International Journal of Safety and Security Engineering
Vol. 14, No. 6, December, 2024, pp. 1871-1881
Journal homepage: http://iieta.org/journals/ijsse

Image Tamper Detection and Correction Using Modified Hamming Code with Eight Data Bits and Four Redundant Bits

Anantha Rao Gottimukkala[1], Anita Pradhan[1], Pramoda Patro[2], J. Hemalatha[3], Ranjan K. Senapati[4], Gandharba Swain[1*]

[1] Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram 522302, India
[2] School of Computer Science and Artificial Intelligence, SR University, Warangal 506371, India
[3] Department of CSE, AAA College of Engineering and Technology, Amathur, Sivakasi 626123, India
[4] Department of ECE, VNR Vignana Jyothi Institute of Engineering & Technology, Bachupally, Nizampet (S.O), Hyderabad 500090, India

Corresponding Author Email: gandharba.swain@kluniversity.in

Copyright: ©2024 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ijsse.140621

Received: 11 October 2024
Revised: 10 December 2024
Accepted: 19 December 2024
Available online: 31 December 2024

Keywords:
data hiding, watermarking, tamper detection, Hamming code, logistic map

ABSTRACT

Tampering an image became more manageable due to the advent of photo editing software. So, it is essential to detect and correct the tampered locations. This article presents an image watermarking scheme that applies a modified version of 8 data bits and 4 redundant bits of Hamming code. It operates over a block of 3 pixels at a time. The 24 bits of the 3 pixels are rearranged and divided into 2 processing units, each comprising 12 bits. Out of 12 bits in a unit, 4 bits are used as redundant bits, and 8 bits are used as data bits. Watermark bits (WBs) are generated from data bits using a logistic map (LM) and stored in redundant bit positions. Experimental results reveal that this proposed technique maintains a fair balance for both capacity and distortion. The measured capacity is 2.67 bits per pixel (bpp) with a 39.21 decibels (dB) distortion. The estimated structural similarity (SSIM) is 0.9844, and the accuracy is 0.9999. Furthermore, it was also found that the proposed technique performs fairly well compared to existing techniques in capacity and distortion.

1. INTRODUCTION

Tampering a watermarked image (WI) is very easy by using photo editing tools. So, accurate tamper detection and correction mechanisms play a vital role in this field of study. Image watermarking can be done in the spatial domain and transform domain. In the spatial domain, the watermarking techniques use block-by-block or pixel-by-pixel for watermark hiding. To ensure the safety of the embedded watermark, we can use a chaotic map (CM) or logistic map (LM). Different kinds of error correction codes can be used in data hiding techniques for error detection and correction [1-3]. Hash functions can also be used to improve attack resistance [4].

Gul and Ozturk [5] proposed a triple data embedding approach for watermarking. Here, the recovery bits (RBs) are created based on the partner block concept with 4×4 size and embedded in the 2 least significant bits (LSBs). Sinhal et al. [6] arranged the color image pixels into 2×4 blocks of color components. A block has 8 bytes. From the most significant bits (MSBs) of the color components, watermark bits (WBs) are generated and stored in LSBs applying a base-9 numbering system. Authors claim they could detect 99% of tampering and correct 80% of it successfully. Qin et al. [7] brought a tamper detection and localization scheme using 3×3 blocks in an overlapped manner. In this technique, WBs and RBs are brought up from the 6 MSBs of the mean value. The WBs replace central pixel LSBs, and the RBs replace the remaining pixel LSBs. This technique effectively recovers the tampered locations.

To improve security, the WBs can be scrambled using CM and LM. Rawat and Raman [8] used Arnold's map for this. In this approach, WBs and bits from the map are made exclusive-or (XORed) and then stored in LSBs. Botta et al. [9] argued that Rawat and Raman could not localize the tampered places. This issue is solved by using 7 MSBs to create the WBs. The LSBs are substituted by the WBs. Prasad and Pal [10] plied LM with shift and mod operators. Here, WBs are computed using 5 MSBs, which are camouflaged plying mod and shift operators. Tampered zones could be localized very effectively in this approach. Sahu [11] coupled LM and XOR functions to bury the WBs. The collage attack, text addition attack, cropping attack, and copy-paste attack cannot affect the watermark in Sahu's technique. Nazari et al. [12] considered 2×2 pixels as a block and applied CM for watermarking. Here, WBs are generated from higher planes and buried in lower planes. The watermark is protected because of CM. Shehab et al. [13] devised a watermarking approach applying singular value factorization and LSB substitution with 4×4 blocks. Used Arnold transformation to decide the insertion of RBs. The authentication bits are plied to protect from vector quantization threats. This technique improves the peak signal-

to-noise ratio (PSNR) and tamper localization accuracy.

Kosuru et al. [14] considered 2×2 pixel blocks and computed quotients and remainders from these 4 pixels of a block. The quotients are treated as leaf nodes of the Merkle tree and the root node is computed from the leaf nodes. The root node is converted to binary and treated as WBs. Gull et al. [15] also used the block concept for tamper localization. For a 4×4 pixel block, the WBs are generated from 4 MSBs of these 16 pixels. Furthermore, RBs are computed from the pixel value average in the block. Here, the payload is only 1 bpp with PSNR 51.94 dB. Laishram and Sing [16] proposed a multipurpose watermarking approach based on the zone of interest in frequency coefficients. It can achieve source authentication, confidentiality, integrity check, ownership identification, and tamper correction. It has built-in recovery against various attacks like scaling, translation, rotation, etc. This technique achieved a PSNR value of 53.17 dB, but the hiding capacity (HC) is very poor.

Rinki et al. [17] used a non-consecutive matrix of pixels and a matrix multiplication process to replace the 3 LSBs of the channels of a color image with the bits of a secret grey image. It achieves content integrity verification and copyright protection. Chen et al. [18] brought a data-hiding technique using Huffman code. Using this code, the MSB plane (4 MSBs) is compressed, and LSBs are buried in vacant MSBs. Thus, the room is created in LSBs to embed the data. The Huffman code performs compression by computing the difference among the higher nibbles. High embedding capacity is achieved. Jana et al. [19] performed watermarking over 4×4 blocks using a similarity matrix. A block may be categorized into one of 2 types: (i) smooth and (ii) complex. WBs are created from 8 MSBs. If the block is in the smooth category, then the block mean is used as RBs. If the block is a complex category, then fuzzy logic and imaging concept is utilized to generate RBs. WBs and RBs are buried in lower-bit planes. Pal et al. [20] applied interpolation to expand a 2×2 block into a 4×4 size. Used local binary pattern and Hamming code to embed the WBs. The error detection is possible at the receiver to some extent. It possesses high embedding capacity.

Patsariya and Dixit [21] proposed watermarking for copyright protection and authenticity using the CM for improved security. They achieved security in addition to robustness, but the resultant correlation values are not appreciable. Chennamma et al. [22] proposed authentication for medical images based on statistical correlations. This approach divides the image into 4 by 4 blocks and in each block estimates statistical correlations. The correlations serve as watermarks and are embedded in lower bit planes. The experimental readings like PSNR and SSIM are all right.

A majority of the articles detect the tamper locations effectively, but are not able to correct those locations accurately. This is a very crucial research issue in image authentication.

2. REVIEW OF RELATED RESEARCH WORK AND AUTHORS' CONTRIBUTIONS

2.1 Review of related work

Figure 1 depicts the basic (7, 4) Hamming code [23]. The redundant bit r_1 is derived by doing an XOR operation over bits f_1, f_2, and f_4. Similarly, r_2 is derived by doing an XOR operation over bits f_1, f_3, and f_4. Furthermore, r_3 is derived by doing XOR operation over bits f_2, f_3, and f_4. Chang et al. [24] plied Hamming code along with LM to generate the WBs from 4 MSBs. The LSBs are replaced by these WBs. They claimed that the tampered pixels are correctly localized. The application of LM protects from different attacks. The security has been increased by the use of LM. The recorded value of PSNR is acceptable, and this value is 37 dB. Wang et al. [25] did watermarking using Hamming code for object verification. In this technique, the watermark has been created using Hamming code and embedded in LSBs. The integrity of the watermark is verified by Hamming code. The original model is not necessary for integrity verification and tamper correction. It is simple and achieves lower distortion. Medical images are very important objects. While transmitted over the internet, errors may occur due to various reasons. Islam et al. [26] did watermarking for medical images plying (8, 4) Hamming code. This method is implemented in an independent processor to be useful for real-life scenarios. Trivedy and Pal [27] used LM with Hamming weight to make the WBs, and these bits are concealed in pixels either by altering their value by ±1 or by keeping their value the same. If a bit of watermark is 1, then the pixel value is altered by ±1; otherwise, the pixel value is not altered. The tamper detection is possible. Hamming code has been used to identify the positions where the WBs can be hidden. This technique possesses very little HC.

Prasad and Pal [28] did watermarking using pixel value differencing (PVD) and Hamming code. In their scheme, a block (P_1, P_2) is considered. The 4 MSBs (2 from each pixel) are the data bits; 3 redundant bits are derived as WBs, which are then hidden in the lower bit plane by the PVD mechanism. It eradicates the pixel boundary problem and gives a higher PSNR value and higher SSIM. Prasad and Pal [29] also detected tampering at the pixel level plying LM and Hamming code. Using a generator matrix. Derived WBs from 4 MSBs. This matrix creates 7 bits, out of which the 3 rightmost bits are picked up as WBs and concealed at 3 LSBs. The PVD concept is applied to hide the 3 WBs in LSBs. Security is surely enhanced when LM is used. Jana et al. [30] also used basic Hamming code for watermarking and error correction using LSBs of 7 pixels. Nguyen and Le [31] performed data hiding over 5 pixels using (5,3) Hamming code. This code is used to detect possible places where the message bits can be buried. Although the HC is only 12.2 bpp, the security is high because the embedding positions are the key positions to secure the embedded data. Ramos et al. [1] used an error-correcting code to develop a fragile watermarking scheme, wherein the WBs are created from redundant bits and hidden in frequency bands. The primary objectives of data hiding techniques are higher HC, lower distortion, and higher security. Khadse and Swain [2] used quotient value differencing to hide data. They generated the authentication bits from MSBs using Merkle tree root and stored them in LSBs. The authentication bits can help detect errors. Kosuru et.al. [3] also used the principle of quotient value differencing for data hiding. They used the Hamming code to create the parity bits from MSBs and store them in LSBs. The error detection and correction happened accurately.

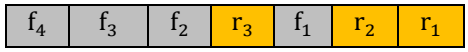


Figure 1. The (7, 4) Hamming code

2.2 Research contribution

There exists a good number of watermarking techniques for tamper detection, but very few of them provide perfect tamper correction. So, it is also essential and interesting work to detect the tampered pixels accurately and correct them perfectly. This article proposes a watermarking technique using a modified Hamming code version. The contribution is briefed below:

- The (12, 8) Hamming code is based on Hamming code (7, 4). Figure 2 can be referred to now. The 4 parity bits (P_4, P_3, P_2, and P_1) are computed from the 8 data bits. To use this in image watermarking, we have modified it by repositioning the redundant bit positions towards the LSB side and altering their computations appropriately.
- Three pixels are taken and divided into 2 units in such a way that the LSBs are used as redundant bits. The computation of redundant bits is modified as compared to the original way in Hamming code.
- The reorganization of redundant bits is performed so that the distortion will be minimal after storing the WBs in redundant bit positions. The modified Hamming code is described in Section 3.2.

Furthermore, we apply the LM sequence with the redundant bits to improve security and generate the WBs. The details of the computation are described in Section 3.1.

f_8	f_7	f_6	f_5	P_4	f_4	f_3	f_2	P_3	f_1	P_2	P_1

Figure 2. The (12, 8) Hamming code

3. PROPOSED TECHNIQUE: MODIFIED HAMMING CODE BASED WATERMARKING (MHCBW)

Section 3.1 describes LM's generation of a random sequence of bits. Section 3.2 explains the generation of WBs and their hiding in the image. Section 3.3 explains the extraction of WBs, localization of tampered pixels, and correction of tampered pixels.

3.1 Used logistic map

The LM generates random values using 2 initial seeds α_0 and γ. These seed values are constrained as follows: $0 \leq \alpha_0 \leq 1$ and $0 < \gamma \leq 4$. Our initial value α_0 is 0.0396, and γ is 3.58. The subsequent values α_i can be computed by Eq. (1).

$$\alpha_i = \gamma \times (1-\alpha_{i-1}) \times \alpha_{i-1} \tag{1}$$

The LM can generate perfect random values if we take $3.57 < \gamma \leq 4$. After obtaining α_i value, we multiply it by 255 to derive another value X_i. This X_i value is rounded up to get Y_i. Eq. (2) depicts this computation. The integer value $Y_i < 255$. Thereafter, representing Y_i value in 8-bit binary, we get S_i. The 4 MSBs of S_i are termed as K and 4 LSBs of S_i as T. So, $K = k_4\ k_3\ k_2\ k_1$ and $T = t_4\ t_3\ t_2\ t_1$.

$$X_i = \alpha_i \times 255 \text{ and } Y_i = Round\ (X_i) \tag{2}$$

3.2 Watermark embedding

The cover image is virtually split into 1×3 size disjoint blocks. Suppose (P_1, P_2, P_3) is a sample block. The various bits of these 3 pixels are designated in Figure 3. The watermark hiding in this sample block is illustrated in the below steps.

---------- $\mathbf{P_1}$ ----------							
f_8	f_7	f_6	f_5	f_4	r_4	r_3	r_2
---------- $\mathbf{P_2}$ ----------							
f_3	f_2	f_1	q_8	q_7	r_1	v_4	v_3
---------- $\mathbf{P_3}$ ----------							
q_6	q_5	q_4	q_3	q_2	q_1	v_2	v_1

Figure 3. Original 1×3 block

Step 1: Partition the 24 bits of the block into 2 units, U_1 and U_2, as shown in Figure 4.

---------- $\mathbf{U_1}$ ----------											
f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	r_4	r_3	r_2	r_1
---------- $\mathbf{U_2}$ ----------											
q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	v_4	v_3	v_2	v_1

Figure 4. Two data units, each 12-bit

Step 2: For U_1 compute r_1', r_2', r_3' and r_4' using Eq. (3) and Eq. (4) respectively.

$$r_1' = f_7 \oplus f_5 \oplus f_4 \oplus f_2 \oplus f_1, \quad r_2' = f_7 \oplus f_6 \oplus f_4 \oplus f_3 \oplus f_1 \tag{3}$$

$$r_3' = f_8 \oplus f_4 \oplus f_3 \oplus f_2, \quad r_4' = f_8 \oplus f_7 \oplus f_6 \oplus f_5 \tag{4}$$

Step 3: For U_2 compute v_1', v_2', v_3' and v_4' using Eq. (5) and Eq. (6) respectively.

$$v_1' = q_7 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1, \quad v_2' = q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_1 \tag{5}$$

$$v_3' = q_8 \oplus q_4 \oplus q_3 \oplus q_2, \quad v_4' = q_8 \oplus q_7 \oplus q_6 \oplus q_5 \tag{6}$$

Step 4: Accept the next 8 bits from the LM generator. Say it is S_i, which is depicted in Eq. (7).

$$S_i = KT, \text{ where } K = k_4\ k_3\ k_2\ k_1 \text{ and } T = t_4\ t_3\ t_2\ t_1 \tag{7}$$

Step 5: Compute the WBs r_4'', r_3'', r_2'', r_1'' for U_1 and v_4'', v_3'', v_2'', v_1'' for U_2 as in Eq. (8) and Eq. (9) respectively.

$$r_i'' = r_i' \oplus k_i, \text{ for } i=1,2,3, \text{ and } 4 \tag{8}$$

$$v_i'' = v_i' \oplus t_i, \text{ for } i=1,2,3, \text{ and } 4 \tag{9}$$

---------- $\mathbf{P_1'}$ ----------							
f_8	f_7	f_6	f_5	f_4	r_4''	r_3''	r_2''
---------- $\mathbf{P_2'}$ ----------							
f_3	f_2	f_1	q_8	q_7	r_1''	v_4''	v_3''
---------- $\mathbf{P_3'}$ ----------							
q_6	q_5	q_4	q_3	q_2	q_1	v_2''	v_1''

Figure 5. The watermarked block

Step 6: To form the watermarked block, replace each r_i by r_i'' in Figure 4, for i=1, 2, 3 and 4. Similarly, replace each v_i by v_i'' in Figure 4, for i=1, 2, 3, and 4. The resultant block after watermarking is shown in Figure 5. It is denoted as (P_1', P_2', P_3'). Diagrammatically, the embedding procedure is shown in Figure 6.

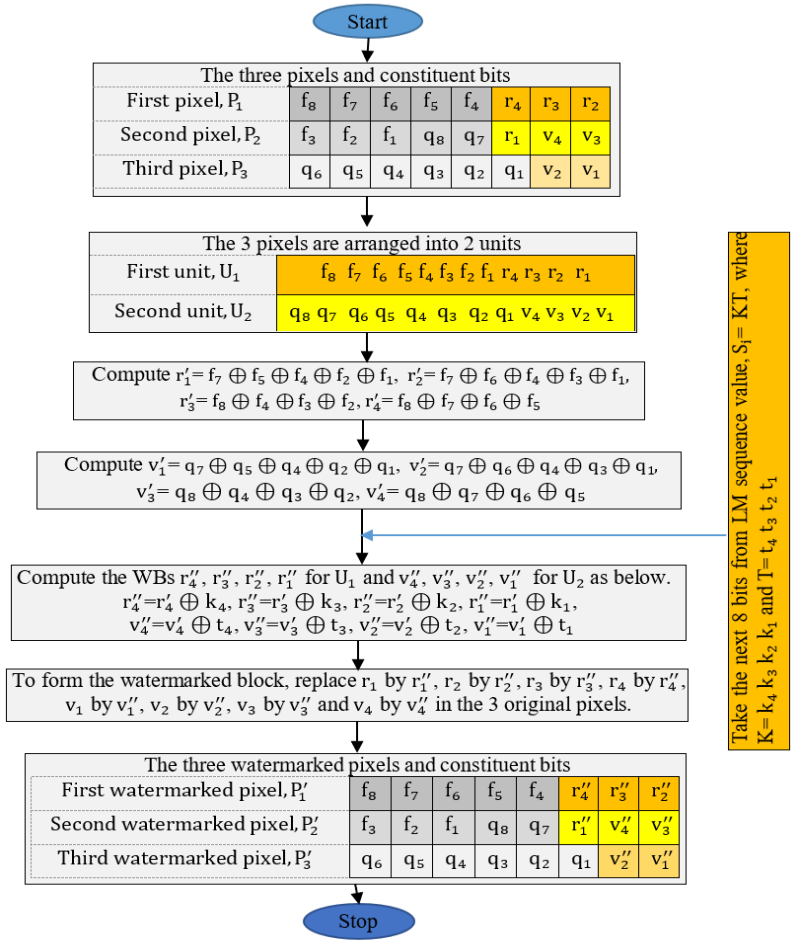


Figure 6. Watermark embedding

3.3 Retrieval of watermark with tamper correction

Divide the WI into 1×3 size blocks. Figure 5 represents a sample block. The extraction of WBs is narrated in the below steps.

Step 1: Partition the block (P_1', P_2', P_3') into 2 units U_1' and U_2' as shown in Figure 7.

Step 2: Take the next 8-bit LM sequence value, S_i and represent it as S_i=KT, where $K=k4\ k_3\ k_2\ k_1$ and $T=t4\ t_3\ t_2\ t_1$. Compute r_4', r_3', r_2', r_1' and v_4', v_3', v_2', v_1' using Eq. (10) and Eq. (11) respectively.

$$r_i'=r_i'' \oplus k_i, \text{ for i=1,2,3, and 4} \tag{10}$$

$$v_i'=v_i'' \oplus t_i, \textit{ for i=1,2,3, and 4} \tag{11}$$

Step 3: Compute r_1^*, r_2^* using Eq. (12) and r_3^*, r_4^* using Eq. (13).

$$\begin{aligned} r_1^* &= r_1' \oplus f_7 \oplus f_5 \oplus f_4 \oplus f_2 \oplus f_1, \\ r_2^* &= r_2' \oplus f_7 \oplus f_6 \oplus f_4 \oplus f_3 \oplus f_1 \end{aligned} \tag{12}$$

$$\begin{aligned} r_3^* &= r_3' \oplus f_8 \oplus f_4 \oplus f_3 \oplus f_2, \\ r_4^* &= r_4' \oplus f_8 \oplus f_7 \oplus f_6 \oplus f_5 \end{aligned} \tag{13}$$

Step 4: Compute v_1^*, v_2^* , v_3^*, v_4^* using Eq. (14) and Eq. (15).

$$\begin{aligned} v_1^* &= v_1' \oplus q_7 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1, \\ v_2^* &= v_2' \oplus q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_1 \end{aligned} \tag{14}$$

$$\begin{aligned} v_3^* &= v_3' \oplus q_8 \oplus q_4 \oplus q_3 \oplus q_2, \\ v_4^* &= v_4' \oplus q_8 \oplus q_7 \oplus q_6 \oplus q_5 \end{aligned} \tag{15}$$

Step 5: If $r_4^* r_3^* r_2^* r_1^* \neq 0000$, then there is tampering in U_1'. The correction is done as shown in Table 1. If $r_4^* r_3^* r_2^* r_1^* = 0000$. Then there has been no tampering in U_1'. Extract the 4 watermarked bits from U_1', these are $r_4'' r_3'' r_2'' r_1''$.

----------U_1'----------											
f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	r_4''	r_3''	r_2''	r_1''
----------U_2'----------											
q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	v_4''	v_3''	v_2''	v_1''

Figure 7. Two data units from the watermarked block

Step 6: If $v_4^* v_3^* v_2^* v_1^* \neq 0000$, then there is tampering in U_2'. The correction is done as shown in Table 2. If $v_4^* v_3^* v_2^* v_1^* = 0000$. Then there has been no tampering in U_2'. Extract the 4 watermarked bits from U_2', these are $v_4'' v_3'' v_2'' v_1''$.

Diagrammatically, the extraction procedure is shown in Figure 8.

Table 1. Correction in U_1'

$r_4^* r_3^* r_2^* r_1^*$	Tampered Location	Correction
0001	r_1'' is tampered	Correct r_1'' by complementing it
0010	r_2'' is tampered	Correct r_2'' by complementing it
0011	f_1 is tampered	Correct f_1 by complementing it
0100	r_3'' is tampered	Correct r_3'' by complementing it
0101	f_2 is tampered	Correct f_2 by complementing it
0110	f_3 is tampered	Correct f_3 by complementing it
0111	f_4 is tampered	Correct f_4 by complementing it
1000	r_4'' is tampered	Correct r_4'' by complementing it
1001	f_5 is tampered	Correct f_5 by complementing it
1010	f_6 is tampered	Correct f_6 by complementing it
1011	f_7 is tampered	Correct f_7 by complementing it
1100	f_8 is tampered	Correct f_8 by complementing it

Table 2. Correction in U_2'

$v_4^* v_3^* v_2^* v_1^*$	Tampered Location	Correction
0001	v_1'' is tampered	Correct v_1'' by complementing it
0010	v_2'' is tampered	Correct v_2'' by complementing it
0011	q_1 is tampered	Correct q_1 by complementing it
0100	v_3'' is tampered	Correct v_3'' by complementing it
0101	q_2 is tampered	Correct q_2 by complementing it
0110	q_3 is tampered	Correct q_3 by complementing it
0111	q_4 is tampered	Correct q_4 by complementing it
1000	v_4'' is tampered	Correct v_4'' by complementing it
1001	q_5 is tampered	Correct q_5 by complementing it
1010	q_6 is tampered	Correct q_6 by complementing it
1011	q_7 is tampered	Correct q_7 by complementing it
1100	q_8 is tampered	Correct q_8 by complementing it

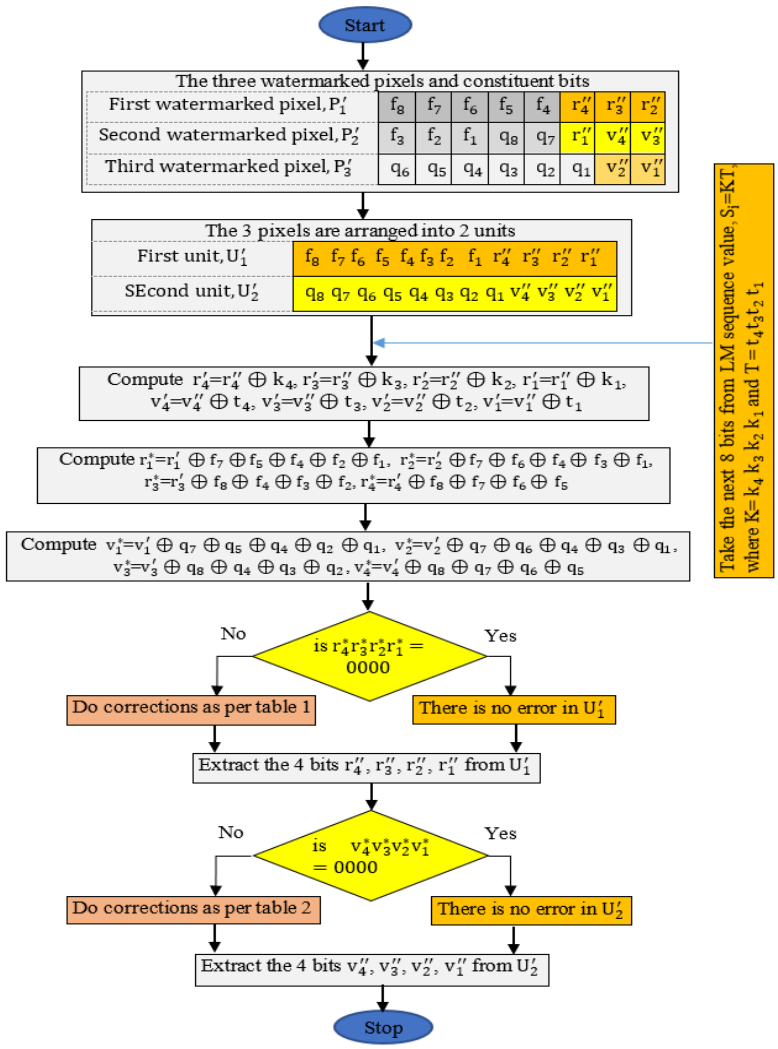


Figure 8. The tamper detection, tamper correction, and watermark extraction procedure

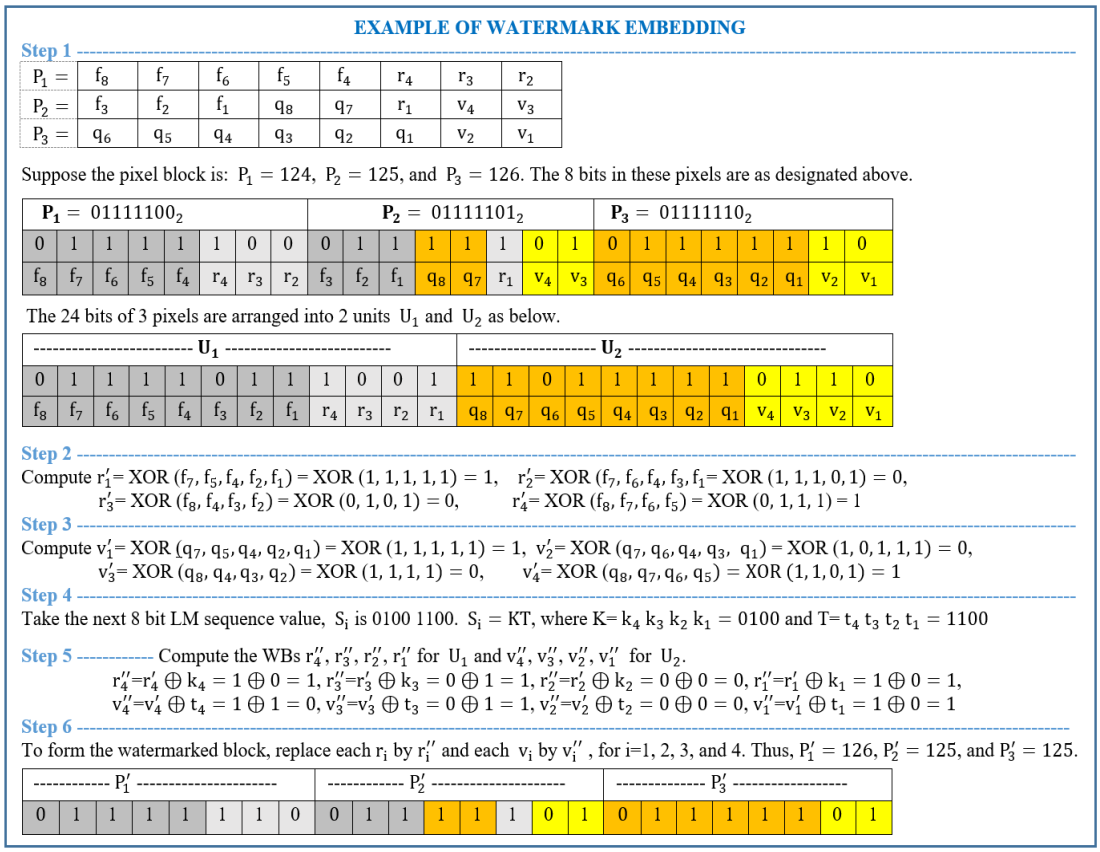
3.4 Example of watermark embedding and watermark extraction

This sub-section presents 3 examples to help you better understand the proposed watermarking technique.

Figure 9 describes an example of watermark embedding in a step-by-step manner.

Figure 10 depicts an example of WB extraction where there is no error. This example presents the extraction procedure with pixel sample values at the bit level with no error case.

Figure 11 illustrates the extractions of WBs with error bits correction. This example presents the extraction procedure with pixel sample values at the bit level and an error correction case.



EXAMPLE OF WATERMARK EMBEDDING

Step 1

P_1 =	f_8	f_7	f_6	f_5	f_4	r_4	r_3	r_2
P_2 =	f_3	f_2	f_1	q_8	q_7	r_1	v_4	v_3
P_3 =	q_6	q_5	q_4	q_3	q_2	q_1	v_2	v_1

Suppose the pixel block is: $P_1 = 124$, $P_2 = 125$, and $P_3 = 126$. The 8 bits in these pixels are as designated above.

$P_1 = 01111100_2$								$P_2 = 01111101_2$								$P_3 = 01111110_2$							
0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0
f_8	f_7	f_6	f_5	f_4	r_4	r_3	r_2	f_3	f_2	f_1	q_8	q_7	r_1	v_4	v_3	q_6	q_5	q_4	q_3	q_2	q_1	v_2	v_1

The 24 bits of 3 pixels are arranged into 2 units U_1 and U_2 as below.

U_1												U_2											
0	1	1	1	1	0	1	1	1	0	0	1	1	1	0	1	1	1	1	1	0	1	1	0
f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	r_4	r_3	r_2	r_1	q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	v_4	v_3	v_2	v_1

Step 2

Compute $r_1' = XOR(f_7, f_5, f_4, f_2, f_1) = XOR(1, 1, 1, 1, 1) = 1$, $r_2' = XOR(f_7, f_6, f_4, f_3, f_1 = XOR(1, 1, 1, 0, 1) = 0$,
$r_3' = XOR(f_8, f_4, f_3, f_2) = XOR(0, 1, 0, 1) = 0$, $r_4' = XOR(f_8, f_7, f_6, f_5) = XOR(0, 1, 1, 1) = 1$

Step 3

Compute $v_1' = XOR(q_7, q_5, q_4, q_2, q_1) = XOR(1, 1, 1, 1, 1) = 1$, $v_2' = XOR(q_7, q_6, q_4, q_3, q_1) = XOR(1, 0, 1, 1, 1) = 0$,
$v_3' = XOR(q_8, q_4, q_3, q_2) = XOR(1, 1, 1, 1) = 0$, $v_4' = XOR(q_8, q_7, q_6, q_5) = XOR(1, 1, 0, 1) = 1$

Step 4

Take the next 8 bit LM sequence value, S_i is 0100 1100. $S_i = KT$, where $K = k_4\ k_3\ k_2\ k_1 = 0100$ and $T = t_4\ t_3\ t_2\ t_1 = 1100$

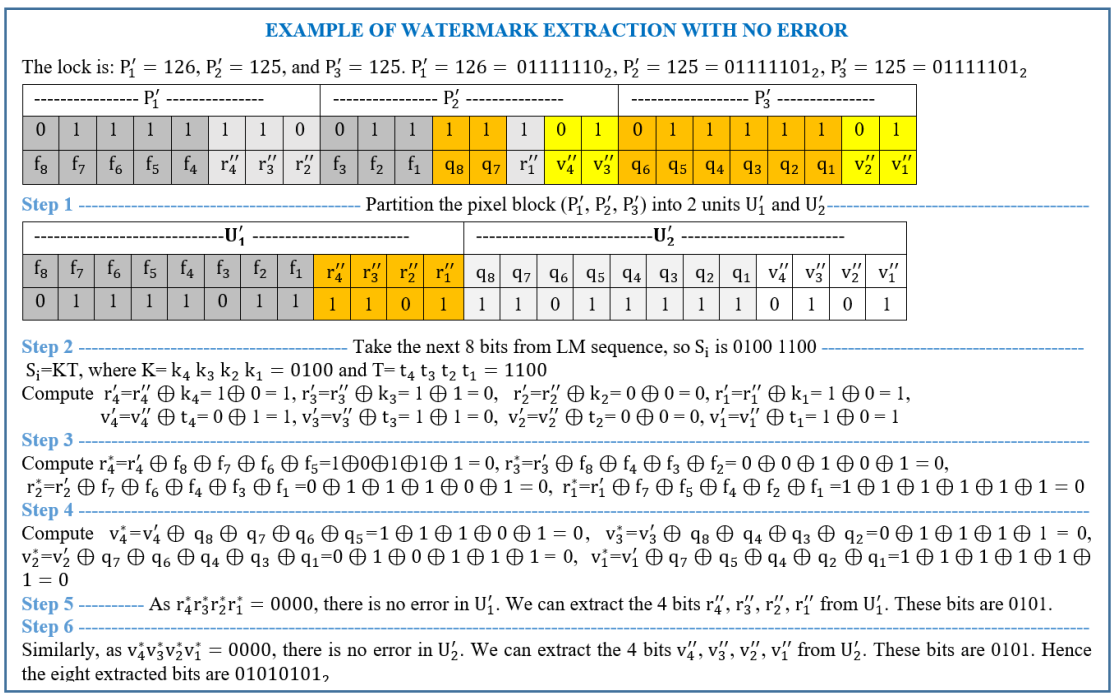
Step 5 Compute the WBs $r_4'', r_3'', r_2'', r_1''$ for U_1 and $v_4'', v_3'', v_2'', v_1''$ for U_2.
$r_4'' = r_4' \oplus k_4 = 1 \oplus 0 = 1$, $r_3'' = r_3' \oplus k_3 = 0 \oplus 1 = 1$, $r_2'' = r_2' \oplus k_2 = 0 \oplus 0 = 0$, $r_1'' = r_1' \oplus k_1 = 1 \oplus 0 = 1$,
$v_4'' = v_4' \oplus t_4 = 1 \oplus 1 = 0$, $v_3'' = v_3' \oplus t_3 = 0 \oplus 1 = 1$, $v_2'' = v_2' \oplus t_2 = 0 \oplus 0 = 0$, $v_1'' = v_1' \oplus t_1 = 1 \oplus 0 = 1$

Step 6

To form the watermarked block, replace each r_i by r_i'' and each v_i by v_i'', for i=1, 2, 3, and 4. Thus, $P_1' = 126$, $P_2' = 125$, and $P_3' = 125$.

P_1'								P_2'								P_3'							
0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1

Figure 9. Example of watermark embedding



EXAMPLE OF WATERMARK EXTRACTION WITH NO ERROR

The lock is: $P_1' = 126$, $P_2' = 125$, and $P_3' = 125$. $P_1' = 126 = 01111110_2$, $P_2' = 125 = 01111101_2$, $P_3' = 125 = 01111101_2$

P_1'								P_2'								P_3'							
0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1
f_8	f_7	f_6	f_5	f_4	r_4''	r_3''	r_2''	f_3	f_2	f_1	q_8	q_7	r_1''	v_4''	v_3''	q_6	q_5	q_4	q_3	q_2	q_1	v_2''	v_1''

Step 1 Partition the pixel block (P_1', P_2', P_3') into 2 units U_1' and U_2'

U_1'												U_2'											
f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	r_4''	r_3''	r_2''	r_1''	q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	v_4''	v_3''	v_2''	v_1''
0	1	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1

Step 2 Take the next 8 bits from LM sequence, so S_i is 0100 1100

$S_i = KT$, where $K = k_4\ k_3\ k_2\ k_1 = 0100$ and $T = t_4\ t_3\ t_2\ t_1 = 1100$

Compute $r_4' = r_4'' \oplus k_4 = 1 \oplus 0 = 1$, $r_3' = r_3'' \oplus k_3 = 1 \oplus 1 = 0$, $r_2' = r_2'' \oplus k_2 = 0 \oplus 0 = 0$, $r_1' = r_1'' \oplus k_1 = 1 \oplus 0 = 1$,
$v_4' = v_4'' \oplus t_4 = 0 \oplus 1 = 1$, $v_3' = v_3'' \oplus t_3 = 1 \oplus 1 = 0$, $v_2' = v_2'' \oplus t_2 = 0 \oplus 0 = 0$, $v_1' = v_1'' \oplus t_1 = 1 \oplus 0 = 1$

Step 3

Compute $r_4^* = r_4' \oplus f_8 \oplus f_7 \oplus f_6 \oplus f_5 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 = 0$, $r_3^* = r_3' \oplus f_8 \oplus f_4 \oplus f_3 \oplus f_2 = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 0$,
$r_2^* = r_2' \oplus f_7 \oplus f_6 \oplus f_4 \oplus f_3 \oplus f_1 = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$, $r_1^* = r_1' \oplus f_7 \oplus f_5 \oplus f_4 \oplus f_2 \oplus f_1 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

Step 4

Compute $v_4^* = v_4' \oplus q_8 \oplus q_7 \oplus q_6 \oplus q_5 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$, $v_3^* = v_3' \oplus q_8 \oplus q_4 \oplus q_3 \oplus q_2 = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$,
$v_2^* = v_2' \oplus q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_1 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$, $v_1^* = v_1' \oplus q_7 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

Step 5 As $r_4^* r_3^* r_2^* r_1^* = 0000$, there is no error in U_1'. We can extract the 4 bits $r_4'', r_3'', r_2'', r_1''$ from U_1'. These bits are 0101.

Step 6

Similarly, as $v_4^* v_3^* v_2^* v_1^* = 0000$, there is no error in U_2'. We can extract the 4 bits $v_4'', v_3'', v_2'', v_1''$ from U_2'. These bits are 0101. Hence the eight extracted bits are 01010101_2

Figure 10. Example of watermark extraction

EXAMPLE OF WATERMARK EXTRACTION WITH ERROR CORRECTION

Suppose the error free pixel block is: $P_1' = 126$, $P_2' = 125$, and $P_3' = 125$. But it reached at the receiver through internet as $P_1' = 254$, $P_2' = 125$, and $P_3' = 127$. It means some of the bits in $P_1'$and P_3' have been modified during transit from sender to receiver.

---------- P_1' ----------								---------- P_2' ----------								---------- P_3' ----------							
1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1
f_8	f_7	f_6	f_5	f_4	r_4''	r_3''	r_2''	f_3	f_2	f_1	q_8	q_7	r_1''	v_4''	v_3''	q_6	q_5	q_4	q_3	q_2	q_1	v_2''	v_1''

Step 1 ---------- Partition the pixel block (P_1', P_2', P_3') into 2 units U_1' and U_2' ----------

----------U_1' ----------												----------U_2' ----------											
f_8	f_7	f_6	f_5	f_4	f_3	f_2	f_1	r_4''	r_3''	r_2''	r_1''	q_8	q_7	q_6	q_5	q_4	q_3	q_2	q_1	v_4''	v_3''	v_2''	v_1''
1	1	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1

Step 2 ---------- Take the next 8- bits from LM sequence, so S_i is 0100 1100 ----------

S_i=KT, where K= $k_4\ k_3\ k_2\ k_1$ = 0100 and T= $t_4\ t_3\ t_2\ t_1$ = 1100
Compute $r_4'=r_4'' \oplus k_4=1\oplus0=1$, $r_3'=r_3'' \oplus k_3=1\oplus1=0$, $r_2'=r_2'' \oplus k_2=0\oplus0=0$, $r_1'=r_1'' \oplus k_1=1\oplus0=1$,
$v_4'=v_4'' \oplus t_4=0\oplus1=1$, $v_3'=v_3'' \oplus t_3=1\oplus1=0$, $v_2'=v_2'' \oplus t_2=1\oplus0=1$, $v_1'=v_1'' \oplus t_1=1\oplus0=1$

Step 3 ----------

Compute $r_4^*=r_4' \oplus f_8 \oplus f_7 \oplus f_6 \oplus f_5=1\oplus1\oplus1\oplus1\oplus1=1$, $r_3^*=r_3' \oplus f_8 \oplus f_4 \oplus f_3 \oplus f_2= 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 =1$
$r_2^*=r_2' \oplus f_7 \oplus f_6 \oplus f_4 \oplus f_3 \oplus f_1 =0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 =0$, $r_1^*=r_1' \oplus f_7 \oplus f_5 \oplus f_4 \oplus f_2 \oplus f_1 =1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

Step 4 ----------

Compute $v_4^*=v_4' \oplus q_8 \oplus q_7 \oplus q_6 \oplus q_5=1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$, $v_3^*=v_3' \oplus q_8 \oplus q_4 \oplus q_3 \oplus q_2=0 \oplus 1 \oplus 1 \oplus 1 \oplus 1=0$
$v_2^*=v_2' \oplus q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_1=1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1=1$, $v_1^*=v_1' \oplus q_7 \oplus k_5 \oplus q_4 \oplus q_2 \oplus q_1=1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$

Step 5 ----------

As $r_4^* r_3^* r_2^* r_1^* = 1100$, there is error in U_1'. Referring the table 1, we notice that d_8 is tampered. So, after correction d_8 value becomes 0. Extract the 4 bits r_4'', r_3'', r_2'', r_1'' from U_1'. These bits are 0101.

Step 6 ----------

Similarly, as $v_4^* v_3^* v_2^* v_1^* = 0010$, there is error in U_2'. Referring the table 2, we notice that v_2'' is tampered. So, after correction v_2'' value becomes 0. Extract the 4 bits v_4'', v_3'', v_2'', v_1'' from U_2'. These bits are 0101. The corrected pixel values are: $P_1' = 126$, $P_2' = 125$, and $P_3' = 125$.

---------- P_1' ----------								---------- P_2' ----------								---------- P_3' ----------							
0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1
f_8	f_7	f_6	f_5	f_4	r_4''	r_3''	r_2''	f_3	f_2	f_1	q_8	q_7	r_1''	v_4''	v_3''	q_6	q_5	q_4	q_3	q_2	q_1	v_2''	v_1''

Figure 11. Example of watermark extraction with error correction

4. RESULTS ANALYSIS

This MHCBW technique is coded in MATLAB on a computing device with an i5 processor. Figure 12 depicts the original images (OIs) from the SIPI database [32]. Although several images are used for testing, only a few samples are shown here. Figure 13 depicts some samples of WIs accordingly. Under each WI, the PSNR and SSIM are mentioned.

The effectiveness of the MHCBW scheme is measured by parameters like embedding time (ET), extraction time (ExT), SSIM, HC, PSNR, and accuracy (ACC). Eq. (16) computes the PSNR, a measure of distortion in WI.

$$\mathrm{PSNR} = 10 \times \log_{10} \frac{m \times n \times 255 \times 255}{\sum_{i=1}^{m} \sum_{j=1}^{n} (P_{ij} - Q_{ij})^2} \qquad (16)$$

HC is the total hiding capacity in all the pixels of the image. Capacity of one pixel is represented as bpp i.e., bits per pixel. Eq. (17) represents structural similarity (SSIM) of WI with OI [26]. P_{ij} is a pixel of OI and Q_{ij} is a pixel of WI. $\overline{P}$ is mean value of all P_{ij} values of OI. $\overline{Q}$ is mean value of all Q_{ij} values of WI. Two constants c_1 and c_2 has been inserted in $(\overline{P}^2 + \overline{Q}^2 + c_1)$ and $(\sigma_x^2 + \sigma_y^2 + c_2)$ to ensure the denominator to be non-zero. The constant $c_1=(K_1 L)^2$, $K_1 \ll 1$ and L value is 255 for grey image. The constant $c_2=(K_2 L)^2$, and $K_2 \ll 1$. Here, variance of OI to WI is σ_{pq}, variance of OI is σ_p^2, and variance of WI is σ_q^2. SSIM value will approach to 1 where the WI is very similar with OI. The ACC has been computed in Eq. (18) [26] using the counts of false negatives (FN), true negatives (TN), false positives (FP) and true positives (TP).

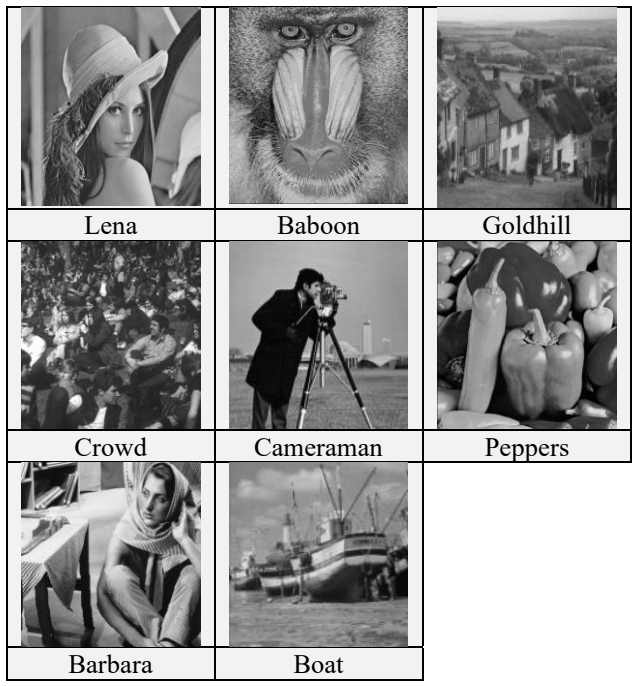


Figure 12. OIs

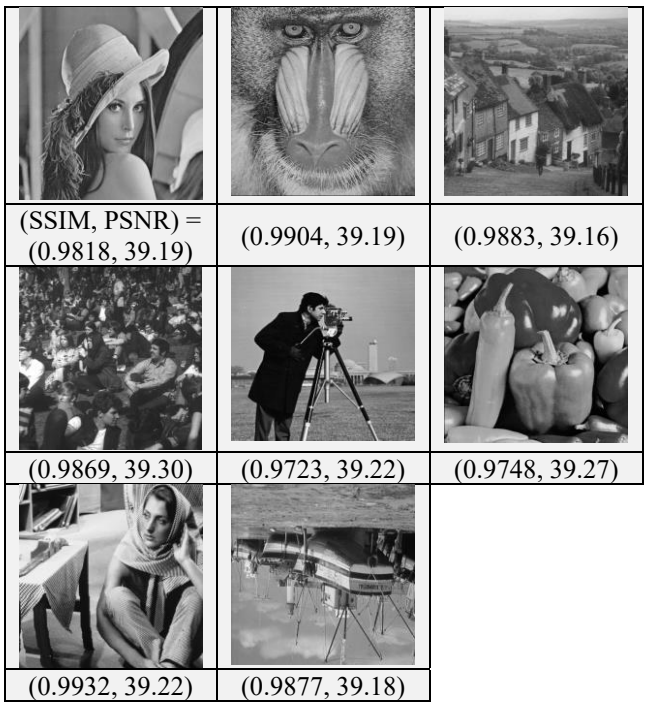


Figure 13. WIs

$$SSIM = \frac{(2\bar{P}\,\bar{Q}+c_1)(2\sigma_{xy}+c_2)}{(\bar{P}^2+\bar{Q}^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \quad (17)$$

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (18)$$

Table 3 records the outcomes of this proposed MHCBW scheme. Table 3 records the PSNR value as 39.21 dB, so it is acceptable. The bpp is 2.67. The average value of SSIM is 0.9844, which indicates that OI and WI are more than 99% alike. ACC value is 0.99, which conveys that the number of tampered pixels is identified with 99% correctness. Furthermore, embedding and extraction of the watermark can be performed in 10.64 and 12.49 secs respectively.

Table 3. Performance measure of MHCBW scheme

512×512 Images	SSIM	PSNR (dB)	ACC	bpp	E×T (secs)	ET (secs)
Lena	0.9818	39.19	0.9999	2.67	11.43	10.35
Baboon	0.9904	39.19	0.9999	2.67	11.29	10.09
Goldhill	0.9883	39.16	0.9999	2.67	11.30	10.02
Crowd	0.9869	39.30	0.9999	2.67	11.34	10.83
Camera-man	0.9723	39.22	0.9999	2.67	11.27	10.96
Pepper	0.9748	39.27	0.9999	2.67	14.77	10.87
Barbara	0.9932	39.22	0.9999	2.67	14.17	10.80
Boat	0.9877	39.18	0.9999	2.67	14.36	11.19
Average	0.9844	39.21	0.9999	2.67	12.49	10.64

Figure 14 conveys the tampered Lena images with tampering % from 5% to 45%. A pixel is tampered with if one or more of its 8 bits are changed. The tampered pixels are localized and made white color for easy reference by the readers. It is shown for only one image. It is performed for all the test images. Table 4 depicts the mean value of bpp, PSNR, SSIM, and ACC over 8 images with tampering percentages starting from 0% to 45%.

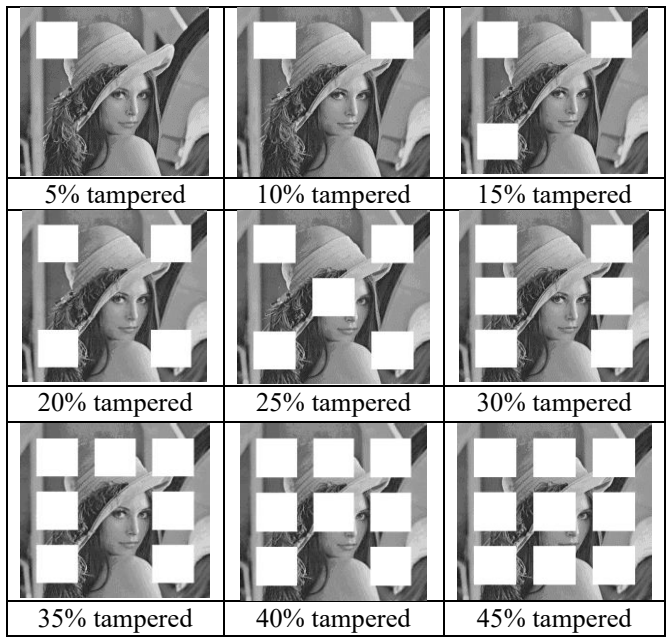
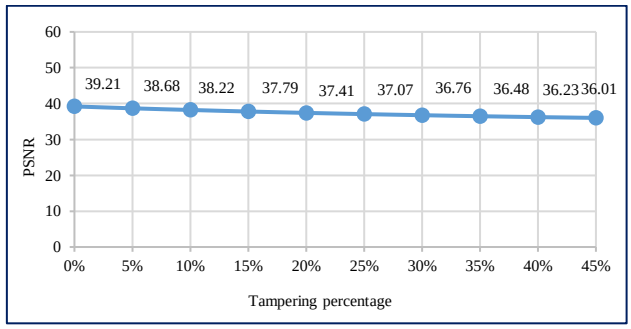
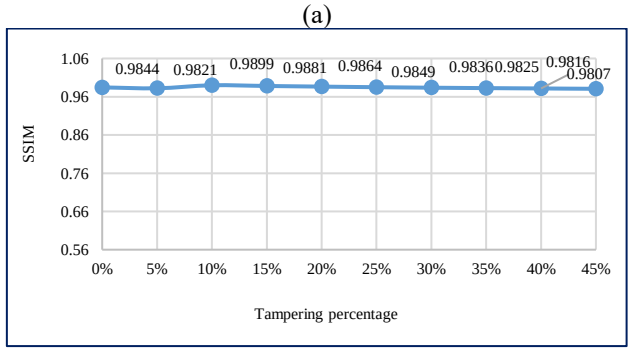


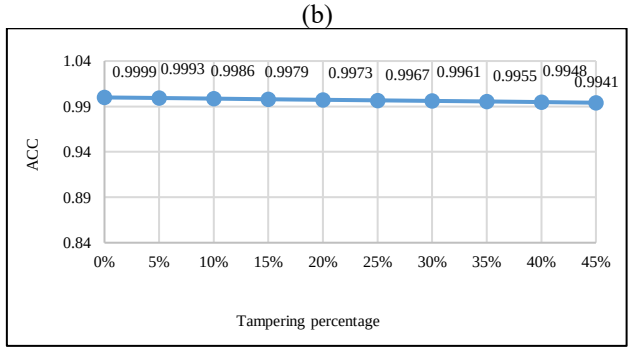
Figure 14. Tampered Lena images (the tampered location pixels are made white color for reference)



(a)



(b)



(c)

Figure 15. (a) Tampering rate versus PSNR, (b) Tampering rate versus SSIM, and (c) Tampering rate versus ACC

Figure 15(a) depicts a graph for the PSNR at tampering rate. It can be observed from this graph that if we further raise the tampering rate after 45%, PSNR can be at least 30 dB. Figure 15(b) depicts a plot for SSIM at different tampering rates, and Figure 15(c) depicts a plot for ACC at different tampering rates. From Figure 15(b), we can achieve an SSIM value greater than 0.98 even after raising the tampering by more than 45%. From Figure 15(c), we can note that the ACC value is greater than 0.99 even after raising the tampering rate beyond 45%.

Table 5 represents a comparison of the average values of different parameters of this MHCBW technique with Nazari et al. [12], Chang et al. [24], Prasad and Pal [28], and Prasad and Pal [29]. Techniques of Chang et al. [24], Prasad and Pal [28], and Prasad and Pal [29] are based on Hamming code. Nazari et al.'s [12] scheme includes a chaotic map. Furthermore, Figure 16 shows a bar graph distinguishing the HC and PSNR of the MHCBW technique with the existing schemes; Figure 17 depicts a bar graph comparing the SSIM and ACC values of the MHCBW technique with existing schemes.

From Figure 16, it may be noted that there is a better trade-off between the HC and PSNR of the MHCBW scheme as compared to the 4 existing schemes. Prasad and Pal's [28] technique possesses very low bpp but higher PSNR. Nazari et al.'s [12] technique possesses both low bpp and low PSNR.

From Figure 17, it may be observed that the SSIM value in MHCBW technique is lesser than the works of Nazari et al. [12], Chang et al. [24], Prasad and Pal [28], and Prasad and Pal [29], but the ACC values of the MHCBW scheme is higher than the works of Nazari et al. [12], Chang et al. [24], Prasad and Pal [28], and Prasad and Pal [29].

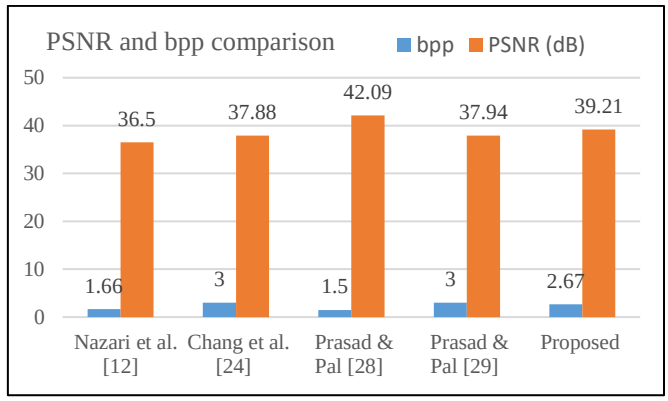


Figure 16. PSNR, bpp comparison

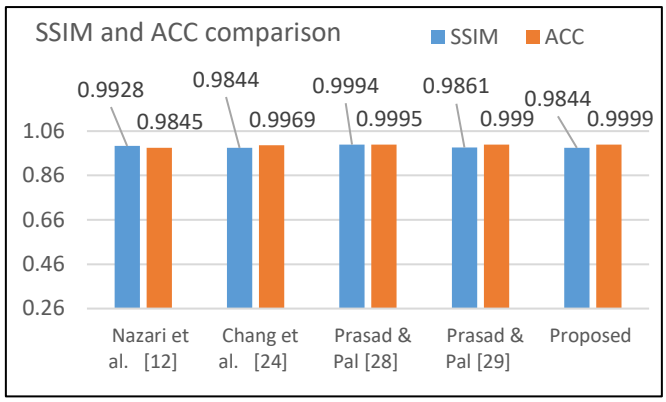


Figure 17. SSIM, ACC comparison

Table 4. Efficacy measurement at varied tampering rates

Efficacy Parameter	Tampering Rate 0%	5%	10%	15%	20%	25%	30%	35%	40%	45%
bpp	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67	2.67
PSNR	39.21	38.68	38.22	37.79	37.41	37.07	36.76	36.48	36.23	36.01
ACC	0.9999	0.9993	0.9986	0.9979	0.9973	0.9967	0.9961	0.9955	0.9948	0.9941
SSIM	0.9844	0.9821	0.9899	0.9881	0.9864	0.9849	0.9836	0.9825	0.9816	0.9807

Table 5. Comparison of average efficacy parameter values

Technique	bpp	PSNR	SSIM	ACC
Nazari et al.'s [12]	1.66	36.50	0.9928	0.9845
Chang et al.'s [24]	3.0	37.88	0.9844	0.9969
Prasad and Pal's [28]	1.5	42.09	0.9994	0.9995
Prasad and Pal's [29]	3.0	37.94	0.9861	0.9990
Proposed MHCBW scheme	2.67	39.21	0.9844	0.9999

5. CONCLUSIONS

This article proposes MHCBW technique for image tamper detection and correction. It uses a modified version of Hamming code for error detection and correction. It uses an LM sequence to improve the security. The WBs are stored in redundant bit positions and can be extracted accurately. Experimental results reveal that this MHCBW technique maintains a fair compromise between HC and PSNR compared to existing techniques. Nazari et al.'s technique produces lesser PSNR and bpp than the proposed one. The other 3 existing techniques produce either higher bpp with lower PSNR or vice-versa compared to the MHCBW technique. The MHCBW scheme's estimated average PSNR is 39.21, and bpp is 2.67. The estimated SSIM value of the MHCBW technique is 0.9844, which indicates that the OIs are structurally similar to the corresponding WIs. The estimated ACC value of the MHCBW technique is 0.9999, which indicates that the WBs could be correctly extracted.

The error detection and correction are applied on 3 pixels, i.e., 24 bits, by spreading them in 2 evacuation units. In each unit, only 1 bit error is identified and corrected by applying the modified Hamming code. The error detection and correction ideas can be extended to 2 or more bits per unit by changing the modified Hamming code or introducing a new error-detecting and correcting code.

REFERENCES

[1] Ramos, A.M., Artiles, J.A.P., Chaves, D.P.B., Pimentel, C. (2023). A fragile image watermarking scheme in DWT domain using chaotic sequences and error-correction codes. Entropy, 25(3): 1-23. https://doi.org/10.3390/e25030508

[2] Khadse, D.B., Swain, G. (2023). Data hiding and integrity verification based on quotient value differencing and Merkle tree. Arabian Journal for

Science and Engineering, 48: 1793-1805. https://doi.org/10.1007/s13369-022-06961-9

[3] Kosuru, S.N.V.J.D., Pradhan, A., Basith, K.A., Sonar, R., Swain, G. (2023). Digital image steganography with error correction on extracted data. IEEE Access, 11: 80945-80957. https://doi.org/10.1109/ACCESS.2023.3300918

[4] Bhalerao, S., Ansari, I.A., Kumar, A. (2021). A secure image watermarking for tamper detection and localization. Journal of Ambient Intelligence and Humanized Computing, 12: 1057-1068. https://doi.org/10.1007/s12652-020-02135-3

[5] Gul, E., Ozturk, S. (2020). A novel triple recovery information embedding approach for self-embedded digital image watermarking. Multimedia Tools and Applications, 79: 31239–31264. https://doi.org/10.1007/s11042-020-09548-4

[6] Sinhal, R., Ansari, I.A., Ahn, C.W. (2020). Blind image watermarking for localization and restoration of color images. IEEE Access, 8: 200157-200169. https://doi.org/10.1109/ACCESS.2020.3035428

[7] Qin, C., Ji, P., Zhang, X., Dong, J., Wang, J. (2017). Fragile image watermarking with pixel-wise recovery based on overlapping embedding strategy. Signal Processing, 138: 280-293. https://doi.org/10.1016/j.sigpro.2017.03.033

[8] Rawat, S., Raman, B. (2011). A chaotic system based fragile watermarking scheme for image tamper detection. International Journal of Electronics and Communications (AEU), 65: 840-847. https://doi.org/10.1016/j.aeue.2011.01.016

[9] Botta, M., Cavagnino, D., Pomponiu, V. (2015). A successful attack and revision of chaotic system based fragile watermarking scheme for image tamper detection. International Journal of Electronics and Communications (AEU), 69(1): 242-245. https://doi.org/10.1016/j.aeue.2014.09.004

[10] Prasad, S., Pal, A.K. (2020). A secure fragile watermarking scheme for protecting integrity of digital images. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 44: 703-727. https://doi.org/10.1007/s40998-019-00275-7

[11] Sahu, A.K. (2022). A logistic map based blind and fragile watermarking for tamper detection and localization in images. Journal of Ambient Intelligence and Humanizes Computing, 13: 3869-3881. https://doi.org/10.1007/s12652-021-03365-9

[12] Nazari, M., Sharif, A., Mollaeefar, M. (2017). An improved method for digital image fragile watermarking based on chaotic maps. Multimedia Tools and Applications, 76: 16107-16123. https://doi.org/10.1007/s11042-016-3897-x

[13] Shehab, A., Elhoseny, M., Muhammad, K., Sangaiah, A.K., Yang, P., Huang, H., Hou, G. (2018). Secure and Robust fragile watermarking scheme for medical images. IEEE Access, 6: 10269-10278. https://doi.org/10.1109/ACCESS.2018.2799240

[14] Kosuru, S.N.V.J.D., Swain, G., Kumar, N., Pradhan, A. (2022). Image tamper detection and correction using Merkle tree and remainder value differencing. Optik, 261(169212): 1-11. https://doi.org/10.1016/j.ijleo.2022.169212

[15] Gull, S., Mansour, R.F., Aljehane, N.O., Parah, S.A. (2021). A self-embedding technique for tamper detection and localization of medical images for smart health. Multimedia Tools and Applications, 80: 29939-29964. https://doi.org/10.1007/s11042-021-11170-x

[16] Laishram, D., Sing, K.M. (2021). A watermarking scheme for source authentication, ownership identification, tamper detection, and restoration for color medical images. Multimedia Tools and Applications, 80: 23815-23875. https://doi.org/10.1007/s11042-020-10389-4

[17] Rinki, K., Verma, P., Singh, R.K. (2022). A novel matrix multiplication based LSB substitution mechanism for data security and authentication. Journal of King Saud University - Computer and Information Sciences, 34: 5510-5524. https://doi.org/10.1016/j.jksuci.2021.01.013

[18] Chen, C.C., Chang, C.C., Chen, K. (2021). High capacity reversible data hiding in encrypted image based on Huffman coding and differences of high nibbles of pixels. Journal of Visual Communication and Image Representation, 76: 103060. https://doi.org/10.1016/j.jvcir.2021.103060

[19] Jana, M., Jana, B., Joardar, S. (2022). Local feature based self-embedding fragile watermarking scheme for tampered detection and recovery utilizing AMBTC with fuzzy logic. Journal of King Saud University - Computer and Information Sciences, 34(10): 9822-9835. https://doi.org/10.1016/j.jksuci.2021.12.011

[20] Pal, P., Jana, B., Bhaumik, J. (2021). An image authentication and tampered detection scheme exploiting local binary pattern along with Hamming error correcting code. Wireless Personal Communication, 121: 939-961. https://doi.org/10.1007/s11277-021-08666-y

[21] Patsariya, S., Dixit, M. (2022). A new block based non-blind hybrid color image watermarking approach using lifting scheme and chaotic encryption based on Arnold cat map. Traitment du Signal, 39(4): 1159-1168. https://doi.org/10.18280/ts.390408

[22] Chennamma, H.R., Basavarajappa, B.K.H., Basavaraju, M., Sowmya, K.N., Kumar, A.V.S. (2024). Medical image authentication using statistical correlations. International Journal of Safety and Security Engineering, 14(5): 1487-1493. https://doi.org/10.18280/ijsse.140516

[23] Hamming, R.W. (1950). Error detecting and error correcting codes. Bell System Technical Journal, 29(2): 147-160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

[24] Chang, C.C., Chen, K.N., Lee, C.F., Liu, L.J. (2011). A secure fragile watermarking scheme based on chaos-and-hamming code. Journal of Systems and Software, 84(9): 1462-1470. https://doi.org/10.1016/j.jss.2011.02.029

[25] Wang, J.T., Chang, Y.C., Yu, C.Y., Yu, S.S. (2014). Hamming code based watermarking scheme for 3D model verification. Mathematical Problems in Engineering, 2014(241093): 1-7. https://doi.org/10.1155/2014/241093

[26] Islam, M.S., Kim, C.H., Kim, J.M. (2015). A GPU-based (8,4) Hamming decoder for secure transmission of watermarked medical images. Cluster Computing, 18: 333-341. https://doi.org/10.1007/s10586-014-0392-x

[27] Trivedy, S., Pal, A.K. (2017). A LM-based fragile watermarking scheme of digital images with tamper detection. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 41: 103-113. https://doi.org/10.1007/s40998-017-0021-9

[28] Prasad, S., Pal, A.K. (2020). A tamper detection suitable

fragile watermarking scheme based on novel payload embedding strategy. Multimedia Tools and Applications, 79: 1673-1705. https://doi.org/10.1007/s11042-019-08144-5

[29] Prasad, S., Pal, A.K. (2020). Hamming code and logistic-map based pixel-level active forgery detection scheme using fragile watermarking. Multimedia Tools and Applications, 79: 20897-20928. https://doi.org/10.1007/s11042-020-08715-x

[30] Jana, B., Giri, D., Mondal, S.K. (2018). Dual image based reversible data hiding scheme using (7,4) Hamming code. Multimedia Tools and Applications, 77(1): 763-785. https://doi.org/10.1007/s11042-016-4230-4

[31] Nguyen, T.D., Le, H.D. (2021). A reversible data hiding scheme based on (5, 3) Hamming code using extra information on overlapped pixel blocks of grayscale images. Multimedia Tools and Applications, 80: 13099-13120. https://doi.org/10.1007/s11042-020-10347-0

[32] University of Southern California (USC) Signal and Image Processing Institute, "USC-SIPI image database," http://sipi.usc.edu/database/database.php?volume=misc/, accessed on December 15, 2022.