Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-19752-1

A side match oriented data hiding based on absolute moment block truncation encoding mechanism with reversibility

Jui-Chuan Liu[1] · **Yijie Lin**[1] · **Ching-Chun Chang**[2] · **Chin-Chen Chang**[1]

Received: 28 September 2023 / Revised: 6 June 2024 / Accepted: 20 June 2024

Abstract
Owing to life style changes during COVID, communication channels were forced to change. Online services ramped up to replace traditional services. It became daily practice for most of us that documents, files, photos, and videos were transmitted through the internet and websites. Information security was one of the big issues requiring more and more attention. Researchers proposed various solutions including steganography, watermarking, and data hiding. Reversible data hiding (RDH) is a commonly and widely accepted method to secure information. A receiver can not only extract secret information from stego media but can also recover the original media. This study focuses on enhancing the embedding capacity based on a known scheme proposed by Chang et al. in 2017. The proposed scheme incorporates absolute moment block truncation coding (AMBTC). In addition to using the high and low values generated from AMBTC to hide information, we introduce a multi-pass index reordering method to embed data into the bitmap, further boosting the embedding capacity. We also apply the side match (SM) technique in the bitmap recovery process. Our experimental results show that the proposed scheme can significantly improve embedding capacity.

Keywords RDH · AMBTC · BM · JNC · Index-reordering · Side match

✉ Chin-Chen Chang
alan3c@gmail.com

Jui-Chuan Liu
p1200318@o365.fcu.edu.tw

Yijie Lin
yjlin.phd@gmail.com

Ching-Chun Chang
ccc@fcu.edu.tw

[1] Department of Information Engineering and Computer Science, Feng Chia University, NO.100 Wenhwa Rd., Seatwen, Taichung 40724, Taiwan

[2] Information and Communication Security Research Center, Feng Chia University, NO.100 Wenhwa Rd., Seatwen, Taichung 40724, Taiwan

Published online: 10 July 2024

1 Introduction

People were prohibited from going into the offices to work and were limited from having face-to-face communications during COVID. Life-styles evolved during the period of time and massive online services started to appear. Online means that data and information in various formats need to be digitized and sent through public channels most of the time. Security issues are raised more and more often and information is at higher risk of being intercepted during data transmission or being downloaded by illegal parties. In order to solve these security issues, there are many studies that emphasize these topics [1–7]. Numerous solutions involving steganography [8], watermarking [9], and data hiding (DH) [10–13] have become popular and widely used technologies applied to multimedia. To guarantee embedded data extraction and to reconstruct the original image are the two goals of steganography. Watermarking embedding should not be detectable by human eyes; therefore, watermarked images need to be able to retain imperceptibility and to stay resistant to attacks. Data hiding conceals secret messages in innocent cover media and has evolved to be an important stream along the road of resolving security issues. However, this generally results in the degrading of image quality after the embedding process. There are many DH schemes [14] which attempt to make the embedding changes in cover images unnoticeable to avoid attracting the attention of malicious attackers. For some applications such as medicine and the military, damaged media which cannot be recovered could lead to wrong judgments. Such high-stakes situations incentivized further studies on the reversibility of stego images after extraction.

Reversible data hiding (RDH) schemes [15, 16] are favored by researchers because certain applications require the recovery of the original media. RDH schemes not only recover secret data but also the original cover media. The performance of a reversible data hiding scheme is usually evaluated by its embedding capacity, visual quality of recovered images, and computational complexity. Image compression is the most famous RDH technique implemented. Image compression can reduce the file size when transmitting media and can keep the image visual quality when using the naked eye. Vector quantization (VQ) [17, 18], Joint Photographic Experts Group (JPEG) [19], absolute moment block truncation coding (AMBTC) [20–26] are techniques commonly used to compress original images.

Sun et al. proposed a scheme using joint neighboring coding (JNC) [12] in 2013 to embed secret information into AMBTC compression code. It revealed two deficiencies: one is that it wasted an extra bit to determine if the difference of pixel values is positive or negative, and the other is that the lengths of the compression codes varied which made the extraction much more complicated. Chang et al. proposed a reversible data hiding method based on AMBTC and XOR join neighborhood coding (XOR-JNC) to improve JNC [13] in 2017. The method achieved high embedding capacity with reasonable stego file size and simplified the embedding process and the extraction process, but there is still room for enhancement. Our study aims to increase embedding capacity and enhance the state-of-the-art AMBTC schemes to be reversible. The proposed scheme increases the embedding capacity by manipulating bitmaps generated from AMBTC on top of using the high and low values. To keep the scheme reversible, side match (SM) [14, 19, 27] technique is applied to reverse pixel values. It is the first study by enhancing AMBTC and combining SM techniques to achieve higher embedding capacity with reversibility. The experiments show promising results and prove we can add more value to the state-of-the-art method.

The contributions of the proposed scheme are:

a. It can further improve the embedding capacity of a capable reversible data hiding method.
b. It introduces a multi-pass index-reorder technique of hiding data in bitmap with reversible capability.
c. It makes full use of the output of AMBTC, including the high values, the low values, and the bitmaps generated for pixel blocks.

Section 2 describes a couple methods related to our proposed scheme. Section 3 details the proposed scheme. The experiment results and analysis are given in Section 4. The conclusion concludes our study and states the possibility of future work in Section 5.

2 Related work

Our proposed scheme involves a few techniques which are commonly used by other data hiding research, such as AMBTC, join neighborhood coding (XOR-JNC) and side match (SM). Block Truncation Coding (BTC) is a compression method using the difference of square values to compress images, but the square calculation is time consuming. Absolute moment block truncation coding (AMBTC) using mean values instead of square values is to overcome the computation complexity achieving shorter time of computation. Traditional AMBTC method is not reversible, XOR join neighborhood coding (XOR-JNC) is then introduced. XOR-JNC using XOR technique to recover the high values and the low values, generated from AMBTC compression, after data embedding. Other than the high/low values, bitmaps are created during AMBTC compression at the same time. They are not used for embedding in XOR-JNC scheme. We researched methods to embed data in bitmaps, looked for means to keep the reversibility for bitmap recovery and proposed our scheme at last. AMBTC, XOR-JNC, and SM techniques are described in the subsections below so that it's easier to understand their fundamental concepts before delegating the proposed scheme.

2.1 Absolute Moment Block Truncation Coding (AMBTC)

Absolute moment block truncation coding was introduced by Lema and Mitchell in 1984 [28] and widely adopted to compress images. Instead of using the difference of square values as in BTC, AMBTC uses the mean in a pixel block to reduce the computation complexity and result in fast computation time. Since the difference between a value and a mean is smaller, the image quality is better after compression.

If there is an $n \times n$ block b_{ij}, the mean m_{ij} of the block b_{ij} is calculated by

$$m_{ij} = \frac{1}{n \times n} \sum_{s=0}^{n-1} \sum_{t=0}^{n-1} p_{st}, \tag{1}$$

where p_{st} is the pixel value at (s, t) in the block. While a bitmap bm_{ij} is generated for block b_{ij},

$$\alpha_{st} = \begin{cases} 1, if\ p_{st} > m_{ij} \\ 0, if\ p_{st} \le m_{ij} \end{cases}, \tag{2}$$

where α_{st} indicates the bit value at (s,t) in bm_{ij} and $0 \le s,t \le n-1$.

While constructing bitmap bm_{ij}, the block can be quantized to a high level hv_{ij} and a low level lv_{ij}. The two levels can be obtained by Eqs. (3) and (4) below.

$$hv_{ij} = \frac{1}{N_1} \sum\nolimits_{\forall(p_{st} > m_{ij})} p_{st} \tag{3}$$

and

$$lv_{ij} = \frac{1}{N_0} \sum\nolimits_{\forall(p_{st} \le m_{ij})} p_{st}, \tag{4}$$

where p_{st} represents the pixel value at (s,t) in the block b_{ij}, N_1 means the number of pixels whose values are greater than the mean of the block m_{ij} and N_0 means the number of pixels whose values are less than or equal to the mean m_{ij}.

After AMBTC process, an $n \times n$ bitmap bm_{ij}, a high value hv_{ij} and a low value lv_{ij} are concatenated to a sequence of bit string for the (i,j)th block. When a receiver receives the concatenated information, the image I' can be recovered by using

$$p'_{st} = \begin{cases} hv_{ij}, \ if\ bm_{ij} = 1 \\ lv_{ij}, \ if\ bm_{ij} = 0 \end{cases}, \tag{5}$$

where $p_{st}{}'$ is the extracted pixel value at (s,t) in the (i,j)th block.

Figure 1 illustrates an AMBTC example. The mean m_{ij} is 102 after calculating the average of the pixel values in the block b_{ij}. Its bitmap bm_{ij} is generated according to the pixel values and the mean m_{ij}. The high value hv_{ij} and the low value lv_{ij} are computed based on the pixel values in the block b_{ij} and the bit values in the bitmap bm_{ij}. During the decoding stage, pixel values in a recovered block $b_{ij}{}'$ are replaced with the high value when the corresponding bit in bitmap is 1 and are replaced with the low value when the corresponding bit is 0.

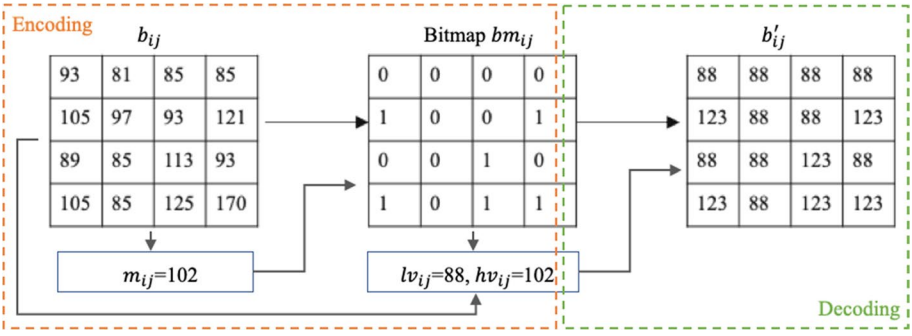


Fig. 1 An example of encoding and decoding using the AMBTC method

2.2 XOR Join Neighborhood Coding (XOR- JNC)

In order to increase the data hiding capacity, Sun et al. [12] embedded secret bits in both the high-level table and the low-level table formed from AMBTC coding. The separation of the level tables is to limit the difference between a pixel value and its neighboring pixel value to enhance the chances of embedding. Chang et al.'s scheme [13] altered the traditional JNC to use an exclusive OR operation for difference calculation for reversibility purposes in 2017.

If there is a given secret message defined as

$$S = \left(s_0, s_1, \cdots, s_{r-1}\right) \text{ where } s_k = \{0, 1\} \; and \; 0 \le k \le r - 1. \tag{6}$$

2.2.1 Embedding of XOR-JNC scheme

When an original image consists of $w \times h$ pixels, both the high value table HT and the low value table LT are with size $\frac{w}{n} \times \frac{h}{n}$. The high value table is formed using the high values from all $n \times n$ blocks and the low value table is formed using the low values. The top row, the left-most column and the right-most column of each value table are reserved as references. Here are the encoding steps using one of the two value tables:

a Extract two bits s_k, s_{k+1} from the binary secret message stream.
b Set current value v_{xy}, find its four neighbor values: $v_{x-1,y}, v_{x-1,y-1}, v_{x,y-1}, v_{x+1,y-1}$ by following the predefined neighboring indices (00, 01, 10, 11) as exhibited in Fig. 2 below.
c Get the pixel value of the neighbor whose neighboring index matches the extracted two secret bits and set the pixel value as the reference value.
d Get the difference d_{xy} between the current value and the reference value by an exclusive OR operation and a decimal conversion of the result.
e Classify the difference according to

$$dc_{xy} = \begin{cases} 11, & if \; 0 \le d_{xy} \le 7 \\ 10, & if \; 8 \le d_{xy} \le 15. \\ 01, & if \; 16 \le d_{xy} \le 31 \\ 00, & if \; 32 \le d_{xy} \le 255 \end{cases}, \tag{7}$$

f. Encode v_{xy} according to the value of dc_{xy} and output v_{xy}' which is a concatenation of two secret bits $s_k \parallel s_{k+1}$, the difference class value dc_{xy}, the difference d_{xy} and additional secret messages.

- When dc_{xy} is 11, 3 bits are reserved for the difference and 5 extra bits extracted from the secret message can be embedded.
- When dc_{xy} is 10, 3 bits are reserved for the difference and 5 extra bits extracted from the secret message can be embedded.

$v_{x-1,y-1}$ (01)	$v_{x,y-1}$ (01)	$v_{x+1,y-1}$ (11)
$v_{x-1,y}$ (00)	v_{xy}	

Fig. 2 Current pixel value v_{xy} and its four neighboring values

- When dc_{xy} is 01, 4 bits are reserved for the difference FVand 4 extra bits extracted from the secret message can be embedded.
- When dc_{xy} is 00, all 8 bits are reserved for the difference.

g. Form a 12-bit compression code for $v_{xy}{}'$
h. Repeat Step a. until the stego value table is completed.

An example is given in Fig. 3 to illustrate the XOR-JNC embedding. Even though the example given uses the high value table, the low value table is embedded with the same method.

2.2.2 Extraction of the XOR-JNC scheme

As for extraction of the XOR-JNC scheme, steps are described below after receiving the stego high value table HT' and the stego low value table LT':

a. Reserve the top row, the left-most column and the right-most column in the stego value table.
b. Extract the first two bits from 12-bit compression code of $v_{xy}{}'$ to get the first two secret bits $s_k \parallel s_{k+1}$.
c. Extract the third and the fourth bits from $v_{xy}{}'$ to get its difference class dc_{xy}.
d. Determine the difference d_{xy} and extract secret bits using Eq. (7)
e. Recover current value v''_{xy} based on

$$v''_{xy} = v_{xy}{}' \oplus d_{xy}. \tag{8}$$

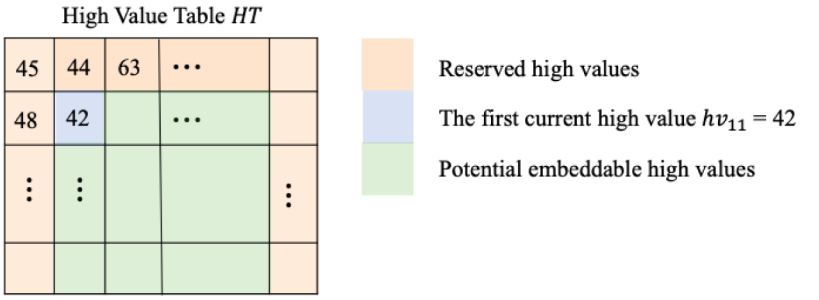
f. Repeat Step b through Step f until the recovered value table is fully retrieved and concatenate the extracted secret bits to form the secret S.

An example is given in Fig. 4 for a better understanding of XOR-JNC extraction. Even though the example given uses the high value table, the low value table is extracted using the exact same way.

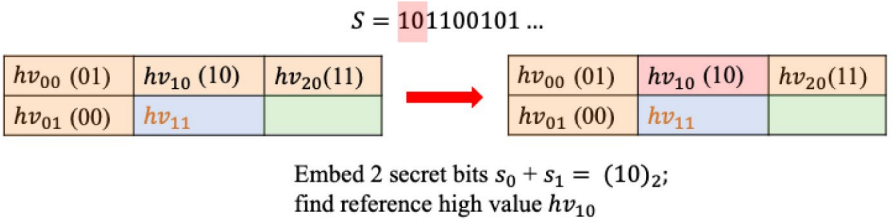
2.3 Side Match Vector Quantization (SMVQ)

Kim proposed using side match on the vector quantization index table in 1992 [11]. It is an algorithm counting on the high correlation among the neighboring blocks in an image. The key principles of a side match method are

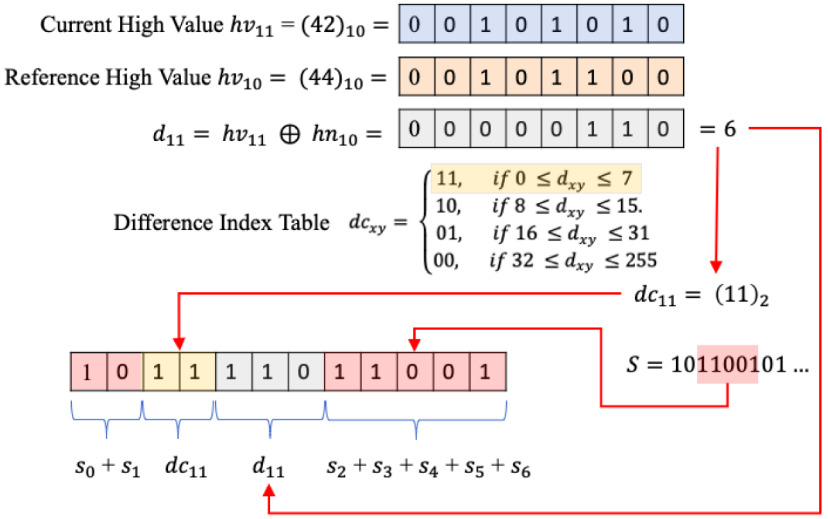
- match the pixel values of the top row in the current block with the pixel values of the bottom row in its upper adjacent block;
- match the pixel values of the left column in the current block with the pixel values of the right column in its left adjacent block.



(a) Reserve the top row, the left-most column and the right-most column.



(b) Find the reference high value according to the first 2 bits of the secret S.



(c) Flow of creating 12-bit compression code

Fig. 3 An example of XOR-JNC high value table embedding

In Fig. 5, C represents the current processed block, U is its upper adjacent block, and L is the left adjacent block. When processing the current block C, blocks U and L have already been processed and all pixel values in both block C and block U are known and unchanged. Since the pixel value $C1$ needs to match both $U13$ and $L4$, $C1 = \frac{U13+L4}{2}$. The other matches are $C2 = U14, C3 = U15, C4 = U16, C5 = L4, C9 = L12$, and $C13 = L16$.

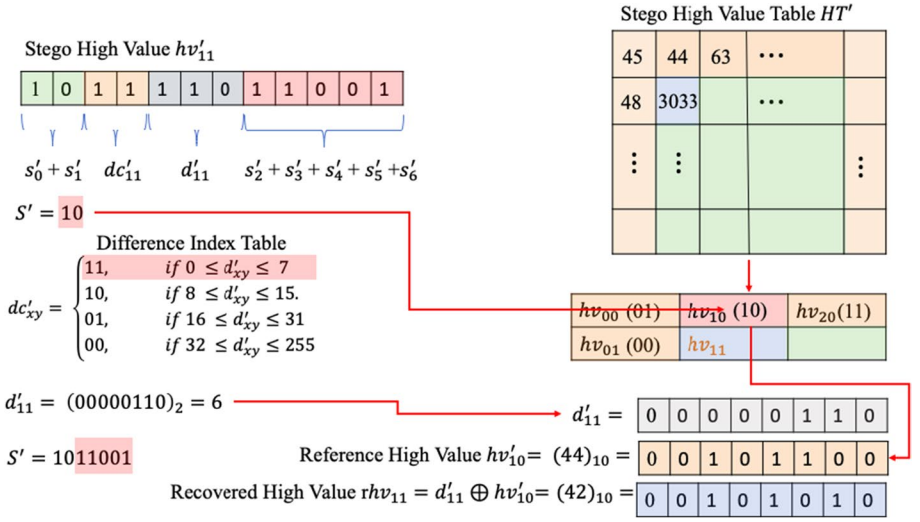


Fig. 4 An example of XOR-JNC high table extraction

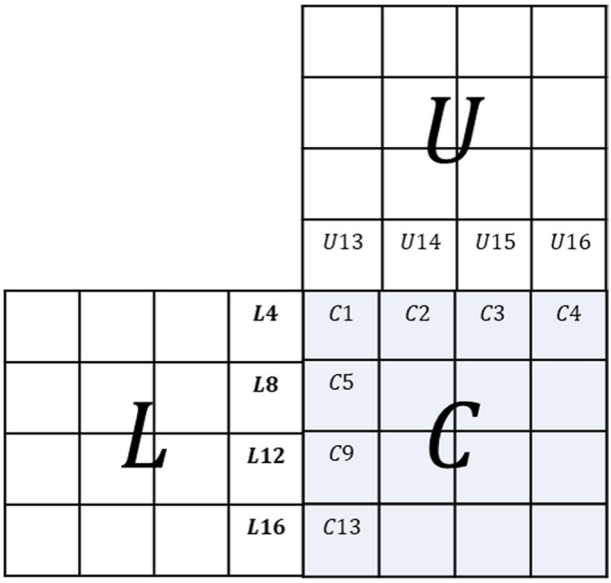


Fig. 5 The concept of SMVQ

3 Proposed scheme

The proposed RDH scheme adopts Chang et al.'s algorithm [13], which combines the AMBTC algorithm with an improved JNC algorithm utilizing XOR operations, to maintain high visual quality of stego images and to keep the sizes of output images reasonable. The goals of the proposed scheme are not only to significantly improve the compression rate but also to keep it a reversible method which can recover the AMBTC compressed images after secret information is extracted.

The proposed scheme makes total usage of data generated after the AMBTC compression in each block. The high values from blocks are collected into a high-value table and the low values from blocks are gathered as a low-value table. They are manipulated by XOR-JNC to embed secret information bits as the first part of the data hiding. The second part of the data hiding is aiming at using bitmaps. The extraction is separated into two parts as well. Figure 6 illustrates the framework of embedding and extraction of our novel scheme.

Subsections are separated into Embedding Process and Extraction Process. Inside each process subsection, it is split into two parts. Part 1 handles the embedding of value tables in 3.1.1 and the extraction of value tables in 3.2.1. Part 2 handles the processes of bitmaps generated from AMBTC. The major enhancements involved in our scheme is the embedding, extraction, and recovery of these bitmaps. The embedding process is described in subsection 3.1.2. The extraction and recovery are detailed in subsection 3.2.2.

3.1 Embedding process

The data embedding procedure in our novel scheme is separated into two parts which are not ordered, but they need to be in the same order as embedding during extraction. In our paper,

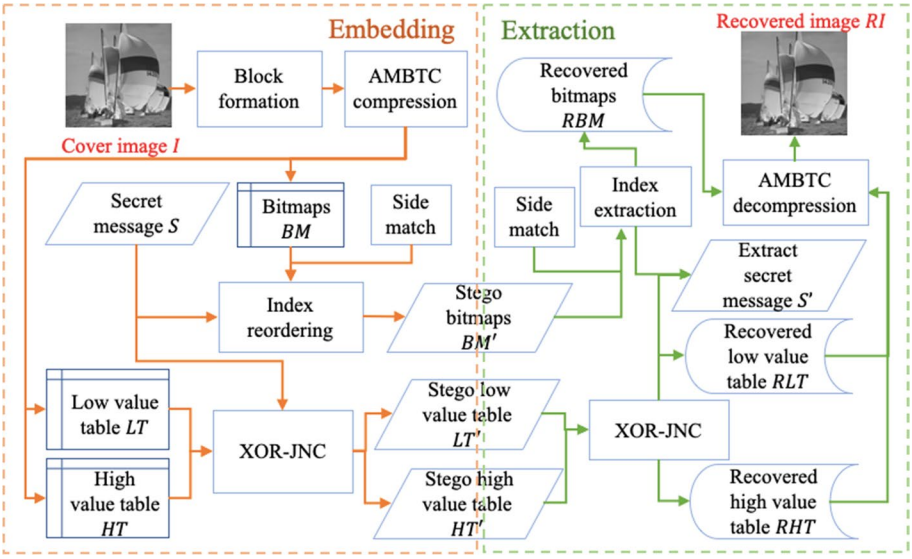


Fig. 6 Data embedding and extraction framework

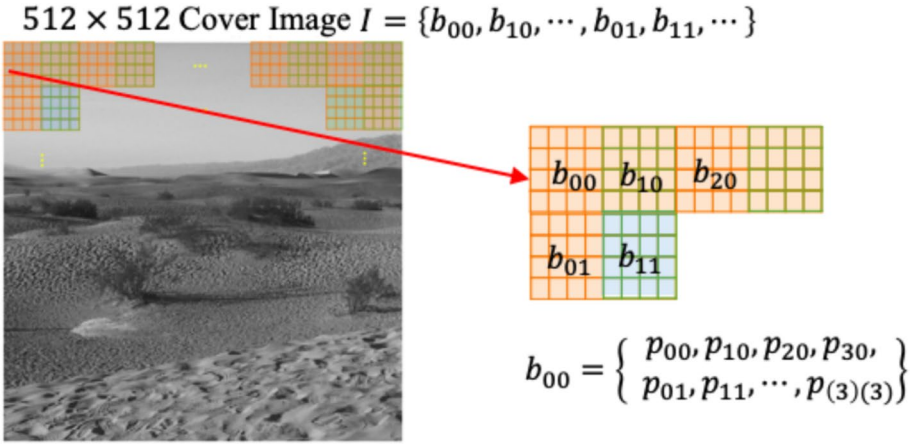


Fig. 7 An example of a covert image and a secret message is represented as

we are going to embed the secret message into the high value table and the low value table first and embed into bitmaps secondly.

Say a given cover image I is $w \times h$ in size, I is broken into B blocks as in Eq. (9) and each block b_{ij} is sized $n \times n$,

$$B = \left(b_{00}, b_{10}, \cdots, b_{(\frac{w}{n}-1)(\frac{h}{n}-1)}\right) = \left(b_{00}, b_{10}, \cdots, b_{(bw-1)(bh-1)}\right). \tag{9}$$

As described in Subsection 2.1, a high value hv_{ij}, a low value lv_{ij}, and a bitmap bm_{ij} are generated for each block b_{ij} after AMBTC compression. All high values are collected into a high table HT and all low values are collected into a low table LT for the image I.

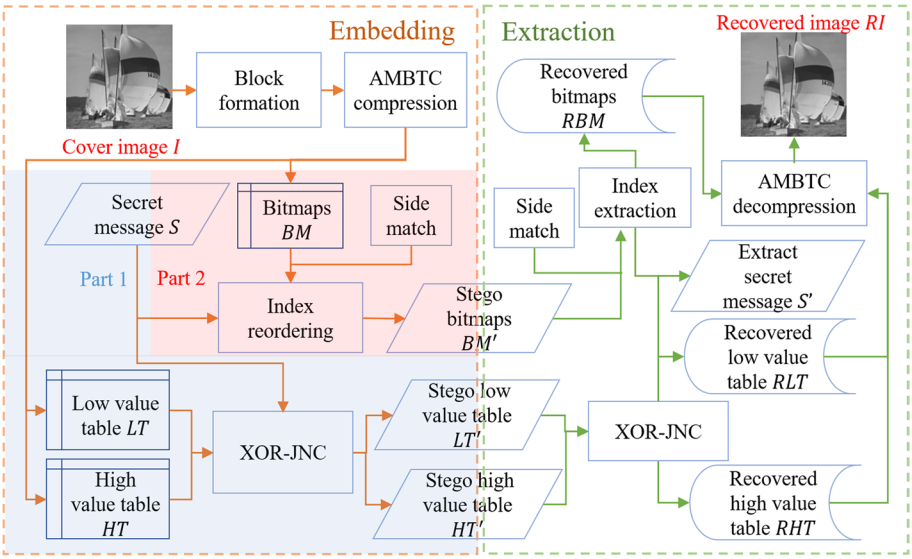


Fig. 8 Embedding consists of part 1- value tables; part 2 – bitmaps

An example of block forming for a cover image I is illustrated in Fig. 7,

$$S = 101100101 \cdots . \tag{10}$$

The details of embedding processes will be explained in the two next subsections Part 1 and Part 2. Figure 8 clarifies the two parts of the embedding process.

3.1.1 Part 1 - Value table embedding

As we mentioned earlier in Subsection 2.2, Chang et al.'s scheme [13] altered the traditional JNC to use an exclusive OR operation for difference calculation to keep the reversibility. Our goal is to improve the algorithm, thus integrating the method as a part of our proposed scheme. The first part of embedding involves the high value table and the low value table generated from AMBTC compression. The XOR join neighborhood coding (XOR-JNC) scheme is used as the first part of hiding the secret data. The concept of XOR-JNC is explained in the Subsection 2.2.1 earlier.

3.1.2 Part 2 - Bitmap embedding

Other than embedding data into the high value table and the low value table, the proposed scheme can further improve the embedding capacity (EC) by using a multi-pass bitmap embedding without disturbing the visual quality of images. It applies an index reordering technique to achieve the embedding purpose. In order to recover the bitmaps after legal receivers obtain the stego image, the side match (SM) technique is applied during the embedding stage. The SM method relies on the similarity of the pixel values along the upper and the left neighboring pixels. The general concept of the first pass of SM is described in Subsection 2.3. The proposed method will use the similarity of the upper and the left bit blocks of the processed block during the first pass, but subsequent passes will use the neighbors within the same processed bit block.

The collection of bitmaps in the cover image I is

$$BM = \left(bm_{00}, bm_{10}, \cdots, bm_{(bw-1)(bh-1)}\right), \tag{11}$$

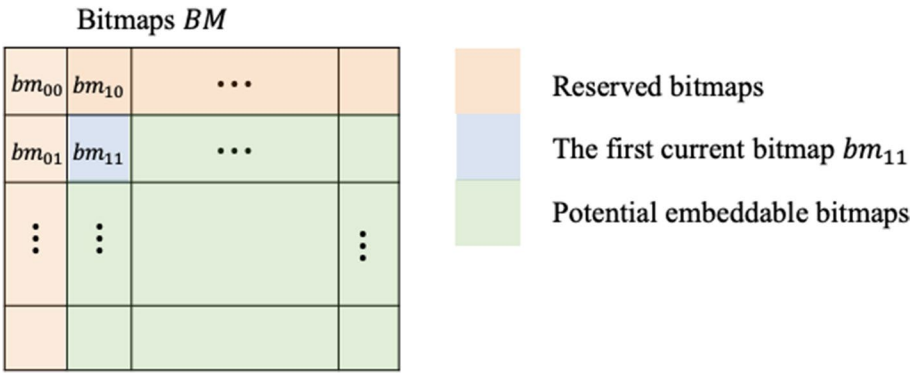


Fig. 9 Reserve the top row and the left-most column for bitmap recovery

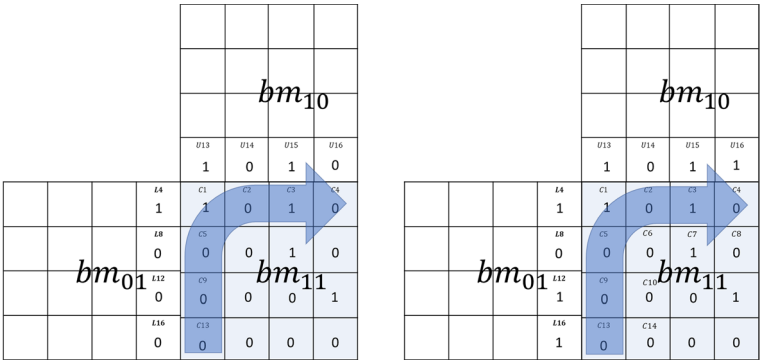


Fig. 10 Determine if a pass is embeddable or not depending on the Euclidean distance between the bits involved in current pass and its reference bits

where $bw = \frac{w}{n}$, $bh = \frac{h}{n}$ and the total number of bitmaps for the image I is $bw \times bh$. The bitmaps at the top row and the left-most column in the bitmap collection BM, as seen in Fig. 9, are reserved for bitmap recoveries.

During the first pass of bitmap embedding, the bits in the current bitmap adjacent to its left and upper neighboring bitmaps are selected for order permutations. These bits are collected into

$$C = \left(c_{00}, c_{10}, \cdots, c_{x0}, c_{01}, c_{02}, \cdots, c_{0y}\right), \tag{12}$$

where $c_{xy} = \{0, 1\}$, $0 \le x \le bw - 1$, and $0 \le y \le bh - 1$. The set can be represented as $(C, cn) = \{0^{cn(0)}, 1^{cn(1)}\}$. The total number of distinct permutations for the $n \times n$ bitmap bm_{ij} in the pass l can be calculated with Eq. (13).

$$P_l = \begin{pmatrix} 2(n-(l-1))-1 \\ cn(0), cn(1) \end{pmatrix} = \frac{2(n-(l-1))-1!}{cn(0)!cn(1)!}. \tag{13}$$

As for the second pass and onward is the same process as the first one used but the number of bits involved in the permutation will be decreased to $(n - l) \times (n - l)$, where l is the current pass. Another exception is that the upper neighboring bits and the left neighboring bits reside in the same block not its upper and left blocks.

When it is a multi-pass embedding process, the total number of bits can be embedded from the first pass and the second pass is

$$nb = \sum_{l=1}^{l=n-2} \left\lfloor \log_2 \left(P_l\right) \right\rfloor. \tag{14}$$

Before each embedding pass, there is a side matching pre-process similar to the traditional SMVQ except we use the Euclidean distance between the neighboring bits and current process bits instead. When the distance is a unique minimum distance, the pass is embeddable; otherwise, it's non-embeddable. To indicate whether a pass is embeddable or not, an extra bit stream is added for these indicators. Figure 10 shows examples of an embeddable pass and a non-embeddable pass.

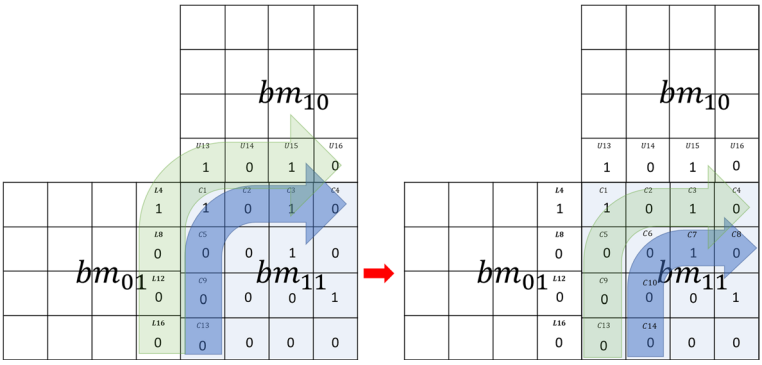
The detail process of the multi-pass bitmap embedding is stepped in Algorithm 1.

Algorithm 1 Multi-Pass Bitmap Embedding

Input:	The collection of bitmaps BM, secret S.
Output:	The collection of embedded bitmaps BM' and embeddable indicators E.
1	$BM' = \{\}$
2	$E = \{\}$
3	FOREACH bm_{ij} IN collection BM
3a	Get the size n of bm_{ij} /* bm_{ij} has the same size as b_{ij} */
3b	$bm' = copy(bm_{ij})$
3c	IF i = 0 OR j = 0 THEN Append bm' to BM' collection Continue to 3 END
3d	FOR l = 1 TO $n-2$
3da	Initialize bit value collection $C = \{\}$
3db	FOR $y = n$ TO l Append bv_{iy} to bit value collection C END
3dc	FOR $x = l$ TO n Append $bv_{x\,l}$ to bit value collection C END
3dd	Set cnt_0 to the number of 0s in collection C Set cnt_1 to the number of 1s
3de	IF cnt_0 = 0 OR $cnt_1 = 0$ THEN /* skip all 0 or 1 */ Continue to 3d END
3df	Generate the distinct permutation table PT according to cnt_0 and cnt_1 Get the size of the permutation table P_l using Eq. (13)
3dg	Collect neighboring reference bits into V
3dh	Get sorted permutation table PT' based on Euclidean distances to V Collect indices with the minimum Euclidean distance into MD

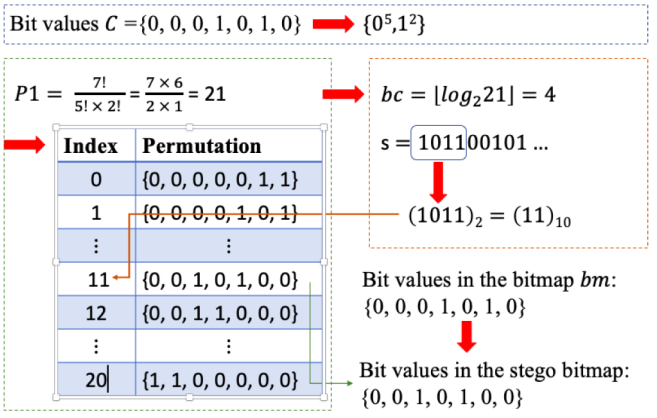
3di	IF size of (MD) = 1 AND $PT'[0] = C$ THEN
	Append 1 to embeddable indicators E
	ELSE
	Append 0 to embeddable indicators E
	Continue to 3d
	END

3dj	Calculate the total embeddable bit count bc for current pass l, $bc = \lfloor \log_2(P_l) \rfloor$
3dk	Extract bc bits from secret S Convert the bc-bit value to a decimal number ds
3dl	Use ds as the index to get the bit sequence C' from the permutation table PT'
3dm	Replace bit values in the stego bitmap with bit values in C' END
3e	Append bm' to BM' collection END
4	Export BM' and E to designated receivers

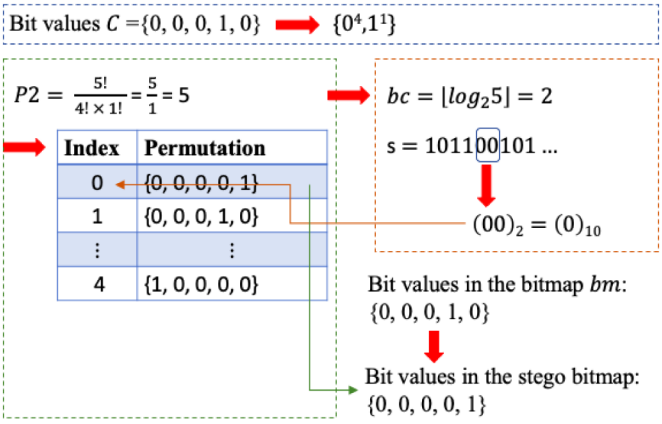


(a) The first pass (b) The second pass

Fig. 11 The first and second data embedding for a 4×4 bitmap



(a) The first pass of data embedding.



(b) The second pass of data embedding.

Fig. 12 Data embedding for bitmap

Figure 11 illustrates the essential idea of the multi-pass hiding process by using an example. Since the block size is 4×4, only the first two passes are considered effective. The last two passes cannot embed any secret bit because there is no extra gain in the embedding capacity. If bm_{11} is the current bitmap, bm_{01} is the left neighboring bitmap and bm_{10} is the upper neighboring bitmap. Figure 11(a) shows a case where the 8 referenced bits under the green arrow are outside of bm_{11} when handling the 8 bits covered by the blue arrow during the first pass. Figure 11(b) shows a case where the 6 reference bits under then green arrow are within bm_{11} itself when processing the 5 bits covered by the blue arrow during the second pass.

The example in Fig. 11 also shows the bit value collection $C = (c13, c9, c5, c1, c2, c3, c4) = (0001010)_2$ for the first pass and the bit value collection $C = (c14, c10, c6, c7, c8) = (00010)_2$ for the second pass. The sorted permutation table is ordered by their Euclidean distances to the referenced bits. The number of distinct permutations $P1$ for the first pass is 21 and the one for the second pass $P2$ is 5. Figure 12(a) and 12(b) illustrate how the two passes embed secret in the index reordering scheme.

3.2 Extraction process

When a receiver obtains the stego image with the embedding key, the secret message can be extracted. The stego high value table HT', the stego low value table LT', and the stego bitmaps BM' can be obtained after getting the AMBTC compression code. To extract secret message as well as to get the recovered high value table RHT, the recovered low table RLT, and the recovered bitmaps RBM are the two parts needed for the extracting process.

The details of embedding processes will be explained in the two next subsections Part 1 and Part 2. Figure 13 illustrates the two parts of the extraction process.

3.2.1 Part 1 - Value table extraction

The first part of extraction involves the stego high value table and the stego low value table immersed after value table embedding. The XOR join neighborhood coding (XOR-JNC) scheme is used to recover this part of the hidden secret. The concept of XOR-JNC is from Chang et al.'s scheme [13] and the extraction method was explained in Subsection 2.2.2.

3.2.2 Part 2 – Bitmap and data extraction

The collection of bitmaps in the stego image I' is

$$BM' = \left(bm_{00}', bm_{10}', \cdots, bm_{bw-1,bh-1}'\right), \tag{15}$$

where $bw = \frac{w}{n}$ and $bh = \frac{h}{n}$. The bitmaps at the top row and the left-most column of the stego image are kept unmodified during embedding. These bitmaps are used as the restoration references during the side matching recovery process. As with the embedding, the extraction is a multi-pass process corresponding to its embedding process. The bits in the current bitmap are selected the same way during the multi-pass process to generate related permutation tables. In order to recover the bitmaps, the side match (SM) technique is applied during the recovery stage as well. The steps in Algorithm II is the detailed process of the multi-pass bitmap and data extraction.

Algorithm 2 Multi-Pass Bitmap and Data Extraction

Input:	The collection of stego bitmaps BM', embeddable indicators E.
Output:	The recovered bitmaps RBM, a recovered secret S'.
1	$S' = \{\}$
2	$RBM = \{\}$
3	FOREACH bm'_{ij} IN collection BM'
3a	Get the size n of bm'_{ij}
3b	$rbm = copy(bm'_{ij})$
3c	IF $i = 0$ OR $j = 0$ THEN Append rbm to RBM collection Continue to 3 END
3d	FOR l = 1 TO $n - 2$
3da	Initialize bit value collection $C' = \{\}$
3db	FOR y = n TO l Append bv'_{ly} to bit value collection C' END
3dc	FOR x = l TO n Append bv'_{xl} to bit value collection C' END

3dd	$cnt_0 = 0$
	$cnt_1 = 0$
	FOREACH bit value c_i IN C'
	IF $c_i = 0$ THEN
	$cnt_0 = cnt_0 + 1$
	ELSE
	$cnt_1 = cnt_1 + 1$
	END

3de	IF $cnt_0 = 0$ OR $cnt_1 = 0$ THEN Continue to 3d
3df	Generate the distinct permutation table PT according to cnt_0, cnt_1 Get the size of the permutation table P_l using Eq. (13)
3dg	Collect neighboring reference bits into V
3dh	Get sorted permutation table PT' based on Euclidean distances to V Collect indices with the minimum Euclidean distance into MD'

3di	IF size of (MD') = 1 THEN
	Match C' with the elements in PT' to get the index of the matched idx
	Convert idx to a binary bit sequence and append it to the recovered secret S'
	Recover corresponding bits of C' in the current bit block rbm using PT'[0]
	ELSE
	Continue to 3d
	END
	END

3e	Append rbm to RBM collection
	END
4	Export RBM and S' to designated receivers

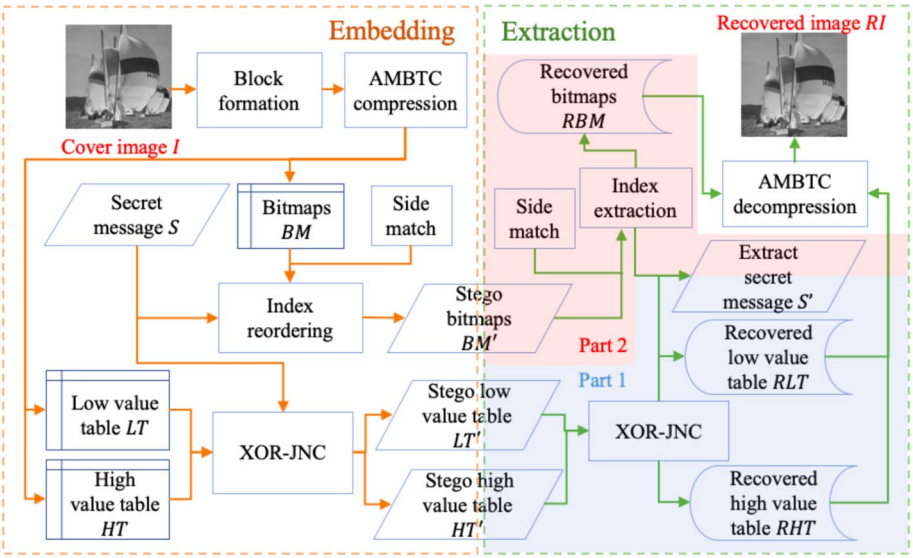


Fig. 13 Extraction consists of part 1- value tables; part 2 – bitmaps

Figure 14 elucidates the extraction details by exercising an example.

The above example shows the bit values collection $C' = (c13', c9', c5', c1', c2', c3', c4') = (0010100)_2$ for the first pass and the bit values collection $C' = (c14', c10', c6', c7', c8') = (0010100)_2$ for the second pass. The sorted permutation table is ordered by their Euclidean distances to the referenced bits. The number of distinct permutations for the first pass is $P1$ and the one for the second pass is $P2$. Figure 15(a) and Figure 15(b) demonstrate the two passes of the double-pass secret extraction.

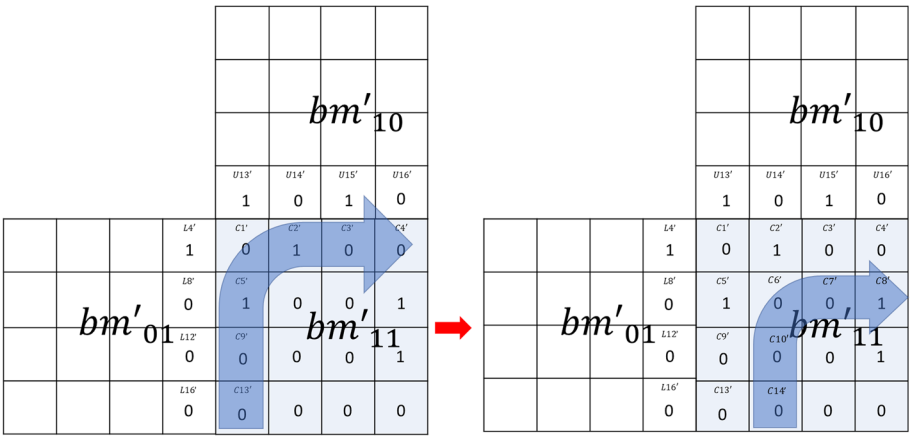
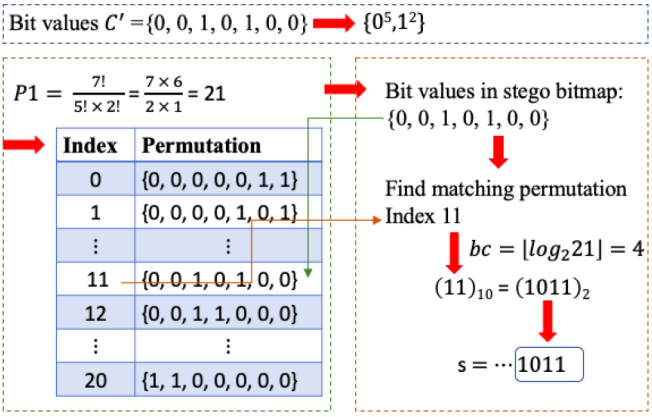
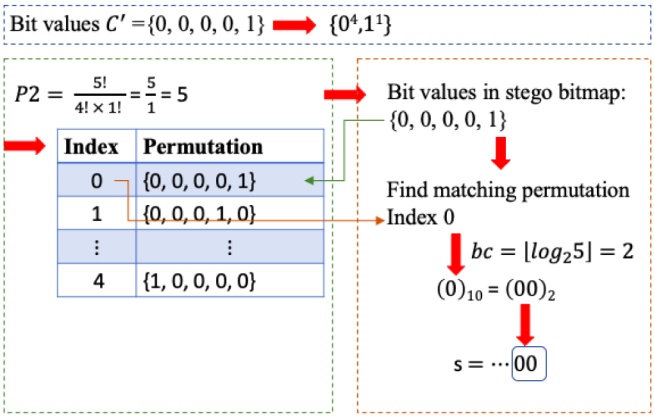


Fig. 14 Bitmap data extraction



(a) The first pass of data extraction.



(b) The second pass of data extraction.

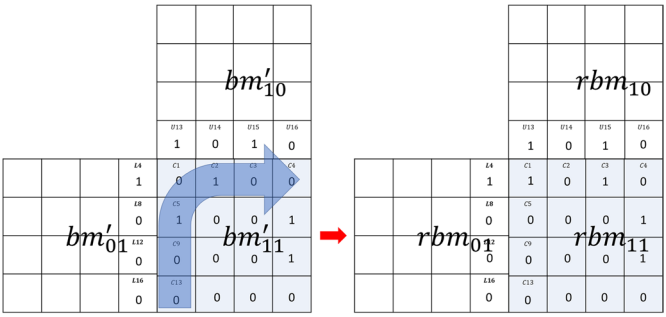
Fig. 15 Data extraction from stego bitmap

Figure 16 illustrates the steps of the double-pass bitmap replacement by exercising an example. The involved bits are set to the bit values of the first entry in the ordered permutation table. The first bit value in the ordered permutation table is the nearest bit vector to the referenced bit vector by their Euclidean distance.

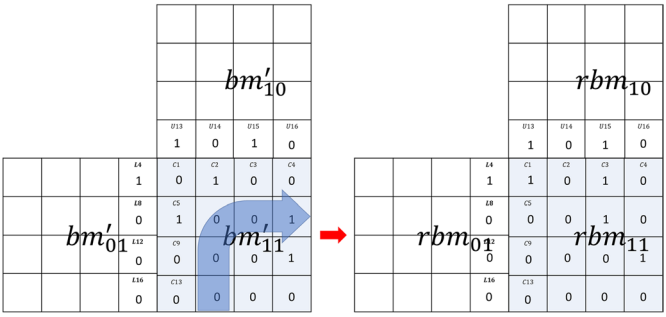
4 Experimental results

4.1 Experimental environment

The experiments were carried out on a Windows 10 laptop with a 3.20 GHz AMD Ryzen 7 CPU and 16 GB of RAM, using MATLAB R2023a.



(a) The first pass of bit value recovery.



(b) The second pass of bit value recovery.

Fig. 16 Bitmap recovery - multi-pass side matching

4.2 Experimental details

As shown in Fig. 17, we used six standard gray images: Airline, Baboon, Boat, Goldhill, Lena, and Peppers for experiments. Different images have different file sizes and embedding capacities.

Equation (16) demonstrates the calculation of the embedding rate (ER). Table 1 displays the file size, embedding capacity, and embedding rate of Chang et al.'s scheme [13] and our scheme in single-pass and multi-pass scenarios. The experimental results demonstrate that using a multi-pass approach effectively enhances the embedding rate.

$$ER = \frac{capacity}{file\ size}. \tag{16}$$

Equation (17) displays the calculation of PSNR, while Eq. (18) displays the calculation of MSE. In these equations, m and n represent the size of the image, while I and K denote the original image and the AMBTC compressed image, respectively. Table 2 and Fig. 18 show the performance comparison between our proposed scheme and other schemes. Chang et al. [13] utilized JNC and XOR implementations. Sun et al. [12] embedded data in both the high mean and low mean tables. Chang et al. [29] employed residual histogram shifting techniques. Lin et al. [11] created four disjoint sets for embeddable blocks, using different combinations of mean and standard deviation to embed the data. Our solution

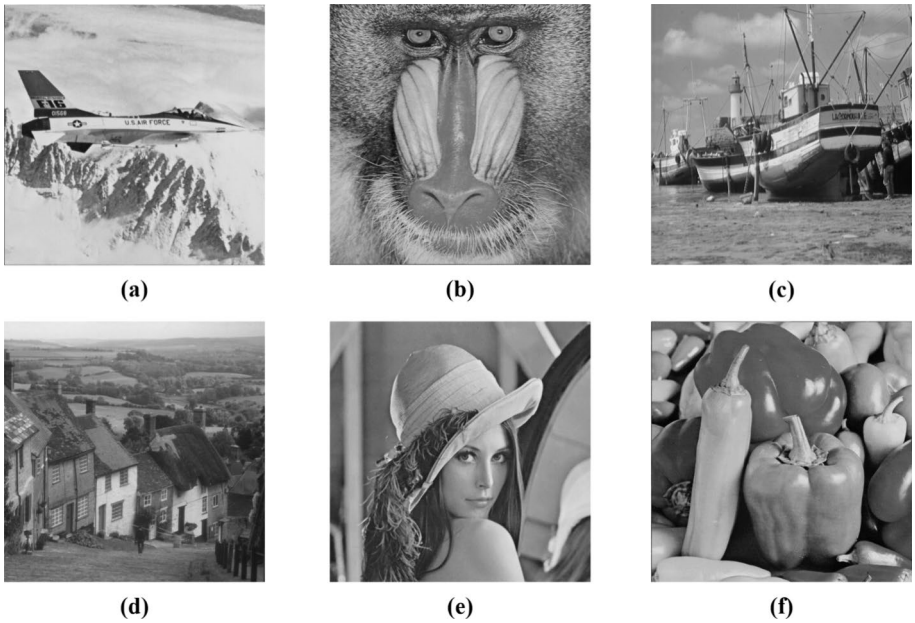


Fig. 17 Standard images sized 512×512 pixels. (**a**) "Airplane". (**b**) "Baboon". (**c**) "Boat". (**d**) "Goldhill". (**e**) "Lena". (**f**) "Peppers"

utilized SM techniques. All the schemes are fully reversible back to the AMBTC images, so the PSNR values are the same. The differences are mainly file sizes and embedding capacity. It can be seen that our scheme outperforms other schemes except for Liu et al. [11]. However, it is worth noting that the file size of Liu's scheme is much larger than ours, resulting in a lower embedding rate. Looking at the table, regardless of the image, the embedding rate of our scheme is always higher than that of other schemes.

$$PSNR = 10 \times log_{10} \frac{255^2}{MSE}. \tag{17}$$

Table 1 Comparison with different pass scenarios

Scheme	Performance	Airplane	Baboon	Boat	Goldhill	Lena	Peppers
Chang et al.'s [13]	File size (bit)	652,304	652,304	652,304	652,304	652,304	652,304
	Capacity (bit)	177,814	151,439	166,786	168,185	175,145	174,222
	ER	27.26%	23.22%	25.57%	25.78%	26.85%	26.71%
single-pass	File size (bit)	659,437	661,527	660,804	661,178	660,139	660,632
	Capacity (bit)	193,138	163,522	181,396	186,545	190,033	185,530
	ER	29.29%	24.72%	27.45%	28.21%	28.79%	28.08%
multi-pass	File size (bit)	664,985	667,264	666,583	667,206	665,588	666,234
	Capacity (bit)	202,772	170,653	189,805	196,618	198,605	191,705
	ER	30.49%	25.58%	28.47%	29.47%	29.84%	28.77%

Table 2 Comparison with other schemes

Scheme	Performance	Airplane	Baboon	Boat	Goldhill	Lena	Peppers
Proposed scheme	PSNR (dB)	33.292	27.782	31.164	33.724	33.722	34.102
	File size (bit)	664,985	667,264	666,583	667,206	665,588	666,234
	Capacity (bit)	202,772	170,653	189,805	196,618	198,605	191,705
	ER	30.49%	25.58%	28.47%	29.47%	29.84%	28.77%
Chang et al.'s [13]	PSNR (dB)	33.292	27.782	31.164	33.724	33.722	34.102
	File size (bit)	652,304	652,304	652,304	652,304	652,304	652,304
	Capacity (bit)	177,814	151,439	166,786	168,185	175,145	174,222
	ER	27.26%	23.22%	25.57%	25.78%	26.85%	26.71%
Sun et al.'s [12]	PSNR (dB)	33.292	27.782	31.164	33.724	33.722	34.102
	File size (bit)	530,856	570,348	549,620	550,088	538,928	538,968
	Capacity (bit)	64,008	64,008	64,008	64,008	64,008	64,008
	ER	12.06%	11.22%	11.65%	11.64%	11.88%	11.88%
Chang et al.'s [29]	PSNR (dB)	33.292	27.782	31.164	33.724	33.722	34.102
	File size (bit)	524,288	524,288	524,288	524,288	524,288	524,288
	Capacity (bit)	4,640	1,512	2,245	2,119	3,286	3,137
	ER	0.89%	0.29%	0.43%	0.40%	0.63%	0.60%
Lin et al.'s [11]	PSNR (dB)	33.292	27.782	31.164	33.724	33.722	34.102
	File size (bit)	1,042,312	1,045,040	1,044,800	1,044,536	1,044,576	1,044,720
	Capacity (bit)	262,016	262,144	262,144	261,968	262,112	262,144
	ER	25.14%	25.08%	25.09%	25.08%	25.09%	25.09%

$$MSE = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[I(i,j) - K(i,j) \right]^2. \tag{18}$$

Experimental results demonstrate that our solution enhances the previous solution by introducing multi-pass embedding to the bitmap, resulting in effective improvement at a lower file size cost. Our solution effectively increases the embedding rate while maintaining an unchanged PSNR.

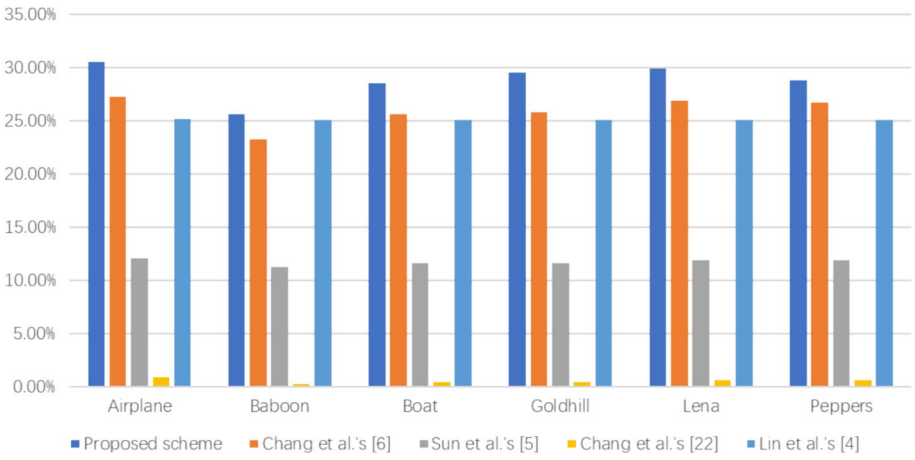


Fig. 18 Comparison of embedding rates with other schemes

5 Conclusion

After studying Chang et al.'s scheme [12], we spotted that we could further improve embedding capacity by utilizing bitmaps generated from the AMBTC compression. We applied index reordering technique to boost embedding quantity. It was desirable to keep the proposed scheme as a reversible method; thus, we applied side match technique to recover the bits altered by index reordering. After observing the experiment results, the novel scheme does offer noticeable increments in data hiding amount with little increase in the file size from Chang et al.'s scheme.

The additional process introduced to the original scheme does require more processing time and power, but we feel the additional embedding capacity provided by our scheme is worth the investment especially for applications required more embedding capacity. Future investigation of our scheme can examine the impact of the additional processing on current applications.

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Gaur VS, Sharma V, McAllister J (2023) Abusive adversarial agents and attack strategies in cyber-physical systems. CAAI Trans Intell Technol 8(1):149–165. https://doi.org/10.1049/cit2.12171
2. Ehis AMT (2023) Optimization of Security Information and Event Management (SIEM) Infrastructures, and Events Correlation/Regression Analysis for Optimal Cyber Security Posture. Archives Adv Eng Sci 1-10. https://doi.org/10.47852/bonviewAAES32021068
3. Singh A, Kumar A, Namasudra S (2024) DNACDS: Cloud IoE big data security and accessing scheme based on DNA cryptography. Front Comput Sci 18(1):181801
4. Zhang D, Shafiq M, Wang L, Srivastava G, Yin S (2023) Privacy-preserving remote sensing images recognition based on limited visual cryptography. CAAI Trans Intell Technol 8(4):1166–1177. https://doi.org/10.1049/cit2.12164
5. Namasudra S, Nath S, Majumder A. (2014) Profile based access control model in cloud computing environment. In 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE) (pp. 1-5). IEEE. https://doi.org/10.1109/ICGCCEE.2014.6921420
6. Ahmad BM, Ahmed SM, Sylvanus DE (2023) Enhancing Phishing Awareness Strategy Through Embedded Learning Tools: A Simulation Approach. Archives Adv Eng Sci 1-14
7. Namasudra S (2018) Taxonomy of DNA-based security models. In Advances of DNA computing in cryptography (pp. 37-52). Chapman and Hall/CRC
8. Tang MW, Zeng SK, Chen XL, Hu J, Du YJ (2016) An adaptive image steganography using AMBTC compression and interpolation technique. Optik 127(1):471–477. https://doi.org/10.1016/j.ijleo.2015.09.216
9. Alattar AM (2004) Reversible watermark using difference expansion of quads. Proc IEEE Int Conf Acoust Speech Signal Process 3:377–380. https://doi.org/10.1109/ICIP.2003.1247008
10. Li F, Bharanitharan K, Chang CC, Mao Q (2016) Bi-stretch reversible data hiding algorithm for absolute moment block truncation coding compressed images. Multimed Tools Appl.https://doi.org/10.1007/s11042-015-2924-7
11 Lin CC, Liu XL, Tai WL, Yuan SM (2015) A novel reversible data hiding scheme based on AMBTC compression technique. Multimed Tools Appl 74:3823–3842. https://doi.org/10.1007/s11042-013-1801-5

12. Sun W, Lu ZM, Wen YC, Yu FX, Shen RJ (2013) High performance reversible data hiding for block truncation coding compressed images. Signal Image Video Process 7:297–306. https://doi.org/10.1007/s11760-011-0238-4
13. Chang CC, Chen TS, Wang YK, Liu YJ (2018) A reversible data hiding scheme based on absolute moment block truncation coding compression using exclusive OR operator. Multimed Tools Appl 77:9039–9053. https://doi.org/10.1007/s11042-017-4800-0
14. Chang CC, Wu WC (2005) A stegangraphic method for hiding secret data using side match vector quantization. IEEE Trans Info Syst 9:2159–2167. https://doi.org/10.1093/ietisy/e88-d.9.2159
15. Sahu AK, Swain G (2019) An optimal information hiding approach based on pixel value differencing and modulus function. Wirel Pers Commun 108(1):159–174. https://doi.org/10.1007/s11277-019-06393-z
16. Sahu AK, Swain G (2020) Reversible image steganography using dual-layer LSB matching. Sens Imaging 21(1):1–21. https://doi.org/10.1007/s11220-019-0262-y
17. Wu WC (2017) Quantization-based image authentication scheme using QR error correction. EURASIP J Image Video Process 2017(1):1–12. https://doi.org/10.1186/s13640-017-0163-8
18. Lee CF, Chen KN, Chang CC, Tsai MC (2011) A hierarchical fragile watermarking with VQ index recovery. J Multimed 6(3):277–284. https://doi.org/10.4304/jmm.6.3.277-284
19. Korus P, Bialas J, Dziech A (2015) Towards practical self-embedding for JPEG-compressed digital images. IEEE Trans Multimedia 17(2):157–170. https://doi.org/10.1109/TMM.2014.2368696
20. Wang HY, Lin HJ, Gao XY, Cheng WH, Chen YY (2019) Reversible AMBTC-based data hiding with security improvement by chaotic encryption. IEEE Access 7:38337–38347. https://doi.org/10.1109/ACCESS.2019.2906500
21. Hong W, Zhou X, Lou DC (2019) A recoverable AMBTC authentication scheme using similarity embedding strategy. Plos ONE 14(2):e0212802. https://doi.org/10.1371/journal.pone.0212802
22. Lin CC, Huang Y, Tai WL (2017) A novel hybrid image authentication scheme based on absolute moment block truncation coding. Multimed Tools Appl 76(1):463–488. https://doi.org/10.1007/s11042-015-3059-6
23. Hong W, Zhou X, Lou DC, Peng C (2018) Detectability improved tamper detection scheme for absolute moment block truncation coding compressed images. Symmetry 10(8):318. https://doi.org/10.3390/sym10080318
24. Chen TH, Chang TC (2018) On the security of a BTC-based-compression image authentication scheme. Multimed Tools Appl 77(10):12979–12989. https://doi.org/10.1007/s11042-017-4927-z
25. Zhong H, Liu H, Chang CC, Lin CC (2016) A novel fragile watermark-based image authentication scheme for AMBTC-compressed images. J Inf Hiding Multimed Signal Process 7(2):362–375
26. Li W, Lin CC, Pan JS (2016) Novel image authentication scheme with fine image quality for BTC-based compressed images. Multimed Tools Appl 75(8):4771–4793. https://doi.org/10.1007/s11042-015-2502-z
27. Kim T (1992) Side match and overlap match vector quantizers for images. IEEE Trans Image Process 1(2):170–185. https://doi.org/10.1109/83.136594
28. Lema MD, Mitchell OR (1984) Absolute moment block truncation coding and its application to color images. IEEE Trans comm COM 32:1148–1157. https://doi.org/10.1109/TCOM.1984.1095973
29. Chang IC, Hu YC, Chen WL, Lo CC (2015) High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding. Signal Process 108:376–388. https://doi.org/10.1016/j.sigpro.2014.09.036