applied sciences

MDPI

Article

Self-Embedding Fragile Watermarking Scheme to Detect Image Tampering Using AMBTC and OPAP Approaches

Cheonshik Kim [1,*,†] and Ching-Nung Yang [2,*,†]

[1] Department of Computer Engineering, Sejong University, Seoul 05006, Korea
[2] Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien 97401, Taiwan
* Correspondence: mipsan@sejong.ac.kr (C.K.); cnyang@gms.ndhu.edu.tw (C.-N.Y.)
† These authors contributed equally to this work.

Abstract: Research on self-embedding watermarks is being actively conducted to solve personal privacy and copyright problems by image attack. In this paper, we propose a self-embedded watermarking technique based on Absolute Moment Block Truncation Coding (AMBTC) for reconstructing tampered images by cropping attacks and forgery. AMBTC is suitable as a recovery bit (watermark) for the tampered image. This is because AMBTC has excellent compression performance and image quality. Moreover, to improve the quality of the marked image, the Optimal Pixel Adjustment Process (OPAP) method is used in the process of hiding AMBTC in the cover image. To find a damaged block in a marked image, the authentication data along with the watermark must be hidden in the block. We employ a checksum for authentication. The watermark is embedded in the pixels of the cover image using 3LSB and 2LSB, and the checksum is hidden in the LSB. Through the recovering procedure, it is possible to recover the original marked image from the tampered marked image. In addition, when the tampering ratio was 45%, the image (Lena) could be recovered at 36 dB. The proposed self-embedding method was verified through an experiment, and the result was the recovered image showed superior perceptual quality compared to the previous methods.

Keywords: watermarking; self-embedding; digital signature; AMBTC; fragile watermarking

check for updates

Citation: Kim, C.; Yang, C.-N. Self-Embedding Fragile Watermarking Scheme against Tampering Image by Using AMBTC and OPAP Approaches. *Appl. Sci.* **2021**, *11*, 1146. https://doi.org/10.3390/app11031146

Academic Editor: Francesco Bianconi
Received: 6 January 2021
Accepted: 22 January 2021
Published: 27 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional clai-ms in published maps and institutio-nal affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the advanced high-speed communication technology, recently, many SNS subscribers freely share the digital contents they have created, and with useful image processing software, digital contents are easily manipulated to create interesting images. In addition, images are deliberately or unintentionally manipulated during transmission, causing many social problems. For this reason, the problem of verifying the integrity of an image is becoming an important area of image security.

To solve such a problem, in the past, several signature-based image authentication schemes [1,2] were proposed for integrity verification. Digital signatures are always stored by third parties in a digital signature-based method. In this approach, the digital signature extracted from the image is compared to a digital signature stored by a third party. Comparing the two signatures may detect if the image has been tampered with [3–5]. This method makes it easy to determine whether an image is authentic or not, but they cannot find the tampered area. Besides, adding signatures requires additional bandwidth and storage space. Digital signatures have these obvious limitations. First of all, to recover a marked image with high quality, a method is required that can accurately detect the tampered area of the marked image. As an alternative to digital signatures, the watermarking technology not only detects tampered areas using watermarks, but it also suggests an alternative to recover marked images and is currently being actively studied.

Watermarking methods are classified as strong watermarking [6–9], semi-fragile watermarking [10–12] and fragile watermarking [13–16]. The strong watermarking method

allows you to extract hidden watermarks from watermarked images, even after image processing (e.g., image compression and filtering). Thus, it can be exploited to verify copyright and intellectual property rights. The fragile watermarking technique can be easily destroyed by simple image processing; thus, it can accurately detect the tampered area. There are currently two types of fragile watermarking techniques. The first type detects only the tampered area from the cover image. The second can detect and find the tampered area as well as recover the area on the image.

Self-embedding is a way of recovering the tampered area with the recovered bits, which are embedded in the pixels of the cover image, where the recovering bits are composed of the feature of the original image. The performance of the self-embedding method based on watermarking technology is generally evaluated by the quality of the recovered image. In most self-embedding methods, the recovery bits of a specific block are always hidden in the other block of the image. A method like this can fail if the block containing the recovery bit has been tampered with. This is called the tampering coincidence problem [17].

The most important factor for image recovery depends on the ability to detect forged areas. Walton [5] proposed the first fragile watermarking method for detection of tampered areas based on inserting checksums in gray levels. Fridrich et al. [18,19] introduced a self-embedding method based on DCT. Here, the DCT is converted to a bitstream, and then it is embedded to pixels of the distant block. The reconstruction quality using Algorithm 1 in this method is 50% quality, which is significantly worse than that for a JPEG compressed image. He et al. [19] proposed an adjacent block-based statistical detection method to accurately identify the tampered block, and they provided an analysis of the tampered detection performance. It has been shown that the statistical detection method can identify the tampered block of the host image. However, there may also be a recovering problem due to statistical error.

Lin et al. [20] introduced a hierarchical-based watermarking method to detect and recover cover image damage. It is effective because the detection is based on a hierarchical structure so that the accuracy of tamper localization can be ensured. The drawback is that it is not possible to check whether the location of the error is an error of a lower block of the current block or an error of a lower block within the same block. Therefore, the scheme [21] proposed a new mechanism to facilitate recovery with a higher probability by inserting a double copy of the watermark into two different blocks.

Zhang and Wang (2008) [22] proposed a new vector coding-based fragile watermarking method that can recover the tampered area without error, as long as it is not too serious. For restoration, recovery and authentication, bits are compressed losslessly and then are embedded in the cover image. The mean PSNR is about 28.70 dB. Moreover, if the tampering rate is less than 3.2%, the tampered area may be totally recovered. To improve the tampering rate, Zhang et al. (2009) [23] proposed another fragile watermarking scheme that restores the content of the original image. While the reference bits are embedded into the entire cover image, the hash bits are embedded into the local blocks.

Qian et al. [24] proposed a fragile watermarking method for high-quality restoration. They first categorized the image into one of six types depending on the degree of smoothness. Complex blocks were compressed into more bits for recovery and smooth blocks were needed fewer bits for recovery. Finally, the recovery bit and authentication bit were embedded into the three number of LSBs of every pixel for the image.

Luo et al. [25] proposed a self-contained watermarking scheme for digital images. The host image was converted into a halftone image using a digital halftone technique, and the converted pixels were used as recovery bits. Halftone can preserve the characteristics of the host image in the most compressed type. The halftoning watermark was used for tamper detection, and the tampered area can be approximately recovered using the extracted watermark. They adopted a simple low-pass filtering approach for inverse halftoning. The reconstructed image based on halftone is not satisfactory from the perspective of the image quality. Hsu and Tu's study [26] used the degree of smoothness to distinguish the types of image blocks, and they employed different watermark embedding, tamper detection and

recovery strategies for different block types to enhance hiding efficiency, authentication and recovery effects.

Yang & Shen [27] proposed a method to detect and recover images tampered with by integrating Wong's watermarking method [28] and vector quantization (VQ). This integration also required a little extra cost, i.e., an increase in codebook size. However, with a codebook, the image is recovered by using VQ if the mapping information for recovery is lost. Due to the limitation of the quality of VQ, the restored image is not of high quality.

In [29–31], they proposed a fragile watermarking technique based on Block Truncation Coding (BTC). In this method, the bitstream compressed with BTC [32] or AMBTC [33] of the original image was hidden in the LSB and 2LSB of the cover image to store the features of the original image. When a part of the image has been tampered, the location is detected, and the information on the tampered area is recovered with AMBTC. Kim et al. [29] adopted a method of improving the image quality through Gaussian filtering after using the reconstructed bits for image restoration. Hemida et al. [30] used a quantum chaos map to escape the tampering attack of the mark. The error rate of tamper detection using the XOR operation between the bitmap of each block and the binary random number may higher than that of using the decimal number. Chang et al. [31] proposed a method to improve the quality of marked images by enhancing the compression performance of AMBTC encoding bits for the original image.

The quality of the reconstructed image was not good due to the loss of recover bits according to compress the bitmaps. They employed an inpainting technique to improve the quality of the recovered image. For the tampered block, the most important thing to recover depends on how to exactly find the tampered location. In this paper, we propose a fragile watermark technique based on self-embedding using AMBTC to restore the tampered cover image. To improve the quality of the cover image, it uses Optimal Pixel Adjustment Process (OPAP) [34] to encode self-embedded data (Watermark). OPAP is introduced to optimize the error in the DH process using LSB replacement, and it is a coding method with excellent performance. In this scheme, we used checksum for authentication of every block and embeded authentication bits in every block to detect forgery. Although the checksum is a simple method, it guarantees relatively accurate performance in detecting whether or not it is a tampered block through threshold comparison. This improves the accuracy of tamper detection and localization.

This scheme has several advantages: (1) high accuracy of tampered detection; (2) the quality of the recovered marked image is guaranteed by the use of high-quality compression bits generated by AMBTC; (3) the quality of marked images is guaranteed because the original image encoded with AMBTC is hidden in the cover-image using OPAP; (4) recovering bits for a block are concealed at a far distance from the current block to prepare for cropping attacks. Experimental results show that the proposed scheme allows high-quality recovery up to a modulation rate of 45%.

The rest of this paper is organized as follows. Section 2 briefly introduces watermark technology. Section 3 presents a review of current and related work. Section 4 introduces the proposed self-embedding watermarking scheme. Section 5 explains the experimental results, and Section 6 provides the conclusions and future work.

2. Conspectus of Watermarking Technology

In this section, an overview of watermarking technology and its main terms is introduced.

2.1. Watermark Requirements

The critical requirements that a watermarking method must have are as follows [35]. First, embedding capacity: it must have capable of storing data of sufficient capacity to protect the copyright of digital contents. Second, robustness: the embedded watermark in a cover image needs to be able to resist various attacks such as compression of images and image processing. Third, security: attackers must not be able to easily access the embedded watermark. Fourth, unrecognizable: It should be possible to hide the presence of

watermarks by preventing distortion of marked images in case of embedding a watermark in the cover image. Fifth, blind: it should be possible to recover the watermark without reference to the original image.

2.2. Watermarking Techniques Classification

Watermarking technology is the most basic application used for copyright protection of digital content such as (color) images [35], video [36] and 3D mesh [37]. That is, the ownership information of the watermark is exploited for identifying copyright ownership and preventing fraud and theft of digital content. In fact, marked digital images can prove ownership when someone claims it by a legitimate owner. In addition, authentication is another watermarking application that aims to verify the integrity of the watermarked digital images and detect attempts to alter the original images. These watermarks are designed to be subject to signal manipulation and are used to indicate the authenticity of digital content.

Watermarking technology can be divided according to several perspectives. First, according to human perception, it is divided into two types: visible watermarking technology and invisible technology. The former means that the watermark is visible from digital images, and the latter means that the watermark cannot be recognized by the human eye. Second, watermarking technology is divided into non-blind and blind, depending on whether the original image is needed for watermark recovery. In the non-blind technique, both the original image and watermark are required during the authentication of the watermark. The blind technique does not require a watermark or original image. Third, it is divided into a spatial domain and a frequency domain based on the work area. The former is done by directly manipulating the pixel values in the original image. The merit of the work based on the spatial domain is its simple implementation and low computational complexity, while its demerit is its weak robustness to compression. In the latter case, you need to convert the host image to an appropriate frequency working domain. Then, the coefficient is adjusted according to the values of a watermark. In general, domain transformation techniques are Discrete Cosine Transform (DCT), Singular Value Decomposition (SVD), and Discrete Wavelet Transform (DWT) [18,38,39]. The frequency domain-based approach is more resilient to compression attacks and image conversion attacks [36].

Fourth, depending on whether the watermark can withstand various attacks, it is classified into a strong watermark, a fragile watermark or a semi-fragile watermark. The strong watermark-based method provides the performance to withstand compression and various image manipulations. To do this, it is characterized by converting the image into the frequency domain. The fragile watermark-based method is vulnerable to image compression and image processing, so the use of this method may be different. This is because the hidden information cannot be restored even with trivial image processing. The semi-fragile watermark-based method provides selective robustness for specific manipulations.

There are two ways to watermark a color image in the conversion domain. The first uses gray level techniques to process each channel individually, and the second treats each pixel in the color image into a quaternion vector to which the transformation is applied.

3. Preliminaries

3.1. AMBTC

The Block Truncation Coding (BTC) [32] is a simple lossy compression method based on moment preserving quantization for blocks of pixels in a grayscale image. Since BTC produces a set of bitmap, mean and standard deviation to represent a block, it gives a CR (size of the original image/size of the compressed image) of 4; hence, the bit rate is 2 bits per pixel for a 4×4 block. Though the BTC method provides good compression without much degradation on the reconstructed images, it shows some artifacts like the staircase effect. Absolute Moment Block Truncation Coding (AMBTC) [33] preserves the higher mean and lower mean of each of the blocks and improves the staircase effect of the

conventional BTC method. Besides, AMBTC is simpler than BTC, thus the computation speed is very fast. The AMBTC algorithm involves the following steps:

Step 1: The original image of size $N \times N$ is divided into non-overlapping blocks (C) of the size $m \times m$ (let $m = 4$), and each block is processed separately. Let $m^2 = k$.

Step 2: For each block, the average pixel value is calculated by Equation (1).

$$\bar{x} = \frac{1}{k} \sum_{i=1}^{k} x_i \tag{1}$$

where x_i represents the ith pixel value of this block with the size of k. All pixels in the block are quantized into a bitmap $b_i(0 \text{ or } 1)$ using Equation (2). That is, if the corresponding pixel x_i is greater than or equal to the average ($\bar{x}$), it is assigned with '1', otherwise it is '0'. Pixels in each block are divided into two groups of '1' or '0'.

$$b_i = \begin{cases} 1, & \text{if } x_i \geq \bar{x}, \\ 0, & \text{if } x_i < \bar{x}. \end{cases} \tag{2}$$

Step 3: The block $\mathcal{M}$ is partitioned into two sets of pixels $\mathcal{M}_0$ and $\mathcal{M}_1$ such that $\mathcal{M} = \mathcal{M}_0 \cup \mathcal{M}_1$ and $\mathcal{M}_0 \cap \mathcal{M}_1 = \phi$ where $\mathcal{M}_0 = \{0_0, 0_1, \ldots, 0_t\}$ and $\mathcal{M}_1 = \{1_1, 1_2, \ldots, 1_{k-t}\}$,, and t and $k - t$ refer to the numbers of pixels in the '0' and '1' groups, respectively. The means Q_1 and Q_2 of the two groups indicate the quantization levels of the groups '0' and '1'. The two quantization levels are calculated by Equations (3) and (4).

$$Q_1 = \left\lfloor \frac{1}{t} \sum_{x_i < \bar{x}} x_i \right\rfloor \tag{3}$$

$$Q_2 = \left\lfloor \frac{1}{k-t} \sum_{x_i \geq \bar{x}} x_i \right\rfloor \tag{4}$$

Step 4: To reconstruct the pixel marked by '0' it will be given the value Q_1, and that marked by '1' will be given the value Q_2. The values Q_1 and Q_2 satisfy the following relation. The compressed block is simply uncompressed by using Equation (5).

$$g_i = \begin{cases} Q_1, & \text{if } b_i = 0, \\ Q_2, & \text{if } b_i = 1. \end{cases} \tag{5}$$

The image block is compressed into two quantization levels Q_1 and Q_2, and a bitmap $\mathcal{M}$, and it can be represented as a $trio(Q_1, Q_2, \mathcal{M})$. A bitmap $\mathcal{M}$ contains the bit-planes that represent the pixels, and the values Q_1 and Q_2 are used to decode the AMBTC-compressed image by using Equation (5). If the block size is 4×4 then it will give the 32-bit compressed data (i.e., the size of the block bitmap is 16 bits; converting Q_1 and Q_1 to binary results in 16 bits), and hence the bit rate is 2 bpp. For $m = 4$, 16 pixels are represented by a $trio(Q_1, Q_2, \mathcal{M})$ of 8 + 8 + 16 = 32 bits, so the compression ratio (CR) is $(16 \times 8)/32 = 4$. For 512×512 pixel images, the file size of 2M-bits can be reduced to 0.5 M-bits.

3.2. LSB Substitution and OPAP

LSB (Least-Significant-Bit) alternative technology is a method of directly concealing the watermark in the LSB of the pixels constituting the cover image. Wang et al. [34] introduced an optimal LSB substitution and genetic algorithms, and it was found that the Worst-case Mean Squared Error (WMSE) (which is a measurement obtained by comparing the original and marked image) is 1/2 of that obtained with simple LSB substitution techniques. Let us look at the DH procedure for the original 8-bit grayscale represented by $x_i \in \{0, 1, \ldots, 255\}$. S denotes n-bit hidden values represented as $S = \{s_k | 0 \leq k < n, s_k \in \{0, 1\}\}$. The mapping between the n-bit secret bits $S = \{s_k\}$ and the embedded bits $S' = s'_k$ can be defined as

follows: $s'_k = \sum_{j=0}^{\delta-1} s_{k\times\delta+j} \times 2^{\delta-1-j}$. The pixel value x_i for embedding the δ-bit s'_k is changed to form the stego-pixel x'_i like $x'_i = x_i - (x_i \bmod 2^\delta) + s'_k$. The δ LSBs of the pixels are extracted by $s_k = x'_i \bmod 2^\delta$.

It has been mathematically proven that OPAP can improve the quality of marked images by reducing WMSE by using LSB replacement based on the minimization rule. Let x_i be the pixel of the cover image, x'_i be the obtained pixel from pixel x_i using the LSB replacement, and x''_i is the optimized pixel derived from x'_i by the OPAP method. The value of Δ_i may be segmented into three intervals. The OPAP modifies x' to form the stego-pixel x'' as the following rules:

1. Rule 1 ($2^{\delta-1} < \Delta_i < 2^\delta$): if $x'_i \geq 2^\delta$, then $x''_i = x'_i - 2^\delta$; otherwise $x''_i = x'_i$;
2. Rule 2 ($-2^{\delta-1} \leq \Delta_i \leq 2^{\delta-1}$): $x''_i = x'_i$;
3. Rule 3 ($-2^\delta < \Delta_i < -2^{\delta-1}$): if $x'_i < 256 - 2^\delta$, then $x''_i = x'_i + 2^\delta$; otherwise $x''_i = x'_i$;

3.3. Luo et al.'s Method

Luo et al. [25] proposed a self-embedding watermarking scheme by using the digital halftoning technique. Here, the tampered image is restored by converting the original image's features into the halftone image and secretly embedding them in the pixels of cover image. If the halftone image composed of 1's and 0's is used as the restoration bits, the size of the watermark is small, but the quality of the restored image is low because it cannot retain sufficient features for the original image.

Suppose I and W denote the host image and the watermark image, respectively, and both are of size $N \times N$. W obtains the enhanced edge by using the error diffusion halftoning algorithm. The watermark W permutes the locations of all pixels constituting the watermark using the key K. The permuted watermark W_p is embedded into the pixels' LSBs in the cover image I (Figure 1).

For reconstruction, recover W' from the marked image I', then divide I' and W' into non-overlapping $m \times m$ block BI_l and $BW_l (l = 1, 2, \ldots, N \times N/m)$ respectively. Compute the difference D between pixel values of each block BI_l and the corresponding block BW_l. If the difference is smaller than threshold T, it is the authenticated block; otherwise, it is not the authenticated block. For the tampered block, it is replaced with the same block of W'.

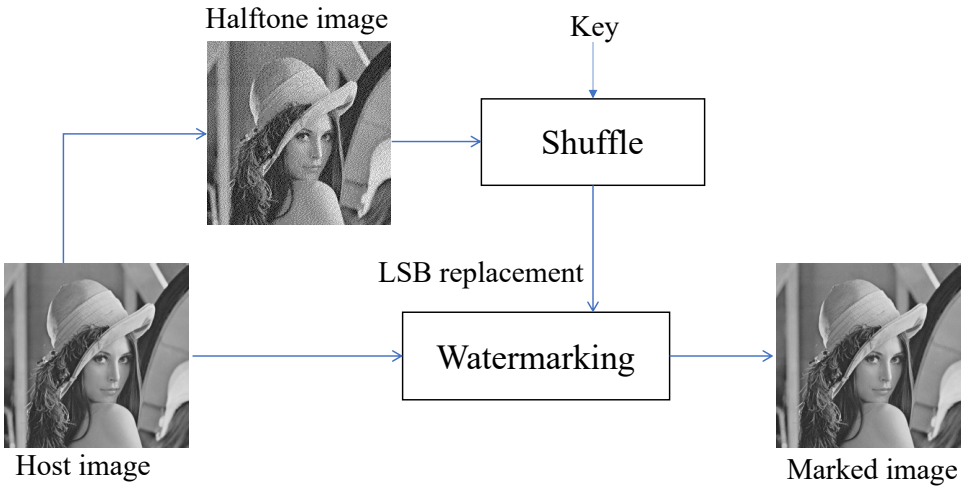


Figure 1. Block diagram of the proposed scheme—watermark generation and embedding.

3.4. Fridrich and Goljan's Method

Fridrich and Goljan [18] described a self-embedding technique. First, the cover image is divided into blocks of 8×8 pixels. Set the LSB of each pixel in a given block to 0, and convert the block into DCT. The quantized matrix is encoded with 64 bits, and the bits are embedded into the LSBs of a distant block. After embedding the watermark, on average it

modifies 5% of pixels in a block, and the quality of the reconstructed image is somewhat worse than 50% of JPEG quality. The following three steps are carried out for each block B:

Step 1: The original image is divided into blocks of 8×8 pixels. All blocks are transformed into the interval $[-127, 128]$, and the LSBs of all pixels are set to zero.

Step 2: Each 8×8 block is transformed into the frequency domain using DCT. The first 11 coefficients (in zig-zag order) are quantized with the following quantization table Q (Fridrich and Goljan [18]) that corresponds to 50% JPEG quality: The quantized values are further binary encoded. The bit lengths of their codes (including the signs) are shown in matrix L (Fridrich and Goljan [18]). Coding based on L will guarantee that the first 11 coefficients from each block will be coded using exactly 64 bits.

Step 3: The binary sequence obtained in Step 2 (e.g., the 64-bit string) is encrypted and inserted into the LSB of the block $B + \overrightarrow{p}$, where $\overrightarrow{p}$ is a vector of length approximately 3/10 of the image size with a randomly chosen direction.

4. Proposed Scheme for Self-Embedding

This section introduces an efficient self-embedding method based on AMBTC. First, the proposed method obtains the feature information of the original image by converting the original image into AMBTC. A basic configuration of the AMBTC image is a trio composed of a bitmap and two quantization levels. Next, the encrypted trios are embedded in their own pixels in the cover image, and the quality of the cover image is somewhat reduced by this procedure.

Figure 2 schematically shows the procedure of obtaining two bitmaps from the original image and the procedure of embedding the checksum and two bitmaps into the cover image after completing the mapping process. The compression method in Section 3.1 is the procedure to obtain the bitmap $\mathbb{M}_1$ and two quantization levels Q_1 and Q_2 per block (Figure 2).

Both quantization levels are converted to binary bitmap $\mathbb{M}_2$ for DH. In preparation for the cropping attack, every block of two $\mathbb{M}_1$ and $\mathbb{M}_2$ moves as far away as possible from the original location of the block using a scramble (mapping) algorithm. Afterward, the watermarks are embedded in the 2LSB and 3LSB of the pixels in the cover image. After creating a checksum for block authentication, it secretly inserts it into the pixels of the cover block. It is used for authentication of the block during the restoration process.

Figure 3 shows a simple schematic explaining the recovery procedure of a tampered watermarked image. To recover the tampered image, the concealed watermarks $\mathbb{M}_1$ and $\mathbb{M}_2$ must first be extracted from 3LSB and 2LSB (OPAP method). The original bitmap is reconstructed by applying Equation (8) to each block of $\mathbb{M}_1$ and $\mathbb{M}_2$. After converting $\mathbb{M}_2$ to a quantization level per block, a grayscale image is restored by performing a decoding process. To check the forged block, we should restore the checksum $\mathcal{V}'_{sum}$ hidden in the LSB and then generate the checksum $\mathcal{V}_{sum}$ for the block by using the key.

The two generated checksums are compared, and the authentication process is executed in block units. In other words, if the two checksums match, it means that there is no forgery attack on the block. If not, forgery has occurred. Therefore, the block is replaced with the corresponding block of the AMBTC image created as a watermark.

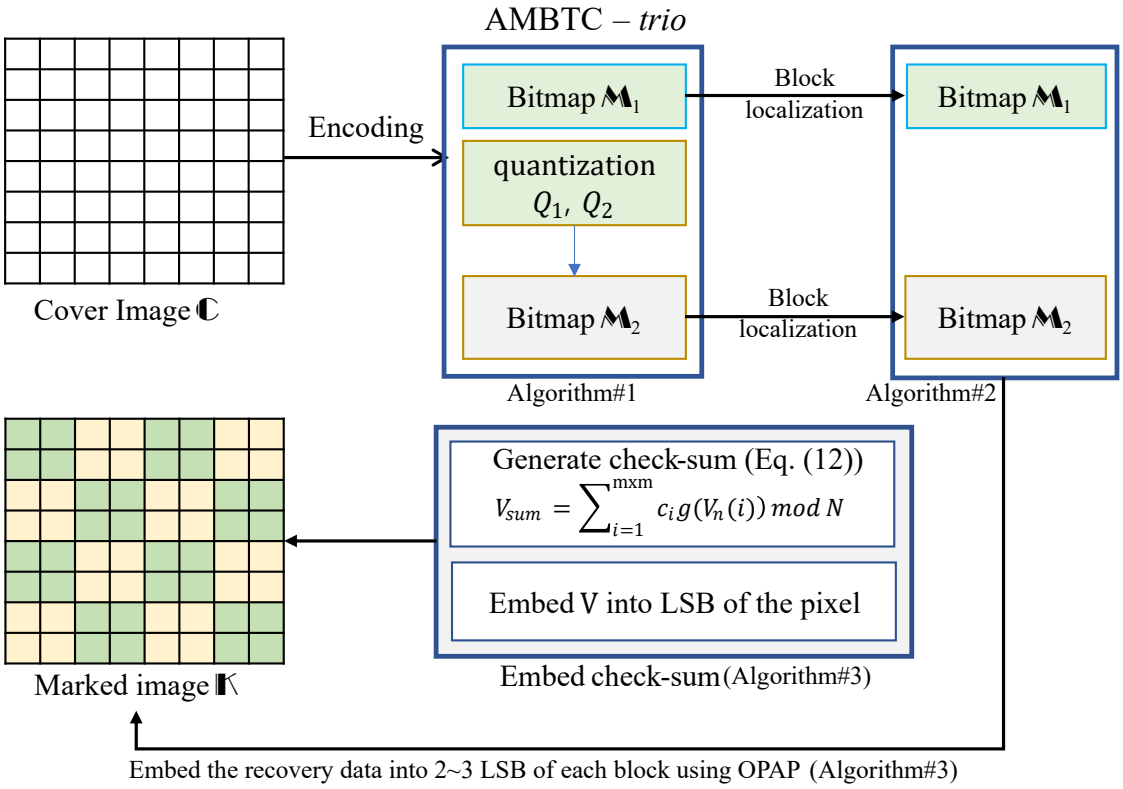


Figure 2. Block diagram for generating a watermarked image using AMBTC and the proposed embedding scheme (See Algorithms 1–3).

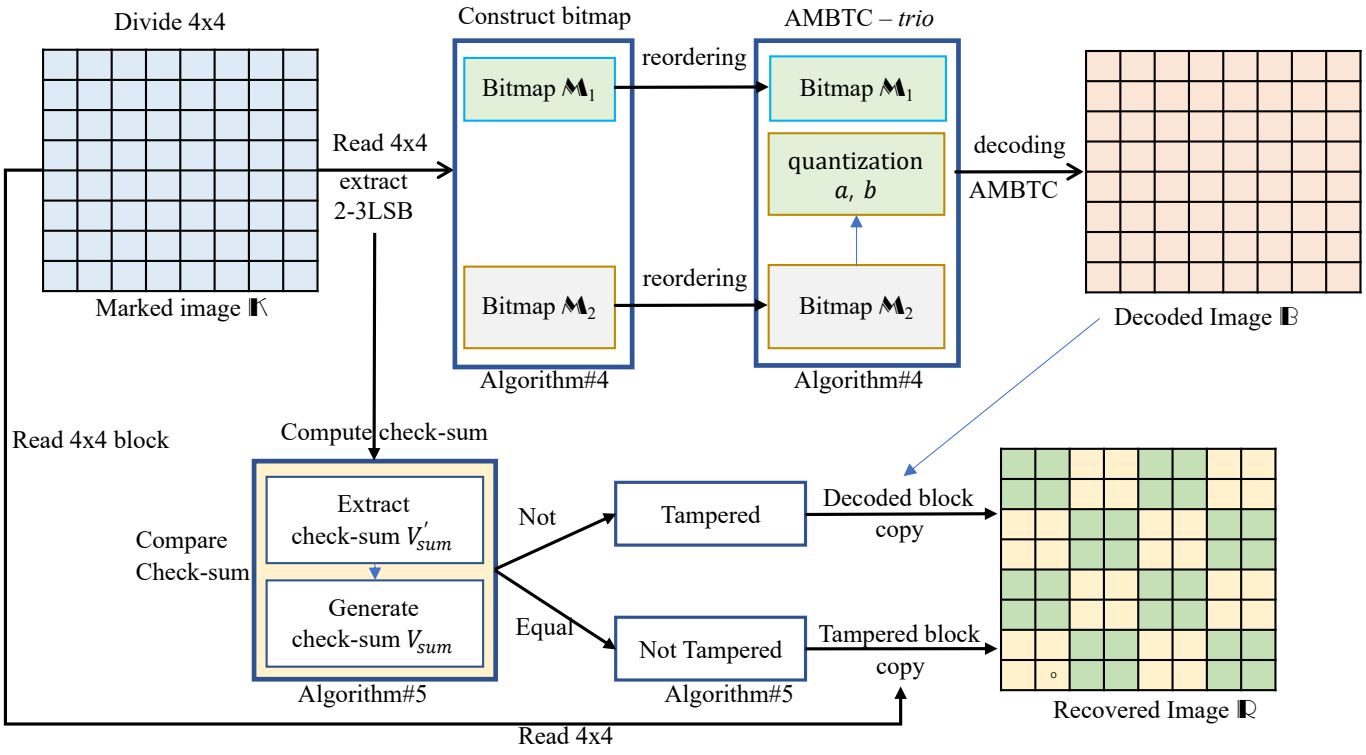


Figure 3. Block diagram for watermark extraction, tampering detection, localization and image recovery (See Algorithms 4 and 5).

4.1. Watermark Generation, Localization and Authentication

Let us suppose that $\mathbb{C}$ and $\mathbb{M}$ denote the cover image and watermark (two maps of AMBTC), respectively, and both have a size of $N \times N$. The two watermark $\mathbb{M}_{1,2}$ is taken by the previously mentioned AMBTC algorithm. The watermark $\mathbb{M}$ is randomly permuted by a key and then is subjected to the embedding procedure.

The step-by-step embedding procedure is as follows:

Algorithm 1 Watermark Generation

Input: A original image $\mathbb{O}$ with a size of $N \times N$

Output: Two bitmap $\mathbb{M}_1$ and $\mathbb{M}_2$ sized $N \times N$.

Step 1: A compressed set $trio(Q_1, Q_2, \mathcal{M})$ is obtained from the original image $\mathbb{O}$ by using the AMBTC algorithm (Section 3.1). The sized $N \times N$ bitmap $\mathbb{M}_1$ and $\mathbb{M}_2$ are initialized with zeros.

Step 2: For given $trio(Q_1, Q_2, \mathcal{M})$, the bitmap $\mathbb{M}_1$ is constructed by adding the block bitmap $\mathcal{M}_n$ to the $\mathbb{M}_1$ using Equation (6), where $n \in \{1 \leq n \leq (N \times N)/(m \times m)\}$ and $m = 4$.

$$\mathbb{M}_1^n = \sum_{n=1} \mathcal{M}_n, \text{ where } \mathcal{M} \in trio(Q_1, Q_2, \mathcal{M})_n, \tag{6}$$

Step 3: After converting the two quantization levels Q_1^n and Q_2^n into 8 bits using Equation (7) respectively, the $b_{i,t}$ are assigined to the block $\mathcal{M}_n$ sequentially. Then, $\mathcal{M}_n$ is added to the bitmap $\mathbb{M}_2$ like Equation (6), i.e., $\mathbb{M}_2^n = \sum_{n=1} \mathcal{M}_n$, where $\mathcal{M} \in trio(Q_1, Q_2, \mathcal{M})_n$.

$$b_{i,t} = \left\lfloor \frac{Q_{1,2}}{2^t} \right\rfloor \bmod 2, t = 0, 1, \ldots, 7. \tag{7}$$

If the watermark is concealed in the same order as the original image, the tampered area cannot be restored when the marked image is damaged. Therefore, the recovery bits, watermarks, are not embedded into the block itself. These watermarks are embedded in LSBs of the mapped block $\mathcal{M}_j$. Here, blocks $\mathcal{M}_i$ and $\mathcal{M}_j$ are chosen such that $\{i \neq j | (i, j) \in [1, 2, \ldots, R]\}$, where R is the total number of blocks in the cover image.

The procedure of watermark localization is as follows:

Algorithm 2 Block Mapping

Input: Two watermark bitmaps, $\mathbb{M}_1$ and $\mathbb{M}_2$ in Algorithm 1, Key ξ

Output: Mapped (scrambled) bitmaps $\mathbb{M}_1$, and $\mathbb{M}_2$

Step 1: Divides the bitmap $\mathbb{M}_1$ into non-overlapping blocks of $m \times m$ pixels. Read a block $\mathcal{M}_n$ from $\mathbb{M}_1$, where $n \in \{1 \leq n \leq R\}$ and $R = (N \times N)/(m \times m)$ and $m = 64$. The optimal position j for the block $\mathcal{M}_n$ is obtained by using Equation (8), where ξ is prime number. Swap the block $\mathcal{M}_n$ with the block $\mathcal{M}_j$ in the map $\mathbb{M}_1$.

$$j = \begin{cases} f(i) = (\xi \times n) \bmod \mathrm{R} \\ \quad j = f(n) + 1; \end{cases} \tag{8}$$

When dividing the image into 16 areas, the block location n and j must not be in the same, and the key ξ must search for its location, which can be the most distant from each other.

Step 2: Read a block $\mathcal{M}_n$ from the bitmap $\mathbb{M}_2$. The optimal position j for the block $\mathcal{M}_n$ is obtained by using Equation (8) and swap the block $\mathcal{M}_n$ with the block $\mathcal{M}_j$ in the map $\mathbb{M}_2$.

Step 3: Repeat Step-1 and Step-2 for all blocks.

4.2. Watermark Embedding Procedure

In Section 4.1, we introduced obtaining bitmaps $\mathbb{M}_1$ and $\mathbb{M}_2$ for the restoration of marked images using Algorithm 1. However, if one block of the marked image has tampered with, $\mathbb{M}_1$ and $\mathbb{M}_2$ of the block have tampered with, so localization is required. Algorithm 2 introduced obtaining localized maps $\mathbb{M}_1$ and $\mathbb{M}_2$. Section 3.2 introduces the procedure of hiding the two maps and the authentication bits in the cover image. Here, the three LSB layers of the cover image are replaced with watermark bits and authentication bits.

The watermark embedding procedure is as follows:

Algorithm 3 Watermark Embedding

Input: cover image $\mathbb{C}$

Output: marked image $\mathbb{K}$

Step 1: Divide the cover image $\mathbb{C}$ and two maps ($\mathbb{M}_1$ and $\mathbb{M}_2$) in Algorithm 2 into non-overlapping blocks sized $m \times m$, where $m = 4$.

Step 2: Read three blocks from the cover image $\mathbb{C}$, two maps $\mathbb{M}_1$ and $\mathbb{M}_2$, respectively, then these blocks are assigned to $\mathcal{P}_n$, $\mathcal{M}_1$, and $\mathcal{M}_2$, where $n \in \{1 \le n \le (N \times N)/(m \times m)\}$ and n is a block index number.

Step 3: Obtain 1LSB (b_i^1), 2LSB (b_i^2), and 3LSB (b_i^3) using Equation (9), where n is the index, and i is the pixel index. After that, OPAP is applied according to the rule of Equation (10), and $\mathcal{M}_1^n$ is hidden in the 3LSB of $\tilde{\mathcal{P}}$. That is, $\tilde{\mathcal{P}}'_n = \sum_{i=1}^{m \times m} f(\tilde{\mathcal{P}}_{n,i}, \mathcal{M}_1^{n,i})$, here, f is a function representing the logic of Equation (10).

$$\begin{cases} \tilde{\mathcal{P}}_{n,i} \xleftarrow[\text{LSB}]{\text{removed}} \lfloor \mathcal{P}_{n,i}/2 \rfloor \\ b_i^1 \xleftarrow{\text{1LSB}} \mathcal{P}_{n,i} \bmod 2 \\ b_i^2 \xleftarrow{\text{2LSB}} \tilde{\mathcal{P}}_{n,i} \bmod 2 \\ b_i^3 \xleftarrow{\text{3LSB}} \lfloor \tilde{\mathcal{P}}_{n,i}/2 \rfloor \bmod 2 \end{cases} \tag{9}$$

$$\tilde{\mathcal{P}}'_n = \begin{cases} \tilde{\mathcal{P}}_{n,i} - 1, & \text{if } (b_i^3 = 0 \text{ and } b_i^2 = 0) \text{ and } \mathcal{M}_1^{n,i} = 1, \\ \tilde{\mathcal{P}}_{n,i} + 1, & \text{if } (b_i^3 = 0 \text{ and } b_i^2 = 1) \text{ and } \mathcal{M}_1^{n,i} = 1, \\ \tilde{\mathcal{P}}_{n,i} - 1, & \text{if } (b_i^3 = 1 \text{ and } b_i^2 = 0) \text{ and } \mathcal{M}_1^{n,i} = 0, \\ \tilde{\mathcal{P}}_{n,i} + 1, & \text{if } (b_i^3 = 1 \text{ and } b_i^2 = 1) \text{ and } \mathcal{M}_1^{n,i} = 0, \\ \text{no change}, & \text{otherwise} \end{cases} \tag{10}$$

Step 4: Embed $\mathcal{M}_2^{n,i}$ into $(b_i^3 \oplus b_i^2)$ of $\tilde{\mathcal{P}}'_n$ using OPAP rule (Equation (11)), where f is the function represented the logic of Eqaution (11). Before applying OPAP, b_i^3 and b_i^2 need to be re-calculated using Equation (9). That is why the LSBs are updated values by using Equation (10).

$$\tilde{\mathcal{P}}''_n = \begin{cases} \tilde{\mathcal{P}}'_n(i) - 1, & \text{if } (b_i^3 = 1 \text{ and } b_i^2 = 1) \text{ and } \mathcal{M}_2^n(i) = 1, \\ \tilde{\mathcal{P}}'_n(i) + 1, & \text{if } (b_i^3 = 1 \text{ and } b_i^2 = 0) \text{ and } \mathcal{M}_2^n(i) = 0, \\ \tilde{\mathcal{P}}'_n(i) - 1, & \text{if } (b_i^3 = 0 \text{ and } b_i^2 = 1) \text{ and } \mathcal{M}_2^n(i) = 0, \\ \tilde{\mathcal{P}}'_n(i) + 1, & \text{if } (b_i^3 = 0 \text{ and } b_i^2 = 0) \text{ and } \mathcal{M}_2^n(i) = 1, \\ \text{no change}, & \text{otherwise} \end{cases} \tag{11}$$

In order to reflect the changed pixel block $\tilde{\mathcal{P}}''_n$ to $\mathcal{P}_n$, the following calculation must be applied. That is, $\mathcal{P}_n(i) = f(\tilde{\mathcal{P}}''_n(i) \times 2) + b_i^1$.

Algorithm 3 *Cont.*

Step 5: For image authentication, we compute a checksum for each block and hide it in a block. First, we choose a large number $\mathcal{G}$ that will be used for calculating the checksums (Equation (12)). For each block, every pixel ($\mathcal{V}_{n,i}$) is generated by a key (ξ) with a pseudo-random number. Here, $g(\mathcal{V}_{n,i})$ is the gray level of the pixel $\mathcal{V}_{n,i}$. We also generate $m \times m$ integers $c_1, c_2, \ldots, c_{m\times m}$ comparable in size to $\mathcal{G}$. The checksum K_{sum} is calculated as

$$\begin{cases} \mathcal{V}_{sum} = \sum_{i=1}^{m\times m} c_i g(\mathcal{V}_n(i)) \bmod \mathcal{G} \\ k_{i,t} = \left\lfloor \dfrac{\mathcal{V}_{sum}}{2^t} \right\rfloor \bmod 2, t = 0, 1, \ldots, m \times m. \end{cases} \tag{12}$$

Finally, the transformed bits k_i from checksum K_{sum} are acquired.

Step 6: Embed k_i into the LSBs of $\mathcal{P}_{n,i}$ using the logic of Equation (13). That is, $\mathcal{P}'_n = \sum_{i=1}^{m\times m} f(\mathcal{P}_{n,i}, k_{i,t})$, where f is the function representing the rule.

$$\mathcal{P}_n(i)' = \begin{cases} \text{no operation, if } k_i = b_i^1 \\ \mathcal{P}_n(i) + 1, \text{ if } (k_i \neq b_i^1) \text{ and } b_i^1 = 0, \\ \mathcal{P}_n(i) - 1, \text{ if } (k_i \neq b_i^1) \text{ and } b_i^1 = 1, \end{cases} \tag{13}$$

4.3. Watermark Extraction and Reconstructing AMBTC

In the method we proposed, the information (checksum) for its authentication is secretly concealed in the LSB, so it is possible to check whether the marked image is tampered with or forged even if there is no original image. The receiver side can find the tampered and forged block by using the validity of the checksum while moving each block. If a modulated block is found, the damaged block can be recovered according to the recovery procedure. Figure 3 shows a block diagram for the content recovery procedure.

Algorithm 4 Watermark Extracting

It extracts two maps, which are watermarks hidden in the marked image $\mathbb{K}$. The restoration process is performed using two maps and the checksum.

Input: A marked image $\mathbb{K}$ (output of Algorithm 3) with a size of $N \times N$, Key ξ.

Output: A reconstructed image $\mathbb{B}$ with a size of $N \times N$.

Step 1: Divide the marked image $\mathbb{K}$ into non-overlapping blocks sized $m \times m$, where $m = 4$. The sized $N \times N$ bitmap $\mathbb{M}_1$ and $\mathbb{M}_2$ are initialized with zeros.

Step 2: Read a block from the marked image $\mathbb{K}$, then this block is assigned to $\mathcal{P}_n$, where n is a block number. After that, the embedded hidden bits in b_i^1(LSB1), b_i^2(LSB2) and b_i^3(LSB3) are obtained from the block $\mathcal{P}_n$ using Equation (9). Then, a block ($\mathcal{M}_{n,i}$) of bitmap $\mathbb{M}_1$ is restored using Equation (14).

$$\mathcal{M}_{n,i} = \sum_{i=1}^{m\times m} \lfloor \tilde{\mathcal{P}}_{n,i}/2 \rfloor \bmod 2 \tag{14}$$

The restored bitmap block $\mathcal{M}_{n,i}$ is assigned to $\mathbb{M}_1$, i.e., $\mathbb{M}_1^n = \mathcal{M}_n$, where n is a block index.

Step 3: After applying Equation (15) to block $\mathcal{P}_n$, the obtained recovery block $\mathcal{M}_{n,i}$ is assigned to $\mathbb{M}_2$. That is, $\mathbb{M}_2^n = \mathcal{M}_n$.

$$\mathcal{M}_{n,i} = \sum_{i=1}^{m\times m} (\lfloor \tilde{\mathcal{P}}_{n,i}/2 \rfloor \bmod 2) \oplus (\tilde{\mathcal{P}}_{n,i} \bmod 2) \tag{15}$$

Algorithm 4 *Cont.*

Step 4: If the procedure of Steps 2 and 3 is repeated, the number of blocks ($(N \times N)/(m \times m)$), the two maps $\mathbb{M}_1$ and $\mathbb{M}_2$ are reconstructed.

Step 5: The mapped blocks $\mathbb{M}_1$ and $\mathbb{M}_2$ constructed by Equation (8) are reconstructed to have their original location. For this, first, divide the bitmap $\mathbb{M}_1$ and $\mathbb{M}_2$ into non-overlapping blocks of $m \times m$ pixels, where m = 64. Repeat Steps 5-1 and 5-2 until $\mathbb{M}_1$ and $\mathbb{M}_2$ are reconstructed.

Step 5-1: Read a block $\mathcal{M}_n$ from $\mathbb{M}_1$, where $n \in \{1 \leq n \leq (N \times N)/(m \times m)\}$. Obtain an index of two blocks needed to be exchanged applying $\mathbb{M}_1$ to Equation (8). That is, $j = \sum_{n=1} f(\mathcal{M}_n, \xi, n)$ where f is the function of the rule of Equation (8) and ξ is the key. Here, the indexes n and j are the indexes of the blocks to be exchanged. Swap the values of the block $\mathcal{M}_n$ and the block $\mathcal{M}_j$ in the map $\mathbb{M}_1$. When the blocks corresponding to the two positions are exchanged, the original positions are returned.

Step 5-2: Read a block $\mathcal{M}_n$ from $\mathbb{M}_2$, where $n \in \{1 \leq n \leq (N \times N)/(m \times m)\}$. Obtain an index to exchange two blocks applying Equation (8) to $\mathbb{M}_1$. That is, $j = \sum_{n=1} f(\mathcal{M}_n, \xi, n)$. Swap the values of the block $\mathcal{M}_n$ and the block $\mathcal{M}_j$ in the map $\mathbb{M}_2$.

Step 6: Divide the bitmap $\mathbb{M}_1$ and $\mathbb{M}_2$ into non-overlapping blocks of $m \times m$ pixels, where m = 4. The sized $N \times N$ AMBTC grayscale image $\mathbb{B}$ are initialized with zeros.

Step 6-1: Read a block $\mathcal{M}_n$ from $\mathbb{M}_2$, where $n \in \{1 \leq n \leq (N \times N)/(m \times m)\}$. The moment values ($Q_1$ and Q_2) are reconstructed from $\mathcal{M}_n$ using Equation (16), where $base = [2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0]^T$.

$$\begin{cases} Q_1^n = \sum_{i=1}^{m \times m/2} (base \cdot \mathcal{M}_{n,i}) \\ Q_2^n = \sum_{i=9}^{m \times m/2} (base \cdot \mathcal{M}_{n,i}) \end{cases} \tag{16}$$

Step 6-2: Read a block $\mathcal{M}_n$ from $\mathbb{M}_1$. Equation (5) is applied to replace a bitmap block $\mathcal{M}_n$ with a grayscale block G_n. That is, $G_{n,i} = \sum_{i=1}^{m \times m} f(\mathcal{M}_n, Q_1, Q_2)$, where f is a function logic of Equation (5) and n is block number. A grayscale block G coding obtained by the decoding is assigned to $\mathbb{B}$, i.e., $\mathbb{B}(n) = G_n$.

Step 7: The image derived from AMBTC is reconstructed as repeating the procedure of Step 6 as much as the number of blocks.

Until now, we explained the restoration of a grayscale image based on AMBTC through extracting watermarks(maps) from the marked image. Next, we will explain how to restore the tampered block after finding the tampered block from the marked image.

Algorithm 5 Watermark Authentication, Tamper Detection and Reconstruct Cover Image

This describes the extracting checksum from the marked image and the restoration procedure of the tampered block using the checksum and the recovered trio.

Input: A marked image $\mathbb{K}$ and a grayscale image $\mathbb{B}$ (output of Algorithm 4) based on AMBTC with a size of $N \times N$, Key ξ.

Output: A reconstructed cover image $\mathbb{R}$ with a size of $N \times N$.

Step 1: Divide the images $\mathbb{K}$ and $\mathbb{B}$ into non-overlapping blocks sized $m \times m$, where $m = 4$. The sized $N \times N$ image $\mathbb{R}$ are initialized with zeros.

Step 2: Read a block of the images $\mathbb{K}$ and $\mathbb{B}$, then this block is assigned to $\mathcal{P}_n$ and $\mathcal{B}_n$, where n is a block number. $\mathcal{P}_{sum}$ embedded in the LSB of the block $\mathcal{P}_n$ is recovered by using Equation (17).

$$\mathcal{P}_{sum} = \sum_{i=1}^{m \times m} (\mathcal{P}_{n,i} \bmod 2) \times 2^i \tag{17}$$

$\mathcal{P}_{sum}$ is an embedded checksum in the block $\mathcal{P}_n$.

Algorithm 5 *Cont.*

Step 3: Generate checksum $\mathcal{V}_{sum}$ using Equation (12) and then discriminate whether the block has been tampered with or not using Equation (18). That is, if $\mathcal{V}_{sum} = \mathcal{P}_{sum}$, this block is a safe block; otherwise, it is a tampered block. If the block is safe, $\mathcal{P}_n$ is assigned to $\mathbb{R}_n$. Meanwhile, if it is tampered, the recovered block $\mathcal{B}_n$ is assigned to $\mathbb{R}_n$.

$$\mathbb{R}_n = \begin{cases} \mathcal{P}_n, & if \ (\mathcal{V}_{sum} = \mathcal{P}_{sum}), \\ \mathcal{B}_n, & otherwise, \end{cases} \tag{18}$$

Step 4: The recovered image $\mathbb{R}$ is made by repeating the procedure of Steps 2 and Step 3 as much as the number of blocks.

5. Experimental Results

In this section, the experiments and analysis are described to prove the performance of the proposed method. The computing platform used in the experiment has a Core i5-8250U processor, 1.60 GHz speed and 8 GB of RAM, and the software for the simulation is MATLAB R2019b. The standard USC-SIPI image database was used in the experiment for image restoration. Of these, some of the original 512×512 grayscale images were selected and used for the experiment. Figure 4 shows a set of test images (e.g., Lena, Pepper, Airplane, Boat, Goldhill, Couple, Baboon, and Zelda) used in the experiment.

For evaluation, Structural Similarity Index Metric (SSIM) and peak signal-to-noise ratio (PSNR) were used to compare the performance of the existing and proposed methods. The quality of the image was measured by the PSNR defined as

$$\text{PSNR} = 10log_{10}\frac{255^2}{\text{MSE}} \tag{19}$$

PSNR is calculated as $10log$ (signal power/noise power), and signal power and noise power are calculated using peak power. The MSE used for PSNR calculation is the difference in average intensity between the marked image and the reference image, and a low MSE value can be evaluated as good image quality. In other words, the MSE is the mean of the squares of the errors $(p_i - p_i')^2$, where p and p' are reference and distorted images, respectively. The MSE is calculated as follows:

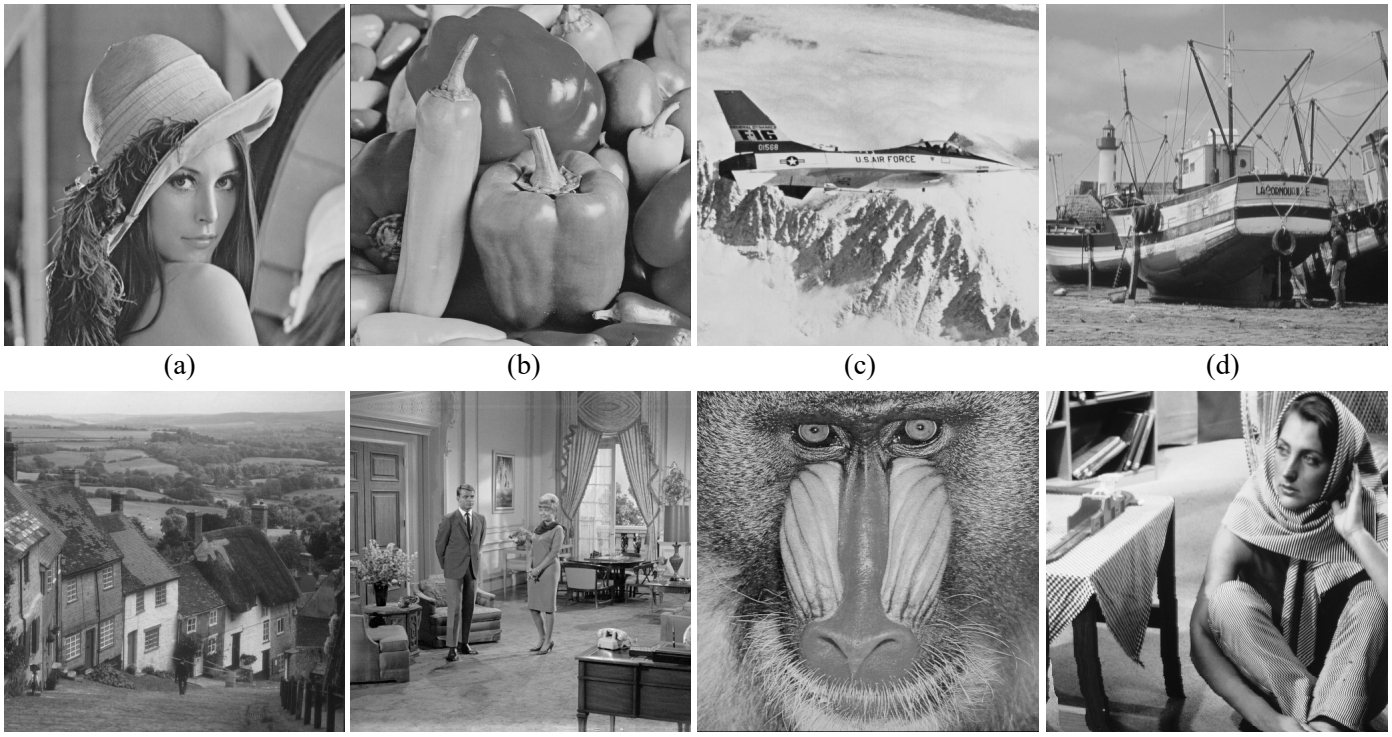
$$\text{MSE}(p, p') = \frac{1}{N}\sum_{i=1}^{N}(p_i - p_i')^2. \tag{20}$$

Here, the allowable pixel intensity is 255^2.

SSIM is a formula that measures the similarity between the original image and the displayed image. It consists of luminance, contrast and structure, and it measures the quality of an image. The range of the SSIM value is limited between 0 and 1, and if the value is close to 1, the image is similar to the cover image. The computation of SSIM is as follows:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2\mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)'} \tag{21}$$

where μ_x and μ_y denote values of cover image x and the marked image y, σ_x and σ_y are standard deviation values of the cover image and the marked image, while $\sigma_{x,y}$ denotes the covariance of both two images. c_1 and c_2 are constants to stabilize the the division.



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

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

Figure 4. Test images used in our experiments: (**a**) Lena, (**b**) Pepper, (**c**) Airplane, (**d**) Boat, (**e**) Goldhill, (**f**) Couple, (**g**) Baboon and (**h**) Zelda.

Figure 5 shows a cropping attack on the marked image with limited ratios and the results of recovering the tampered images using the proposed method. The ratio was limited to 10% to 45%. For the cropping attack, the visual difference between the original image and the restored images as applying the method proposed in Section 4 (see Figure 5b) was very similar. Objective evaluations such as PSNR and SSIM of Figure 5 can be found in Tables 1 and 2.

The merit of our proposed method is that it manages PSNR (Table 1) and SSIM (Table 2) about marked images and recovered images reasonably. In the case of Lena image, when the ratio of cropping attacks is 5% and 45%, the difference between the two PSNRs is only 3.4573 dB. While, in the case of the Barbara image, the difference was highest among the comparison PSNRs in Table 1, i.e., it was 6.3909 dB.

In Table 2, the reason for introducing SSIM to measure image quality is that SSIM was high in the case of Baboon images with low PSNR (in Table 1), and subjective evaluation of image quality like the human visual system is possible, unlike PSNR measurement results. Overall, it means that SSIM was more accurate than PSNR in subjective terms. Therefore, two measurements are required for complementarity. As a result, it showed very good performance compared to other existing methods.

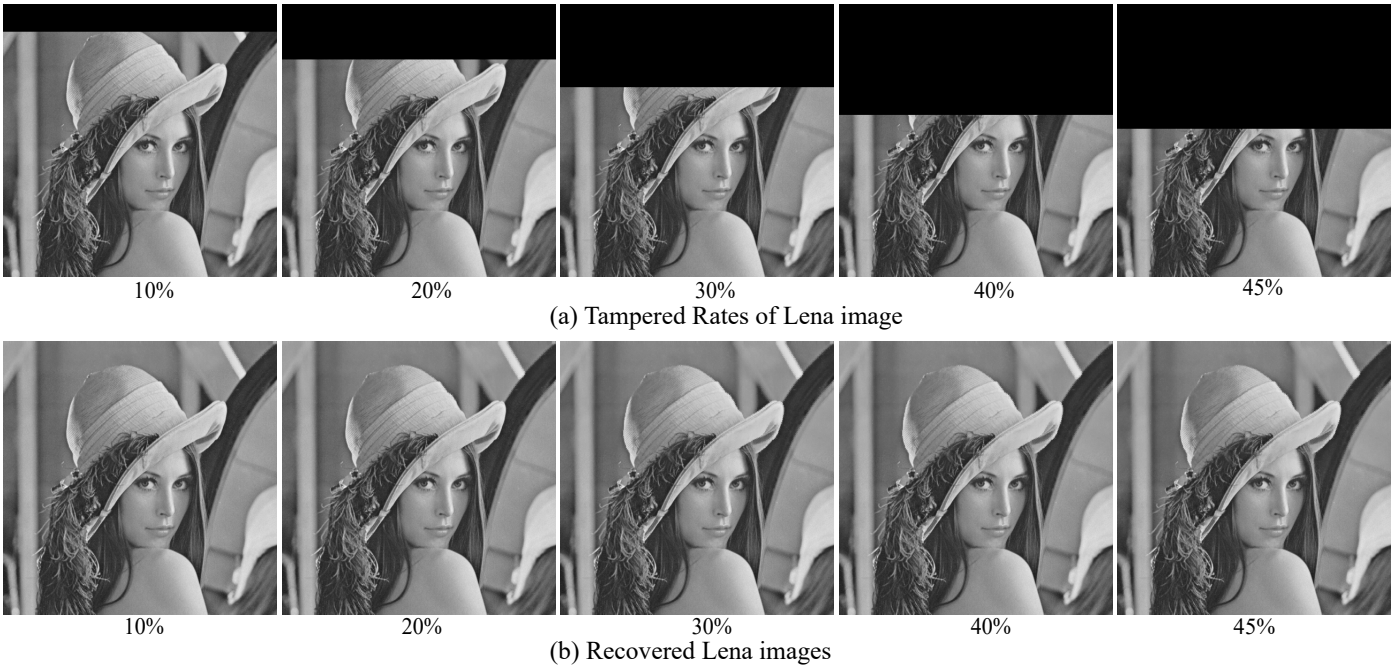


Figure 5. Cropping attack according to various ratios on Lena images and reconstructed Lena images applying the method we proposed.

Table 1. PSNR comparisons between original images and recovered images according to tampered rates.

Cover Image	Tampering Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Lena	40.0894	40.0741	39.8066	39.2411	38.7654	38.2036	37.6783	36.754	36.6321
Pepper	39.1559	38.8223	38.258	37.8034	37.4989	37.0547	36.7648	36.1718	36.0495
Airplane	39.9916	40.0421	39.8804	39.0067	39.087	38.9265	39.1673	35.7464	35.7072
Boat	40.1693	39.9564	39.6507	38.9742	38.4153	37.7102	37.5841	35.8619	35.7968
Goldhill	39.7814	39.6703	39.5971	38.0752	37.8333	37.5566	37.4104	36.2497	35.8103
Couple	38.6335	37.9082	36.0183	34.9058	33.5838	33.1666	32.4731	32.1509	31.7149
Baboon	34.9313	33.6695	32.1257	31.5393	31.0723	30.4577	29.6961	29.551	28.6731
Zelda	40.0645	39.7914	39.0381	38.847	38.4935	37.7322	38.0708	36.3148	37.4341
Barbara	37.8367	37.0637	35.9489	35.8704	35.4713	34.1328	33.2672	31.9851	31.4458
Average	38.96151	38.55533	37.81376	37.14034	36.6912	36.10454	35.79023	34.53173	34.36264

Table 2. SSIM comparisons between original images and recovered images according to tampered rates.

Cover Image	Tampering Rate								
	5%	10%	15%	20%	25%	30%	35%	40%	45%
Lena	0.9508	0.9517	0.9534	0.9506	0.951	0.9516	0.9521	0.9495	0.9507
Pepper	0.9504	0.9505	0.9504	0.9495	0.9497	0.9484	0.948	0.9458	0.9463
Airplane	0.9477	0.9492	0.9509	0.953	0.955	0.9573	0.9599	0.9546	0.9571
Boat	0.9582	0.9601	0.9625	0.9638	0.9644	0.9647	0.9668	0.9608	0.9632
Goldhill	0.9665	0.9668	0.9674	0.9604	0.9593	0.958	0.9572	0.9502	0.9502
Couple	0.9664	0.9658	0.9604	0.9609	0.9552	0.9513	0.9465	0.9441	0.94
Baboon	0.9776	0.9732	0.9655	0.9615	0.9572	0.9516	0.9454	0.9421	0.9359
Zelda	0.9494	0.9486	0.947	0.9462	0.9446	0.943	0.9443	0.9398	0.9421
Barbara	0.969	0.9686	0.967	0.9689	0.9682	0.9644	0.9618	0.9585	0.9565
Average	0.959556	0.959389	0.958278	0.9572	0.956067	0.954478	0.953556	0.949489	0.949111

Figure 6 shows the quality of the restored image as PSNR when the cropping attack ratio was applied from 5% to 45% for the marked image. As the attack rate increased, PSNR

decreased, showing a downward trend. However, the line was relatively smooth. In the case of the Lena image, the maximum performance was 40 dB or more at 5% and 36 dB or more at 45%.

In the case of the Pepper image, it showed about 39 dB at 5% and more than 36 dB at 45%, showing the lowest performance. The texture of the Pepper image has a smoother characteristic than that of the Lena image, and the surface of the cover image is very bright. Such features seem to degrade the quality of the recovered image during the reconstruction process using AMBTC. That can be a minor weakness of the method we have proposed.

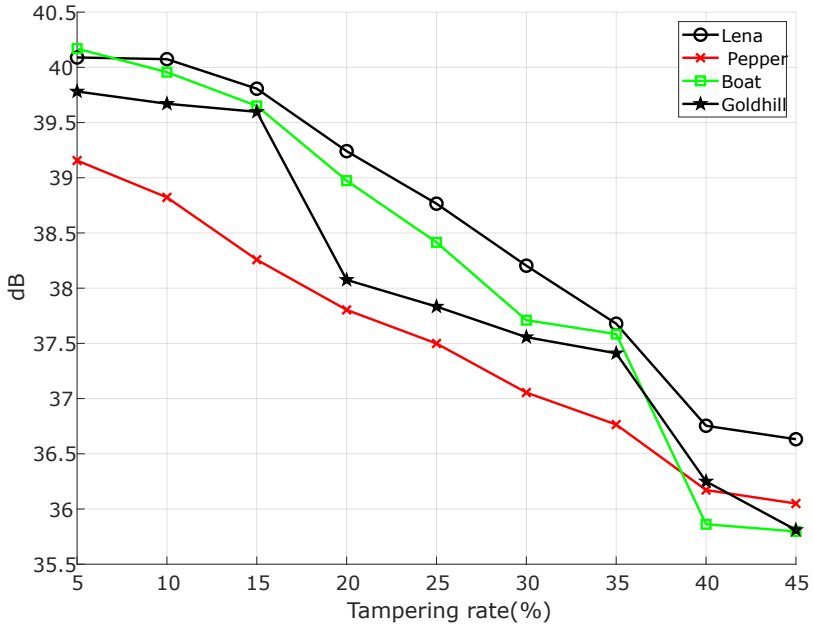


Figure 6. Tampered images at various tampering rates (%).

Since the recovered image using the proposed method used AMBTC derived from the BTC, this staircase effect could be reduced to some extent. The staircase effect may tend to appear larger in the brighter parts of the image. As a result, as the tampering rate increased, the staircase effect on the nose of Barbara's image was slightly revealed. Overall, the quality of the image restored with the method we proposed was excellent.

Figure 7 compares the performance of the existing self-embedding methods (i.e., Zhang et al. [17], Luo et al. [25], Yang & Shen [27], Hemida & He [30]) with our proposed method. The Tampering Rate (TR) for the marked cover-image (Lena) ranged from 5% to 45%. Both methods proposed by Hemida & He [30] and Luo et al. [25] measured about 35dB when the TR was 5%, and there was a slight difference in the performance of the two methods until the TR reached about 20%. However, the PSNR of the two showed similar reduction, and when TR was 20%, it was about 30 dB.

Then, the PSNR of Luo et al. [25] decreased to a smooth descending curve and was about 24 dB when TR = 45%, while the method proposed by Hemida & He [30] drastically decreased to about 11 dB when TR = 45%. One of the reasons Hemida& He's method [30] did not perform well is because it uses simple binary operations to detect the tampering area. This seems to be due to the threshold error according to the use of binary operations. Luo et al. [25] used the 7MSB binary value of each block for image authentication. Although Luo et al.'s authentication method performed better than Hemida & He [30], authentication failures may occur due to errors caused using binary numbers and may affect performance.

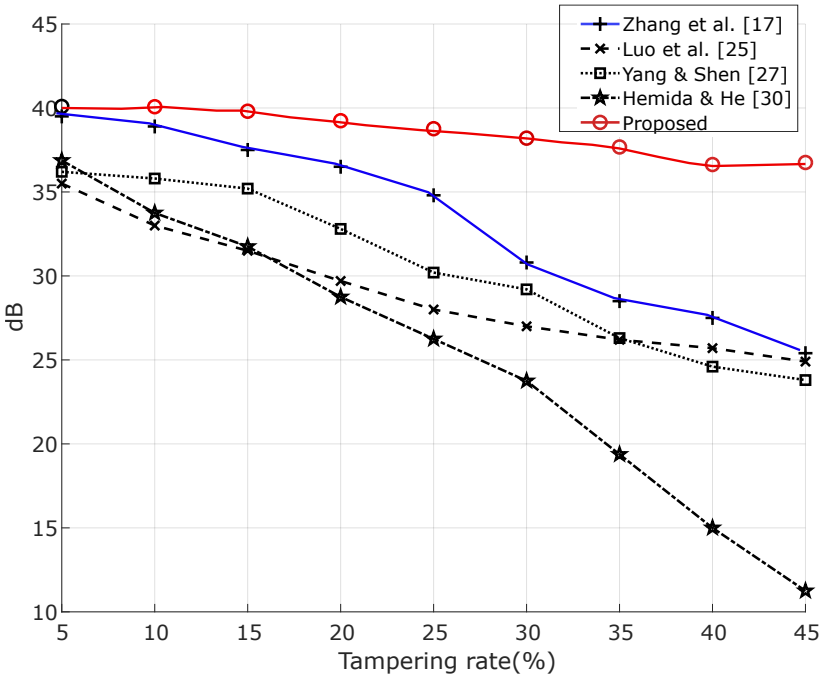


Figure 7. Comparison of the performance between previous methods and our proposed method.

Using halftones for image restoration is important to investigate. The image quality obtained by restoration using halftones was around 30 dB, which is a bit insufficient to ensure high image quality. The reason is that in the process of converting halftone to a grayscale image, it is visually observed that the quality of the image is clearly different from the texture of the original image. This means that the halftone could not be enough for the recovery bits. The PSNR of Yang & Shen [27] was limited from about 35 dB (highest) to about 24 dB (lowest), but it showed a relatively stable PSNR performance. For tampering authentication, they used Wong's watermarking technique [28], and its plan had a good impact on performance.

When the TR was 5%, the PSNR of Zhang et al.'s method [17] and the proposed method were shown at about 39 dB and about 40 dB, respectively. The proposed method decreased slowly until TR = 45%, and when TR = 45%, PSNR was 36 dB. On the other hand, the PSNR of Zhang et al. [17] was about 25 dB (TR = 45%), which appeared as a slightly steeper curve than ours. However, the advantage of this method is that it is designed to prevent tampering coincidence problems, which has a positive effect on performance and shows superior performance among existing methods. Nevertheless, the proposed method shows good performance among self-embedding methods.

The factors for improving the performance of the proposed method are as follows: first, the recovery bits for a specific block were stored in a long block located as far away as possible. Second, there was correct detection of tampered blocks. Since our proposed method was faithful in this perspective, we find that it had a positive effect on performance.

Table 3 shows a comparison of the PSNR of the marked image and the restored image obtained after applying various self-embedding watermarking methods to the Lena image. Looking at the PSNR of the reconstructed image in Table 3, we show that the proposed method was superior to the previous methods. In the case of He et al.'s method [19], some blocks at the boundary of the tampered region were erroneously identified. Zhang et al.'s method [23] showed that the quality of the recovered image was high when the tampered domain was less than 35% of the entire image. Qian et al.'s method did the authentication bit and reference bit in the three LSB layers of the image. Therefore, the restored image was excellent under limited conditions. Yang & Shen [27] produced an index table of the original image through vector quantization (VQ), and the obtained data were hidden in the cover image for image restoration. If the VQ is lost with a tamper attack, the restoration

of the lost VQ area is impossible. In Kim et al.'s method [29], when the image damaged area was less than 50%, the quality of the reconstructed image was high (33.6). This was improved through image filtering after image restoration. In conclusion, our proposed method had the best restored image quality, but the quality of marked images was not the best. This is because up to 3LSB was used for restoration performance.

Table 3. Comparisons of PSNR of marked image and recovered image among different schemes.

Methods	Marked Images (PSNR)	Recovered Images (PSNR)	Criteria of Restoration
He et al. [19]	51.1	32.2	Tampered areas must be reserved
Zhang et al. [23]	37.9	29.9	<59%
Qian et al. [24]	37.9	35.0	<35%
Yang and Shen [27]	40.7	32.0	<50%
Kim et al. [29]	43.7	33.6	<50%
The proposed method	40.0	36.6	<45%

Table 4 shows the PSNR and NCC (Normalized Cross-Correlation) of the watermarked images, and it shows their embedding times (seconds). The PSNRs of the marked images were at levels difficult to discriminate with the human visual system. Therefore, the marked image made by the proposed method was very good in the aspect of the images' quality. In addition, the time performance measured with MATLAB was not bad, but the reason why the performance was not high is due to the performance of MATLAB, and it seems that there will be no problem in terms of time when developing in C language.

Table 4. Measuring PSNR, SSIM, NCC and time (second) of marked images based on the proposed method.

Cover Image	PSNR (dB)	SSIM	NCC	Embedding Time (s)
Lena	40.0076	0.9488	0.9996	1.0396
Pepper	40.0121	0.9516	0.9997	0.9516
Airplane	40.0306	0.9469	0.9996	0.8567
Boat	40.0093	0.9549	0.9996	0.9976
Goldhill	39.9983	0.9662	0.9995	1.2315
Couple	40.0196	0.9691	0.9996	1.2363
Baboon	40.0095	0.9841	0.9996	1.2099
Zelda	39.9805	0.9480	0.9994	0.8555

6. Conclusions

In this paper, we present a productive, fragile, self-embedding watermarking method based on AMBTC. Here, we concealed the recovery bits in LSB2 and LSB3 and the checksum bits in LSB in block units using the OPAP method. In addition, a checksum was introduced for accurate block authentication. In the existing method, binary bits were used for authentication, but there was a lack of precision, so checksum was used. Since the proposed method is a fragile watermarking method, the watermark information may be destroyed by image processing such as compression and filtering. However, in case of a partial cropping attack, it is possible to restore the tampered area to the level of the marked area before the forgery by using the hidden restoration information. The limitation of our proposed method is that if the damaged area of the image is large, the restoration information is also removed, so that an area that cannot be restored may occur. In the future we would like to find a way to solve the problem of corruption of recovery information that occurs when the size of the damaged image area is more than 50%.

Author Contributions: Conceptualization, C.-N.Y.; writing—original draft preparation, C.K.; Writing—review & editing, C.K.; Validation, C.-N.Y., and C.K.; formal analysis, C.-N.Y.; methodology, C.K.; data

curation, C.K.; funding acquisition, C.-N.Y., C.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Ministry of Science and Technology (MOST), under Grant 108-2221-E-259-009-MY2 and 109-2221-E-259-010, and by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by (2015R1D1A1A01059253), and it was supported under the framework of international cooperation program managed by NRF (2016K2A9A2A05005255). This work was supported by the faculty research fund of Sejong University in 2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Thank you to the reviewers who reviewed this paper and the MDPI editor who edited it professionally.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

$\mathbb{M}_1$	first bitmap image
$\mathbb{M}_2$	second bitmap image
$\mathcal{V}_{sum}$	generated checksum
$\mathbb{C}$	cover image
$\mathbb{M}$	watermark (two maps of AMBTC)
$\mathbb{O}$	original image
$\mathcal{P}_n$	storing $m \times m$ pixel in a block
$\tilde{\mathcal{P}}_{n,i}$	a block which is removed LSB from $\mathcal{P}_n$
$trio(Q_1, Q_2, \mathcal{M})$	Q is a quantization level and $\mathcal{M}$ is a bitmap block
N	the size of original image
m	the size of a block
ξ	Key for block mapping in Algorithm 2
G_n	a grascale block
BTC	Block Truncation Coding
AMBTC	Absolute Moment Block Truncation Coding
OPAP	Optimal Pixel Adjustment Process
DH	Data Hiding
XOR	Exclusive-OR
CR	Compression Ratio
PSNR	Peak Signal-to-Noise Ratio
MSE	Mean Squared Error
WMSE	Worst-case Mean Squared Error
LSB	Least-Significant-Bit
SSIM	Structural Similarity Index Metric

References

1. Lou, D.-C.; Liu, J.-L. Fault resilient and compression tolerant digital signature for image authentication. *IEEE Trans. Consum. Electron.* **2000**, *46*, 31–39.
2. Umamageswari, A.; Suresh, G.R. Secure medical image communication using ROI based lossless watermarking and novel digital signature. *J. Eng. Res.* **2014**, *2*, 87–108. [CrossRef]
3. Tsai, P.; Hu, Y.; Chang, C. Novel image authentication scheme based on quadtree segmentation. *Imaging Sci. J.* **2005**, *53*, 14–162. [CrossRef]
4. Ababneh, S.; Ansari, R.; Khokhar, A. Iterative compensation schemes for multimedia content authentication. *J. Vis. Commun. Image Represent.* **2009**, *20*, 303–311. [CrossRef]
5. Walton, S. Image authentication for a slippery new age. *Dr. Dobb's J.* **1995**, *20*, 18–26.
6. Mishra, A.; Agarwal, C.; Sharma, A.; Bedi, P. Optimized gray-scale image watermarking using DWT-SVD and firefly algorithm. *Expert Syst. Appl.* **2014**, *41*, 7858–7867. [CrossRef]

7. Parah, S.A.; Sheikh, J.A.; Loan, N.A.; Bhat, G.M. Robust and blind watermarking technique in DCT domain using inter-block coefficient differencing. *Digit. Signal Process.* **2016**, *53*, 11–24. [CrossRef]
8. Di, Y.-F.; Lee, C.-F.; Wang, Z.-H.; Chang, C.-C.; Li, J. A robust and removable watermarking scheme using singular value decomposition. *KSII Trans. Internet Inf. Syst.* **2016**, *12*, 5268–5285.
9. Zear, A.; Singh, A.K.; Kumar, P. A proposed secure multiple watermarking technique based on dwt, DCT and SVD for application in medicine. *Multimed. Tools Appl.* **2018**, *77*, 4863–4882. [CrossRef]
10. Preda, R.O. Semi-fragile watermarking for image authentication with sensitive tamper localization in the wavelet domain. *Measurement* **2013**, *46*, 367–373. [CrossRef]
11. Al-Otum, H.M. Semi-fragile watermarking for grayscale image authentication and tamper detection based on an adjusted expanded-bit multiscale quantization-based technique. *J. Vis. Commun. Image Represent.* **2014**, *25*, 1064–1081. [CrossRef]
12. Qi, X.J.; Xin, X. A singular-value-based semi-fragile watermarking scheme for image content authentication with tamper localization. *J. Vis. Commun. Image Represent.* **2015**, *30*, 312–327. [CrossRef]
13. Tong, X.J.; Liu, Y.; Zhang, M.; Chen, Y. A novel chaos-based fragile watermarking for image tampering detection and self-recovery. *Signal Process. Image Commun.* **2013**, *28*, 301–308. [CrossRef]
14. Chen, F.; He, H.J.; Tai, H.M.; Wang, H.X. Chaos-based self-embedding fragile watermarking with flexible watermark payload. *Multimed. Tools Appl.* **2014**, *72*, 41–56. [CrossRef]
15. Ansari, I.A.; Pant, M.; Ahn, C.W. SVD based fragile watermarking scheme for tamper localization and self-recovery. *Int. J. Mach. Learn. Cybern.* **2016**, *7*, 1225–1239. [CrossRef]
16. Singh, D.; Singh, S.K. Effective self-embedding watermarking scheme for image tampered detection and localization with recovery capability. *J. Vis. Commun. Image Represent.* **2016**, *38*, 775–789. [CrossRef]
17. Zhang, X.; Wang, S.; Qian, Z.; Feng, G. Reference Sharing Mechanism for Watermark Self-Embedding. *IEEE Trans. Image Process.* **2011**, *20*, 485–495. [CrossRef]
18. Fridrich, J.; Goljan, M. Images with self-correcting capabilities. In *Proceedings of International Conference on Image Processing (ICIP)*; IEEE: Korbe, Japan, 1999; pp. 792–796.
19. He, H.; Zhang, J.; Chen, F. Adjacent-block based statistical detection method for self-embedding watermarking techniques. *Signal Process.* **2009**, *89*, 1557–1566. [CrossRef]
20. Lin, P.L.; Hsieh, C.K.; Huang, P.W. A hierarchical digital watermarking method for image tamper detection and recovery. *Pattern Recogn.* **2005**, *38*, 2519–2529. [CrossRef]
21. Lee, T.-Y.; Lin, S.D. Dual watermark for image tamper detection and recovery. *Pattern Recogn.* **2008**, *41*, 3497–3506. [CrossRef]
22. Zhang, X.; Wang, S. Fragile watermarking with error-free restoration capability. *IEEE Trans. Multimed.* **2008**, *10*, 1490–1499 [CrossRef]
23. Zhang, X.; Wang, S.; Feng, G. Fragile Watermarking Scheme with Extensive Content Restoration Capability. In Proceedings of the IWDW 2009, Guildford, UK, 24–26 August 2009; Volume 5703.
24. Qian, Z.; Feng, G.; Zhang, X.; Wang, S. Image self-embedding with high-quality restoration capability. *Digit. Signal Process.* **2011**, *21*, 278–286. [CrossRef]
25. Luo, H.; Chu, S.-C.; Lu, Z.-M. Self Embedding Watermarking Using Halftoning Technique. *Circuits Syst. Signal Process.* **2008**, *27*, 155–170. [CrossRef]
26. Hsu, C.-S.; Tu, S.-F. Image tamper detection and recovery using adaptive embedding rules. *Measurement* **2016**, *88*, 287–296. [CrossRef]
27. Yang, C.-W.; Shen, J.-J. Recover the tampered image based on VQ indexing. *Signal. Process.* **2010**, *90*, 331–343. [CrossRef]
28. Wong, P.W.; Memon, N. Secret and public key image watermarking schemes for image authentication and ownership verification. *IEEE Trans. Image Process.* **2001**, *10*, 1593–1601. [CrossRef]
29. Kim, C.; Shin, D.; Yang, C.-N. Self-embedding fragile watermarking scheme to restoration of a tampered image using AMBTC. *Pers. Ubiquit. Comput.* **2018**, *22*, 11–22. [CrossRef]
30. Hemida, O.; He, H. A self-recovery watermarking scheme based on block truncation coding and quantum chaos map. *Multimed. Tools Appl.* **2020**, *79*, 18695–18725. [CrossRef]
31. Chang, C.; Lin, C.; Su, G. An effective image self-recovery based fragile watermarking using self-adaptive weight-based compressed AMBTC. *Multimed. Tools Appl.* **2020**, *79*, 24795–24824. [CrossRef]
32. Delp, E.; Mitchell, O. Image compression using block truncation coding. *IEEE Trans. Commun.* **1979**, *27*, 1335–1342. [CrossRef]
33. Lema, M.D.; Mitchell, O.R. Absolute moment block truncation coding and its application to color images. *IEEE Trans. Commun.* **1984**, *COM-32*, 1148–1157. [CrossRef]
34. Wang, R.Z.; Lin, C.F.; Lin, J.C. Hiding data in images by optimal moderately significant-bit replacement. *IEE Electron. Lett.* **2000**, *36*, 2069–2070. [CrossRef]
35. Hsu, L.Y.; Hu, H.T. Blind watermarking for color images using EMMQ based on QDFT. *Expert Syst. Appl.* **2020**, *149*, 1–16. [CrossRef]
36. Hammami, A.; Hamida, A.B.; Amar, C.B. Blind semi-fragile watermarking scheme for video authentication in video surveillance context. *Multimed. Tools Appl.* **2020**. [CrossRef]
37. Hamidi, M.; Chetouani, A.; Haziti, M.E.; Hassouni, M.E.; Cherifi, H. Blind robust 3D mesh watermarking based on mesh saliency and wavelet transform for copyright protection. *Information* **2019**, *10*, 67. [CrossRef]

38. Sadek, R.A. SVD based image processing applications: State of the art, contributions and research challenges. *Int. J. Adv. Comput. Sci.* **2012**, *3*, 26–34
39. Joshi, A.M.; Gupta, S.; Girdhar, M.; Agarwal, P.; Sarker, R. Combined DWT-DCT-based video watermarking algorithm using arnold transform technique. In *Proceedings of the International Conference on Data Engineering and Communication Technology*; Springer: Singapore, 2017; pp. 455–463.