symmetry

MDPI

Article

High-Precision Authentication Scheme Based on Matrix Encoding for AMBTC-Compressed Images

Guo-Dong Su [1,2,3], Chin-Chen Chang [2] and Chia-Chen Lin [4,*]

1 The School of Electronic and Information Engineering, Fuqing Branch of Fujian Normal University, Fuzhou 350300, China
2 Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan
3 Engineering Research Center for ICH Digitalization and Multi-Source Information Fusion, Fujian University, Fuzhou 350300, China
4 Department of Computer Science and Information Management, Providence University, Taichung 433, Taiwan
* Correspondence: mhlin3@pu.edu.tw

Received: 2 July 2019; Accepted: 28 July 2019; Published: 3 August 2019

Abstract: In this paper, a high-precision image authentication scheme for absolute moment block truncation coding (AMBTC)-compressed images is presented. For each block, two sub-bitmaps are conducted using the symmetrical separation, and the six-bit authentication code is symmetrically assigned to two sub-codes, which is virtually embedded into sub-bitmaps using the matrix encoding later. To overcome distortion caused by modifications to the bitmap, the corresponding to-be-flipped bit-location information is recorded instead of flipping these bits of the bitmap directly. Then, the bit-location information is inserted into quantization levels based on adjusted quantization level matching. In contrast to previous studies, the proposed scheme offers a significantly improved tampering detection ability, especially in the first hierarchical tampering detection without remediation measures, with an average tampering detection rate of up to 98.55%. Experimental results show that our approach provides a more stable and reliable tampering detection performance and sustains an acceptable visual quality.

Keywords: image authentication; AMBTC; matrix encoding; adjusted quantization levels matching

1. Introduction

Recently, as engineering technology has rapidly developed, the performance of computers has become increasingly stronger in terms of computing ability, storage capacity, etc. At the same time, the higher transmission capacity offered by wired/wireless networks allows users to share data anywhere, anytime. Obviously, digital images can be conveniently and transparently transmitted via the Internet; however, the Internet cannot always promise reliable and secure transmission, since it is openly accessible. In other words, it is easy for an intruder to intercept data transmitted over the Internet and then corrupt them, intentionally or non-intentionally. For example, attackers can insert vulgar words into a digital image in an imperceptible manner. Such malicious behavior presents a huge challenge to the security, usability, and integrity of personal information. Therefore, it is urgent to protect the integrity and verifiability of the digital image. Under this scenario, it is expected that a technique is provided to solve this problem.

Researchers have conducted a series of scientific studies on image authentication. Roughly, authentication methods can be classified into two categories: Digital signature-based methods [1–4] and digital watermark-based methods [5–31]. Generally, the digital signature-based method performs

the encryption of the hashed results of the features of the image using a private key to form a unique signature, which will be used later for authentication. The process of authentication can be implemented by comparing the hashing result of the image under question and the original hashed version, which can be decrypted from the signature using an associated public key. By this way, the digital signature-based method performs well on image authentication because it is extremely sensitive to any kind of modification to the image, even if only one bit has been modified. Sometimes, the digital image may be allowed a little distortion in some applications, as long as the tamper regions can be localized precisely. Unfortunately, the digital signature-based method is not workable under this scenario.

The digital watermark-based method imperceptibly embeds relevant or irrelevant information called the authentication code (AC) into a digital image. The digital watermark-based method includes three categories: Fragile, semi-fragile, and robust watermarking. Of these, fragile watermarking embeds the authentication code into the cover image and makes it very sensitive to the modification of the image so that it can be used to verify its authenticity. In the authentication phase, receiver(s) can judge whether the image has been tampered with by comparing the extracted and recalculated authentication code. If they are the same, the received image has not been tampered with, and vice versa.

In recent decades, several forms of fragile watermarking-based image authentication have been proposed. An early fragile watermarking-based method was proposed by [5], where the authentication code is generated from the parity check of the pixel value and embedded into the least significant bit (LSB) of the original image. A possibility of false judgement exists since the tampered 1 LSB could be the same as the computed parity check value of the tampered 7 most significant bits (MSBs). Other fragile watermarking methods based on cryptographic theory were presented by [6–8]. Here, authentication codes are derived from a hashing function with various inputs, such as the image content, image index, block index, and pseudo-random number. These methods offer acceptable tampering detection performance. However, some of them could not withstand a vector quantization attack [9] or the tampering coincidence problem [10]. To overcome those problems, other fragile watermarking strategy-based block mappings were proposed by [11–16]. In these schemes, the original image is divided into non-overlapping blocks and the authentication code is generated by employing different kinds of technologies for each block, including the discrete cosine transform (DCT)-based method [11,12], the singular value decomposition (SVD)-based method [13,14], and the coding-based method [15,16]. Then, the authentication code of block is scrambled, mapped to the other block, and then inserted into it. The block mapping is one-to-one. Verification is conducted by comparing the extracted and recalculated authentication code. These schemes [11–15] also adopt a multi-hierarchical tampering detection strategy to improve the tampering detection rate; for example, a first hierarchical tampering detection strategy is used to initially identify the tampered area and a second hierarchical tampering detection strategy serves as a remediation measure. As a consequence, these approaches have high precision in tampering detection. Hence, their schemes make it more probable that the tampering region can be restored with respect to satisfactory recovered image quality.

Image authentication technology is also widely used in the compression domain. The aim of image compression algorithms, such as joint photographic experts group (JPEG) [17,18], vector quantization (VQ) encoding [19,32], and absolute moment block truncation coding (AMBTC) [33], is to reduce the size of an image to alleviate the burden of data communication. Of these, AMBTC is a variation of block truncation coding (BTC) [34]. Considering the BTC family is simple and less computationally complex while AMBTC offers better image quality than BTC, many scholars have proposed image authentication schemes for either BTC- or AMBTC-compressed images in the last decade. In 2004, Tu and Hsu [20] proposed a copyright protection scheme for digital images based on BTC. The authentication code, called the ownership share, is constructed by combining the determined authentication code with the binary image generated from the permuted host image using BTC. It is stored by a trusted third party for future authentication. On average, this scheme can extract the authentication code at around

92.13%. In 2009, Jiang et al. [21] proposed a fragile watermarking method that inserts the authentication code into the host image according to the parity of the reconstruction levels of the BTC quantizer. In 2011, Yang and Lu [22] proposed an image authentication method using BTC. Their authentication code is embedded into the block according to the odevity of the number of '1's in the bitmap. If the authentication code bit is '1', the number of '1's in the bitmap is made odd by changing, at most, the three-pixel value of the block. If the authentication code bit is '0', the number of '0's in the bitmap is modified following a similar rule. In the tampering detection phase, the authentication code can be extracted according to the odevity of the number of '1's in each block. In 2013, Hu et al. [23] proposed a fragile watermarking method based on AMBTC. In their approach, the authentication code is derived using a pseudo-random generator. For each block, the corresponding bitmap is further divided into k sub-bitmaps with the same size. Then, based on the idea of bit-flipping in [35], each sub-bitmap is used to carry a one-bit authentication code by adjusting the parity of the number of '1's to make it equal to that of the one-bit authentication code. Moreover, to achieve the better image quality, the most suitable flipping bit was determined using the least distortion criterion [24]. Besides, two quantization levels are recomputed to further improve the quality of the compressed image block. The renewed AMBTC compression codes are further compressed using the linear prediction technique and the Huffman coding technique to cut down the storage cost of the AMBTC compression codes. Among these methods [22,23], weaknesses have been noted, in that changing the bitmap may further distort reconstructed image quality.

To solve this problem, Hu et al. [25] in 2013 proposed another image authentication scheme for BTC-compressed images. For each image block, the AC was thus embedded into quantization levels by adjusting the k-bit parity value of their difference to be the same as the k-bit AC. In 2014, Nguyen et al. [26] discovered the mean square error provided by scheme [25] was increased because of adjustment of the quantization levels. Thus, a reference table was designed to carry the AC to achieve a better image quality. The *PSNR* provided by their scheme is around 32.43 dB. In the same year, Lin et al. [27] adopted the odevity of the bitmap of AMBTC compression codes to derive the authentication code and then inserted the authentication code into the quantization levels. To enhance the security of the authentication code, the embedding position of the authentication code would be selected with the aid of a pseudo-random sequence. For tampering detection, a two-hierarchy tampering detection strategy is employed to increase performance. In the end, 15 of 16 tampered blocks can be successfully detected when each block carries a four-bit authentication code. Compared to Hu et al. [23], the method proposed by Lin et al. [27] has better visual quality and good detection accuracy. In 2016, Li et al. [28] proposed a novel image authentication scheme to verify the integrity of the AMBTC-compressed image. For each block, the authentication code is inserted into the quantization levels according to the determined reference matrix. The length of the to-be-inserted authentication code can be flexibly decided as the user requires. In this way, their true detection ratio is close to 93.75%, while the authentication code is designed as four bits. For these schemes [25–28], there is room for improvement in detecting the compression codes' attack and collage attack.

To achieve this goal, in 2017, Lin et al. [29] proposed a hybrid image authentication method to protect the integrity of the AMBTC-compressed image. To begin with, they considered the bitmap of the smooth area rather than the complex area as more suitable for parity-check coding [36]. Hence, their scheme first classified the image's blocks into two groups: Smooth and complex. For the smooth group, they forced the parity of the sub-bitmap to match that of the to-be-embedded authentication bit using the bit-flipping technique. For the complex block, on the other hand, the authentication code is embedded into the quantization levels according to a reference table. The different traversal sequence, decided by the number of '1's in a bitmap, is chosen as the hiding sequence to carry the authentication code. In the tampering detection phase, a hybrid strategy is used to ensure superior localization accuracy along with better visual image quality. Experimental results confirm Lin et al.'s scheme outperforms previous schemes on the image quality of watermarked images and tamper detection. However, it is a little regrettable that the scheme [29] did not completely solve the compression

codes' attack. Hence, in 2018, Hong et al. [30] proposed an efficient image authentication method for AMBTC-compressed images using adaptive pixel pair matching. In their scheme, image blocks are classified into edge and non-edge blocks using a predetermined threshold. For each block, the bitmap and location information are inputted into a hashing function to generate the authentication code. The length of the authentication code ranges from one to four bits and can be flexibly determined, according to the type of image block. Then, the authentication code is embedded into the quantization levels using an adaptive reference table. Their scheme significantly reduces image distortion caused by embedding the authentication code and provides a lower false detection rate, averaging at 0.17%. However, their embedding strategy could break the natural relationship between high and low quantization levels; thus, it can only confirm the authenticity of AMBTC compression codes rather than being effective for AMBTC-compressed images. Additionally, their embedding strategy does not always guarantee the minimum distortion for an embedded edge block because a predetermined distance between two quantization levels must be maintained after authentication code embedding.

The same year, Hong et al. [31] proposed two image authentication schemes, i.e., LSBP and MSBP, for tampering detection for AMBTC compression codes. For each block, their schemes can embed an $(a + b)$-bit authentication code generated from the bitmap and quantization levels' MSBs. LSBP is a strategy that embeds the a-bit authentication code into a high quantization level and the b-bit authentication code into a low quantization level using LSBs replacement. Due to the rough embedding strategy, MSBP is suggested to minimize distortion using an MSBs perturbation technique. Their schemes have the ability to achieve a tampering detection rate of more than 93.75%. However, in a few cases, their scheme fails to authenticate the watermarked image due to having broken the natural correlation between quantization levels.

Table 1 gives summaries of those authentication schemes [23,25–31]. In short, some of them [26,28–30] need to store a reference matrix during the AC embedding and authentication phase, and some schemes [23,25–29] have a limitation against the compression codes' attack or collage attack. Also, most existing methods [23,26–30] have the weakness that the upper bound of their first hierarchical tampering detection accuracy is around 93.75%. Hence, most of them employ a second hierarchical tampering detection strategy, such as neighborhood elimination, to improve the tampering detection rate. To overcome those problems, this paper proposes a novel image authentication scheme that protects the integrity of both AMBTC compression codes and AMBTC-compressed images. The proposed scheme does not need a reference matrix during AC embedding and extraction and can resist both the compression codes' attack and collage attack. Our approach achieves a higher tampering detection rate in the first hierarchical tampering detection round without a remediation mechanism and sustains acceptable visual quality.

Table 1. Comparisons of the proposed scheme and other schemes [23,25–31].

Methods	Tampering Detection Result for First Stage (%)	Requirement of Reference Matrix	Main Limitation
Scheme in [23]	93.75	No	Compression codes' attack, Collage attack
Scheme in [25]	96.87	No	
Scheme in [26]	93.75	Yes	
Scheme in [27]	93.75	No	
Scheme in [28]	93.75	Yes	
Scheme in [29]	93.75	Yes	Compression codes' attack
Scheme in [30]	93.75	Yes	More computation
Scheme in [31]	98.50/99.61	No	Collage attack
Proposed scheme	99.85	No	-

The rest of this paper is organized as follows: We briefly review related works in Section 2, including AMBTC compression technology and matrix encoding; in Section 3, we describe the proposed

scheme in detail; in Section 4, we perform a series of experiments to show the performance of our approach; finally, we provide conclusions in Section 5.

2. Related Works

In this section, we first introduce the AMBTC compression technique for an image in Section 2.1 and then look at the matrix encoding for data hiding in Section 2.2.

2.1. AMBTC Compression

In some cases, to increase the speed of transmission, users have to reduce the size of the digital image in advance with compression techniques, whether lossy or lossless. AMBTC is a widely used compression technology due to its simple computation [33]. Meanwhile, AMBTC is a variation of BTC [34] with better reconstructed image quality. Assume there is a to-be-compressed image, I, of $X \times Y$ pixels. The image, I, is thus divided into $x \cdot y$ non-overlapping blocks, with the size of each block being $w \times w$ pixels. In general, w is set to 4 in both BTC and AMBTC. Assume $p_{(1,1)}, p_{(1,2)}, \cdots, p_{(1,w)}$; $p_{(2,1)}, p_{(2,2)}, \cdots, p_{(2,w)}; \cdots; p_{(w,1)}, p_{(w,2)}, \cdots, p_{(w,w)}$ are the pixel values of each block. The detailed process of AMBTC compression is described as follows. First, the average, μ, of those pixels' values in a block is calculated by:

$$\mu = \frac{1}{w \cdot w} \sum_{r=1}^{w} \sum_{c=1}^{w} p_{(r,c)}, \tag{1}$$

where $p_{(r,c)}$ is the pixel value of a block in the to-be-compressed image, I.

According to the average value, μ, the pixels in a block can be partitioned into two subgroups according to the following rules:

$$\begin{cases} p_{(r,c)} \in G_{S0}, \text{ if } p_{(r,c)} < \mu, \\ p_{(r,c)} \in G_{S1}, \text{ if } p_{(r,c)} \geq \mu, \end{cases} \tag{2}$$

where $1 \leq r, c \leq w$, G_{S0} is a subgroup that contains all pixels whose values are lower than the average value, μ, and G_{S1} is a subgroup that contains the pixels not included in G_{S0}.

Later, the average value for each subgroup is calculated as follows:

$$l = \frac{1}{k_0} \sum_{p_{(r,c)} \in G_{S0}} p_{(r,c)}, \tag{3}$$

$$h = \frac{1}{k_1} \sum_{p_{(r,c)} \in G_{S1}} p_{(r,c)}, \tag{4}$$

where k_0, k_1 are the numbers of pixels in the subgroups, G_{S0}, G_{S1}, respectively, and $k_0 + k_1 = k$, $k = w \cdot w$. Here, h is the high quantization level and l is the low quantization level. A bitmap (bm) for the current block is generated based on the following rules: (1) If the pixel belongs to the subgroup, G_{S0}, its corresponding position in the bitmap is marked '0'; (2) if the pixel belongs to the subgroup, G_{S1}, its corresponding position in the bitmap is marked '1'. The generation rule for the bitmap is described as follows:

$$bm_{(r,c)} = \begin{cases} 0, p_{(r,c)} \in G_{S0} \\ 1, p_{(r,c)} \in G_{S1} \end{cases}, \text{ for } 1 \leq r, c \leq w. \tag{5}$$

In this way, the AMTBC compression code for one block is derived in the form of a trio (h, l, bm). On the receiving side, the decoding process is quite simple. A reconstructed block can be derived based on the following rules:

$$p'_{(r,c)} = \begin{cases} l, \text{ if } bm_{(r,c)} = 0 \\ h, \text{ if } bm_{(r,c)} = 1 \end{cases}, \text{ for } 1 \leq r, c \leq w, \tag{6}$$

where $p'_{(r,c)}$ is the pixel value of the reconstructed image. Once all blocks have been processed by Equation (6), the reconstructed image is obtained.

2.2. The Matrix Encoding

The (7, 4) Hamming code [37,38] was invented in 1950 by Richard Hamming as a linear error-correction code. The basic idea of the (7, 4) Hamming code is that some attached information, i.e., three parity check bits, are added to the original four-bit data. As a result, the recipient can detect and correct a single-bit error with the help of the parity check matrix, H, as shown in Equation (7). The result of the equation $(H \times RCW^T)^T$ indicates where the error bit occurs, with RCW representing the received codeword. Assume a codeword that meets the verification rules is CW = (1101001), and the received codeword is RCW = (1100001), and the syndrome vector, $z^T = (100)^T$, is equal to the fourth column of the parity check matrix, H. This means that a single-bit error is detected and the error bit in this RCW is in the fourth position.

$$\begin{aligned} z &= (H \times RCW^T)^T \\ &= \left(\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right)^T \\ &= (100). \end{aligned} \tag{7}$$

Based on the (7, 4) Hamming code, Ron Crandall proposed an efficient embedding method known as matrix encoding [39]. In matrix encoding, for a (1, n, k) code, the n modifiable bit-places are used to carry the k-bit secret message by flipping at most one modifiable place, where $n = 2^k - 1$. In this paper, we pay more attention to the (1, 7, 3) code.

First, a coset of the (7, 4) Hamming code with a parity check matrix, H, is constructed, as shown in Table 2. To show the embedding process intuitively, an example is given as follows.

Table 2. The cosets of the (7, 4) Hamming code with the parity check matrix, H.

ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Syndrome	000	001	010	011	100	101	110	111
Coset leader	0000000	1000000	0100000	0010000	0001000	0000100	0000010	0000001

Let us assume the to-be-embedded message is $S = (s_1, s_2, s_3)$ and the cover vector is $CV = (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$. The details of the embedding process are as follows.

Step 1: Compute $M = \left(H \times CV^T\right)^T$ to derive vector M.

Step 2: Calculate the syndrome vector, $z = M \oplus S$.

Step 3: Search for the same syndrome value as z in Table 2, then the gth column can be located, where $0 \le g \le 7$. The corresponding identifier (ID) is also g, which is the to-be-flipped bit-location later in this paper. At the same time, the corresponding coset leader vector is mapped as e_g.

Step 4: Change the gth bit of the cover vector CV by $CV' = CV \oplus e_g$. Note that if the syndrome value, z = (000) or g = 0, then there is nothing to be changed; that is, $CV' = CV$.

Finally, the embedding process ends and CV' carries the three-bit secret message. The decoding process is simple, using $S' = \left(H \times (CV')^T\right)^T$, where S' is the extracted message and is the same as the secret message S, as long as no error occurred.

For example, assume S = (101) and CV = (1101001). The computed M is (000) and the syndrome z is (101). By searching Table 2, the fifth column is located, ID = 5 is found, and its corresponding coset leader, e_5 = (0000100), is determined. Finally, the embedding process is carried out and the cover vector, CV, is changed to CV' = (1101101). For the recipient, the secret, S', can be extracted by $S' = \left(H \times (CV')^T\right)^T = (101)$. That is to say, the seven-bit cover vector can carry the three-bit message by changing at most one bit.

3. Proposed Scheme

To increase the tampering localization accuracy after the first hierarchical tampering detection, the bitmap carries more authentication code for each block using the matrix encoding scheme. Meanwhile, to overcome the distortion caused by changing the bitmap, the corresponding to-be-flipped bit-location information, G, is recorded instead of flipping this bit of the bitmap directly. Then, this bit-location information is embedded into the quantization levels based on adjusted quantization level matching. Finally, a flowchart of the authentication code generation and embedding is shown in Figure 1. Besides, to better present the proposed scheme, the main symbols used in this paper and their definitions are listed in Table 3.

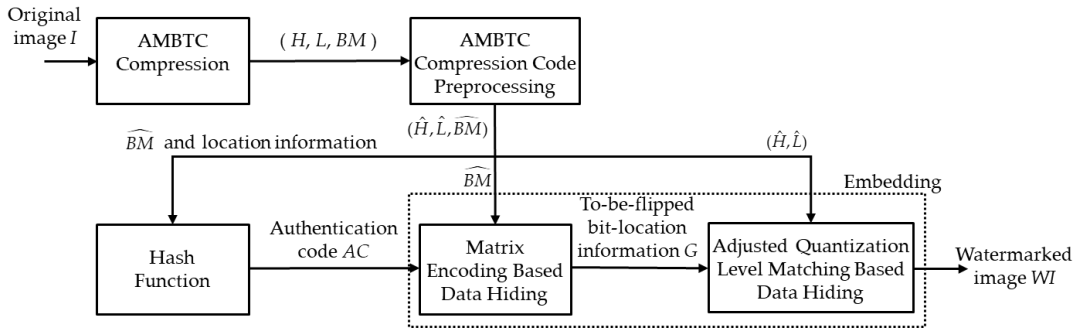

Figure 1. Flowchart of the authentication code generation and embedding.

Table 3. Main symbols used in this paper and their definitions.

Symbols	Definitions	Symbols	Definitions
I	Original image	WI	Watermarked image
H	Parity check matrix	TM	Tampered map
h	High quantization level for a block in I	H	A set of h
l	Low quantization level for a block in I	L	A set of l
bm	Bitmap for a block in I	BM	A set of bm
$\hat{h}$	Preprocessed high quantization level	$\hat{H}$	A set of $\hat{h}$
$\hat{l}$	Preprocessed low quantization level	$\hat{L}$	A set of $\hat{l}$
$\hat{bm}$	Preprocessed bitmap	$\hat{BM}$	A set of $\hat{bm}$
h'	High quantization level for a block in WI	H'	A set of h'
l'	Low quantization level for a block in WI	L'	A set of l'
bm'	Bitmap for a block in WI	BM'	A set of bm'
g_1, g_2	Bit-location information of a block	ac	Authentication code for a block

3.1. AMBTC Compression and Preprocessing Phase

In this paper, the original image, I, is divided into non-overlapping blocks, and then the AMBTC compression technique as mentioned in Section 2.1 is performed for image block compression. As a result, AMBTC compression codes are derived and denoted as (H, L, BM), where H, L, and BM represent the sets of high quantization level, low quantization level, and bitmap, respectively. For convenience, let us assume (h, l, bm) is a triple of AMBTC compression codes for one block. The preprocessing of the AMBTC compression code for a block is done as follows:

$$\begin{cases} \hat{l} = l,\ \hat{h} = h+1,\ \hat{bm} = bm \,\&\, bm_{(u,v)} = 0, & \text{if } l = h = 0, \\ \hat{l} = l-1,\ \hat{h} = h,\ \hat{bm} = bm \,\&\, bm_{(u,v)} = 0, & \text{if } 0 < l = h \le 255, \\ \hat{l} = l, \quad \hat{h} = h,\ \hat{bm} = bm, & \text{if } l \ne h, \end{cases} \tag{8}$$

where u and v are constants, and $\hat{h}$, $\hat{l}$, and $\hat{bm}$ are the high quantization level, low quantization level, and bitmap of a preprocessed AMBTC compression code, respectively. After all blocks have been preprocessed in the same way, the preprocessed AMBTC compression codes $(\hat{H}, \hat{L}, \hat{BM})$ were obtained. Note that the $bm_{(u,\,v)}$ should not be used in the embedding procedure. In our experiments, (u, v) is set to (3, 1), based on experimental results.

We preprocess to check whether the h is equal to l. If it is, the bits in the bm are all equal to '1' and the preprocessing should thus be conducted; otherwise, nothing should be done. Preprocessing ensures the recipient can obtain the same AMBTC compression codes from the reconstructed image, as long as no malicious attack occurs. Two examples of preprocessing are the red dotted squares shown in Figure 2. In Figure 2a, since h is not equal to l, the AMBTC compression code remains the same $\left(\hat{h},\ \hat{l},\ \hat{bm}\right) = (h, l, bm)$. Meanwhile, in Figure 2b, since h is equal to l, the AMBTC compression code is preprocessed as $\left(\hat{h},\ \hat{l},\ \hat{bm}\right) = \left(h, l-1, bm \,\&\, bm_{(3,1)} = 0\right)$.

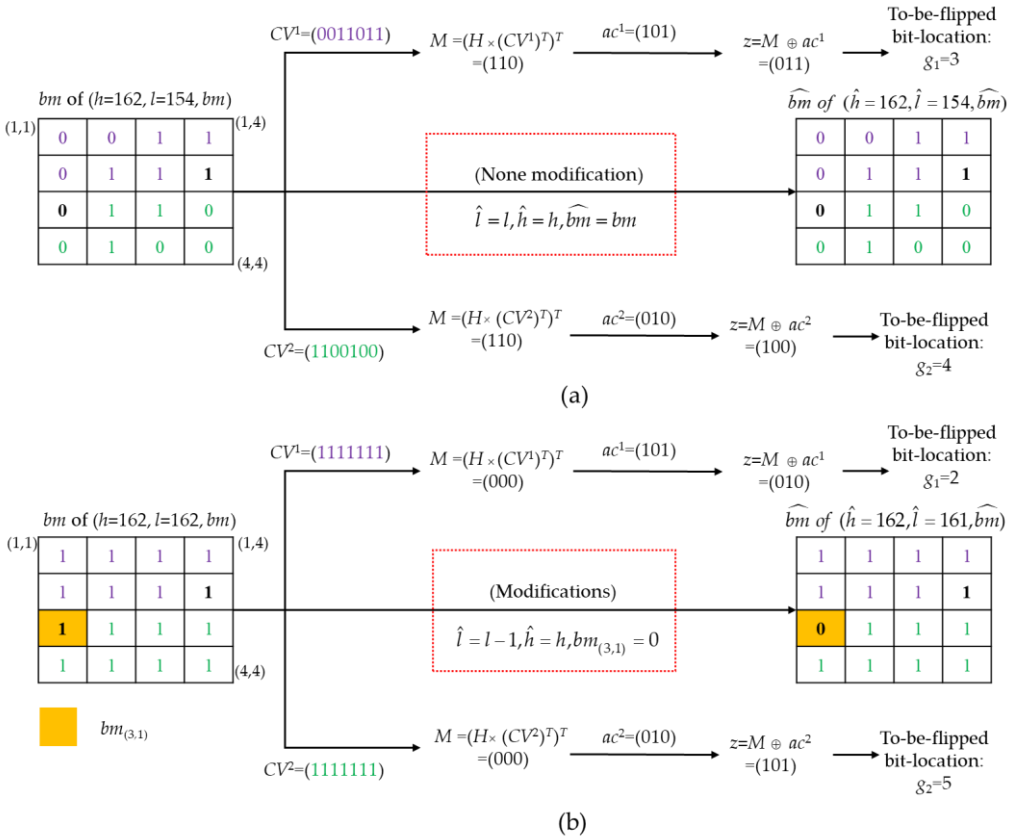

Figure 2. Examples of preprocessing and generation of the to-be-flipped bit-location information when (**a**) $h \ne l$; (**b**) $h = l$ & $l \ne 0$.

3.2. Authentication Code Generation Phase

After the processed AMBTC compression code is generated, for each block, our approach generates a six-bit authentication code by feeding the block's $\hat{bm}$ and location information into a hash function. This six-bit authentication code is further separated into two three-bit sub-codes, which are embedded into the bitmap using matrix encoding. Let us assume that the binary form of a six-bit

authentication code is $ac = (ac_1, ac_2, ac_3, ac_4, ac_5, ac_6)$; the first three-bit sub-code is then defined as $ac^1 = \left(ac_1^1, ac_2^1, ac_3^1\right) = (ac_1, ac_2, ac_3)$ and the second three-bit sub-code as $ac^2 = \left(ac_1^2, ac_2^2, ac_3^2\right) = (ac_4, ac_5, ac_6)$.

3.3. Embedding Phase

The flowchart of the authentication code embedding phase is shown in Figure 1. First, the authentication code is embedded into the bitmap. In fact, we only record the to-be-flipped bit-location information. No modification has been done in *bm*, as will be described in Section 3.3.1. Second, the to-be-flipped bit-location information is inserted into the two quantization levels, as will be described in Section 3.3.2.

3.3.1. The Matrix Encoding Based Data Hiding

After authentication code generation and AMBTC compression codes' preprocessing, the authentication code will be embedded into the bitmap for each block. The algorithm of the embedding process is described as follows.

Input: Original image, *I*.

Output: To-be-flipped bit-location information, *G*.

Step 1. Divide the original image, *I*, into non-overlapping blocks by a size of 4×4 pixels.

Step 2. Generate the AMBTC compression codes (*H*, *L*, *BM*) for the image, *I*, as explained in Section 2.1. For each block, assume its preprocessed AMBTC compression code is $(\hat{h}, \hat{l}, \hat{bm})$.

Step 3. Generate the authentication code, *AC*, as explained in Section 3.2. For each block, assume the first three-bit sub-code is $ac^1 = \left(ac_1^1, ac_2^1, ac_3^1\right)$ and the second three-bit sub-code is $ac^2 = \left(ac_1^2, ac_2^2, ac_3^2\right)$.

Step 4. Extract the first seven bits of *bm* as the first cover vector, $CV^1 = \left(cv_1^1, cv_2^1, cv_3^1, cv_4^1, cv_5^1, cv_6^1, cv_7^1\right) = (bm_{(1,1)}, bm_{(1,2)}, bm_{(1,3)}, bm_{(1,4)}, bm_{(2,1)}, bm_{(2,2)}, bm_{(2,3)})$. Extract the last seven bits of *bm* as the second cover vector, $CV^2 = \left(cv_1^2, cv_2^2, cv_3^2, cv_4^2, cv_5^2, cv_6^2, cv_7^2\right) = (bm_{(3,2)}, bm_{(3,3)}, bm_{(3,4)}, bm_{(4,1)}, bm_{(4,2)}, bm_{(4,3)}, bm_{(4,4)})$.

Step 5. Perform matrix encoding, as explained in Section 2.2, to embed the first three-bit sub-code, ac^1, into the first cover vector, CV^1, then derive the to-be-flipped bit-location, g_1. The second three-bit sub-code, ac^2, is embedded into the second cover vector, CV^2, and then used to derive the to-be-flipped bit-location, g_2. Note that we only record the to-be-flipped bit-location information but nothing is modified for the *bm*. Hence, for each block, two-tuple location information (g_1, g_2) can be derived.

Step 6. Perform Steps 2 to 5 until all blocks have been processed.

Step 7. Output all location information (g_1, g_2) for each block to the to-be-flipped bit-location information, *G*.

Step 8. End.

After matrix encoding is completed, the to-be-flipped bit-location information, *G*, is obtained. We provide two examples to demonstrate the generation of the to-be-flipped bit-location information, as shown in Figure 2. In Figure 2a, the first seven bits of *bm* are recombined as the first cover vector, CV^1, shown in purple. The last seven green bits comprise the second cover vector, CV^2. Later, matrix encoding is conducted, as described in Section 2.2, and then the two-tuple location information ($g_1 = 3$, $g_2 = 4$) will be derived and used as the to-be-hidden message in the next section. In this case, the AMBTC compression code is not changed. Similarly, Figure 2b presents a special case, i.e., *h* is equal to *l* and all bits in *bm* are 1. Thus, the preprocessing of the AMBTC compression code should be done, as described in Section 3.1; i.e., *l* is decreased by 1 and the $bm_{(3,1)}$ is set to 0. In this case, *l* and *bm* are slightly modified. Also, two-tuple location information ($g_1 = 2$, $g_2 = 5$) will be derived and used as the to-be-hidden message in the following phase. The yellow grid represents the chosen $bm_{(3,1)}$, as described in Section 3.1.

3.3.2. Adjusted Quantization Levels Matching Based Data Hiding

After the matrix encoding procedure, the two-tuple to-be-flipped bit-location information, G, is individually inserted into two quantization levels. The algorithm of the adjusted quantization level matching-based data hiding is described as follows.

Input: Preprocessed AMBTC compression codes $(\hat{H}, \hat{L}, B\hat{M})$, to-be-flipped bit-location information, G.
Output: Watermarked image, WI.

Step 1. Get a triple $(\hat{h},\ \hat{l},\ \hat{bm})$ for one block from preprocessed AMBTC compression codes $(\hat{H}, \hat{L}, B\hat{M})$.
Step 2. Get the corresponding to-be-flipped bit-location information (g_1, g_2) for this block from G.
Step 3. Calculate the factors for $\hat{h}$ and $\hat{l}$ for the current block by:

$$\begin{cases} f_h = \left\lfloor \frac{\hat{h}}{2^3} \right\rfloor, \\ f_l = \left\lfloor \frac{\hat{l}}{2^3} \right\rfloor, \end{cases} \tag{9}$$

where f_h and f_l represent the factors of $\hat{h}$ and $\hat{l}$, respectively.
Step 4. Calculate the remaining values for $\hat{h}$ and $\hat{l}$ for the current block by:

$$\begin{cases} rv_h = \hat{h} \bmod 2^3, \\ rv_l = \hat{l} \bmod 2^3, \end{cases} \tag{10}$$

where rv_h and rv_l represent the remainder values of $\hat{h}$ and $\hat{l}$, respectively.
Step 5. Perform adjusted quantization level matching-based data hiding. If the remainder, rv_h, is equal to g_1, the matching work of $\hat{h}$ is done; otherwise, the candidates at the quantization level, $\hat{h}$, should be adjusted by:

$$\begin{cases} h_1 = f_h \cdot 2^3 + g_1, \\ h_2 = \begin{cases} h_1 + 2^3, & \text{if } h_1 < \hat{h}, \\ h_1 - 2^3, & \text{if } h_1 > \hat{h}, \\ h_1, & \text{if } h_1 = \hat{h}, \end{cases} \end{cases} \tag{11}$$

where h_1 and h_2 are the candidates at the quantization level, $\hat{h}$. Here, if h_2 is lower than zero or higher than 255, h_2 must be set to h_1. Similarly, if the remainder, rv_l, is equal to g_2, the matching work of $\hat{l}$ is done; otherwise, the candidates at the quantization level, $\hat{l}$, should be adjusted by:

$$\begin{cases} l_1 = f_l \cdot 2^3 + g_2, \\ l_2 = \begin{cases} l_1 + 2^3, & \text{if } l_1 < \hat{l}, \\ l_1 - 2^3, & \text{if } l_1 > \hat{l}, \\ l_1, & \text{if } l_1 = \hat{l}, \end{cases} \end{cases} \tag{12}$$

where l_1 and l_2 are the candidates at quantization level $\hat{l}$. l_2 must be set to l_1 if l_2 is lower than zero or higher than 255.
Step 6. Let $h_{cf} = h_a$ and $l_{cf} = l_b$ be the final, selected solution under the constraint of the least distortion, $dist_{(a,b)}$, which can be calculated by:

$$dist_{(a,b)} = \sum_{r=1}^{4} \sum_{c=1}^{4} \begin{cases} (p_{(r,c)} - h_a)^2 \text{ if } bm_{(r,c)} = 1, \\ (p_{(r,c)} - l_b)^2 \text{ if } bm_{(r,c)} = 0, \end{cases} \tag{13}$$

where $a, b \in [1, 2]$, $h_a \in [h_1, h_2]$, $l_b \in [l_1, l_2]$. Besides, in a few cases, this situation may happen, that is, h_{cf} will be equal to l_{cf}, then h_{cf} or l_{cf} will be forced to be justified once again by the following rule:

$$\begin{cases} l_{cf} = l_{cf} - 2^3, & \text{if } l_{cf} > 247, \\ h_{cf} = h_{cf} + 2^3, & \text{otherwise.} \end{cases} \tag{14}$$

Step 7. Perform Steps 1 to 6 until all blocks have been processed.

Step 8. Output the watermarked AMBTC compression code for each block, and the watermarked image, *WI*, is achieved.

Step 9. End.

An example is given to show the detailed processing of the adjusted quantization level matching-based data hiding in Figure 3. Here, for convenience, we assume $dist_{(1,1)}$ is the least distortion after calculation using Equation (13) in this example.

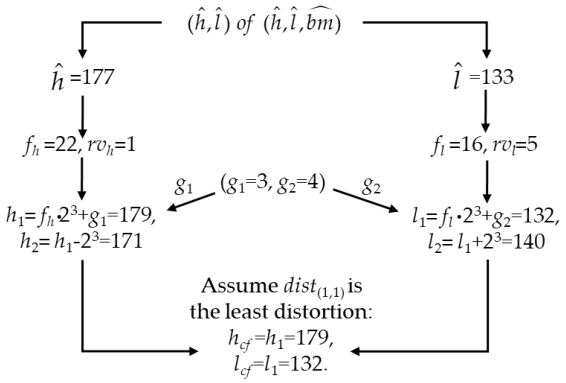

Figure 3. Example of adjusted quantization levels matching based data hiding.

3.4. Tampering Detection Phase

Once the recipient receives the watermarked image, *WI*, or the watermarked AMBTC compression codes, the precise tampering detection can be done in respect to whether the tampering occurred or not. Figure 4 depicts the flowchart of tampering detection. First, the bit-location information is extracted from the AMBTC compression codes. Then, the authentication code is calculated and reconstructed. Finally, the extracted and recalculated authentication code are compared to determine whether each block has been tampered with.

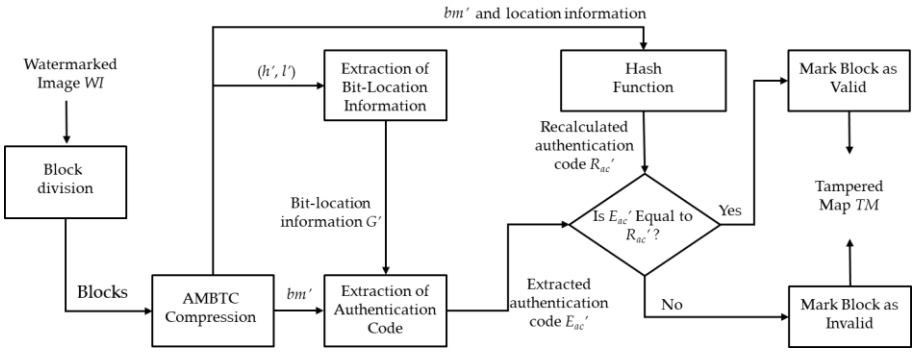

Figure 4. Flowchart of tampering detection.

3.4.1. Extraction of Bit-Location Information

The algorithm for bit-location information extraction is described as follows.

Input: Watermarked image, *WI*.

Output: Bit-location information, *G'*.

Step 1. Divide the watermarked image, *WI*, into non-overlapping blocks by a size of 4 × 4 pixels.

Step 2. Perform the AMBTC compression technique for one block to derive the corresponding AMBTC compression code (h', l', bm').

Step 3. Extract the bit-location information (g'_1, g'_2) for this block by the following equation:

$$\begin{cases} g'_1 = h' \bmod 2^3, \\ g'_2 = l' \bmod 2^3, \end{cases} \tag{15}$$

Step 4. Perform Steps 2 and 3 until all blocks have been processed.

Step 5. Output all location information (g'_1, g'_2) for each block to provide the bit-location information, G'.

Step 6. End.

An example of bit-location information extraction is given in Figure 5a.

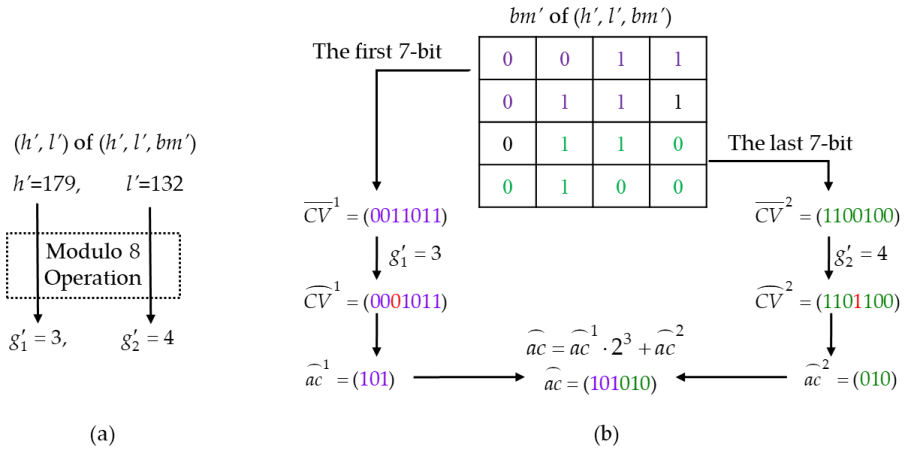

Figure 5. Examples: (**a**) example of extracting bit-location information, (**b**) example of extracting authentication code.

3.4.2. Extraction of the Authentication Code

The algorithm of authentication code extraction is described as follows.
Input: AMBTC compression codes (H', L', BM'), bit-location information, G'.
Output: Extracted authentication code, EAC.

Step 1. Get a triple (h', l', bm') for one block from AMBTC compression codes (H', L', BM').

Step 2. Get a tuple bit-location information (g'_1, g'_2) for the corresponding block from G'.

Step 3. Recombine the first seven bits of bm' as the first cover vector, $\overline{CV}^1 = \left(cv_1^1, cv_2^1, cv_3^1, cv_4^1, cv_5^1, cv_6^1, cv_7^1\right)$. Recombine the last seven bits of bm' as the second cover vector, $\overline{CV}^2 = \left(cv_1^2, cv_2^2, cv_3^2, cv_4^2, cv_5^2, cv_6^2, cv_7^2\right)$.

Step 4. Flip the g'_1th bit for the first cover vector, $\overline{CV}^1$, and denoted as $\widehat{CV}^1$, then flip the g'_2th bit for the first cover vector, $\overline{CV}^2$, and denoted as $\widehat{CV}^2$. Note that if g'_1 or g'_2 is equal to zero, the corresponding flipping operation is skipped. In Figure 5b, the red digit represents the flipped bit-location.

Step 5. Extract the authentication code. Two three-bit sub-codes $(\widehat{ac}^1, \widehat{ac}^2)$ can be computed by:

$$\begin{cases} \widehat{ac}^1 = \left(H \times \left(\widehat{CV}^1\right)^T\right)^T, \\ \widehat{ac}^2 = \left(H \times \left(\widehat{CV}^2\right)^T\right)^T. \end{cases} \tag{16}$$

Then, the six-bit authentication code, $\widehat{ac}$, can be derived by:

$$\widehat{ac} = \widehat{ac}^{1} \cdot 2^{3} + \widehat{ac}^{2}. \quad (17)$$

Step 6. Perform Steps 1 to 5 until all blocks have been processed.

Step 7. Output the authentication code, $\widehat{ac}$, for each block to provide the extracted authentication code, *EAC*.

Step 8. End.

Figure 5b shows an example of authentication code extraction.

3.4.3. Tampering Detection

The algorithm of the tampering detection is described as follows.
Input: Extracted authentication code, *EAC*.
Output: Tampered map, *TM*.

Step 1. Generate the authentication code, as mentioned in Section 3.2, and denote it as *RAC*.

Step 2. Get an authentication code for one block from *RAC* and denote it as $R_{ac}{}'$.

Step 3. Get an authentication code for the corresponding block from *EAC* and denote it as $E_{ac}{}'$.

Step 4. Mark the tampered map, *TM*, according to the comparison results of $R_{ac}{}'$ and $E_{ac}{}'$. If they are equal, the corresponding position of *TM* is marked as '0', which means the current block is valid; otherwise, it is marked as '1', which indicates the current block is invalid.

Step 5. Perform Steps 2 to 4 until all blocks have been processed.

Step 6. Output the tampered map, *TM*.

Step 7. End.

Without loss of generality, a second hierarchical tampering detection algorithm [29,30] is also provided in our experiments. This improves the tampering detection rate, even if excellent detection performance is already offered by the first hierarchical tampering detection algorithm.

4. Experimental Results

In this section, a series of experiments and analyses are performed to demonstrate the performance of the proposed method. All experiments were implemented in Matlab R2017a on a PC with Intel® Core (TM) i7-3770 CPU @3.4 GHz, 8 GB RAM (Intel Corporation, Santa Clara, CA, USA). Nine classic grayscale images with the size of 512 × 512 served as test images, as shown in Figure 6. Several attacks were employed to test the performance of our proposed scheme on tampering detection, including cropping attack, constant average attack, collage attack, and AMBTC compression codes' attack. All test images were compressed using the AMBTC compression technique with the size of 4 × 4 pixels.

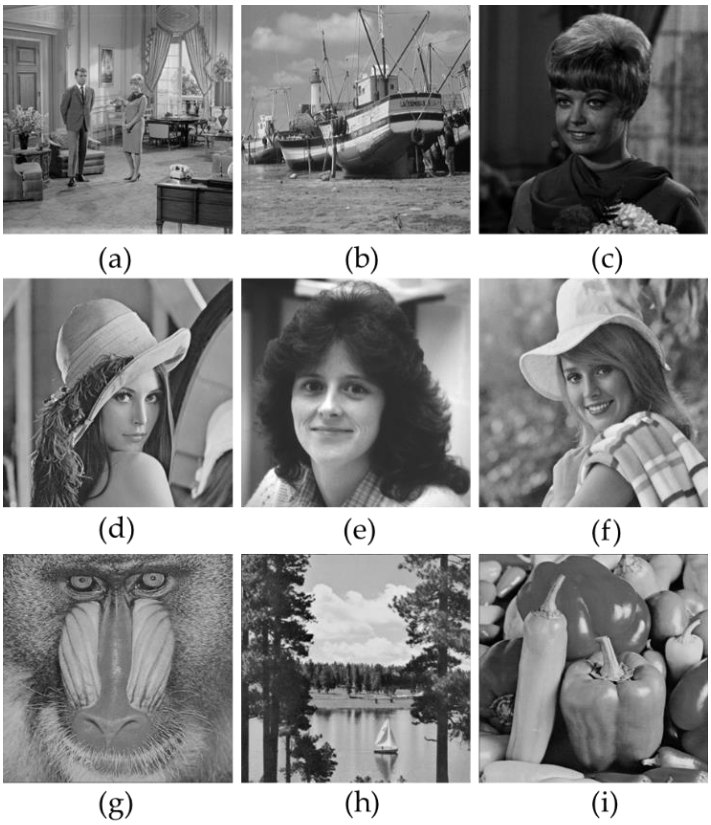

Figure 6. Nine 512 × 512 grayscale images. (**a**) Couple; (**b**) boat; (**c**) Zelda; (**d**) Lena; (**e**) woman; (**f**) Elaine; (**g**) baboon; (**h**) lake; (**i**) peppers.

4.1. Statistical Metrics

The following statistical metrics were utilized to demonstrate the superior performance of our approach.

(a) Peak signal-to-noise ratio (*PSNR*) [14]: Measures the difference between the watermarked image and the original image:

$$PSNR = 10 \cdot \log_{10}\left(\frac{255^2}{\frac{1}{X \cdot Y} \cdot \sum_{i=1}^{X} \sum_{j=1}^{Y} \left(p_{wi}(i,j) - p_{oi}(i,j)\right)^2}\right), \tag{18}$$

where $X \times Y$ is the size of an image; $p_{wi}(i,j)$ represents the pixel value of the watermarked image, and $p_{oi}(i,j)$ represents the pixel value of the original image.

(b) Tampering detection rate (*TDR*) [14]: Measures the percentage of tampered pixels detected in a tampered area:

$$TDR = \frac{\text{No. of detected tampered pixels}}{\text{All pixels in tampered region}}. \tag{19}$$

(c) False positive rate (*FPR*) [14]: Measures the percentage of non-tampered pixels misjudged as tampered, among all tampered pixels:

$$FPR = \frac{\text{No. of false misjudged pixels}}{\text{All tampered pixels}}. \tag{20}$$

(d) False negative rate (*FNR*) [14]: Measures the percentage of tampered pixels misjudged as non-tampered, among all non-tampered pixels:

$$FNR = \frac{\text{No. of false misjudged pixels}}{\text{All non} - \text{tampered pixels}}. \quad (21)$$

4.2. Tampering Detection Analyses

In our experiments, we performed more than 14 kinds of attacks to show the stable and reliable localization ability and superior performance of the first hierarchical tampering detection algorithm. The details are as follows.

4.2.1. Cropping Attack

We used three grayscale images, Lena, Elaine, and woman, to simulate the cropping attack. Their watermarked images are shown in Figure 7a–c, with *PSNRs* of 31.99, 32.41, and 35.03 dB, respectively. The tampered Lena image is shown in Figure 7d, with some inside blocks cropped with a rectangle. The corresponding tampering detection result using the first hierarchical tampering detection method is shown in Figure 7g, with a *TDR* of 98.75%. The tampered Elaine image is attacked by outside cropping within the larger area, as shown in Figure 7e. The corresponding tampering detection result delivered by the first hierarchical tampering detection algorithm is shown in Figure 7h, with a *TDR* of 98.60%. Finally, we employed an inside cropping attack with an irregular shape to replace some blocks of the watermarked woman image, as shown in Figure 7f. Certainly, the proposed scheme achieves a high tampering detection rate, with a *TDR* of 98.99%. Thus, our approach achieves superior detection and localization to both the inside cropping attack and the outside cropping attack.

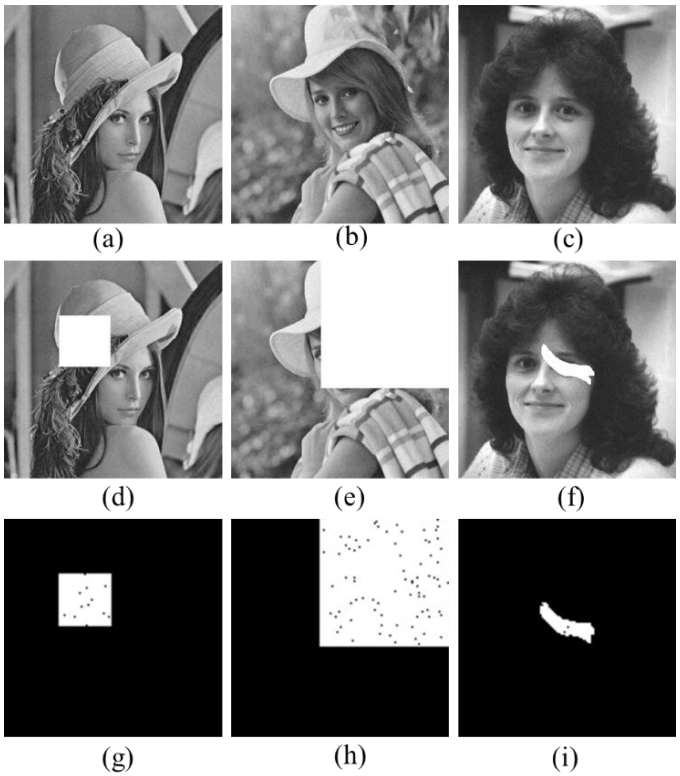

Figure 7. Cropping attack: watermarked images for (**a**) Lena (*PSNR* = 31.99 dB); (**b**) Elaine (*PSNR* = 32.41 dB); (**c**) woman (*PSNR* = 35.03 dB). Tampered images for (**d**) Lena, (**e**) Elaine, (**f**) woman. Tampering detection results after first hierarchy for (**g**) *TDR* = 98.75%; (**h**) *TDR* = 98.60%; (**i**) *TDR* = 98.99%.

Furthermore, the tampering detection performance for the cropping attack is demonstrated in Table 4. As we can see, after the first hierarchical tampering detection algorithm, the number of tampered blocks for the tampered Lena, Elaine, and woman images are 949, 5620, and 293, respectively. Our approach achieves a high *TDR*, with an average of 98.78%, and a low *FPR* and *FNR*, with averages of 0 and 0.3356, respectively. Additionally, a second hierarchical tampering detection strategy was used in this paper, and the corresponding detection performance is shown in Table 4. Here, the *TDR* is close to 100% and the *FNR* and *FPR* are close or equal to 0.

Table 4. Tampering detection performance of the cropping attack.

Original Images	Number of Tampered Blocks (4 × 4)			The First Hierarchical Tampering Detection Results (%)			The Second Hierarchical Tampering Detection Results (%)		
	Total	First Hierarchy	Second Hierarchy	*TDR*	*FPR*	*FNR*	*TDR*	*FPR*	*FNR*
Lena	961	949	961	98.75	0	0.0777	100	0	0
Elaine	5700	5620	5698	98.60	0	0.7432	99.96	0	0.0187
Woman	296	293	296	98.99	0	0.0186	100	0	0
Average				98.78	0	0.3356	99.99	0	0.0063

4.2.2. Constant Average Attack

Two watermarked images using our proposed scheme, Zelda and baboon, are shown in Figure 8a,d, with *PSNRs* of 35.07 and 27.78 dB, respectively. The constant average attack is an attack that replaces all pixels of one block with the average value of the current block. Figure 8b shows the tampered Zelda image. Within a 2 × 2 block, all pixels are modified by the mean value. As we can see, it is difficult to identify the change visually. The corresponding tampering detection result using the first hierarchical tampering detection method is shown in Figure 8c, with a *TDR* of 96.42%. Figure 8e shows the tampered baboon image under the constant average attack. Within a 4 × 4 block, all pixels are replaced by the mean value. The corresponding tampering detection result from the first hierarchical tampering detection algorithm is shown in Figure 8f, with a *TDR* of 99.02%.

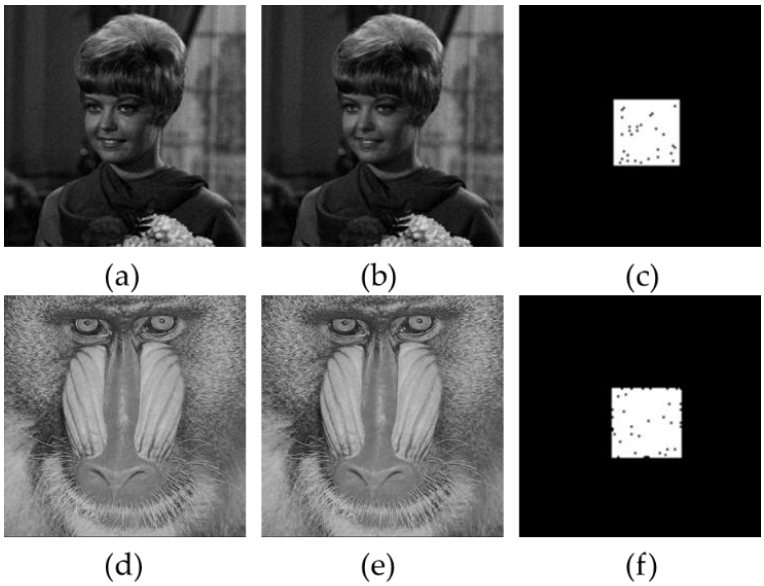

Figure 8. Constant average attack: watermarked images for (**a**) Zelda (*PSNR* = 35.07 dB); (**d**) baboon (*PSNR* = 27.78 dB). Tampered images for (**b**) Zelda (2 × 2); (**e**) baboon (4 × 4). Tampering detection results after the first hierarchy for (**c**) *TDR* = 96.42%; (**f**) *TDR* = 99.02%.

Meanwhile, Table 5 shows the tampering detection performance of the constant average attack. As we can see, after the first round of hierarchical tampering detection, the *TDR* of the four experiments is higher than 96.42% and there is low *FNR* and *FPR*. The second hierarchical tampering detection strategy was also employed, and the corresponding detection performance is rather good. For both, the *TDR* is close to 100% and the *FPR* and *FNR* are close to 0.

Table 5. Tampering detection performance of the constant average attack.

Size of Blocks	Original Images	Number of Tampered Blocks			The First Hierarchical Tampering Detection Results (%)			The Second Hierarchical Tampering Detection Results (%)		
		Total	First Hierarchy	Second Hierarchy	*TDR*	*FPR*	*FNR*	*TDR*	*FPR*	*FNR*
4×4	Zelda	1225	1196	1225	97.63	0	0.1909	100	0	0
	Baboon	1225	1213	1225	99.02	0	0.0791	100	0	0
2×2	Zelda	1369	1320	1367	96.42	0	0.3253	99.85	0	0.0133
	Baboon	1369	1333	1368	97.37	0	0.2392	99.93	0	0.0067
		Average			97.61	0	0.2086	99.95	0	0.0050

4.2.3. Collage Attack

In this section, an image is tampered with by copying image blocks from other watermarked images and pasting them into arbitrary positions in the watermarked image; this is called a collage attack. In Figure 9, we can see the watermarked images of lake, peppers, boat, and couple, with *PSNRs* of 29.88, 32.15, 30.89, and 30.38 dB, respectively. The first example of collage attack, shown in Figure 9c, is generated by copying the vegetables from Figure 9b into Figure 9a while preserving their relative spatial locations. After the first hierarchical tampering detection, the corresponding tampering detection result is shown in Figure 9d, with a *TDR* of 98.45%. Similarly, we copied some blocks from the watermarked couple image, shown in Figure 9f, into the watermarked boat image, shown in Figure 9e on the bottom left. The tampered boat image is presented in Figure 9g. Verification was performed using our tampering detection approach, with a *TDR* of 98.59%, as shown in Figure 9h.

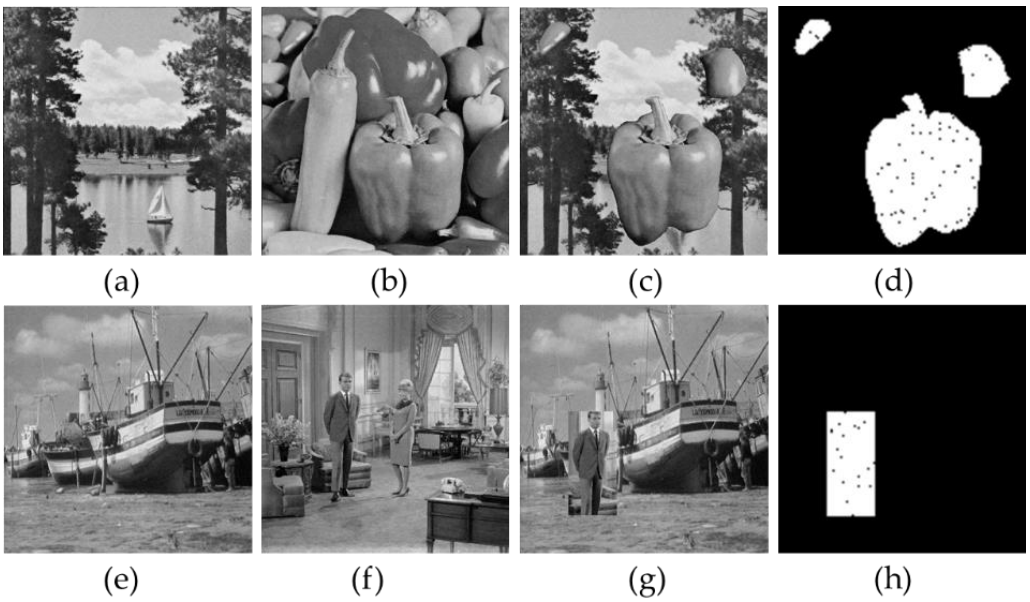

Figure 9. Collage attack: watermarked images for (**a**) lake (*PSNR* = 29.88 dB); (**b**) pepper (*PSNR* = 32.15 dB); (**e**) boat (*PSNR* = 30.89 dB); (**f**) couple (*PSNR* = 30.38 dB). Tampered images for (**c**) lake; (**g**) boat. Tampering detection results after the first hierarchy for (**d**) *TDR* = 98.45%; (**h**) *TDR* = 98.59%.

Table 6 lists the detailed tampering detection performance of the proposed scheme. The tampering detection rate reaches 98%, with an *FPR* of 0 and a lower *FNR*. Furthermore, a two-hierarchy tampering detection strategy was also employed on this kind of attack, resulting in superior tampering detection, as shown in the last three columns of Table 6.

Table 6. Tampering detection performance of the collage attack.

Original Mages	Number of Tampered Blocks (4 × 4)			The First Hierarchical Tampering Detection Results (%)			The Second Hierarchical Tampering Detection Results (%)		
	Total	First Hierarchy	Second Hierarchy	*TDR*	*FPR*	*FNR*	*TDR*	*FPR*	*FNR*
Lake	4126	4062	4121	98.45	0	0.5194	99.88	0.0485	0.0408
Boat	1350	1331	1350	98.59	0	0.1262	100	0	0
Average				98.52	0	0.3228	99.94	0.0243	0.0204

4.2.4. AMBTC Compression Codes' Attacks

In this section, kinds of experiments are performed to test the tampering detection performance of the AMBTC compression codes' attack. We applied these attacks to the original 512 × 512 Lena image.

(a) Attack the quantization levels

We first tampered with either the high quantization level or the low quantization level of the 120 × 120 AMBTC compression codes, illustrated in Figure 10a. The modification is so natural that it is impossible to identify visually. Figure 10e shows the detection result obtained after the first hierarchical tampering detection, with a *TDR* of 100%. The corresponding experimental results appear in Table 7.

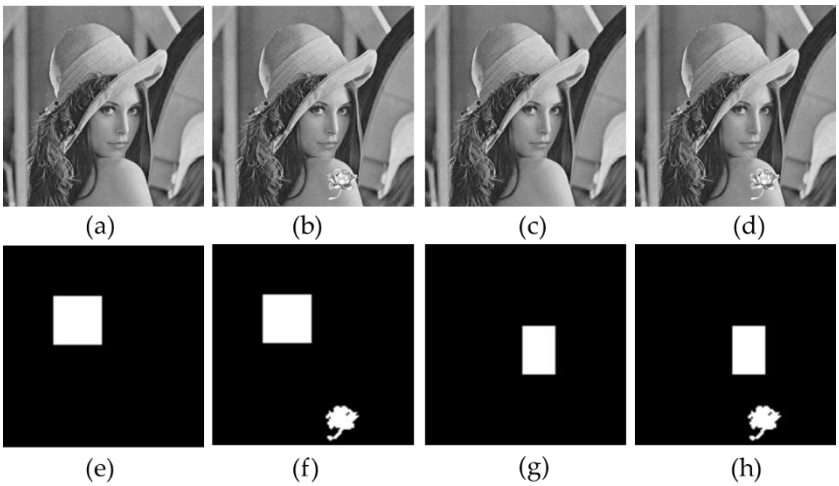

Figure 10. AMBTC compression codes' attacks: Tampered Lena images by kinds of attacks: (**a**) *H* or *L*; (**b**) (*H* or *L*) and flower; (**c**) *BM*; (**d**) *BM* and flower. Tampering detection results after the first hierarchy for kinds of attacks: (**e**) *TDR* = 100%; (**f**) *TDR* = 99.42%; (**g**) *TDR* = 100%; (**h**) *TDR* = 99.22%.

(b) Attack the quantization levels with a flower

Secondly, based on the experiment of attacking the quantization levels, at the same time, we tried to copy and paste a flower onto Lena's shoulder, as shown in Figure 10b. It is not easy to detect the attack under this scenario. The corresponding tampering detection result using the first hierarchical tampering detection method is shown in Figure 10f, with a *TDR* of 99.42%. Other statistical parameters are listed in the fourth row of Table 7.

Table 7. Tampering detection performance of the AMBTC compression codes' attack.

Original image	Kinds of Attacks	Number of Tampered Blocks (4 × 4)			The First Hierarchical Tampering Detection Results (%)			The Second Hierarchical Tampering Detection Results (%)		
		Total	First Hierarchy	Second Hierarchy	*TDR*	*FPR*	*FNR*	*TDR*	*FPR*	*FNR*
Lena	(*H* or *L*)	961	961	961	100	0	0	100	0	0
	(*H* or *L*) and flower	1211	1204	1208	99.42	0	0.0461	99.75	0.0827	0.0198
	BM	651	651	651	100	0	0	100	0	0
	BM and flower	901	894	898	99.22	0	0.0452	99.67	0.1112	0.0194
Average					99.66	0	0.0228	99.86	0.0485	0.0098

(c) Attack the bitmap

We also simply tampered with the bitmap of the 120 × 80 AMBTC compression codes, shown in Figure 10c. Again, it is hard to notice any change to the Lena image. The tampered area is around Lena's face, and Figure 10g shows the tampering detection result after the first hierarchical tampering detection, with a *TDR* of 100%. Some experimental results are listed in the fifth row of Table 7.

(d) Attack the bitmap and copy/paste a flower

We now tampered with the Lena image by modifying the bitmap and adding a flower near her shoulder, as illustrated in Figure 10d. The tampering detection result delivered by our approach is shown in Figure 10h, with a *TDR* of 99.22%. In Table 7, the sixth row shows the detailed detection results, with lower *FPR* and *FNR*.

In summary, we implemented multiple experiments to test the performance of our proposed tampering detection scheme, including the cropping attack, constant average attack, collage attack, and AMBTC compression codes' attacks. After the first hierarchical tampering detection, our proposed scheme can offer a high *TDR* along with low *FPR* and *FNR* for all kinds of attacks.

4.3. Performance Comparisons

Table 8 compares the *PSNRs* of the reconstructed images by using the original AMBTC compression codes and watermarked AMBTC compression codes generated through our proposed scheme. The watermarked images have a few more distortions than the image reconstructed using the original AMBTC compression codes. The average *PSNR* of the watermarked image using our approach is around 31.66 dB; in other words, we sustained acceptable visual quality while achieving highly accurate tampering detection.

Table 8. *PSNRs* of the reconstructed image by the different schemes with block size 4 × 4.

Original Images	*PSNRs* (dB)	
	AMBTC	Proposed Scheme
Couple	31.27	30.38
Boat	31.87	30.89
Zelda	37.99	35.07
Lena	33.23	31.99
Woman	37.98	35.03
Elaine	33.91	32.41
Baboon	28.30	27.78
Lake	29.88	29.21
Peppers	33.43	32.15
Average	33.10	31.66

Table 9 compares the tampering detection performance among three existing schemes and our proposed scheme. The upper bounds of the number of authentication bits in Lin et al.'s scheme [29], Hong et al.'s scheme [30], and the proposed scheme are four, four, and six, respectively. Thus, the proposed scheme obtains higher tampering detection accuracy than the other two with the assistance of one six-bit authentication code per block. The experimental results provide the expected effect; that is, the average *TDR* of our proposed scheme is about 98.55% in the first hierarchical tampering detection procedure, which is about 4.8% higher than those of Lin et al.'s scheme [29] and Hong et al.'s scheme [30]. In other words, our approach can offer a more stable and reliable tampering detection performance. In Lin et al.'s scheme [29] and Hong et al.'s scheme [30], the stego-images' quality with the four-bit authentication code has average *PSNRs* of 33.07 and 32.33 dB, respectively. Also, we can observe that Hong et al.'s scheme [31] proposes two embedding strategies that can embed an eight-bit authentication code into each block, resulting in a *TDR* of 99.61% and average *PSNRs* of 28.92 and 29.84 dB. Meanwhile, with the same size of the authentication code, i.e., six bits, our proposed scheme and the Hong et al. scheme [31] have the same effect with respect to the *TDR*. The average *PSNR* of our approach is 0.39 dB higher than that of the LSBP scheme and 0.05 dB lower than that of the MSBP scheme. Hong et al.'s scheme tries at least six possible combinations to find the minimal distortion during authentication code embedding with MSBP. Therefore, it can be concluded they maintain image quality at the cost of execution efficiency. Also, a large proportion of image authentication methods employ multi-hierarchical tampering detection strategies as a remedial measure to improve their *TDR*. In Table 9, we report the detection performance offered by Lin et al.'s scheme [29], Hong et al.'s scheme [30,31], and the proposed scheme. The four schemes had very similar tampering detection rates after the second hierarchical remediation measure. However, our proposed scheme outperforms the others when considering tampering detection performance and the image quality of the watermarked images from the first hierarchical tampering detection.

Table 9. Tampering detection performance of different schemes.

Methods	Number of Authentication Bits	Hierarchy of Detection Strategies	*TDR* of First Hierarchy (%)	*TDR* of Multi-Hierarchy (%)	Average *PSNRs* (dB)
Lin et al. [29]	4	2	93.75	98.19	33.07
Hong et al. [30]	4	2	93.75	99.83	32.33
Hong et al. [31] (LSBP)/(MSBP)	6	2	98.50	99.66	31.27/31.73
	8		99.61		28.92/29.84
Proposed scheme	6	2	98.55	99.85	31.66

Table 10 summarizes comparisons between the proposed scheme and other work [29–31]. It can be observed that Lin et al.'s scheme generates an authentication code by using a pseudo-random generator, while Hong et al.'s scheme [30] and the proposed scheme do so by hashing the bitmap and location information, and Hong et al.'s scheme [31] generates the authentication by hashing the bitmap and quantization levels' MSBs. Lin et al.'s scheme [29] can resist the cropping attack, constant attack, and collage attack, but is weak against tampering with the bitmap of the complex blocks because it only embeds the authentication code into the quantization levels. The two schemes proposed by Hong et al. [30,31] are sensitive to AMBTC compression codes' attacks since they embed the authentication code generated to refer to the bitmap into the quantization levels. However, in some cases, the natural relationship between two quantization levels is broken for some blocks, making it only possible to verify the authenticity of the AMBTC compression codes and not that of the AMBTC-compressed image. Hence, in their schemes, it seems likely that only professional technicians will be able to implement the cropping attack, constant average attack, or collage attack.

Also, Hong et al.'s scheme [31] is poor against the collage attack. In our approach, we embed the authentication code in the bitmap in a virtual manner and insert the bit-location information into the quantization levels later. This strategy maintains effective interlocking among the authentication code, the quantization levels, and the bitmap. If any slight modifications are encountered in the quantization levels or bitmap, the correlation between them will be broken. Thus, the proposed scheme can not only resist AMBTC compression codes' attacks but also detect the cropping attack, constant average attack, and collage attack. Undoubtedly, the experimental results show that our approach provides more stable and reliable tampering detection performance by simply conducting first hierarchical tampering detection and also sustains acceptable visual quality.

Table 10. Comparisons with various schemes.

Compared Lists	**Lin et al.'s Scheme [29]**	**Hong et al.'s Scheme [30]**	**Hong et al.'s Scheme [31]**	**Proposed Scheme**
Components to embed *AC*	Quantization levels or bitmap	Quantization levels	Quantization levels	Quantization levels and bitmap
Generation of *AC*	Pseudo-random generator	Hash function	Hash function	Hash function
Detection of the special modification of bitmap	No	Yes	Yes	Yes
Detection of the special modification of quantization levels	No	Yes	Yes	Yes
Detection of the cropping attack	Yes	Yes	Yes	Yes
Detection of the constant average attack	Yes	Yes	Yes	Yes
Detection of the collage attack	Yes	Yes	No	Yes
Authentication for AMBTC compression codes	Yes	Yes	Yes	Yes
Authentication for AMBTC compressed image	Yes	No	No	Yes

Table 11 summarizes comparisons among the proposed scheme and the authentication methods for compressed images using other compression techniques. Herein, the smaller the detectable block size is, the precision the detecting unit is, and the larger the size of AC is, the more tampering detection accuracy it has. We can observe that the JPEG-based authentication schemes [17,18] embed the AC in the frequency domain. Although their schemes provide a better visual quality of the watermarked image, they have a weakness in their tampering detection accuracy because the size of the AC they embedded is one bit and three bits, respectively. It is also apparent that the schemes [18,19] provide a detectable block size of 8×8 pixels, which means that a lower precision of the detecting unit they are in. Besides, we also can find that our approach and scheme [19] work on the spatial domain. Our approach provides a better performance in tampering detection and image quality than that of the VQ-based scheme [19]. Moreover, the tamper detection performance of our proposed scheme is significantly better than the existing three schemes. Besides, we also can find that our approach and scheme [19] work on the spatial domain. Our approach provides a better performance in tampering detection and image quality than that of the VQ-based scheme [19].

Table 11. Comparisons with other schemes based on different compression techniques.

Methods	Compression Methodology	Domain	Detectable Block Size	Length of AC for a Block	*PSNR* (dB)
Scheme in [17]	JPEG	Frequency	4×4	1	[40.33, 44.12]
Scheme in [18]	JPEG	Frequency	8×8	3	44.63
Scheme in [19]	VQ	Space	4×4, 8×8	[1, 3]	≈[29.00, 31.50]
Proposed scheme	AMBTC	Space	4×4	[1, 6]	[31.66, 33.10]

5. Conclusions

This paper proposed a novel precise image authentication scheme to protect the integrity of AMBTC-compressed images. The authentication code is generated by hashing the processed bitmap and block's location information. For each block, a six-bit authentication code is virtually inserted into the bitmap using matrix encoding. Instead of changing the bitmap, we only recorded the to-be-flipped bit-location information without modifying the bitmap in each block. This bit-location information is embedded into two quantization levels based on adjusted quantization levels matching. Experiments were performed to evaluate the performance of our approach. The results showed that our approach provides more stable and reliable tampering detection performance than previous work and sustains acceptable visual quality. In the future, we will try to use the bitmap adequately to provide strong tampering detection performance while improving image quality. Moreover, we will also pay more attention to the authentication schemes for compression codes.

Author Contributions: C.-C.C. conceived and designed the experiments; G.-D.S. performed the experiments and wrote the paper; and C.-C.L. analyzed the data.

Funding: This work was supported by the Education-Scientific research Project for Middle-aged and Young of Fujian Province under Grant No. JAT160574 and JT180621, and Ministry of Science and Technology: MOS 108-2410-H-126 -021.

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

References

1. Schneider, M.; Chang, S.F. A robust content based digital signature for image authentication. In Proceedings of the 3rd IEEE International Conference on Image Processing, Lausanne, Switzerland, 19 September 1996; pp. 227–230.
2. Tabatabaei, S.A.H.; Ur-Rehman, O.; Zivic, N.; Ruland, C. Secure and robust two-phase image authentication. *IEEE Trans. Multimed.* **2015**, *17*, 945–956. [CrossRef]
3. Yan, C.P.; Pun, C.M. Multi-scale difference map fusion for tamper localization using binary ranking hashing. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2144–2158. [CrossRef]
4. Chen, Z.; Li, L.; Peng, H.; Liu, Y.; Yang, Y. A novel digital watermarking based on general non-negative matrix factorization. *IEEE Trans. Multimed.* **2018**, *20*, 1973–1986. [CrossRef]
5. Walton, S. Image authentication for a slippery new age. *Dr. Dobb's J.* **1995**, *20*, 18–26.
6. 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] [PubMed]
7. Haouzia, A.; Noumeir, R. Methods for image authentication: A survey. *Multimed. Tools Appl.* **2008**, *39*, 1–46. [CrossRef]
8. Ozyurt, F.; Tuncer, T.; Avci, E. A novel probabilistic image authentication method based on universal hash function for RGB images. In Proceedings of the 2018 International Conference on Computing Sciences and Engineering (ICCSE), Kuwait City, Kuwait, 11–13 March 2018; pp. 1–6.
9. Holliman, M.; Memon, N. Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes. *IEEE Trans. Image Process.* **2000**, *9*, 432–441. [CrossRef] [PubMed]
10. Zhang, X.; Wang, S.; Qian, Z.; Feng, G. Reference sharing mechanism for watermark self-embedding. *IEEE Trans. Image Process.* **2011**, *20*, 485–495. [CrossRef] [PubMed]

11. Zhang, X.; Xiao, Y.; Zhao, Z. Self-embedding fragile watermarking based on DCT and fast fractal coding. *Multimed. Tools Appl.* **2015**, *74*, 5767–5786. [CrossRef]
12. Chang, C.C.; Lu, T.C.; Zhu, Z.H.; Tian, H. An effective authentication scheme using DCT for mobile devices. *Symmetry* **2018**, *10*, 13. [CrossRef]
13. Dadkhah, S.; Manaf, A.A.; Hori, Y.; Hsssanien, A.E.; Sadeghi, S. An effective SVD-based image tampering detection and self-recovery using active watermarking. *Signal Process. Image Commun.* **2014**, *29*, 1197–1210. [CrossRef]
14. Shehab, A.; Elhoseny, M.; Muhammad, K.; Sangaiah, A.K.; Yang, P.; Huang, H.; Hou, G. Secure and robust fragile watermarking scheme for medical images. *IEEE Access* **2018**, *6*, 10269–10278. [CrossRef]
15. Qin, C.; Ji, P.; Wang, J.; Chang, C.C. Fragile image watermarking scheme based on VQ index sharing and self-embedding. *Multimed. Tools Appl.* **2017**, *76*, 2267–2287. [CrossRef]
16. Fan, M.; Wang, H. An enhanced fragile watermarking scheme to digital image protection and self-recovery. *Signal Process. Image Commun.* **2018**, *66*, 19–29. [CrossRef]
17. Qi, X.; 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]
18. Preda, R.O.; Vizireanu, D.N. Watermarking-based image authentication robust to JPEG compression. *Electron. Lett.* **2015**, *51*, 1873–1875. [CrossRef]
19. Chuang, J.C.; Hu, Y.C. An adaptive image authentication scheme for vector quantization compressed image. *J. Vis. Commun. Image Represent.* **2011**, *22*, 440–449. [CrossRef]
20. Tu, S.F.; Hsu, C.S. A BTC-based watermarking scheme for digital images. *Int. J. Inf. Secur.* **2004**, *15*, 216–228. [CrossRef]
21. Jiang, M.F.; Zhu, N.B. Image fragile watermarking algorithm based on BTC domain. *Sci. Technol. Eng.* **2009**, *9*, 717–720.
22. Yang, C.N.; Lu, Z.M. A Blind image watermarking scheme utilizing BTC bitplanes. *Int. J. Digi. Crime Forensics* **2011**, *3*, 42–53. [CrossRef]
23. Hu, Y.C.; Lo, C.C.; Chen, W.L. Joint image coding and image authentication based on absolute moment block truncation coding. *J. Electron. Imaging* **2013**, *22*, 013012. [CrossRef]
24. Yang, H.; Kot, A.C. Pattern-based data hiding for binary image authentication by connectivity-preserving. *IEEE Trans. Multimed.* **2007**, *9*, 475–486. [CrossRef]
25. Hu, Y.C.; Lo, C.C.; Wu, C.M.; Chen, W.L.; Wen, C.H. Probability-based tamper detection scheme for BTC-compressed images based on quantization levels modification. *Int. J. Secur. Its Appl.* **2013**, *7*, 11–32.
26. Nguyen, T.S.; Chang, C.C.; Chung, T.F.A. Tamper-detection scheme for BTC-compressed images with high-quality images. *KSII Trans. Internet Inf. Syst.* **2014**, *8*, 2005–2021.
27. Lin, C.C.; Huang, Y.; Tai, W.L. A high-quality image authentication scheme for AMBTC-compressed images. *KSII Trans. Internet Inf. Syst.* **2014**, *8*, 4588–4603.
28. Li, W.; Lin, C.C.; Pan, J.S. Novel image authentication scheme with fine image quality for BTC-based compressed images. *Multimed. Tools Appl.* **2016**, *75*, 4771–4793. [CrossRef]
29. Lin, C.C.; Huang, Y.; Tai, W.L. A novel hybrid image authentication scheme based on absolute moment block truncation coding. *Multimed. Tools Appl.* **2017**, *76*, 463–488. [CrossRef]
30. Hong, W.; Chen, M.; Chen, T.S.; Huang, C.C. An efficient authentication method for AMBTC compressed images using adaptive pixel pair matching. *Multimed. Tools Appl.* **2018**, *77*, 4677–4695. [CrossRef]
31. Hong, W.; Zhou, X.; Lou, D.C.; Huang, X.; Peng, C. Detectability improved tamper detection scheme for absolute moment block truncation coding compressed images. *Symmetry* **2018**, *10*, 318. [CrossRef]
32. Nasrabadi, N.M.; King, R.A. Image coding using vector quantization: A review. *IEEE Trans. Commun.* **1988**, *36*, 957–971. [CrossRef]
33. Lema, M.; Mitchell, R. Absolute moment block truncation coding and its application to color images. *IEEE Trans. Commun.* **1984**, *32*, 1148. [CrossRef]
34. Delp, E.J.; Mitchell, O.R. Image compression using block truncation coding. *IEEE Trans. Commun.* **1979**, *27*, 1335. [CrossRef]
35. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
36. Nguyen, D.V.; Vasic, B. Two-bit bit flipping algorithms for LDPC codes and collective error correction. *IEEE Trans. Commun.* **2014**, *62*, 1153–1163. [CrossRef]
37. Hamming, R.W. Error Detecting and Error Correcting Codes. *Bell Syst. Tech. J.* **1950**, *29*, 147–160. [CrossRef]

38. Chen, K.; Chang, C.C. Real-time error-free reversible data hiding in encrypted images using (7, 4) hamming code and most significant bit prediction. *Symmetry* **2019**, *11*, 51. [CrossRef]
39. Liu, S.; Fu, Z.; Yu, B. Rich QR codes with three-layer information using hamming code. *IEEE Access* **2019**, *7*, 78640–78651. [CrossRef]