 (or -1), respectively. The advantage of this method is that it has very high stego image quality, but its embedding capacity is still low. The PVO method has attracted the attention of many researchers and recently there have been many improvements to enlarge the embedding capacity as in [39–45].

Since the RIA methods often detect the tampered regions of an image in a block-wise manner, the block-based RDH techniques are therefore suitable to be used in the RIA methods. Lo and Hu [12] proposed an RIA method in which a sequence of authentication codes (AC) is generated by a random seed. Each 4 × 4 pixel block is used to embed an AC bit using a PEHS technique. Later, Nguyen et al. [13] proposed an RIA method based on adaptive prediction-error expansion (PEE).

Yin et al. [15] also proposed an RIA method in which the pixels of an original image are visited by the Hilbert curve and divided into non-overlapped blocks of size 4 × 4. Since four consecutive pixels in the Hilbert curve are always within a 2 × 2 sub-block, the Yin method essentially embeds the authentication bits on the 2 × 2 blocks using the IPVO technique. This method used two thresholds to classify blocks into three types: flat, normal, and rough. At most eight bits and four bits are embedded in a flat and a normal block, respectively. Whereas rough blocks are skipped, and no bit is embedded in them. Nguyen and Vo [14] used an idea similar to in [13] to create an RIA method in which an original image is divided into non-overlapped blocks with the size of 3 × 3. The pixel in the centre of the block is selected as a prediction value of pixels on the left and right sides. This method has the same stego image quality as in [13], but an embedding capacity smaller.

All four above methods used a pseudo-random bit sequence as authentication information. So the validation content will remain the same even if the pixel values have been changed. This is one reason for the inability to detect attacked blocks. Moreover, the authentication bits only are embedded in embeddable blocks. That means non-embeddable blocks will not be protected. Hong et al. [11] proposed an RIA method in which, the above two disadvantages have been overcome. This method used the same embedding algorithm as in [15], namely an original image is divided into non-overlapped blocks with the size 4 × 4, then at most eight bits are embedded in four 2 × 2 sub-blocks by the IPVO technique. Authentication code (AC) bits of a block are generated by using the hash function MD5 of the block's pixel values and location information. The non-embeddable blocks are not ignored. In each such block one AC bit is embedded by the LSB replacement technique. Due to these improvements, the method of Hong et al. has a higher ability to detect and locate tampered blocks than the aforementioned methods.

In addition to the RDH methods, dual-image RDH techniques have also been proposed, that generate two stego images after embedding data bits into a single host image. Because of using two stego images, the dual-image RDH methods have a larger embedding capacity and higher security than the RDH methods. Without two stego images being simultaneously obtained, hackers can't extract the complete secret data. A dual-image RDH scheme was first proposed by Chang et al. [46], in which a 256 × 256 modulus function matrix is used. Qin et al. [47] used Exploiting Modification Direction (EMD) to construct a dual-image RDH method in which four bits are embedded in a pixel pair. Lu et al. [48] applied the LBS matching [17] to enhance stego image quality in which four bits are embedded in a pixel pair (x, y). Lee and Huang [49] proposed a dual image RDH method using 25 orientation combinations to embed more than 4 bits in a pair of two pixels. Lu et al. [50] proposed a dual image RDH method by using the center folding strategy (CFS) to reduce the distortion of the image. Yao et al. [51] proposed an improvement of Lu et al.'s method in which an extra bit is embedded when a bit sequence embedded in a pixel consists of all bits equal to 1. Recently, Chen et al. [52] proposed an efficient dual-image reversible data hiding method by using EMD.

To correctly authenticate a pixel block, it is important to embed multiple authentication bits on the block. If the number of embedded bits is n, the conclusion that the block is attacked has an average accuracy rate of $1 - (\frac{1}{2})^n$. Since dual-image RDH methods have a very large embedding capacity, applying them in RIA schemes will achieve high detection accuracy. Peng et al. [53] proposed a new reversible image authentication scheme based on a dual image RDH method developed from Yin et al.'s irreversible hiding technique [54] in which four bits are embedded in a pixel pair. In this scheme, AC bits are made from pseudo-random numbers and the original image is divided into non-overlapped pixel blocks with the size of 4×4. Two stego images are used to embed data bits independently. the 16 AC bits and the 16 data bits used to restore the original image are embedded alternately in each 4×4 block of two stego images. Since only half of the pixel pairs of each watermarking image are used to embed the AC bits, if pixel pairs that do not contain the AC bits are hacked, then this attack cannot be detected. This is why Peng et al.'s scheme is not able to detect tampered blocks with high accuracy.

This paper has the following contributions:

+ Based on Lu et al.'s paper [50], proposed a new dual-image reversible data-hiding algorithm that can embed about $m \times n \times log_2 5 = m \times n \times 2.32$ AC bits in an $m \times n$ sized block. This embedding capacity is higher than related methods.
+ Using the above hiding algorithm to construct the RIA method in which an original image is divided into $m \times n$ non-overlapping blocks. The AC bits are generated from the $MD5$ hash with input data including pixel values and the position of each block. Both watermarking images are used to embed the AC bits.
+ The proposed method can detect forged blocks with 100% accuracy with 3×3 and 4×4 blocks in all attack types. Existing methods have only an attack detection ability for 4×4 block sizes with about 0% to 85.91 % accuracy depending on every case.

The paper consists of an Introduction and the following sections. Section 2 introduces related works and gives some remarks on their disadvantages and advantages. The proposed method is presented in Section 3. The experimental results comparing the embedding capacity, watermarking image quality, and ability to detect tampered blocks between methods are shown in Section 4. Finally, Section 5 provides some conclusions.

2 Related works

2.1 Lu et al.'s method

Lu et al. [50] used the central folding strategy to create a dual image RDH method in which embedding s bits $b_1, .., b_s$ in a pixel x of an original image I to obtain the two watermarking pixels x' and x''. In the case of $s = 2$, the embedding algorithm is performed as follows. First, convert b_1, b_2 into a decimal value d with $0 \leq d \leq 3$. Fold d in the centre by the operation $d - 2$, then embed $d - 2$ in the pixel x to get x' and x'' as follows

$$x' = x + \lfloor \frac{d-2}{2} \rfloor, \quad x'' = x - \lceil \frac{d-2}{2} \rceil. \tag{1}$$

When two stego pixels x' and x'' are known, the original pixel x is restored by the formula:

$$x = \lceil \frac{x' + x''}{2} \rceil, \tag{2}$$

and two embedded bits are extracted as follows:

$$d = (x' - x'' + 2) \rightarrow b_1 b_2. \tag{3}$$

That means converting the decimal number d into the two binary bits.

2.2 Yao et al.'s method

Yao et al.'s method [51] is an improvement to Lu et al.'s algorithm for embedding an extra bit when both first bits are equal to one. Specifically, when $d \leq 2$ Yao et al. embed two bits by formula (1). But when $d = 3$, i.e. $b_1 b_2 = 11$, Yao et al. have embedded an extra as follows:

$$x' = x, x'' = x - 1, \text{ if } b_3 = 0,$$

and

$$x' = x + 1, x'' = x - 1, \text{ if } b_3 = 1.$$

When knowing x' and x'', the original pixel x is restored by the formula (2), and embedded bits are extracted as follows. Compute $d = x' - x'' + 2$. If $d \leq 2$, two bit $b_1 b_2$ are extracted by formula (3). Otherwise, the three bits are extracted according to formulas:

$$b_1 b_2 b_3 = \begin{cases} 110, \text{ if } d = 3, \\ 111, \text{ if } d = 4. \end{cases}$$

To avoid underflow/overflow, only pixels with values between 1 and 254 are used for embedding. Pixels with a value of 0 or 255 are ignored. The embedding ratio of Lu et al.'s algorithm is 2 bits per pixel (2 bpp), while the embedding ratio of Yao et al.'s algorithm is 2.25 bpp. The maximal magnitude of the difference between each watermarking pixel and the corresponding original pixel of both methods is equal to one. That proves that both methods have high image quality.

2.3 Image authentication method of Yin et al.

Yin et al. [15] divided the original image I into 4×4 non-overlapped pixel blocks $\{B_k\}_{k=1}^{N}$. Then embed AC bits in four consecutive pixels of the Hilbert Curve. As these four pixels are in a sub-block with the size of 2×2, so the embedding technique used in Yin et al.'s method is IPVO in sub-blocks of 2×2 pixels. Yin et al. used two thresholds T_1 and T_2 to classify 4×4 blocks into three categories: flat, normal and rough. The flat and normal blocks can embed at most 8 AC bits and 2 AC bits, respectively. While the rough blocks will be omitted in the embedding procedure. When both thresholds T_1 and T_2 are equal to 255, all 4×4 blocks can embed at most 8 bits, then the method has the highest embedding capacity.

Consider a 4×4 pixel block B_k that consists of four sub-blocks B_k^t with $t = 1, ..., 4$. Suppose that B_k^t has four pixels $B_{k,i}^t$, $i = 1, ..., 4$. The embedding algorithm IPVO in sub-blocks B_k^t is carried out as follows. First sort the sequence of pixels $B_{k,i}^t$ in ascending order to get:

$$B_{k,\sigma_1}^t \leq B_{k,\sigma_2}^t \leq B_{k,\sigma_3}^t \leq B_{k,\sigma_4.}^t$$

Then compute d_{min} and d_{max} by formulas:

$$d_{k,min}^t = \begin{cases} B_{k,\sigma_1}^t - B_{k,\sigma_2}^t, \text{ if } \sigma_1 < \sigma_2, \\ B_{k,\sigma_2}^t - B_{k,\sigma_1}^t, \text{ if } \sigma_2 < \sigma_1, \end{cases}$$

$$d_{k,max}^t = \begin{cases} B_{k,\sigma_3}^t - B_{k,\sigma_4}^t, \text{ if } \sigma_3 < \sigma_4, \\ B_{k,\sigma_4}^t - B_{k,\sigma_3}^t, \text{ if } \sigma_4 < \sigma_3. \end{cases}$$

Next a bit w_1 is embedded in B^t_{k,σ_1} and a bit w_2 is embedded in B^t_{k,σ_4} (if possible) to obtain C^t_{k,σ_1} and C^t_{k,σ_4} as follows:

$$C^t_{k,\sigma_1} = \begin{cases} B^t_{k,\sigma_1} - w_1, \text{ if } d^t_{k,min} = 0 \text{ or } d^t_{k,min} = 1, \\ B^t_{k,\sigma_1} - 1, \text{ and no bit is embedded, otherwise,} \end{cases}$$

$$C^t_{k,\sigma_4} = \begin{cases} B^t_{k,\sigma_4} + w_2, \text{ if } d^t_{k,max} = 0 \text{ or } d^t_{k,max} = 1, \\ B^t_{k,\sigma_4} + 1, \text{ and no bit is embedded, otherwise.} \end{cases}$$

In Yin et al.'s method, authentication code bits (AC bits) are a sequence of N pseudo random bits generated by a secret key, where N is the number of 4 × 4 blocks obtained from the original image I. Non-embeddable blocks are ignored in the embedding procedure, and as such, they are not protected. An AC bit is embedded repeatedly in each embeddable block by mentioned above formulas. After embedding AC bits in the image I is completed, the watermarking image I' consisting of 4 × 4 blocks $C_k, k = 1, ..., N$ is obtained. During authentication, if all bits extracted from an embeddable block C_k are equal to the embedded AC bit, the k-th block is considered unchanged. On the contrary, it is considered an attacked block.

Remark of Yin et al.'s method

It is easily noted that if a 4 × 4 block C_k of the watermarking image I' is modified in the following ways, then Yin et al.'s method cannot detect.

1. All pixels of a sub-block C^t_k with $t \in \{1, ..., 4\}$ are increased or decreased by the same value. Because then the result of extracting information according to IPVO does not change.
2. The last sub-blocks of a 4 × 4 block C_k are modified such that these sub-blocks become non-embeddable. Because then C_k becomes non-embeddable or all bits extracted are equal to the embedded AC bit.
3. A non-embeddable 4 × 4 block C_k is modified so that it remains non-embeddable.
   Moreover, because the number of embedded AC bits is not large, the accuracy of the detection is not high.

2.4 Image authentication method of Hong et al.

Hong et al. [11] used the same block division and embedding technique as in [15]. So symbols of [11] are used here. The original image I is partitioned into non-overlapped 4 × 4 pixel blocks $\{B_k\}_{k=1}^N$. Each 4 × 4 block B_k is divided into 4 sub-blocks with the size of 2 × 2 B^t_k with $t = 1, ..., 4$. Embedding data bits are implemented in 2 × 2 sub-blocks by using the technique IPVO as in Section 2.3. Thus each 4 × 4 block can embed at most 8 bits. A block with the number of embedded bits greater than zero is called embeddable, otherwise is called non-embeddable.

In this method, AC bits are generated for each 4 × 4 block. Specifically, eight values B^t_{k,σ_2} and B^t_{k,σ_3} with $t = 1, ..., 4$ and the location position of the block B_k are used as the input data of the hash function MD5 to generate 128 AC bits for block B_k. To enlarge detection capacity, an AC bit is embedded into the LSB of the first pixel of a key-selected sub-block of each 4 × 4 non-embeddable block by using the LSB replacement technique. The LSB of pixel used in the LSB replacement technique and AC bits will be embedded in embeddable 4 × 4 blocks by formulas described in Section 2.3. After embedding AC bits in

4×4 blocks B_k of the original image I, the watermarking image I' consists of 4×4 blocks C_k is obtained.

Compared with Yin et al.'s method, Hong et al. introduced two critical improvements: AC bits are generated from the features of each block and an AC bit is embedded in a non-embeddable block.

Remark of Hong et al.'s method

Although the attack detection ability of Hong et al.'s method is significantly improved compared to Yin et al.'s method, it still cannot detect the change of blocks in the following cases:

1. For a non-embeddable 4×4 block C_k consisting of four sub-block C_k^t, each C_k^t has four pixels $C_{k,\sigma_1}^t \leq C_{k,\sigma_2}^t \leq C_{k,\sigma_3}^t \leq C_{k,\sigma_4}^t$ if decreasing C_{k,σ_1}^t or/and increasing C_{k,σ_4}^t by an even value for at least $t \in \{1, 2, 3, 4\}$, then C_k is still non-embeddable and the bit extracted from C_k is still equal to the AC embedded bit. Therefore, in this case, the modification of C_k cannot be detected.
2. For an embeddable 4×4 block C_k, if Pixels C_{k,σ_1}^t and C_{k,σ_4}^t with $t = s, ..., 4$ (where s is a number between 2 and 4) are modified so that sub-blocks C_k^t, $t = s, .., 4$ become non-embeddable, but C_k is still embeddable. Then the bits extracted from C_k are still equal to the AC bits. Therefore in this case the change of C_k cannot be detected.

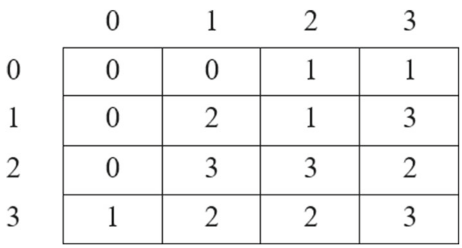
In this method, the number of embedded AC bits is not large, so the accuracy of the detection is not high. In addition, since an embeddable block contains the LSB value of a pixel in a non-embeddable block, a change in the latter block can lead to false conclusions about the change in the former block. For example, suppose C_s is embeddable that can embed two bits. In the embedding process, an AC bit and an LSB from a non-embeddable C_r are embedded in C_s. Now if C_r is attacked so that it becomes embeddable, then two bits embedded in C_s are considered as two AC bits. Then the bits extracted from C_s can be different from the AC bits. This leads to the false conclusion that C_s was attacked.

2.5 Image authentication method of Peng et al.

In Peng et al.'s scheme [53], authentication information is the pseudo random bits which are generated by keys. The original image I with pixels x_i is copied into two images denoted as I_1 with pixels p_i and I_2 with pixels q_i, that means $x_i = p_i = q_i$ with $i = 1, ..., H \times W$, where $H \times W$ is the size of the image. Peng et al. improved the irreversible data hiding algorithm SOS [54] of Yin et al. to get a reversible data hiding method in two images and then use this method to create a reversible image authentication scheme.

First, four AC bits b_1, b_2, b_3, b_4 are embedded in the pair of two pixels p_1, p_2 of I_1 based on matrix MB (Fig. 1) as follows. Convert b_1, b_2 and b_3, b_4 into two decimal numbers d_1

Fig. 1 Matrix MB of size 4 × 4



	0	1	2	3
0	0	0	1	1
1	0	2	1	3
2	0	3	3	2
3	1	2	2	3

and d_2 with values between 0 and 3. Define two stego pixels p'_1, p'_2 from p_1, p_2 and d_1, d_2 by formulas

$$\begin{aligned}
&MB(p'_1\%4, p'_2\%4) = d_1,\\
&MB(p'_1\%4, (p'_2 + 1)\%4) = d_2,\\
&p_1 - 1 \le p'_1 \le p_1 + 2,\\
&p_2 - 1 \le p'_2 \le p_2 + 2.
\end{aligned}$$

To be able to recover p_1, p_2 from p'_1, p'_2 can calculate values:

$$e_1 = 2 + (p_1 - p'_1),\ e_2 = 2 + (p_2 - p'_2),$$

where $0 \le e_1, e_2 \le 3$. Embed e_1, e_2 in the pair of two pixels q_1, q_2 of the image I_2 according to the same formulas above. That means

$$\begin{aligned}
&MB(q'_1\%4, q'_2\%4) = e_1,\\
&MB(q'_1\%4, (q'_2 + 1)\%4) = e_2,\\
&q_1 - 1 \le q'_1 \le q_1 + 2,\\
&q_2 - 1 \le q'_2 \le q_2 + 2.
\end{aligned}$$

To ensure that both images $I_1^{'}$ and $I_2^{'}$ contain the same number of authentication bits, the embedding of the authentication bits and the recovery information are performed alternately on the I_1 and I_2 images. That is, use the pair of pixels q_3, q_4 of I_2 to embed the 4 AC bits and the pair of pixels p_3, p_4 of I_1 to embed recovery information, and so on.
It is easy to see that only the pairs (p_i, p_{i+1}) as well as (q_i, q_{i+1}) satisfy the condition

$$1 \le p_i, p_{i+1}, q_i, q_{i+1} \le 253,$$

can be used to embed AC bits without causing underflow/overflow.
In Peng et al.'s scheme, the original is divided into non-overlapped pixel blocks with the size of 4 × 4, so each block of I'_1 and I'_2 contains at most 16 AC bits.
For $i = 1, 5, 9, ...,$, extracting the AC bits and restoring the original pixels (x_i, x_{i+1}) from (p'_i, p'_{i+1}) and (q'_i, q'_{i+1}) are carried out as follows:

$$\begin{aligned}
e_i &= MB(q'_i\%4, q'_{i+1})\%4,\\
e_{i+1} &= MB(q'_i)\%4, q'_{i+1} + 1)\%4,\\
x_i &= p'_i + e_i - 2,\\
x_{i+1} &= p'_{i+1} + e_{i+1} - 2,
\end{aligned}$$

and

$$\begin{aligned}
d_i &= MB(p'_i\%4, p'_i\%4),\\
d_{i+1} &= MB(p'_i\%4, (p'_{i+1} + 1)\%4).
\end{aligned} \tag{4}$$

Convert d_i and d_{i+1} into 4 AC bits:

$$d_i = (b_1, b_2)_2,\ d_{i+1} = (b_3, b_4)_2 \tag{5}$$

For $i = 3, 7, 11, ...$, extracting the AC bits and restoring the original pixels is done similarly but with a change of roles between p'_i and q'_i.

Remark of Peng et al.'s method
Although the number of embedded AC bits is large, this method has some limitations in its ability to detect attacks as follows.

1. From (4) and (5) , it follows that if p'_i or/and p'_{i+1} (with i=1,5,9,...) are incremented or decremented by a positive integer that is a multiple of four, then the same AC bits are extracted, so the change of I'_1 is undetectable.
2. Since with $i = 3, 7, 11, .., p'_i$ và p'_{i+1}do not contain AC bits, so if they are modified in any way, this change will not be detected.
3. From (a) and (b), deduce that if each pixel of $I_1^{'}$ is increased or decreased by a positive integer equal to a multiple of 4, such modification is also undetected.

The limitations in attack detection ability for the watermarking image $I_2^{'}$ are similar to the image $I_1^{'}$.

3 Proposed reversible image authentication scheme

3.1 Improvements of Lu et al.'s and Yao et al.'s algorithms

The algorithm used in the proposed scheme is an improvement of Lu et al.'s [50] and Yao et al.'s [51] algorithms by embedding a 5-nary digit d in a pixel x according to the following formula:

$$x' = x + \left\lfloor \frac{(d-2)}{2} \right\rfloor, \; x'' = x - \left\lceil \frac{(d-2)}{2} \right\rceil. \tag{6}$$

Consider a block X of size $m \times n$ consisting of $k = m \times n$ pixels with a value between 1 and 254. Then can embed k 5-nary digits in the block X. Now we will evaluate the embedding capacity of this technique. Set:

$$\begin{aligned} &k5Max = 5^k - 1,\\ &bk5Max: \text{ binary representation of } k5Max,\\ &l: \text{length of } bk5Max, \end{aligned}$$

where $k5Max$ is the maximum 5-nary value of k digits and is the maximum value that can be embedded in the block X. Denote

$$l2Min = 2^{l-1}, \; l2Max = 2^l - 1,$$

are the minimum $l-bit$ binary value and the maximum $l-bit$ binary value, respectively. Let $emValue$ be a embedded binary number of l bits, then it can be divided into the following cases:

1. $emValue$ is between $l2Min$ and $k5Max$, $emValue$ can be embedded in block X. In this case, l bits are embedded.
2. $emValue$ is greater than $k5Max$, then can not embed $emValue$ in X. In this case, can only embed the first $(l-1)$ bits of $emValue$.
3. $emValue$ is smaller than $l2Min$, then the first bit of $emValue$ is equal to zero. Although we can embed all l bits of $emValue$, but to avoid ambiguity with case 2, only the first $(l-1)$ bits of $emValue$ are embedded.

Since can consider that values of $emValue$ are uniformly distributed over the domain from zero to $l2max$, so it is deduced that the average number of embedded bits in X is:

$$Cap(X) = [(k5Max - l2Min + 1) \times l + (l2Max + l2Min - k5Max) \\ \times (l-1)]/(l2Max + 1).$$

For block X with sizes 2×2, 3×3 and 4×4 corresponding to 4, 9 and 16 pixels, the value of $Cap(X)$ is equal to 9.1104, 20.4313 and 37.0551, respectively. Finally, the embedding ratio of the proposed algorithm is 2.2776, 2.2701 and 2.3159 corresponds to block sizes 2×2, 3×3 and 4×4. All these embedding ratios are higher than Lu et al.'s embedding ratio of 2 bpp and Yao et al.'s embedding ratio of 2.25 bpp.

Our other improvement is using the pixels with the value of 0 or 255 to embed, while these pixels are ignored in the methods of Lu et al. and Yao et al. Namely, a bit b is embedded in the pixel x with the value of 255 as follows:

$$\begin{aligned} x' = 255, x'' = 255 \text{ if } b = 0, \\ x' = 254, x'' = 255 \text{ if } b = 1. \end{aligned} \tag{7}$$

For a block X consisting of all pixels with the value of 0 ($x(i) = 0$, $i = 1, ..., k$), one-bit b is embedded in the first pixel of the block by formulas:

$$\begin{aligned} x'(1) = 4, x''(1) = 0, \text{ if } b = 0, \\ x'(1) = 3, x''(1) = 0, \text{ if } b = 1, \\ x'(i) = x''(i) = 0, i = 2, ..., k. \end{aligned} \tag{8}$$

From formulas (6)-(8) it is easy to see that the embedding algorithms in cases: $0 < x < 255$, $x = 255$, and $x = 0$ will not give ambiguous results.

3.2 Embedding data bits in a pixel block

Let X be a block of size $m \times n$ consisting of k (with $k = m \times n$) pixels $x_i = 1, ..., k$, and $B = b_1, b_2, ...$ be a given bit sequence. For embedding B in X, consider three cases:

The first case: All pixels $x(i)$ of X with a value of 0. In this case, one bit of B is embedded in $x(1)$ by formula (8) to get $x'(i)$ and $x''(i)$, $i = 1, ..., k$.

The second case: There is at least a pixel of X with a value of 255, the remaining pixels with a value of 0. In this case, a bit of B is embedded in each pixel of X with a value of 255 by formula (7). For $x(i) = 0$, no bit is embedded, and $x'(i) = x''(i) = 0$.

The third case: X has at least a pixel with a value between 1 and 254. Suppose there are s such pixels.

First, scan by rows pixels of X. Each pixel with a value of 255 (if any) is used to embed one bit according to formula (7). For a pixel $x(i) = 0$, no bit is embedded, and $x'(i) = x''(i) = 0$. Then scan again X to find s pixels with a value between 1 and 254. The subsequent bits will be embedded in these s pixels according to Section 3.1. Namely, calculate the $s5Max = 5^s - 1$, the length l of the binary representation of $s5Max$. Then compute $l2Min = 2^{l-1}$. Select l bits from B to generate a binary number called $emValue$. Compare $emValue$ with $s5Max$ and $l2Min$. If $emValue$ is in the domain between $l2Min$ and $s5Max$ then $emValue$ is preserved, otherwise, $emValue$ is adjusted by obtaining only the first $(l - 1)$ bits. Convert $emValue$ to a 5-nary number consisting of s digits: $D = (d1, d2, .., ds)$ with $0 \leq dj \leq 4$. Embed each digit of D in a pixel of X with a value from 1 to 254 according to the formula (6).

The illustrating examples:

1. Consider the block:

$$X = \begin{bmatrix} 255 & 255 \\ 255 & 0 \end{bmatrix},$$

and : B = (1,0,1). Scan pixels by rows, as far as the second case goes, there are:

$$X' = \begin{bmatrix} 254 & 255 \\ 254 & 0 \end{bmatrix},\ X'' = \begin{bmatrix} 255 & 255 \\ 255 & 0 \end{bmatrix}.$$

2. Consider the block:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

and : B = (0). By the formulas in the first case we get:

$$X' = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix},\ X'' = \begin{bmatrix} 0 & 0 \\ 0 & 0. \end{bmatrix}.$$

3. Consider the block:

$$X = \begin{bmatrix} 0 & 6 \\ 255 & 4 \end{bmatrix},\ \text{and} : B = (1, 1, 0, 1, 1, 1).$$

First, embed one bit in a pixel x_3 with a value of 255. Since $b_1 = 1$, so $x_3^{'} = 254$, and $x_3^{''} = 255$. Two pixels $x_2 = 6$ and $x_4 = 4$ will be used to embed next bits. In this case, $s = 2$, $s5Max = 5^2 - 1 = 24$, $bs5Max = 11000$, $l = 5$, and $l2Min = 10000$. Select the next 5 bits: 10111, it is between $l2Min$ and $s5Max$, so five of these bits can be embedded in pixels x_2 and x_4. Convert this binary value to a 5-nary number to get $emValue = (43)_5$. Embed numbers 4 and 3 in the pixels x_2 and x_4 according to formula (6), we obtain:

$$x_2' = 7,\ x''_2 = 5$$
$$x_4' = 4,\ x''_4 = 3$$

Thus, the obtained watermarking blocks are:

$$X' = \begin{bmatrix} 0 & 7 \\ 254 & 4 \end{bmatrix},\ X'' = \begin{bmatrix} 0 & 5 \\ 255 & 3 \end{bmatrix}.$$

4. Consider the block:

$$X = \begin{bmatrix} 3 & 5 \\ 6 & 5 \end{bmatrix},\ \text{and } B = (1, 0, 0, 0, 1, 0, 0, 0, 0, 1).$$

This block belongs to the third case with $s = 4$. In this case, no pixel has a value of 255, so all bits will be embedded in pixels with values 1 to 254. We have $s5Max = 5^4 - 1 = 624$, $bs5Max = 1001110000$, $l = 10$, and $l2Min = 2^{l-1} = 2^9$. Select 10 bits of $B : emValue = 1000100001$. It is clear that $emValue$ is smaller than $s5Max$ and greater than $l2Min$, so convert this binary value to a 5-nary number to get $(4140)_5$. Embed the 5-nary digits 4, 1, 4 and 0 in pixels with the values 3, 5, 6 and 5 of X according to formula (6), respectively, we get:

$$X' = \begin{bmatrix} 4 & 4 \\ 7 & 4 \end{bmatrix},\ X'' = \begin{bmatrix} 2 & 5 \\ 5 & 6 \end{bmatrix}.$$

3.3 Extracting and restoring from a pair of two watermarking blocks

Suppose $X^{'}$ and $X^{''}$ are a pair of known watermarking blocks of size $m \times n$. The process of extracting embedded bits and restoring the original block is performed by the following steps:
First restoring the original block $X = (x_1, ..., x_k)$, with $k = m \times n$ by formulas:

$$x_i = \begin{cases} 0, \text{ if } i = 1\ , x_1^{'} \in \{3, 4\} \text{ and } x_1^{''} = 0, \\ \lceil \frac{x_i^{\prime}+x_i{}^{\prime\prime}}{2} \rceil, \text{ otherwise.} \end{cases} \tag{9}$$

where $i = 1, ..., k$. Then we divide X into three cases as in Section 3.2. *In the first case*, one bit is extracted by the formula:

$$b = \begin{cases} 0, \text{ if } x_1^{'} = 4, \\ 1, \text{ if } x_1^{'} = 3. \end{cases} \tag{10}$$

In the second case, one bit is extracted from each $x' \in \{254, 255\}$ by the formula:

$$b = \begin{cases} 0, \text{ if } x' = 255, \\ 1, \text{ if } x' = 254. \end{cases} \tag{11}$$

In the third case, first scan X' and X'' by rows to get pairs of $x^{'}$ and $x^{''}$. If $x^{'}$ and $x^{''}$ correspond to a pixel x with a value of 255, extract a bit by formula (11). Then scan again $X^{'}$ and $X^{''}$, for each pair of pixels x' and x'' corresponds with an original pixel x with a value between 1 and 254, extract a 5-nary digit by the formula:

$$d = x^{'} - x^{''} + 2. \tag{12}$$

Suppose have s such original pixels. Compute values $s5Max$, l, and $l2Min$ as in Section 3.2. Convert s of obtained 5-nary digits into a decimal number denoted $emValue$. If $emValue$ is between $l2Min$ and $s5Max$, convert $emValue$ in to l binary bits. Otherwise, convert $emValue$s into $l - 1$ binary bits. The sequence of embedded bits is obtained by collecting bits extracted. The illustrating example:

1. Consider a pair of two watermarking blocks (example 4 of 3.2)

   $$X^{'} = \begin{bmatrix} 4 & 4 \\ 7 & 4 \end{bmatrix},\ X^{''} = \begin{bmatrix} 2 & 5 \\ 5 & 6 \end{bmatrix}.$$

   First, restore the original block X by (9), obtain:

   $$X = \begin{bmatrix} 3 & 5 \\ 6 & 5 \end{bmatrix}$$

   We see that the original pixel block belongs to case 3. In this case, have $s = 4$, $s5Max = 624$, $bs5Max = 1001110000$, $l = 10$, $l2Mins = 512$. Extract four 5-nary digits by formula (12), we obtain a 5-nary number: $(4, 1, 4, 0)_5$. Convert it into a decimal number denoted $emValue$, we have $emValue = 545$. Since $emValue$ is between $l2Min$ and $s5Max$, so convert $emValue$ to a sequence of l bits.We obtain 1000100001, that are the bits to be extracted.

2. Consider two watermarking blocks (example 2 of 3.2)

$$X^{'} = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}, \; X^{''} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

From formulas (9) and (10) it follows

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

and an embedded bit is 0.

3.4 Embedding AC bits in an original image

Let I be an original image of size $H \times W$. Divide I into non-overlapped blocks $I_{r,c}$ with the size of $m \times n$ where $r = 1, ..., \lfloor H/m \rfloor, c = 1, ..., \lfloor W/n \rfloor$. The process of embedding authentication information on image I is carried out according to the following steps:

Step 1: For each block $I_{r,c}$ with pixels $I_{r,c}(i), i = 1, ..., k$ (with $k = m \times n$), first generate 128 AC bits by applying hash function MD5 as follows

$$\begin{aligned} I_{r,c}AC &= hashMD5(r, c, I_{r,c}(1), ..., I_{r,c}(k)) \\ &= (ac(1), ..., ac(128)). \end{aligned} \tag{13}$$

Step 2: Choose AC bits defined in the previous step as the embedded bits:

$$(b(1), ..., b(128)) = (ac(1), ..., ac(128)). \tag{14}$$

Embed bits $b(1), b(2), ..., b(\alpha)$ in the block $I_{r,c}$ by formulas in Section 3.2 to get two watermarking blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$, where α is a maximum number of bits that can be embedded.

Step 3: Repeat steps 1 and 2 for all blocks $I_{r,c}$ with $r = 1, ..., \lfloor H/m \rfloor$ and $c = 1, ..., \lfloor W/n \rfloor$ to get all watermarking blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$.

Step 4: Combine blocks $I^{'}_{r,c}$ to get the watermarking image $I^{'}$ and blocks $I^{''}_{r,c}$ to get the watermarking image $I^{''}$. The watermarking images $I^{'}$ and $I^{''}$ are sent to a certain recipient. This person needs to check if each block of the watermarking images is hacked or not. If a pair of two watermarking blocks is not attacked, then from those two blocks we can restore the corresponding original block correctly. Therefore, if both watermarking images are not attacked or the number of tampered block pairs is small, the recovered image is still available.

3.5 Authenticating the watermarking images

After embedding AC bits in the original image I of size $H \times W$ as in Section 3.4 we get two watermarking images $I^{'}$ and $I^{''}$ of the same size as I. These images are sent, but the recipient receives images $\widetilde{I^{'}}$ and $\widetilde{I^{''}}$ that can be equal to or different from $I^{'}$ and $I^{''}$. If images $I^{'}$ and $I^{''}$ are attacked, then $\widetilde{I^{'}}$ and $\widetilde{I^{''}}$ are different from $I^{'}$ and $I^{''}$, otherwise, they are equal. To detect and locate attacked regions, first divide images $\widetilde{I^{'}}$ and $\widetilde{I^{''}}$ into non-overlapped blocks $\widetilde{I^{'}_{r,c}}$ and $\widetilde{I^{''}_{r,c}}$ with the size of $m \times n$ as in Section 3.4. Then the process of authenticating each pair of blocks $\widetilde{I^{'}_{r,c}}$ and $\widetilde{I^{''}_{r,c}}$ is carried out according to the following steps:

Step 1: Preliminary Check: First from Section 3.4, it is easy to see that the pair of two blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$ has the following two properties:

1. All pixels of $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are between 0 and 255.
2. Except for the case $I^{'}_{r,c}(1) \in \{3, 4\}$, $I^{''}_{r,c}(1) = 0$, and $I^{'}_{r,c}(i) = I^{''}_{r,c}(i) = 0, i = 2, ..., k$. In other cases, the values of pixels $I^{'}_{r,c}(i)$ and $I^{''}_{r,c}(i)$ differ by no more than 2, that is $abs(I^{'}_{r,c}(i) - I^{''}_{r,c}(i)) \leq 2$, with $i = 1, ..., k$.

Therefore, we check if $\widetilde{I^{'}_{r,c}}$ and $\widetilde{I^{''}_{r,c}}$ satisfy these two properties. If the condition is not satisfied, then it can be confirmed that $I^{'}_{r,c}$ and $I^{''}_{r,c}$ have been attacked, and skipped steps 2-5. Otherwise, go to step 2.

Step 2: From the pair of two blocks $\widetilde{I^{'}_{r,c}}$ and $\widetilde{I^{''}_{r,c}}$, restore the block $\widetilde{I_{r,c}}$ by formula (9). Note that if blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are not attacked, then $\widetilde{I_{r,c}}$ is the original block, that is: $\widetilde{I_{r,c}} = I_{r,c}$.

Step 3: Generate AC bits for $\widetilde{I_{r,c}}$ by formula (13):

$$\begin{aligned}\widetilde{I_{r,c}}AC &= hashMD5(r, c, \widetilde{I_{r,c}}(1), ..., \widetilde{I_{r,c}}(k)) \\ &= (\widetilde{ac}(1), ..., \widetilde{ac}(128))\end{aligned}$$

Step 4: From blocks $\widetilde{I^{'}_{r,c}}$ and $\widetilde{I^{''}_{r,c}}$ extract bits by formulas in Section 3.3. Denote extracted bits as $\tilde{B} = (\tilde{b}(1), ..., \tilde{b}(\alpha))$.

Step 5: Compare $\tilde{b}(i)$ with $\widetilde{ac}(i))$, $i = 1, ..., \alpha$. If all are equal, we conclude that $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are not attacked, that is $\widetilde{I^{'}_{r,c}} = I^{'}_{r,c}$ and $\widetilde{I^{''}_{r,c}} = I^{''}_{r,c}$. Otherwise consider as $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are attacked.

Step 6: Repeat steps 1-5 to authenticate and locate tampered pairs of two watermarking blocks. In the case $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are not attacked then block $\widetilde{I_{r,c}}$ defined in step 2 is the original block $I_{r,c}$. Thus, we can locate the attacked blocks and recover the original blocks correctly in case of no attack.

3.6 Accuracy of the authentication algorithm

First, it is noted that almost original pixel blocks $I_{r,c}$ satisfy the condition:

$$I_{r,c}(i) \in \{1, ..., 254\} \tag{15}$$

with $i \in \{1, ..., k\}$. For such pixel blocks, restoring original pixels and extracting embedded bits are performed according to formulas (9) and (12), respectively.

For simplicity, in this section only blocks satisfying condition (15) are considered. The authentication accuracy of the proposed method in the below three attack types of a block will be discussed. .

The first type of attack: Together increase or decrease $I^{'}_{r,c}(i)$ and $I^{''}_{r,c}(i)$ by the same value, that is,

$$\widetilde{I^{'}_{r,c}}(i) = I^{'}_{r,c}(i) + \Delta(i),\ \widetilde{I^{''}_{r,c}}(i) = I^{''}_{r,c}(i) + \Delta(i),\ i = 1, ..., k$$

with at least one $\Delta(i) \neq 0$. Then by formula $(9)_b$, pixels $\widetilde{I_{r,c}}(i)$ restored by step 2 of Section 3.5 differ from the original pixels. That means $(\widetilde{I_{r,c}}(1), ..., \widetilde{I_{r,c}}(k)) \neq (I_{r,c}(1), ..., I_{r,c}(k))$. So, AC bits obtained in step 3 of the authentication procedure

(Section 3.5) are not equal to AC bits created in the process of embedding AC bits (Section 3.4), which means

$$(\widetilde{ac}(1), ..., \widetilde{ac}(128)) \neq (ac(1), ..., ac(128)). \tag{16}$$

While from (12), the bits extracted in step 4 of Section 3.5 are the same as the embedded bits, which means

$$\widetilde{b}(i) = b(i), \ i = 1, ..., \alpha.$$

From this and (14) it follows that

$$\widetilde{b}(i) = ac(i), \ i = 1, ..., \alpha. \tag{17}$$

By formulas (16)-(17), deduce that

$$(\widetilde{b}(1), ..., \widetilde{b}(\alpha)) \neq (\widetilde{ac}(1), ..., \widetilde{ac}(\alpha))$$

So, by step 5 of Section 3.5, we conclude that blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are attacked.
However, when the value of α is small, from (16) it can still deduce that

$$(\widetilde{ac}(1), ..., \widetilde{ac}(\alpha)) = (ac(1), ..., ac(\alpha)) \tag{18}$$

From this and (17) we have $(\widetilde{b}(1), ..., \widetilde{b}(\alpha)) = (\widetilde{ac}(1), ..., \widetilde{ac}(\alpha))$. So, by step 5 of Section 3.5, we conclude that blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are not attacked. Thus, we get the wrong conclusion. However, from (16) it follows that the probability of occurrence of the condition (18) is $\frac{1}{2^{\alpha}}$. When α is greater than 10, this probability is very low.

The second type of attack: Increase $I^{'}_{r,c}(i)$ by one value and decrease $I^{''}_{r,c}(i)$ by the same value or vice versa, that is

$$\widetilde{I^{'}_{r,c}}(i) = I^{'}_{r,c}(i) + \Delta(i), \ \widetilde{I^{''}_{r,c}}(i) = I^{''}_{r,c}(i) - \Delta(i), \ i = 1, .., k,$$

with at least one $\Delta(i) \neq 0$. Similar to the first case, by formula (9), have

$$\widetilde{I_{r,c}}(i) = I_{r,c}(i), \ i = 1, .., k.$$

It follows that

$$\widetilde{ac}(i) = ac(i), \ i = 1, .., 128.$$

On the other hand, from (12), deduce that

$$\widetilde{b}(i) \neq b(i),$$

with at least one i. From this and (14), deduce $(\widetilde{b}(1), ..., \widetilde{b}(\alpha)) \neq (\widetilde{ac}(1), ..., \widetilde{ac}(\alpha))$. So, by step 5 of Section 3.5, we conclude that blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are attacked. Similar to the first case, the probability of a false conclusion in this case is $\frac{1}{2^{\alpha}}$.

The third type of attack: This attack type differs from both the first type and the second type. Then we have

$$(\widetilde{ac}(1), ..., \widetilde{ac}(128)) \neq (ac(1), ..., ac(128),$$
$$(\tilde{b}(1), ..., \tilde{b}(\alpha)) \neq (b(1), ..., b(\alpha)).$$

If

$$(\widetilde{ac}(1), ..., \widetilde{ac}(\alpha)) = (\tilde{b}(1), ..., \tilde{b}(\alpha)) \tag{19}$$

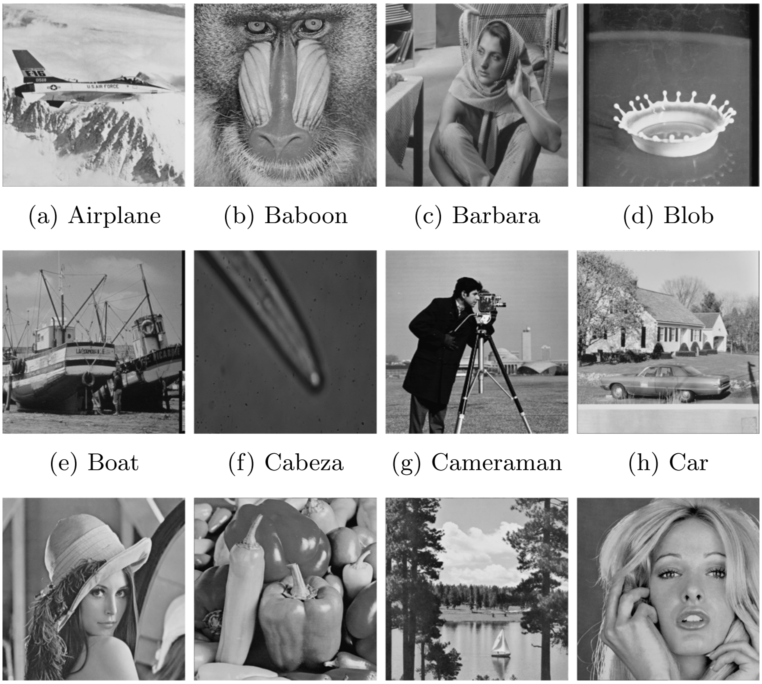
According to step 5 of Section 3.5, we conclude that two blocks $I^{'}_{r,c}$ and $I^{''}_{r,c}$ are not attacked. In this case, bits $\widetilde{ac}(i)$ and $\tilde{b}(i)$ do not have any relationship, so the probability of occurrence of the condition (19) is equal to is $\frac{1}{2^{\alpha}}$. In summary, for the proposed method, the true attack detection rate is $(1 - \frac{1}{2^{\alpha}})$, with α is the number of AC bits embedded in a pixel block. This rate depends on the size of the block $I_{r,c}$. For the block sizes of 4×4, 3×3 and 2×2, the embedding capacities α are 37, 20 and 9, respectively. In all of these cases, the correct attack detection rate is close to 100%.

4 Experimental results

This section presents experiment results to compare the detection effectiveness of the proposed method with the methods of Yin et al. [15], Hong et al. [11], and Peng et al. [53]. The computations of methods have been performed using MATLAB running on Lenovo computer IdeaPad Slim 5 15 ITL 05 with Intel Core i5, 8-GB Ram, 512-GB SSD. The experiments are performed on the 12 standard grayscale images of size 512 × 512 as shown in Fig. 2.

4.1 Embedding capacity and image quality comparisons between methods

The embedding capacity of each method is measured by the average number of AC bits embedded in 12 standard grayscale images. Similarly, stego-image quality is calculated by the average PSNR value obtained in these 12 test images. The comparison results are presented in Table 1. The information in the second column is the number of non-overlapping blocks


(a) Airplane (b) Baboon (c) Barbara (d) Blob

(e) Boat (f) Cabeza (g) Cameraman (h) Car

(i) Lena (j) Pepper (k) Sailboat (l) Tiffany

Fig. 2 Experimental images

Table 1 Embedding capacity and image quality comparisons

Methods	Number of blocks	Number of Embedable blocks	Number of AC bits Embedded in a block	PSNR
Yin	16384	13418	3.3060	51.9780
Hong	16384	13418	3.3060	51.1981
Peng	16384	16384	15.9851	46.3635
Proposed 4x4	16384	16384	37.0350	50.3362
Proposed 3x3	28900	28900	20.4219	50.2876
Proposed 2x2	65536	65536	9.1045	50.2889

partitioned from each 512 × 512 test image. For methods using 4 × 4 blocks, such as Yin et al., Hong et al., Peng et al., and the proposed method with 4 × 4 blocks, the number of blocks in each test image is 16384. With the proposed method using 3 × 3 and 2 × 2 blocks, the numbers of blocks are 28900 and 65536, respectively. The third column presents the average number of embeddable blocks in 12 test images. This number for methods of Yin et al. and Hong et al. is 13418 over 16384 blocks. For the remaining methods, all blocks are embeddable.

The fourth column introduces the embedding capacity of methods. In Sections 2.3 and 2.4, it is noted that the methods of Yin et al. and Hong et al. used the IPVO technique in 2 × 2 sub-blocks. So, in each block of 4 × 4 pixels at most 8 AC bits are embedded and at least no pixel is embedded, the average number is 4 bits. The average number of embedded bits in each 4 × 4 block of these two methods is 3,3060 as shown in Table 1 which is in agreement with the theoretical analysis. Section 2.5 showed that in each 4 × 4 block of 16 pixels with values from 1 to 253, the method of Peng et al. can embed 32 bits, but only half of the bits are AC bits, the remaining bits are used to restore the original image. The mean number of embedded AC bits of Peng et al's method in a block is 15.9851. It is shown in Section 3.1 that for a pixel block X of sizes 2 × 2, 3 × 3, and 4 × 4, the proposed method can embed a number of 9.1104, 20.4313 and 37.0551 bits, respectively. Thus, the embedding capacity of the proposed method presented in Table 1 is consistent with the theoretical analysis.

The 5th column presents the average PSNR values of the methods. In Peng et al.'s method and the proposed method, PSNR is the mean of PSNR1 and PSNR2. For Yin et al.'s method, only 50% number of pixels in an image is increased or decreased by one, so its average PSNR is the largest. Compared with Yin et al.'s method, Hong et al.'s method must perform embedding in non-embeddable blocks. Therefore, the PSNR of Hong et al.'s method is smaller than that of Yin et al.'s method. In the proposed method, each pixel is also increased or decreased by one, but the number of modified pixels is larger than that of Yin et al.'s method. Therefore, the PSNR of the proposed method is slightly smaller than the PSNR of Yin et al.'s method. The pixels in Peng et al.'s method are modified at most two units, so the PSRN coefficient of this method is the smallest.

4.2 Detection capacity comparison between methods

4.2.1 Small and uniform distribution attack

In this attack manner, first, a noise matrix of size 512 × 512, denoted by NM, is created. This matrix consists of non-overlapped 4 × 4 blocks, denoted by $NM_{r,c}$ with $1 \leq r, c \leq 128$. All

$$\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

Fig. 3 The first part of noise matrix NM

blocks are equal to zero except blocks $NM_{r,c}$ with r, c satisfy the condition:

$$(r-1)\ mod\ 4 = 0 \text{ and } (c-1)\ mod\ 2 = 0 \tag{20}$$

Each of these blocks has two elements with values belonging to the set $\{-2, -1, 1, 2\}$, other elements are equal to zero. The positions and values of non-zero elements are determined randomly. For example, a sub-matrix consisting of the first 20 rows and 20 columns of NM is shown in Fig. 3. The matrix NM used for attacking a watermarking image $I^{'}$ of size 512×512 to get an attacked image $\widetilde{I}^{'}$ as follows:

$$\widetilde{I}^{'}(i,j) = \begin{cases} I^{'}(i,j) + NM(i,j), & \text{if } 0 \leq I^{'}(i,j) + NM(i,j) \leq 255 \\ I^{'}(i,j) - NM(i,j), & \text{otherwise.} \end{cases} \tag{21}$$

For Peng et al.'s method and the proposed method, the second watermarking image $I^{''}$ will be modified according to formula (21) to get an attacked image $\widetilde{I}^{''}$ but with a noise matrix $NM2$ that only differs NM in positions of non-zero elements in each block $NM_{r,c}$. These attacks make a small modification to the watermarking images. Even this change is difficult to perceive with the naked eye. Figure 4 below illustrates a watermarking image $I^{'}$ and an attacked image $\widetilde{I}^{'}$ obtained by attacking it in the above manner.

The average detection ability of attacked blocks on 12 test images of the methods is presented in Table 2.

We now explain some notations in Table 2. The TP in column 2 is the exact number of blocks detected as fake. In other words, TP is the number of blocks that are confirmed to have been hacked and that has in fact been modified. FN is the number of blocks that are confirmed to be unchanged but in fact, they have been tampered with. TN is the exact number of blocks detected as not being hacked. FP is the number of blocks detected as hacked but in reality, they have not changed at all. TPR is the ratio of the exact number of blocks detected as hacked to the total number of hacked blocks. This coefficient evaluates the ability to detect hacked blocks. If $TPR = 1$, it means that 100% of hacked blocks are detected. The FPR is the ratio of the number of blocks that are falsely detected as hacked to the total number

(a) Marked (b) Attacked

Fig. 4 Watermarking image and attacked image by the small and uniform distribution attack

of blocks that are not attacked. If $FPR = 0$, it means that 100% of not-hacked blocks are correctly detected. According to Table 2, Yin et al.'s method detects 42.30% and Hong et al.'s method detects 64.59% of the number of tampered blocks. Peng et al.'s method can detect 85.91 % of tampered blocks. The attack detection ratio of the proposed method is 99.91% if 2 × 2 block sizes are used and is always 100% if 3 × 3 and 4 × 4 block sizes are used. It should also be mentioned that for all methods except Hong's method, every block that is not attacked is detected as unmodified. Particularly for Hong's method, $FPR = 0.05\%$. This means there are some blocks that do not change, but the method of Hong et al. concluded that they have been attacked. This is explained in Section 2.4.

4.2.2 A special attack on the watermarking images of Sailboat image

This section presents a special attack that is performed in each 4 × 4 block $bI^{'}$ of the watermarking image $I^{'}$ independently. Denote $mi^{'}$ ($ma^{'}$) as the smallest (largest) value of pixels in the $bI^{'}$. Now we want to define a quantity, denoted $deV^{'}$, that is a multiple of 4 with

Table 2 Attack detection ability comparison between methods

Methods	Correct attack detected block number (TP)	False not-attack detected block number (FN)	Correct not-attack detected block number (TN)	False attack detected block number (FP)	TPR= TP/(TP+ FN) (%)	FPR= FP/(FP+ TN) (%)
Yin	866.25	1181.75	14336.00	0	42.30	0
Hong	1322.75	725.25	14328.75	7.25	64.59	0.05
Peng 2	1759.50	288.50	14336.00	0	85.91	0
Proposed 4x4	2048.00	0	14336.00	0	100	0
Proposed 3x3	5191.00	0	23709.00	0	100	0
Proposed 2x2	6655.00	5.00	58876.00	0	99.91	0

as large a value as possible, such that after decreasing all pixels of $bI^{'}$ by $deV^{'}$, the decreased pixels are retained in segment [0, 255]. Obviously, $deV^{'}$ is determined by the formula:

$deV^{'} = 4 \times \lfloor (mi^{'}/4) \rfloor$.

Similarly, if define

$inV^{'} = 4 \times \lfloor (255 - ma^{'})/4) \rfloor$,

then $inV^{'}$ is the largest possible multiple of 4 such that after increasing all the pixels of $bI^{'}$ by $inV^{'}$, the increased pixels will be retained in the [0,255] segment.

To make $bI^{'}$ as variable as possible, $deV^{'}$ is chosen If $deV^{'} \geq inV^{'}$, and $inV^{'}$ is chosen, if otherwise. The following are some illustrative examples. Consider a block:

$$bI^{'} = \begin{bmatrix} 103 & 150 & 152 & 154 \\ 208 & 156 & 160 & 134 \\ 200 & 150 & 170 & 144 \\ 228 & 158 & 180 & 124 \end{bmatrix},$$

In this case, $mi^{'} = 103$, $ma^{'} = 228$, $deV^{'} = 100$, $inV^{'} = 24$. Choose $deV^{'} = 100$ to attack $bI^{'}$. In other words, all pixels of $bI^{'}$ are decreased by $deV^{'}$. The attacked block is equal to:

$$\widetilde{bI}^{'} = \begin{bmatrix} 3 & 50 & 52 & 54 \\ 108 & 56 & 60 & 34 \\ 100 & 50 & 70 & 44 \\ 128 & 58 & 80 & 24 \end{bmatrix},$$

Consider a block:

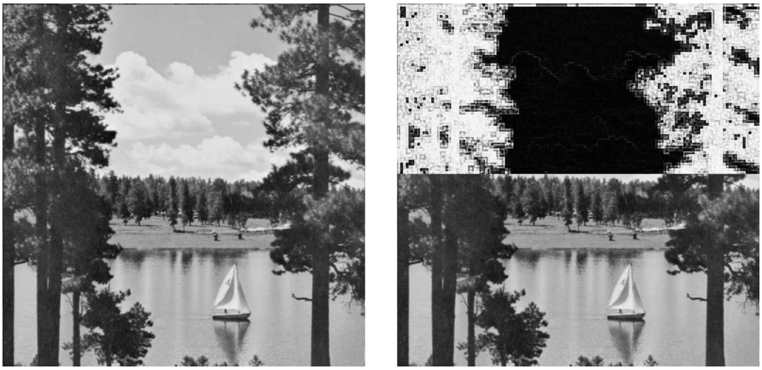
$$bI^{'} = \begin{bmatrix} 23 & 150 & 152 & 150 \\ 108 & 153 & 140 & 134 \\ 100 & 150 & 130 & 144 \\ 128 & 148 & 120 & 124 \end{bmatrix},$$

In this case, $mi^{'} = 23$, $ma^{'} = 153$, $deV^{'} = 20$, $inV^{'} = 100$. Choose $inV^{'} = 100$ to attack $bI^{'}$. In other words, all pixels of $bI^{'}$ are increased by $inV^{'}$. The attacked block is equal to:

$$\widetilde{bI}^{'} = \begin{bmatrix} 123 & 250 & 252 & 250 \\ 208 & 253 & 240 & 234 \\ 200 & 250 & 230 & 244 \\ 228 & 248 & 220 & 224 \end{bmatrix},$$

In Peng et al.'s method and the proposed method, there are two watermarking images $I^{'}$ and $I^{''}$. Therefore, for a pair of two blocks $bI^{'}$ and $bI^{''}$. First, need to compute $deV^{'}$, $inV^{'}$ of $bI^{'}$, and $deV^{''}$, $inV^{''}$ of $bI^{''}$. Then define $deV = min(deV^{'}, deV^{''})$, and $inV = min(inV^{'}, inV^{''})$. Finally, if $deV \geq inV$, using deV to attack both blocks $bI^{'}$ and $bI^{''}$ simultaneously. Otherwise, inV is used. Figure 5 illustrates a watermarking image of the Sailboat image and an image obtained after attacking this watermarking image by modifying 7680 blocks of size 4 × 4 located in the first 240 rows of the image.

This attack makes significant changes to the watermarking images that we can easily perceive with the naked eye. However, according to Table 3, both Yin et al.'s and Peng et al.'s methods failed to detect any block that is attacked. This has been explained in Sections 2.3 and 2.5. The method of Hong et al. detected 55.29% of 4 × 4 blocks being hacked.



(a) Marked (b) Attacked

Fig. 5 Watermarking image and attacked image by the special attack

While the proposed method with block sizes 4 × 4, 3 × 3 and 2 × 2 have a detection ratio of 100%, 100% and 99.80% respectively.

In summary, the experimental results confirmed that the proposed method has superior attack detection ability compared to other methods, while always maintaining high watermarking image quality (PSNR more than 50).

4.3 Computational complexity comparison

The proposed method and the method of Hong et al. used the characteristics of each block as input data to the MD5 hash function to generate AC bits for each block. Both of these methods achieve higher attack detection capabilities than the other two methods. Therefore their computational complexity is also higher. The running time of the proposed method and the method of Hong et al. are similar. This conclusion holds for the methods of Peng et al. and Yin et al. Specifically, the running time in the AC bit embedding phase is 1.0552, 0.7716, 4.9216 and 4.8619 (seconds) for the methods of Peng et al., Yin et al., Hong et al., and the proposed method. The running time in the hacked block detection phase is 0.8741, 1.0636, 4.9498 and 5.1961 for the above methods. Thus, the proposed method has a significantly higher ability to detect attack blocks than existing methods, but its computational complexity is only approximately that of Hong et al.'s method.

Table 3 Detection ability comparison for a special attack on the Sailboat image

Method	Number of blocks	Number of attacked blocks	TP	FN	TN	FP	TPR (%)
Yin	16384	7680	0	7680	8704	0	0
Hong	16384	7680	4274	3406	8704	0	55.29
Peng	16384	7680	0	7680	8704	0	0
Proposed 4x4	16384	7680	7680	0	8704	0	100
Proposed 3x3	28900	13600	13600	0	15300	0	100
Proposed 2x2	65536	30720	30661	59	34816	0	99.80

5 Conclusion

In this paper, we have improved the reversible data hiding methods on two images of Lu et al. and Yao et al. by embedding a 5-nary number of *k* digits on *k* pixels with values from 1 to 254. The average embedding ratios of our improvement are 2.3159, 2.2701 and 2.2776 bpp for block sizes of 4×4, 3×3 and 2×2 respectively, which is larger than the embedding ratios 2 and 2.25 bpp of Lu et al.'s and Yao et al.'s methods. Due to the large embedding capacity and the AC bits generated for each block individually by using the MD5 hash function on the information about the value and location of the block, the proposed method can detect any type of attack with high accuracy. For the attack type of modifying slightly 2048 blocks that are distributed uniformly out of a total of 16384 blocks of size 4×4, the correct detection ratios of Yin et al.'s, Hong et al.'s and Peng et al.'s methods are 42.30%, 64.59%, and 85.91%, respectively, while the correct detection ratios of the proposed method with block sizes of 4×4, 3×3 and 2×2 are 100%, 100%, and 99.91%. In the second attack, all 7680 blocks of size 4×4 in the first 240 rows of a watermarking image are transformed by increasing or decreasing all pixels of each block 4×4 by the same value which is the largest possible multiple of 4. For this attack, the methods of Yin et al. and Peng et al. can not detect any hacked block, Hong et al.'s method has a correct detection rate of 55.29%, while the proposed method with block sizes of 4×4, 3×3 and 2×2 has a correct detection rate of 100%, 100% and 99.80%, respectively. In summary, the proposed reversible image authentication method has a superior attack detection ability compared to existing methods, while maintaining high watermarking image quality (values of PSNR greater than 50 dB).

Data Availability No data were used to support this study.

Declarations

Conflict of interest Authors declare that they have no conflict of interest.

References

1. Lou DC, Liu JL (2000) Fault resilient and compression tolerant digital signature for image authentication. IEEE Trans Consum Electron 46(1):31–39
2. Zhang YD, Tang S, Li JT (2007) Secure and incidental distortion tolerant digital signature for image authentication. J Comput Sci Technol 22(4):618–625
3. El Bakrawy LM, Ghali NI, ella Hassanien A, Kim Th (2011) A rough k-means fragile watermarking approach for image authentication. In: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 19–23. IEEE
4. Zhang H, Wang C, Zhou X (2017) Fragile watermarking based on lbp for blind tamper detection in images. Journal of Information Processing Systems 13(2):385–399
5. Chang CC, Chen KN, Lee CF, Liu LJ (2011) A secure fragile water marking scheme based on chaos-and-hamming code. J Syst Softw 84(9):1462–1470
6. Hu YC, Lo C-C, Chen W-L, Wen C-H (2013) Joint image coding and image authentication based on absolute moment block truncation coding. J Electron Imaging 22(1):013012
7. Li W, Lin CC, Pan JS (2016) Novel image authentication scheme with fine image quality for btc-based compressed images. Multimedia tools and applications 75(8):4771–4793
8. Lin CC, Huang Y, Tai WL (2017) A novel hybrid image authentication scheme based on absolute moment block truncation coding. Multimedia tools and applications 76(1):463–488
9. Tsaur WJ, Li JH, Lee WB (2012) An efficient and secure multi-server authentication scheme with key agreement. J Syst Softw 85(4):876–882

10. Wang SJ, Lin IS, Lin LC, Chiang WY (2009) A secret sharing authentication scheme for digital images. J Comput 20(1):50–62
11. Hong W, Chen M, Chen TS (2017) An efficient reversible image authentication method using improved pvo and lsb substitution techniques. Signal Processing: Image Communication 58:111–122
12. Lo CC, Hu YC (2014) A novel reversible image authentication scheme for digital images. Signal Process 98:174–185
13. Nguyen TS, Chang CC, Shih TH (2016) A high-quality reversible image authentication scheme based on adaptive pee for digital images. KSII Transactions on Internet and Information Systems (TIIS) 10(1):395–413
14. Nguyen TS, Vo PH (2021) Reversible image authentication scheme based on prediction error expansion. Indonesian Journal of Electrical Engineering and Computer Science 21(1):253–262
15. Yin Z, Niu X, Zhou Z, Tang J, Luo B (2016) Improved reversible image authentication scheme. Cogn Comput 8(5):890–899
16. Hong W, Chen TS (2012) A novel data embedding method using adaptive pixel pair matching. IEEE Trans Inf Forensics Secur 7(1):176–184
17. Mielikainen J (2006) Lsb matching revisited. IEEE Signal Process Lett 13(5):285–287
18. Wu DC, Tsai WH (2003) A steganographic method for images by pixel value differencing. Pattern Recogn Lett 24(9–10):1613–1626
19. Zhang X, Wang S (2006) Efficient steganographic embedding by exploiting modification direction. IEEE Commun Lett 10(11):781–783
20. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Trans Circuits Syst Video Technol 13(8):890–896
21. Ni Z, Shi YQ, Ansari N, Su W (2006) Reversible data hiding. IEEE Trans Circuits Syst Video Technol 16(3):354–362
22. Alattar AM (2004) Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans Image Process 13(8):1147–1156
23. Khodaei M, Faez K (2010) Reversible data hiding by using modified difference expansion. In: 2010 2nd International conference on signal processing systems, vol. 3, pp. 31–34. IEEE
24. Lee CC, Wu HC, Tsai CS, Chu YP (2008) Adaptive lossless stegano graphic scheme with centralized difference expansion. Pattern Recogn 41(6):2097–2106
25. Hu Y, Lee HK, Li J (2008) De-based reversible data hiding with improved overflow location map. IEEE Trans Circuits Syst Video Technol 19(2):250–260
26. Kamstra L, Heijmans HJ (2005) Reversible data embedding into images using wavelet techniques and sorting. IEEE Trans Image Process 14(12):2082–2090
27. Liu M, Seah HS, Zhu C, Lin W, Tian F (2012) Reducing location map in prediction-based difference expansion for reversible image data embedding. Signal Process 92(3):819–828
28. Sachnev V, Kim HJ, Nam J, Suresh S, Shi YQ (2009) Reversible water marking algorithm using sorting and prediction. IEEE Trans Circuits Syst Video Technol 19(7):989–999
29. Bharanitharan K, Chang CC, Rui YH, Wang ZH (2016) Efficient pixel prediction algorithm for reversible data hiding. Int. J. Netw. Secur. 18(4):750–757
30. Dragoi IC, Coltuc D (2014) Local-prediction-based difference expansion reversible watermarking. IEEE Trans Image Process 23(4):1779–1790
31. Thodi DM, Rodriguez JJ (2004) Prediction-error based reversible water marking. In: 2004 International conference on image processing, 2004. ICIP'04., vol. 3, pp. 1549–1552. IEEE
32. Thodi DM, Rodríguez JJ (2007) Expansion embedding techniques for reversible watermarking. IEEE Trans Image Process 16(3):721–730
33. Chen X, Sun X, Sun H, Xiang L, Yang B (2015) Histogram shifting based reversible data hiding method using directed-prediction scheme. Multimedia Tools and Applications 74(15):5747–5765
34. Kukreja S, Kasana SS, Kasana G (2019) Histogram based multilevel reversible data hiding scheme using simple and absolute difference images. Multimedia Tools and Applications 78(5):6139–6162
35. Li YC, Yeh CM, Chang CC (2010) Data hiding based on the similarity between neighboring pixels with reversibility. Digital Signal Processing 20(4):1116–1128
36. Lin CC, Tai WL, Chang CC (2008) Multilevel reversible data hiding based on histogram modification of difference images. Pattern Recogn 41(12):3582–3591
37. Shin SY, Yoo HM, Suh JW (2014) Reversible watermarking based on histogram shifting of difference image between original and predicted images. In: IARIA, pp. 147–150
38. Li X, Li J, Li B, Yang B (2013) High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. Signal Process 93(1):198–205
39. Li J, Wu YH, Lee CF, Chang CC (2018) Generalized pvo-k embedding technique for reversible data hiding. Int. J. Netw. Secur. 20(1):65–77

40. Ou B, Li X, Zhao Y, Ni R (2014) Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion. Signal processing: image communication 29(7):760–772
41. Ou B, Li X, Li W, Shi YQ (2018) Pixel-value-ordering based reversible data hiding with adaptive texture classification and modification. In: International Workshop on Digital Watermarking, pp. 169–179. Springer
42. Peng F, Li X, Yang B (2014) Improved pvo-based reversible data hiding. Digital Signal Processing 25:255–265
43. Sao NK, Hoa NN, Van At P (2020) An effective reversible data hiding method based on pixel-value-ordering. Journal of Computer Science and Cybernetics 36(2):139–158
44. Weng S, Js Pan, Li L (2016) Reversible data hiding based on an adaptive pixel-embedding strategy and two-layer embedding. Inf Sci 369:144–159
45. Weng S, Shi Y, Hong W, Yao Y (2019) Dynamic improved pixel value ordering reversible data hiding. Inf Sci 489:136–154
46. Chang CC, Kieu TD, Chou YC (2007) Reversible Data Hiding Scheme Using Two Steganographic Images. In: TENCON 2007-2007 IEEE Region 10 Conference, pp. 1–4. IEEE
47. Qin C, Chang CC, Hsu TJ (2015) Reversible data hiding scheme based on exploiting modification direction with two steganographic images. Multimedia Tools and Applications 74(15):5861–5872
48. Lu TC, Tseng CY, Wu JH (2015) Dual imaging-based reversible hiding technique using lsb matching. Signal Process 108:77–89
49. Lee CF, Huang YL (2013) Reversible data hiding scheme based on dual stegano-images using orientation combinations. Telecommun Syst 52(4):2237–2247
50. Lu TC, Wu JH, Huang CC (2015) Dual-image-based reversible data hiding method using center folding strategy. Signal Process 115:195–213
51. Yao H, Qin C, Tang Z, Tian Y (2017) Improved dual-image reversible data hiding method using the selection strategy of shiftable pixels' coordinates with minimum distortion. Signal Process 135:26–35
52. Chen X, Hong C (2021) An efficient dual-image reversible data hiding scheme based on exploiting modification direction. Journal of Information Security and Applications 58:102702
53. Peng Y, Niu X, Fu L, Yin Z (2018) Image authentication scheme based on reversible fragile watermarking with two images. Journal of information security and applications 40:236–246
54. Yin ZX, Chang CC, Xu Q, Luo B (2015) Second-order steganographic method based on adaptive reference matrix. IET Image Proc 9(4):300–305

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Van-At Pham[1] · **Ngoc-Hung Nguyen**[2] · **Quang-Hoa Le**[3] · **Kim-Sao Nguyen**[4] · **Thi-Luyen Cao**[4] · **Minh-Thai Pham**[1]

Van-At Pham
phamvanat83@gmail.com

Quang-Hoa Le
hoa.lequang1@hust.edu.vn

Kim-Sao Nguyen
saonkoliver@utc.edu.vn

Thi-Luyen Cao
luyenct@utc.edu.vn

Minh-Thai Pham
pmthai@uneti.edu.vn

[1] Faculty of Information Technology, University of Economics - Technology for Industries, 456 Minh Khai Street, Hanoi, Vietnam

[2] Department of Telematics, Institute of Information Technology, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet Street, Hanoi, Vietnam

[3] School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, 1 Dai Co Viet Street, Hanoi, Vietnam

[4] Faculty of Information Technology, University of Transport and Communications, 3 Cau Giay Street, Hanoi, Vietnam