A Data Hiding Scheme Based on Turtle-shell for AMBTC Compressed Images

Chin-Feng Lee[1], Chin-Chen Chang[2,3,*], Guan-Long Li[2]
[1] Department of Information Management, Chaoyang University of Technology
Taichung, 41349, Taiwan
[Email: lcf@cyut.edu.tw]
[2] Department of Information Engineering and Computer Science, Feng Chia University
Taichung, 40724, Taiwan
[Email: alan3c@gmail.com]
[3] School of Computer Science and Technology, Hangzhou Dianzi University
Hangzhou 310018, China
*Corresponding author: Chin-Chen Chang

Received December 29, 2019; revised March 26, 2020; accepted April 12, 2020; published June 30, 2020

Abstract

Data hiding technology hides secret information into the carrier, so that when the carrier is transmitted over network, it will not attract any malicious attention. Using data compression, it is possible to reduce the data size into a small compressed code, which can effectively reduce the time when transmitting compressed code on the network. In this paper, the main objective is to effectively combine these two technologies. We designed a data hiding scheme based on two techniques which are turtle-shell information hiding scheme and absolute moment block truncation coding. The experimental results showed that the proposed scheme provided higher embedding capacity and better image quality than other hiding schemes which were based on absolute moment block truncation coding.

Keywords: Data hiding, image compression, absolute moment block truncation coding, turtle-shell

1. Introduction

Due to the development of the Internet, transferring data over the Internet has become very common. For a secure transmission, data hiding technology makes use of a reliable carrier such as video, audio, image or text to hide secret information. This makes it more difficult for the attacker to detect the existence of a secret message. The current data hiding techniques can be divided into two categories: reversible data hiding and irreversible data hiding. Reversible data hiding enables the recipient to recover the original carrier after extracting the secret information. This technique is carried out by shifting the pixel values. Users can embed secret data in the space emptied after shifting pixel values. However, in order for the carrier to recover original image, the reversible data hiding technique cannot have a large amount of embedding capacity. In the case of irreversible data hiding, it does not recover the original image, however, it has a higher embedding capacity.

Previous researchers [1-13] have proposed various embedding methods in order to achieve irreversible data hiding,. These methods can be further divided into three categories: Least Significant Bit (LSB) [4-6], Pixel Value Difference (PVD) [7-9], and Exploiting Modification Direction (EMD) [11-13]. LSB [4-6] embeds the data by modifying the lowest number of bit in the pixel, demonstrating a high embedding capacity. Depending on the number of LSBs per pixel, the embedding capacity ranges from 1 to 8 bits per pixel (bpp). However, using this method causes large damage to the image which is easily detectable by the attacker. PVD [7-9] determines the embedding capacity by calculating the pixel value difference. It improves the distortion rate of the image, even though it has a small amount of embedding capacity, which is about 0.2bpp. EMD [11-13] uses a reference table or embedding function to embed secret data. It embeds data by changing pixels to match secret data in a reference table or embedding function. The embedding capacity of the method has a minimum of 1 bpp. It can hide more secret data having the same image quality. Each of the three types of embedding techniques mentioned above demonstrate different performance levels in terms of image quality and embedding capacity.

The main goal of data hiding is to obtain a high embedding capacity and a high image quality. However, it is seen that an increase in embedding capacity tends to reduce the image quality. Therefore, researchers have been carrying out in-depth research in order to provide improvisations to the existing methods. Turner [5] introduced the concept of LSB replacement in 1989. Although this method was simple and had good embedding capacity, the image quality decreased rapidly as the amount of embedded data increased. In 2002, Tseng et al. [10] proposed a binary image block hiding scheme. This method used block partitioning to get more embedding space and hid secret data into digital matrix of the image to reduce image distortion. In 2006, Zhang et al. [11] proposed a new data hiding method. This method used a magic matrix to create a reference table. The reference table made it easier to embed secret data and later extract it. In 2014, Chang et al. [12] proposed a data hiding scheme based on a turtle-shell. The turtle-shell was mapped to the magic matrix, where each turtle-shell contained 8 digital numbers. Due to the nature of the turtle-shell, each pixel pair could embed 3 bits of secret data. This method obtained a higher image quality compared to Sudoku. Taking Lena as an example, Sudoku's image quality was 44.9dB, and the image quality of this method was 49.4 dB. Also the pixel value difference between the stego image and the original image was very small. In 2016, Liu et al. [13] also proposed a high-capacity data hiding scheme based on turtle-shell. It had a location table which showed relationships between all the numbers present on the turtle-shell matrix. Pixels

could embed more data using a location table. This scheme had a high embedding capacity and similar image quality. After hiding secret data, the size of the stego image remained same as the original image.

In order to reduce the amount of data that is transmitted, some experts have proposed image compression techniques which compress the size of the original image before transmission. An image compression technique converts images into compressed codes using algorithms. Using such techniques, the number of bits required for each pixel gets reduced. Also, the file requires less resources when it is stored. The sender needs to transmit the compression code and the parameters required by the algorithm to the receiver on the other end. The receiver generates a compressed image when they receive the compression code and other parameters. In the past, several researchers have proposed different methods to execute image compression [14-18]. For example, in 1979, Delp and Mitchell [17] proposed an image compression scheme using block truncation coding (BTC). It did not require additional parameters in the generation of compressed images and was better than other compression techniques that require several parameters. It first divided the original image into blocks of $n \times n$ pixels and calculated the mean and standard deviation of each block. The mean and standard deviation were then used to calculate the quantization levels and the bitmap for each block. The quantization levels and bit maps of all blocks were combined into the compressed code. The receiver used compression code to recover the image. The block mean and standard deviation of the recovered image were similar to the original mean and standard deviation, so the image quality of the recovered image was same as the original image. In 1984, Lema and Mitchell [18] proposed absolute moment block truncation coding (AMBTC). This method was based on BTC image compression technique. For each image block, AMBTC did not use the standard deviation but instead used the first absolute moment of a block and its mean. This method was computationally simpler than block truncation coding, and the error of the image was also smaller. Taking the airplane as an example, the mean square error of BTC was 32, and the mean square error of AMBTC was 30.

The main objective of this paper is to combine both image compression and information hiding, so that we can hide the secret data into a compressed code. Recently, several researchers [19-25] have proposed different hiding methods in compressed images. In 2015, Ou et al. [22] proposed an AMBTC-based data hiding scheme (referred to as AMBTC_HPISMD). In their technique, data was only embedded in the bitmap of the smoothing block. The method set a threshold, and the block was classified as a smooth block depending on whether the difference d=A-B of the quantization level indicator (A, B) was less than the threshold or not. This method only changed the value of the smooth block, so the peak signal-to-noise ratio (PSNR) of the image was very good. In 2017, Huang et al. [23] proposed an AMBTC-based data hiding scheme (referred to as AMBTC_Hybrid). It embedded data in the bitmap of the smooth and complex blocks. So the image was 50,000 bits higher than the AMBTC_HPISMD scheme of Ou et al. [22]. In 2017, Malik et al. [24] proposed a data hiding scheme based on modified AMBTC (referred to as AMBTC_Modify). Their method upgraded the bitmap to 2 bits and the quantization level to four. AMBTC_Modify enhanced the PSNR of the image by sacrificing a part of the compression ratio and additionally using two quantization level indicators. In addition, AMBTC_Modify could hide more secret bits in each block. However, the additional quantization level indicator and the 2-bit bitmap made the compression ratio only half of AMBTC. AMBTC_Modify also determined whether the block was a smooth block or a complex block through the threshold. In the smooth block, the secret data replaced the bit map of the block to form a new bitmap. In a complex block, the secret data replaced the LSBs of its bitmap. In 2019, Kumar et al.

[25] proposed an AMBTC-based data hiding scheme (abbreviated as AMBTC_HDPVD). Their method used Hamming distance and PVD to embed secret data. The blocks were first classified into three categories, smooth blocks, low complexity blocks and high complexity blocks. AMBTC_HDPVD used a replacement strategy to embed data into smooth blocks. Low complexity blocks used Hamming distance to embed secret data, whereas highly complex blocks used PVD to embed secret data.

Different data hiding techniques perform differently under the same image compression techniques. In order to obtain better embedding capacity and high image quality, this paper proposes a data hiding method in the compression domain. The proposed scheme uses absolute moment block truncation coding (AMBTC) and modified absolute moment block truncation coding (M-AMBTC), respectively. The secret data is embedded into the quantization levels and bitmap of each compressed block based on the turtle-shell data hiding strategy.

The rest of this paper is organized as follows. We will briefly review AMBTC and the turtle-shell-based data hiding scheme in Section II. The data hiding scheme proposed in this paper will be described in detail in Section III. The experimental results are given in Section IV. Finally, the conclusion will be provided in Section V.

2. Related Works

In this section, we will first introduce absolute moment block truncation coding (AMBTC), which was proposed by Lema and Mitchell [18]. It was used for compression of grayscale images. Thereafter, we will introduce more methods such as the modified AMBTC proposed by Malik et al. [24] and the data hiding scheme based on turtle-shell, proposed by Liu et al. [13].

2.1 Absolute Moment Block Truncation Coding

AMBTC is a lossy image compression scheme for grayscale images. It does not require additional parameters to aid in the generation of the compressed image. In this method, when an image is divided into blocks, each image block has two quantization levels and a bitmap. The bit map consists of 0s and 1s and is used to record the position of each pair of quantization levels in each block. First, a block mean is obtained by taking average of all the pixels in a block. Accordingly, the pixels smaller than or equal to the block mean are treated as group 1. On the other hand, pixels greater than the block mean are treated as group 2. Then, a small mean and a large mean are calculated by taking the averages of all the pixels in groups 1 and 2 respectively. The small mean and large mean in each block are called lower quantization level and higher quantization level. These two quantization levels mainly record two block mean values for the purpose of recovery. AMBTC has two steps, an image encoding step and an image decoding step.

In the encoding procedure, the digital image of the $W \times H$ pixels is first divided into blocks of $n \times n$ pixels. The size of the block determines the image quality and the compression ratio. If n is large, the image will have a high compression ratio, but the image quality becomes poorer. Users can adjust the size of n according to their preferences. The numbers in each block are added and divided by the number of pixels in the block. Then sample mean μ for each block is calculated using Eq. (1).

$$\mu = \frac{1}{n \times n} \sum_{i=1}^{n \times n} x_i, \tag{1}$$

where x_i represents the pixel value and $0 \leq x_i \leq 255$. After doing the calculation, a mean of the blocks will be obtained. Each pixel in the block is compared to the mean value. The pixel values in the block can be divided into two categories: smaller than the mean value, and larger than or equal to the mean.

Thereafter, the bitmap is created. The creation rule is: When the pixel belongs to a value less than the mean value μ, it is marked as "0" on the bitmap; otherwise, its value is marked as "1" on the bitmap.

After creating the bitmap, we need to calculate the quantization level of each block. The pixel values in the block are mapped to a bitmap and the average of the two categories is calculated. The average value labeled "0" is the quantization level *A*, and the average value labeled "1" is the quantization level *B*. In quantization level *A*, all pixel values labeled "0" are added and divided by the number labeled "0." We get the quantization level *A* as in Eq. (2).

$$A = (1/\sum_{x_i<\mu} 1) \sum_{x_i<\mu} x_i \quad . \tag{2}$$

In quantization level *B*, all pixel values labeled "1" are added and divided by the number labeled "1." We get the quantization level *B*, using Eq. (3).

$$B = (1/\sum_{x_i\geq\mu} 1) \sum_{x_i\geq\mu} x_i \quad . \tag{3}$$

After the bitmap generation and quantization level calculations, the image is compressed into a compressed code using AMBTC. When it is necessary to decompress the image, the average (*A*, *B*) needs to fill in the position of (0, 1) on the bitmap to form an image.

At the stage of image decoding, the receiver acquires the compressed code. The cover image can be recovered by filling the bitmap with the quantization level. The position marked 0 in the bitmap is filled with the quantization level *A*, and the position marked 1 in the bitmap is filled with the quantization level *B*. All quantization levels are populated at the corresponding locations in the bitmap. It can thereafter be recovered to a similar image. Due to the AMBTC recording quantization levels (*A*, *B*) and the bitmap, the image cannot recover the complete original image. However, the human eye still cannot distinguish image differences.

The modified AMBTC and AMBTC proposed by Malik et al. [24] have similar encoding and decoding procedures. The difference is that AMBTC has a higher compression ratio. The compression ratio of AMBTC is 0.25, while the compression ratio of modified AMBTC is 0.5. First, the method calculates the average of the blocks and divides the pixel values in the block into two categories. The average of the two classes is calculated and the pixel values in the block are divided into four categories. The bitmap is created in this step. The pixel values are marked in the bit map from small to large in order of "00," "01," "10," and "11." After creating a bitmap, we need to calculate the quantization level of each block. Four quantization levels are obtained by calculating the average of each class. After the bitmap generation and quantization level calculation, the image is compressed into a compressed code using the modified AMBTC. When the image needs to be decompressed, the four quantization levels are filled into the positions on the bitmap to form an image.

2.2 The Data Hiding Scheme Based on Turtle-shell

In 2016, Liu et al. [13] proposed a high-capacity data hiding scheme based on turtle-shells. It uses a turtle-shell matrix to hide secret data and a location table to increase the embedding

capacity. Liu et al.'s scheme could hide 4 bits of secret data in pixel pairs, which means that each pixel hides 2 bits of secret data on an average. In order to hide secret data with a size of two digits, it needs a location table. The location table is constructed based on the relationship of the shell matrix. The secret bits refer to the location table to determine the change in pixel values. The pixel pair is then changed corresponding to the turtle-shell matrix to hide secret data. The location table lists all the relationships in the turtle-shell matrix.

Because secret data requires a shell matrix to hide values, first we need to create a shell matrix of size 256×256. The turtle-shell matrix is mainly composed of a number of continuous hexagons, each of which is called as turtle-shell. Each turtle-shell contains 8 numbers, including two numbers on the back and six edges. The range of numbers is from 0 to 7. The difference between two adjacent numbers of each column alternates between 2 and 3. Similarly, the difference between two adjacent numbers for every row is 1. Part of the turtle-shell matrix is shown in **Fig. 1**.

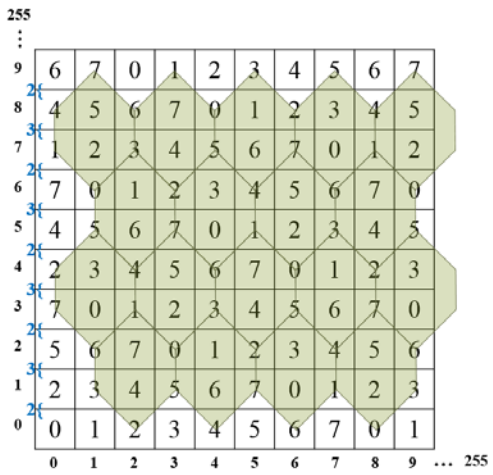


Fig. 1. An example of turtle-shell matrix

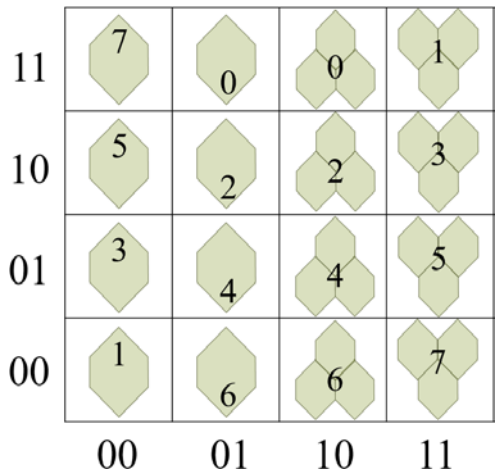


Fig. 2. An example of location table

All the elements in the turtle-shell matrix can be divided into two categories: general and special elements. The general element is the element on the edge of the turtle or inside the turtle-shell. The special element is the element outside the turtle-shell. We need to collect all the possibilities of the general elements and each of the secret data that may be representative to form a location table. The location table is shown in **Fig. 2**. The secret data is hidden in different ways for different elements.

For general elements, if the extracted pixel pair is a number contained in the turtle-shell, the secret data S_i and S_{i+1} can be extracted respectively. The i^{th}secret data corresponds to the value of the row of the location table, and the $(i+1)^{th}$ secret data corresponds to the value of the column of the location table. All the secret messages can be obtained from the location table. In order to match the value of the position table, it is necessary to find the value of the nearest pixel after matching the value. When the pixel is modified, it contains the embedded secret message.

For special elements, if the extracted pair of pixels does not contain the number in the turtle-shell, the pair of pixels is replaced by the number itself or closest value to the secret bit.

Due to the nature of the turtle-shell matrix, nearby numbers contain all the possibilities. First, the pixel values are mapped to the turtle-shell matrix. Then nearby pixel pairs are looked to match to the secret message and the original pixel pair is modified. The modified pixel values are mapped to the turtle-shell matrix. The value obtained is the secret message. The value of the pixel pair is modified to complete hiding the secret message.

With respect to the extraction procedure, the secret data is extracted from the stego pixel pair. The stego pixel pairs are then mapped into the turtle-shell matrix to get the position of the value in the matrix. Then the position is mapped to the location table and the secret message is extracted using the location table.

Suppose we have a pixel pair (2, 2) and want to hide the secret message $(01\ 10)_2$ into the pixel pair. The secret message is mapped to the location table. It indicates that the modified value is 2 as shown in **Fig. 3(b)**, and the value is on the back of the turtle-shell. Then we map the pixel pair (2, 2) into the turtle-shell matrix. The value of the pixel pair (2, 2) is 7. It means that we need to find the nearby value. We find that the pixel pair (3, 3) is matching pixel pair. Pixel pair (2, 2) is then modified to the pixel pair (3, 3) as shown in **Fig. 3(a)**. When the secret message needs to be extracted, pixel pair (3, 3) is mapped to the location table to get the binary secret message $(01\ 10)_2$.

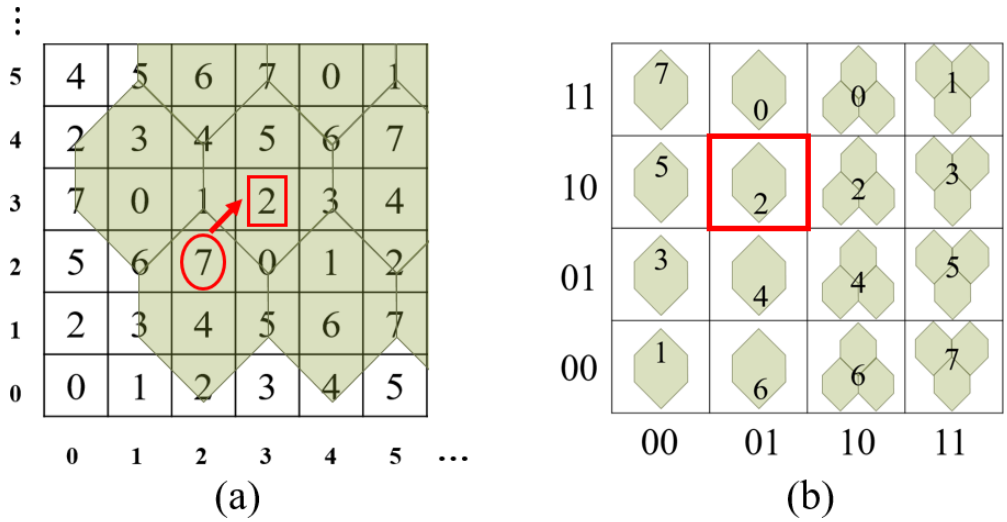


Fig. 3. An example of embedding procedure based on turtle-shell

3. The Proposed Scheme

In this section, we present a data hiding scheme for the AMBTC and the modified AMBTC. In AMBTC-based compressed images, we use AMBTC to compress images. The grayscale image is divided into non-overlapping blocks of *n*×*n* pixels. Each block calculates its own mean and quantization levels. We use the modified AMBTC to compress the image. The quantization levels are extended to 4 quantization levels rather than 2. Accordingly, the bitmap is expanded to a 2-bit bitmap. When we get the quantization levels and the bitmap, we can hide the secret data into the quantization levels and the bitmap. Due to the characteristics of the turtle-shell-based scheme, each pixel can carry 2 bits of secret information, and each pixel moves by no more than 3, thus achieving higher storage and good image quality. Based on the AMBTC compressed image, our method embeds the secret

data in the turtle-shell matrix composed of the quantization levels. In addition, after the secret data have been embedded in the quantization levels, we used threshold *TH* to classify the block as a smooth or a complex block. The secret data is further hidden in the bitmap of the smooth block and the order of the quantization levels is used to hide secret data. The flowchart of the proposed method is shown in **Fig. 4**.

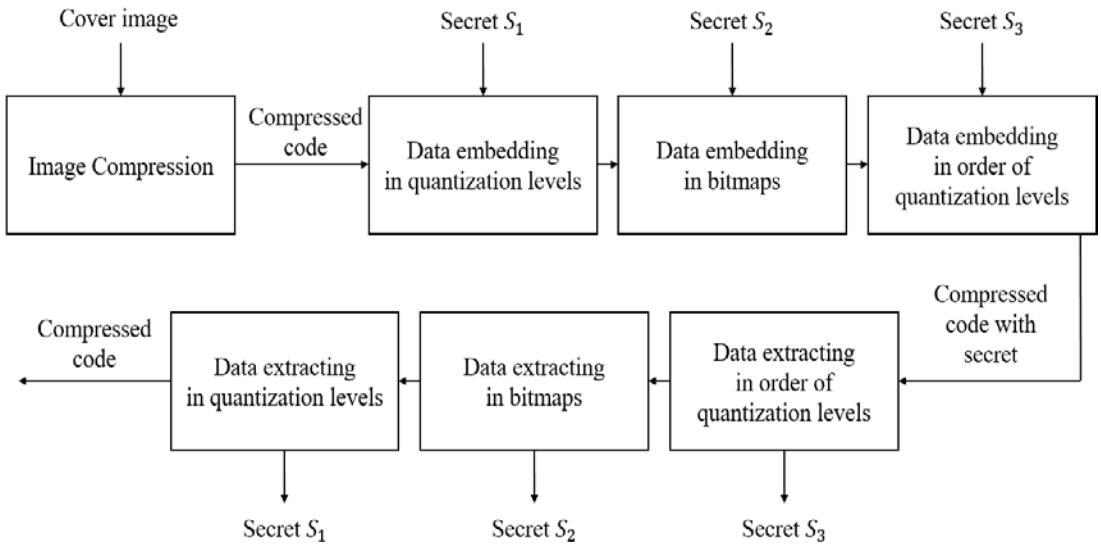


Fig. 4. A flowchart of the proposed procedure

The proposed technique has two procedures: data embedding and data extraction which will be presented as follows.

3.1. Embedding procedure

In the data embedding procedure, we have three stages to hide secret data into the quantization levels of each block as well as the corresponding bitmap. First, we need to build a turtle-shell matrix. The size of the turtle-shell matrix is 256×256. The size of the matrix is determined by the range of pixel intensity. For grayscale images, the pixel value is an 8-bit data value (with a range of 0 to 255). The turtle-shell matrix is a 2D hyper-cubes (squares), each of which corresponds to a pair of gray-values. The turtle-shell matrix consists of hexagonal turtle-shells. Each turtle-shell contains 8 numbers, including 2 numbers on the back of the turtle-shell and 6 numbers on the edge of the turtle-shell. The range of numbers is from 0 to 7. In the turtle-shell matrix, the difference between two adjacent columns alternates between 2 and 3. The difference between adjacent numbers in the same row is 1. When the turtle-shell matrix is created, we can embed the secret data into the quantization levels. We use the quantization levels (A, B) of each block, treated as a pixel pair in the turtle-shell. The value A maps to the row vector on the turtle-shell matrix and B maps to the column vector on the turtle-shell matrix. Thereby, the position on the turtle-shell matrix is obtained. The pixel pair (A, B) is mapped to the position table. When the value at the position table (A', B') matches the secret data $S_1 \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, ($A$, B) is modified to (A', B') to complete embedding the secret data S_1 hidden in the quantization levels.

In order to embed the secret data, association set of the location must be found. The association set can be found according to the following rules.

Rule 1. If the quantization levels are the numbers on the turtle-shell, the association set is a

collection of all numbers in a 5×5 size sub-block centered at the quantization level (A, B). Due to the architectural properties of the turtle-shell, all possibilities of the position table are included in the sub-block.

Rule 2. If the quantization level is not included in any turtle-shell, the association set is a collection of all numbers in a 3×3 sub-block. The quantization level must be in the sub-block. Due to the architectural properties of the turtle-shell, all sub-blocks contain 8 possibilities.

After the secret data has been embedded into quantization levels, we need to check the absolute difference $D=|A'- B'|$ of the new quantization level (A', B'). We use threshold TH to classify the differences. In the smooth block, the difference D between the quantization levels A' and B' is small and less than the pre-determined threshold. Therefore, embedding secret data in the corresponding bitmap with small differences is not obvious to the image. When the threshold TH is large, the embedding capacity increases. Conversely, when the threshold TH is small, the image quality improves. The threshold TH can be set by the user as needed. We do not hide secret data in the bitmap of a complex block because the distortion in the complex blocks is easy to be detected by the human eye. When the block is a smooth block, the secret data S_2 of length block size in bits can be embedded into the bitmap using bit replacement.

After embedding, the secret data is substituted in the bitmap. We can further hide another one-bit secret data S_3 in the order of the new quantization level (A', B'). Since the secret data is embedded in the quantization level, we must first ensure the order of A' and B'. When $A'> B'$, the binary value of the location is additionally recorded. After the recording is complete, we exchange A' and B' to ensure that A' is less than or equal to B'. Then we can hide the secret data according to the order of the quantization levels. First, the record of the location is hidden in the order of (A', B'), and other secret bits are hidden in the remaining space. When the secret bit is 0, the order of A' and B' does not change. When the secret bit is 1, the position of (A', B') is exchanged. The embedding procedure is presented in detailed steps below:

Input: The original image I of size $M\times N$ pixels, threshold TH, S_1: secret data to-be-hidden in quantization levels, S_2: secret data to-be-hidden in bitmap, S_3: secret data to-be-hidden in the order of new quantization levels, and a location table.

Output: the AMBTC compressed codes with secret data.

Step 1: The original image I is imported and AMBTC is used to form the AMBTC compression code.

Step 2: A turtle-shell matrix is generated and the quantization levels (A, B) of each block are mapped into the matrix.

Step 3: Mapping S_1 to the position table yields the corresponding value, and the value is mapped to the position (A', B') closest to the quantification of the shell matrix. Modify (A, B) to (A', B') to match the secret data S_1.

Step 4: When the secret data S_1 are embedded, the difference value $D = |A'-B'|$ is calculated to determine whether the to-be-processed block is smooth or complex.

Step 5: When $D<TH$, the block is a smooth block. Therefore, the bitmap Bm of the block can hide the secret data. Substituting S_2 for the original bitmap Bm forms a new bitmap Bm'.

Step 6: Ensure the order of A' and B' after secret data S_1 have been embedded. When $A' > B'$, a binary bit “1” corresponding to the location (A', B') is recorded which indicates that

we will exchange A' and B' to ensure that A' is less than or equal to B'.

Step 7: Embed S_3 into the order of the quantization levels. When S_3 is 0, do not change the order of A' and B'. When S_3 is 1, the order of A' and B' is modified to (B', A'). When A' is equal to B', S_3 is not embedded in the order of this quantization level.

3.2. Extraction procedure

In the data extraction phase, we receive the quantization levels, the bitmap and the threshold *TH*. First, we check the order of the quantization levels (A, B). If A is less than B, the one-bit secret data S_3 is 0. If A is greater than B, the secret data is 1. When A is equal to B, there is no secret data. Thereafter, we can recover the original order based on the overhead information to recover the positions of (A, B). Then we check the absolute difference $D=|A'-B'|$ in the quantization levels. If the difference D is less than or equal to the threshold *TH*, then the bitmap of the block has a secret message S_2. We can get secret data in order from this bitmap. After all the secret data is extracted from the bitmap, the quantization level can be mapped to the turtle-shell matrix to obtain the position of the value according to each pair of quantization levels. The location then maps the location table to obtain the secret data S_1. Through these methods, all secret data $S_1 \| S_2 \| S_3$ or $S_1 \| S_2$ can be extracted. The extraction procedure is explained in detailed steps as below:

Input: AMBTC compressed codes with secret, *TH*: threshold, location table

Output: S_1: secret data in quantization levels, S_2: secret data in bitmap, S_3: secret data in the order of quantization levels

Step 1: Check the order of the quantization levels. If A is less than B, the secret message S_3 is 0. If A is greater than B, the secret message S_3 is 1. When A is equal to B, this order of the quantization levels does not have the secret data S_3. And the value of the position of the exchanged order is obtained.

Step 2: After all the secret data S_3 is obtained, the order of the quantization levels (A, B) is recovered by the value of the position. The quantization level (A, B) is modified to (A', B').

Step 3: Calculate the absolute difference value $D= |A'- B'|$ of the quantization levels. If the difference D is less than or equal to the threshold *TH*, then the bitmap of the block carries the secret data S_2.

Step 4: After the secret data S_2 is obtained, the quantization levels are mapped to the turtle-shell matrix to obtain the position of the value. Map the location to the location table to get the corresponding secret data S_1.

3.3. Illustration of the proposed scheme

In this section, we will illustrate an example using the proposed data hiding scheme. **Fig. 5** shows the embedding procedure of the secret data in the original block. We set the threshold *TH*=10. The secret data stream $S = S_1 \| S_2 \| S_3$ that needs to be embedded is $(1100\ 1110101001100100\ 1)_2$. The secret data S_1 = {1100} is the first 4 bits of S. Secret data S_2 = {1110101001100100} has a total of 16 bits and S_3 = {1}. Suppose we have a block of 4×4 as shown in **Fig. 5(a)**. This block contains the original pixel values {2, 5, 5, 3, 1, 4, 3, 2, 3, 4, 5, 6, 2, 5, 6, 8}. It is compressed using AMBTC technique which generates a compression code (A, B, Bm) = (2, 5, Bm), where Bm is a 4×4 bitmap as shown in the **Fig. 5(b)**. In the process of data hiding, we first compare S_1 = {1100} with the position table (as shown in **Fig. 2**) of the turtle-shell. It indicates that the quantization levels (A, B) = (2, 5) need to be modified. The quantization levels (2, 5) correspond to the value on the edge of the turtle-shell. We

search in the turtle-shell matrix. It gives the closest point as (A', B') = (3, 5). So the new compression code becomes (A', B', Bm) = (3, 5, Bm) as shown in **Fig. 5(c)**. Then we check the absolute difference D= |A'- B'| of the quantization levels A' and B'. Because D= |A' - B'|=|3-5|<TH, we replace the original bitmap Bm with the secret 16-bit secret data to get the new bitmap Bm' as shown in **Fig. 5(d)**. In addition, we confirm $A' < B'$, i.e., 3<5. It shows that secret data can be hidden in the order of quantization levels. Since S_3 is 1, the quantization levels (3, 5) must be exchanged. Finally, the new compression code is (5, 3, Bm') as shown in **Fig. 5(e)**. At the time of data extraction procedure, we follow the same process, however in the reverse order.

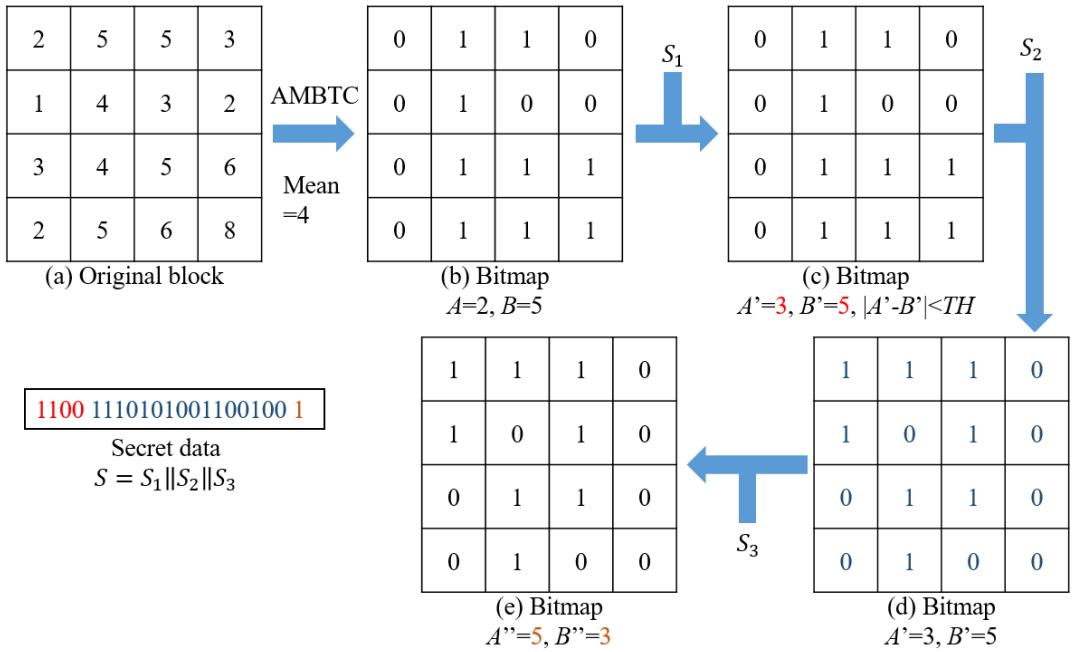
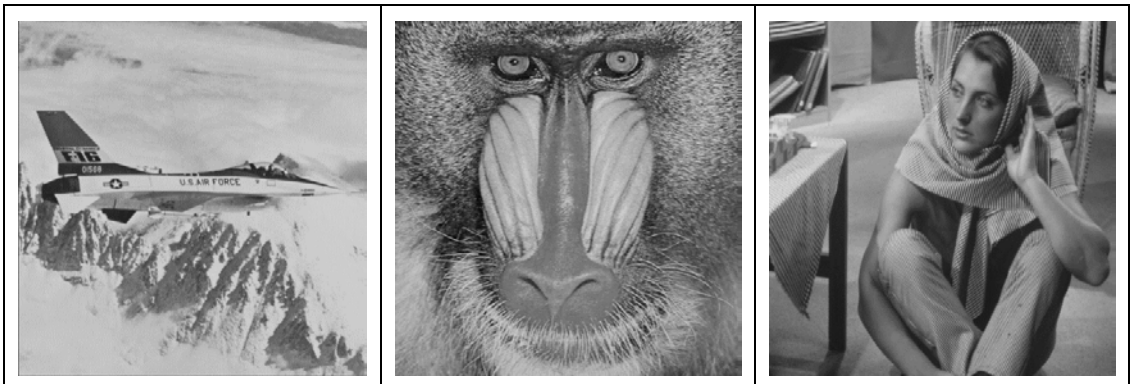


Fig. 5. An example of the proposed procedure based on AMBTC

4. Experimental Results



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

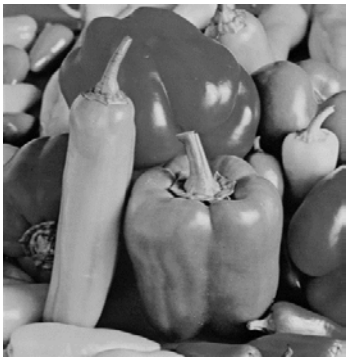
		
(d) Boats	(e) Lena	(f) Peppers

Fig. 6. Cover images of the size 512×512

We carried out the experiment using six grayscale images of size 512×512. As shown in the **Fig. 6**, the test images used were Airplane, Baboon, Barbara, Boat, Lena and Peppers. The secret data used time function to generate a pseudo-random number. The secret data was in the binary format, which consisted of 0s and 1s. The environment used in this experiment was Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 8 GB RAM PC. The operating system used was Windows 10 Professional.

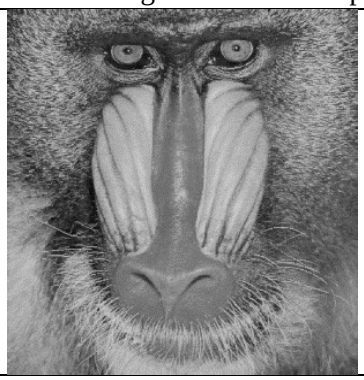
We used peak signal-to-noise ratio (PSNR) to evaluate image quality of the processed image as defined in Eq. (8).

$$\mathrm{PSNR} = 10\log_{10}\frac{255^2}{MSE} \ . \tag{8}$$

The mean square error (MSE) is defined in Eq. (9).

$$\mathrm{MSE} = \frac{1}{N\times N}\sum_{i=1}^{N}\sum_{j=1}^{N}(I_{ij} - I'_{ij})^2, \tag{9}$$

where I_{ij} is the pixel location of the original image I, which is located in the i-th row and the j-th column. And I'_{ij} is the pixel position of the secret image I', which is located in the i-th row and the j-th column. When the PSNR is 30 dB or higher, the human eye usually cannot recognize the difference between original and the compressed image.

		
(a) Airplane using AMBTC PSNR: 33.26 Compression ratio: 25%	(b) Baboon using AMBTC PSNR: 27.77 Compression ratio: 25%	(c) Barbara using AMBTC PSNR: 29.85 Compression ratio: 25%

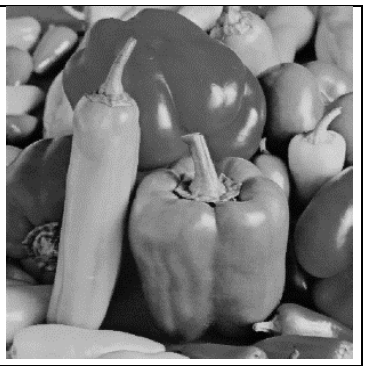
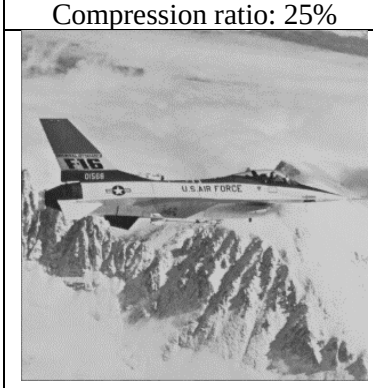
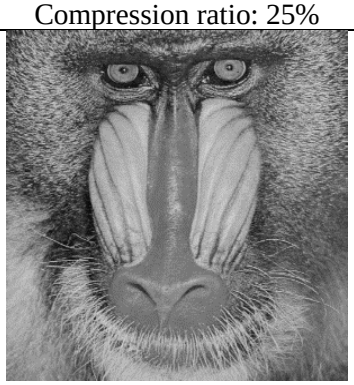
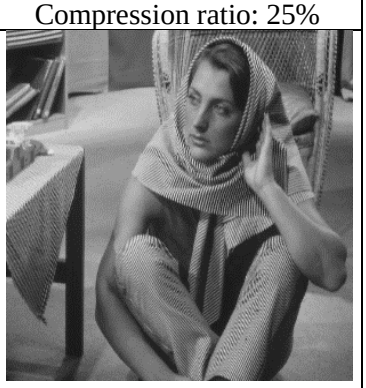
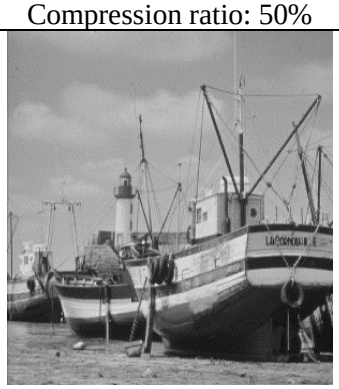
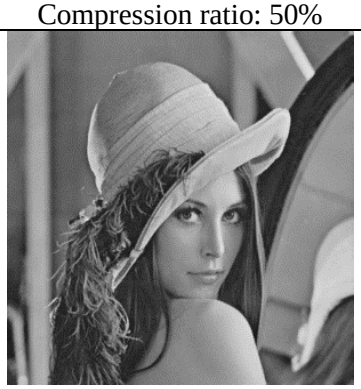
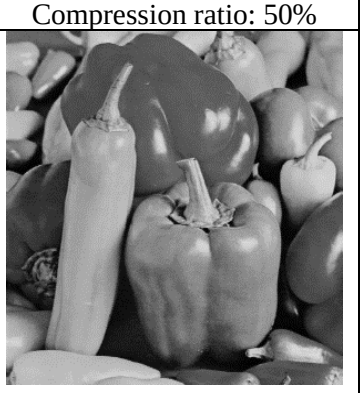
		
(d) Boats PSNR: 31.96 Compression ratio: 25%	(e) Lena PSNR: 33.69 Compression ratio: 25%	(f) Peppers PSNR: 34.06 Compression ratio: 25%
		
(g) Airplane using Modified AMBTC PSNR: 40.22 Compression ratio: 50%	(h) Baboon using Modified AMBTC PSNR: 34.34 Compression ratio: 50%	(i) Barbara using Modified AMBTC PSNR: 36.64 Compression ratio: 50%
		
(j) Boats using Modified AMBTC PSNR: 38.95 Compression ratio: 50%	(k) Lena using Modified AMBTC PSNR: 40.25 Compression ratio: 50%	(l) Peppers using Modified AMBTC PSNR: 40.7 Compression ratio: 50%

Fig. 7. Compressed images of the size 512×512

The images were compressed using the original AMBTC technique as shown in **Fig.s 7(a)-7(f)**. On the other hand, the images were compressed using modified AMBTC technique as shown in **Fig.s 7(g)-7(1)**. Even though the human eye cannot perceive the differences in

compressed images between the two compression techniques, the compressed image using modified AMBTC technique showed better image quality. The modified AMBTC technique improved image quality by reducing the compression ratio. The 2-bit bitmap and the 4 quantization levels had a more accurate recovery effect on the images. After compression, we the quantization levels and a corresponding bitmap were obtained. The threshold was predetermined to decide whether the block is smooth or complex. The value of the threshold had an impact on image quality and the embedding capacity. As the threshold increases, the embedding capacity increased. However, at the same time, the image quality decreased.

In the AMBTC-based scheme, different thresholds were used to test the scheme on various images as shown in **Fig.s 7 (a)-(f)**. The experimental results are shown in **Table 1** and **Fig. 8**. They show PSNR, embedding capacity, and the number of bits hidden by each pixel point (bpp). In the experiment, the larger the threshold, the higher will be the embedding capacity. As the amount of embedding capacity increased, the image quality decreased. We set the threshold values range from 0 to 30 to classify the blocks into smooth and complex blocks. It was found that with a threshold of more than 30, the PSNR of all images was below 30 dB. The human eye is able to perceive the differences in images when the PSNR is below 30 dB. Therefore, the experiment was not carried out with higher thresholds. According to the experimental results, the image “Pepper” performed well in terms of embedding capacity. Pepper's embedding capacity increased up to 1.22 bpp as the threshold increased. However, its PSNR dropped to 29.85 dB at *TH*=30. It was found that the image “Baboon” showed bad performance. When the threshold was 0, the data was only hidden in the quantization level (*A*, *B*). We also discovered that only when the block was a single color block, the data was hidden in the bitmap. Baboon's PSNR was less than 30 dB and its embedding capacity was only 0.31 bpp. Even when the threshold was set to 30, the amount of capacity was only 0.88 bpp. On an average, each pixel of the image could hide secret data that was close to 1 bit. It was found that smooth images performed better, and complex images had smaller reserves due to a large number of complex blocks. The user could set the trade-off between image quality and embedding capacity according to the requirements of the application.

Table 1. Experimental results of the proposed scheme on AMBTC-compressed images with different threshold *TH*

Image	**Metrics**	***TH*** = 0	***TH*** = 10	***TH*** = 20	***TH*** = 30
Airplane	PSNR	33.078	32.287	31.012	29.76
	capacity	95690	262202	293178	307194
	bpp	0.36	1.00	1.11	1.17
Baboon	PSNR	27.72	27.5	26.627	25.271
	capacity	82010	134522	191962	232714
	bpp	0.31	0.51	0.73	0.88
Barbara	PSNR	29.768	29.339	28.52	27.145
	capacity	85985	206657	241025	268177
	bpp	0.32	0.78	0.91	1.02
Boat	PSNR	31.819	31.252	29.924	28.452
	capacity	96065	238897	277265	299697
	bpp	0.36	0.91	1.05	1.14
Lena	PSNR	33.482	32.376	30.766	29.336
	capacity	92015	257935	299247	317055
	bpp	0.35	0.98	1.14	1.20

Pepper	PSNR	33.842	32.452	31.067	29.851
	capacity	88895	271535	307359	320943
	bpp	0.33	1.03	1.17	1.22
Average	**PSNR**	**31.61**	**30.86**	**29.65**	**28.30**
	capacity	**90110**	**228624**	**268339**	**290963**
	bpp	**0.34**	**0.87**	**1.02**	**1.10**

As shown in **Fig.s 7 (g)-(l)**, the scheme based on modified AMBTC was implemented using different thresholds. The experimental results are shown in **Table 2** and **Fig. 9**. The table shows the PSNR values with different images and thresholds, the embedding capacity and the number of bits hidden by each pixel point (bpp). According to the experiment, the "Airplane" image performed very well. As the threshold increased, the Airplane's capacity increased to 2.15 bpp. When the threshold was 30, its PSNR was 32.27 dB. However, Baboon's performance was still bad. When the threshold was 0, Baboon's PSNR was only 34 dB. When the threshold was 30, the capacity was only 1.33 bpp. On an average, each pixel of the image carried data that was close to 2 bits.

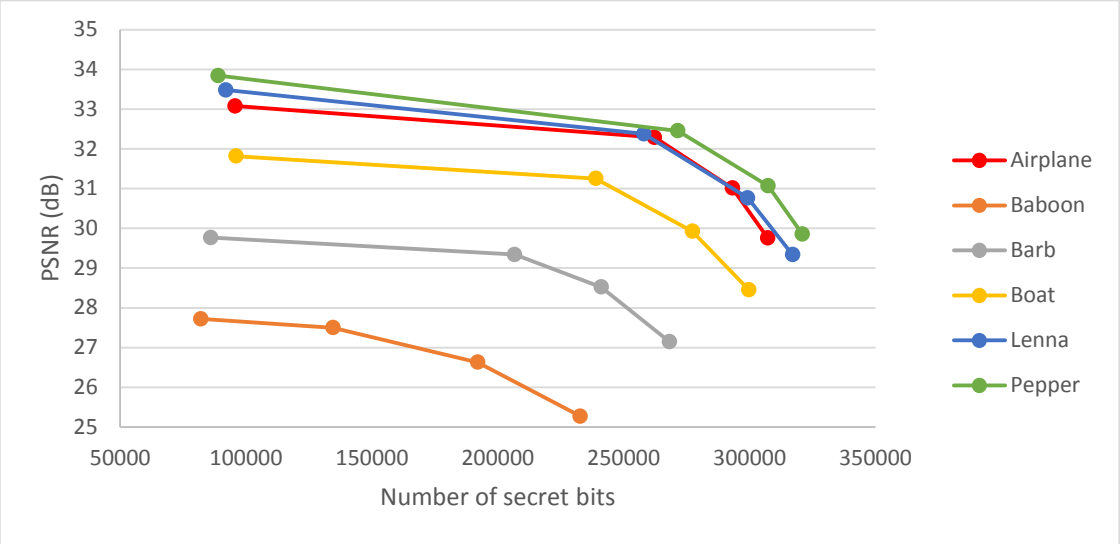


Fig. 8. Comparison of performance between images using the proposed scheme based on AMBTC technique

Table 2. Experimental results of the proposed scheme on modified AMBTC-compressed images with different threshold *TH*

Image	Metrics	***TH*** = 0	***TH*** = 10	***TH*** = 20	***TH*** = 30
Airplane	PSNR	38.978	35.755	34.032	32.277
	capacity	161778	456690	527858	566130
	bpp	0.61	1.74	2.01	2.15
Baboon	PSNR	34.103	33.854	32.156	30.174
	capacity	147487	182879	282463	350847
	bpp	0.56	0.69	1.07	1.33

Barbara	PSNR	36.236	34.488	32.932	31.343
	capacity	149006	331886	417518	458958
	bpp	0.56	1.26	1.59	1.75
Boat	PSNR	38.258	36.509	34.523	32.044
	capacity	154803	416307	482643	533779
	bpp	0.59	1.58	1.84	2.03
Lena	PSNR	39.29	34.152	32.549	31.061
	capacity	153036	420908	530956	577804
	bpp	0.56	1.60	2.02	2.20
Pepper	PSNR	39.607	34.589	32.711	31.459
	capacity	150153	426889	553929	594025
	bpp	0.57	1.62	2.11	2.26
Average	PSNR	37.745	34.891	33.150	31.393
	capacity	152710	372593	465894	513590
	bpp	0.58	1.42	1.77	1.95

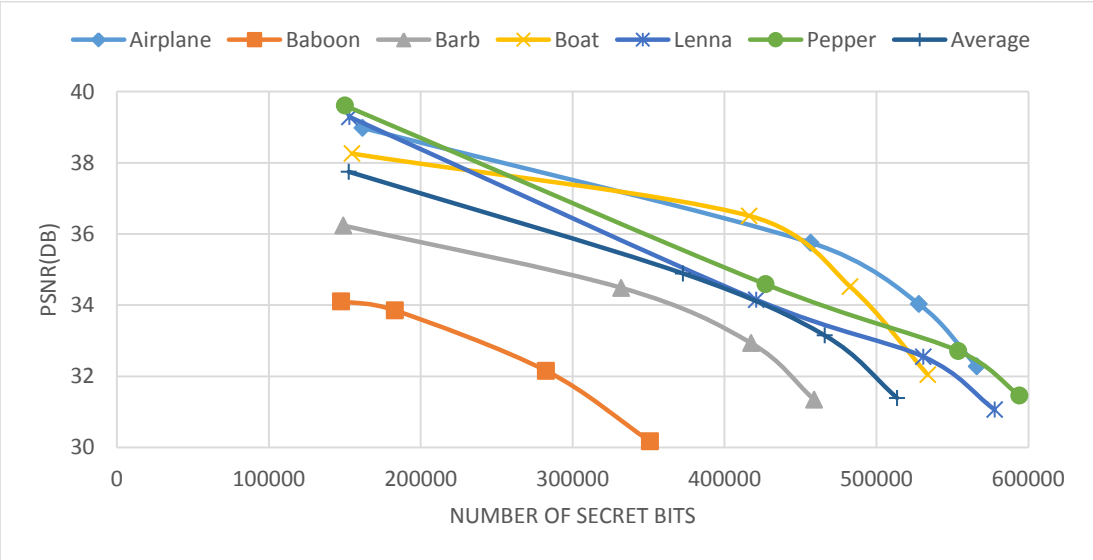
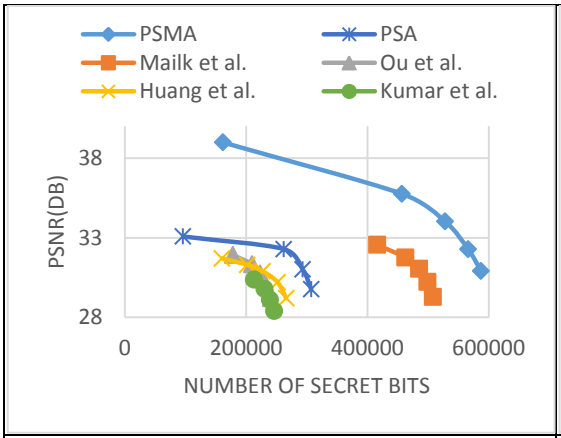


Fig. 9. Comparison of performance between images using the proposed scheme based on modified AMBT technique

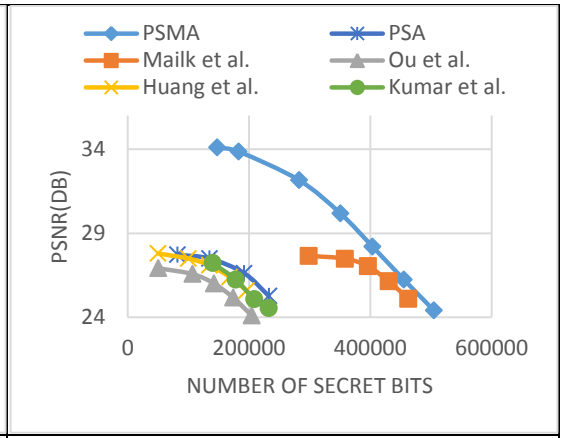
In addition to experimenting with different images using the proposed scheme, we also compared AMBTC-based solutions with other AMBTC-based data hiding schemes (eg Ou et al. [22], Huang et al. [23] and Kumar et al. [25]). In this paper, all AMBTC-based data hiding schemes had a compression ratio of 0.25. We mainly compared the performance of different thresholds. As can be seen in **Fig. 10**, the proposed scheme based on AMBTC (PSA) was superior to the scheme of Huang et al. [23]. The values of PSNR and embedding capacity were better. In **Fig. 10**, we used PSA and Kumar et al.'s scheme [25] to test the images. The PSA applied different thresholds from 0 to 30. The thresholds of Huang et al. [23] and

Kumar et al. [25] ranged from 10 to 50. In the PSA, the embedding capacity between 0 and 10 varied greatly. When the threshold was 0, the secret message was mainly embedded in the quantization levels. However, when the threshold was 0, it had a higher PSNR. As it can be seen from **Fig. 10**, when the threshold was from 0 to 10, the images of "Airplane," "Boats," and "Lena" changed more than "Baboon." Since the number of smooth blocks in the "Baboon" image was less than those of the other three images, the bitmap of the "Baboon" image retained less secret data. In addition, when the threshold was small, the schemes of Ou et al. [22] and Huang et al. [23] performed better than PSA method using the images of "Lena" or "Pepper. When the threshold was 0, the proposed scheme mainly embedded data in the quantization levels. Only a small amount of data was embedded in the bitmap. When the threshold was large, PSA was still better than Ou et al. [22], Huang et al. [23] and Kumar et al. [25]. We can say that PSA's data hiding ability is better than the recent AMBTC-based data hiding schemes.

The difference between the modified AMBTC and AMBTC is the amount of quantization levels and bitmap. When the image was compressed using modified AMBTC, it generated four quantization levels and a two-bit bitmap. In this paper, all data hiding schemes based on modified AMBTC had a compression ratio of 0.5. The modified AMBTC compression ratio was only half of that of AMBTC, but the modified AMBTC compressed image had a higher PSNR than the AMBTC compressed image. Using the modified AMBTC, the proposed method concealed only two quantization levels (*Am*, *Bm*). In the smooth block, the secret data replaced the original bitmap. In a complex block, there was no secret data in the bitmap. The secret data and the abnormal location were embedded into the order of the quantization levels. The proposed scheme based on modified AMBTC (PSMA) was compared to the method of Malik et al. [24]. They had proposed a data hiding scheme based on the modified AMBTC in 2017. In **Fig. 10**, the images were tested using PSMA and Malik et al. [24]. The threshold for PSMA was set 0 to 40. The threshold for Malik et al. [24] was 10 to 50. Although the thresholds of the two methods were different, the two methods had similar embedding capacity in this range. When the *TH* of PSMA was 0, the image quality of PSMA was much higher than that of Malik et al. [24]. Since there were fewer hidden secret messages, the stego image had less distortion and the PSNR was very similar to the modified AMBTC compressed image. It was also found that the image quality of PSMA was higher than that of Malik et al. [24]. Compared to the scheme of Malik et al. [24], PSMA was closer to the original image.



(a) Airplane



(b) Baboon

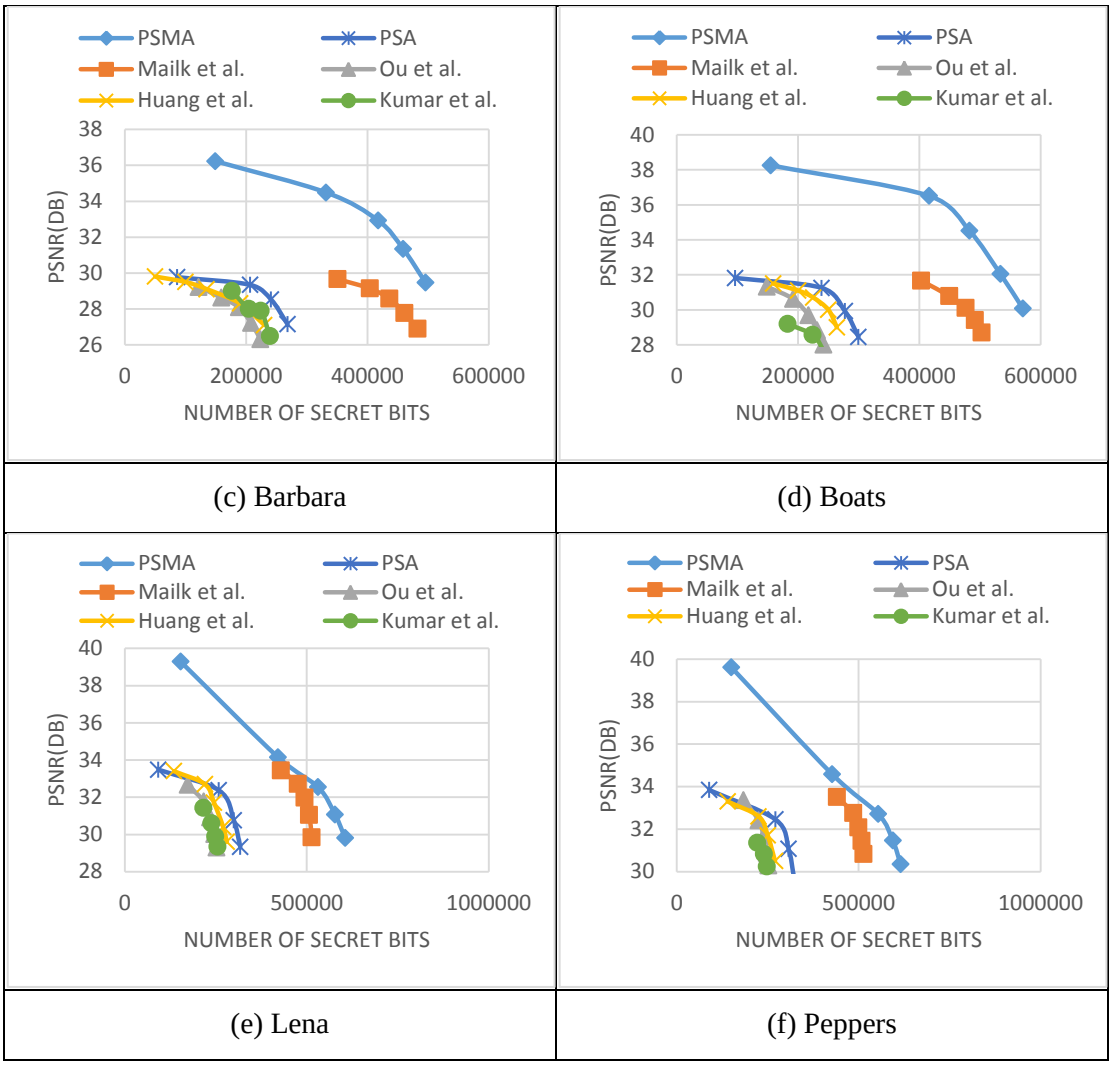


Fig. 10. Comparison of performance between the proposed scheme and other schemes

The stego image of the proposed scheme is shown in **Fig. 11**. It can be seen that images based on the AMBTC scheme had a better compression ratio, whereas, images based on the modified AMBTC scheme had a better image quality. It was also found that at the same threshold levels, images based on the modified AMBTC scheme could carry more secret data as compared to other scheme.

(a) The proposed scheme based on AMBTC PSNR: 33.69 dB Embedding capacity: 0.98 bpp Compression ratio: 0.25	(b) The proposed scheme based on modified AMBTC PSNR: 40.25 dB Embedding capacity: 1.60 bpp Compression ratio: 0.5

Fig. 11. Stego images of the size 512×512 of the proposed scheme with TH=10

Suppose the attacker knows that the embedding method uses a hexagonal turtle shell to carry secret information, he/she must obtain the key to generate the reference matrix. Since the value of the hexagonal turtle-shaped shell ranges from 0 to 7, the probability of a hexagonal turtle-shell matrix is 8! i.e. 40320 different types. Besides, the proposed method adopted a location table based on the relationship of the shell matrix to increase the embedding capacity. There are 4! * 4! = 567 kinds of tables. Accordingly, even the attacker knows the secret data is hidden using the turtle-shell method, he/she may have 8! * 567, which is 22861440 chances using brute force to guess the reference matrix and location table.

Secure multimedia distribution is a practical application while transmitting multimedia content from the sender to the receivers. When transmitting multimedia content from the sender to the receiver, secure multimedia distribution is a practical application, and confidentiality is one of the necessary conditions. Thus it is rational to encrypt the secret message before it is embedded and transmitted.

5. Conclusions

In this paper, we proposed a compression-based data hiding scheme. The scheme involves embedding secret data into the compressed block using the turtle-shell method. The compressed image can be generated by using either of the two schemes such as AMBTC or modified AMBTC. After obtaining the compressed images, we used the data embedding technique of turtle-shell. The quantization levels of each block carried at least 4 bits of secret data. A large amount of data was hidden in the bitmaps of smooth blocks as well as the quantization levels. The turtle-shell's data embedding scheme made the image less distorted. When the threshold was low, the PSNR of the stego image was close to the AMBTC

compressed image. This scheme performed well on both compression techniques of AMBTC or modified AMBTC. In terms of image quality and data hiding, the proposed scheme was superior compared to other data hiding schemes using AMBTC compression and modified AMBTC compression schemes.

Acknowledgment

This research was partially supported by the Ministry of Science and Technology of the Republic of China under the Grants MOST 108-2221-E-324-014.

References

[1] C. Qin, C. C. Chang, Y. H. Huang and L. T. Liao, "An Inpainting Assisted Reversible Steganographic Scheme Using a Histogram Shifting Mechanism," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, pp. 1109-1118, Jul. 2013. Article (CrossRef Link)

[2] K. H. Jung and K. Y. Yoo, "Data Hiding Scheme in Binary Images Based on Block Masking for Key Authentication," *Information Sciences*, vol. 277, pp. 188-196, Sept. 2014. Article (CrossRef Link)

[3] A. Westfeld, "F5: A Steganographic Algorithm," *Lecture Notes in Computer Science*, vol. 2137, pp. 289-302, 2001. Article (CrossRef Link)

[4] J. Mielikainen, "LSB Matching Revisited," *IEEE Signal Processing Letters*, vol. 13, pp. 285-287, May 2006. Article (CrossRef Link)

[5] L. F. Turner, "Digital Data Security System," *Patent IPN, WO 89/08915*, available on Jan. 2019. Article (CrossRef Link)

[6] M.U. Celik, G. Sharma, A.M. Tekalp and E. Saber," Lossless generalized-LSB data embedding," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 253-266, Feb. 2005. Article (CrossRef Link)

[7] D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recognition Letters*, vol. 24, no. 9–10, pp. 1613-1626, June 2003. Article (CrossRef Link)

[8] C. M. Wang, N.I. Wu, C. S. Tsai and M. S. Hwang, "A high quality steganographic method with pixel-value differencing and modulus function," *Journal of Systems and Software*, vol. 81, no. 1, pp. 150-158, January 2008. Article (CrossRef Link)

[9] W. Luo, F. Huang and J. Huang, "A more secure steganography based on adaptive pixel-value differencing scheme," *Multimedia Tools and Applications*, vol. 52, no. 2–3, pp 407–430, April 2011. Article (CrossRef Link)

[10] Y. C. Tseng, Y. Y. Chen and H. K. Pan, "A Secure Data Hiding Scheme for Binary Images," *IEEE Transactions on Communications*, vol. 50, pp. 1227-1231, Jul. 2002. Article (CrossRef Link)

[11] X. Zhang, S. Wang, "Efficient Steganographic Embedding by Exploiting Modification Direction," *IEEE Communications Letters*, vol. 10, no.11, pp.781-783, November 2006. Article (CrossRef Link)

[12] C. C. Chang, Y. Liu and T. S. Nguyen, "A Novel Turtle-shell Based Scheme for Data Hiding," in *Proc. of Tenth Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing(IIHMSP'14), Kitakyushu, Japan*, pp.89-93, 27-29 August 2014. Article (CrossRef Link)

[13] Y. Liu, C. C. Chang and T. S. Nguyen, "High Capacity Turtle-shell-based Data Hiding," *IET Image Processing*, vol. 10, pp.130-137, Feb. 2016. Article (CrossRef Link)

[14] D. J. Healy and O. R. Mitchell, "Digital Video Bandwidth Compression Using Block Truncation Coding," *IEEE Transactions on Communications*, vol. 29, pp.1809-1817, Dec 1981. Article (CrossRef Link)

[15] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, Volume 23, Issue 3, pp.337-343, May 1977. Article (CrossRef Link)

[16] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp.1158-1170, Jul 2000. Article (CrossRef Link)

[17] E. Delp and O. Mitchell, "Image Compression Using Block Truncation Coding," *IEEE Transactions on Communications*, vol. 27, no. 9, pp.1335-1342, 1979. Article (CrossRef Link)

[18] M. Lema and O. Mitchell, "Absolute Moment Block Truncation Coding and Its Application to Color Images," *IEEE Transactions on Communications*, vol. 32, no. 10, pp.1148-1157, 1984. Article (CrossRef Link)

[19] S. S. Maniccam and N. Bourbakis, "Lossless Compression and Information Hiding in Images," *Pattern Recognition*, vol. 37, no. 3, pp. 475-486, March 2004. Article (CrossRef Link)

[20] C. Qin, C. C. Chang and Y.P. Chiu, "A Novel Joint Data-Hiding and Compression Scheme Based on SMVQ and Image Inpainting," *IEEE Transactions on Image Processing*, vol. 23, no. 3, pp. 969-978, March 2014. Article (CrossRef Link)

[21] C. C. Lin, X. L. Liu, W. L. Tai and S. M. Yuan, "A Novel Reversible Data Hiding Scheme Based on AMBTC Compression Technique," *Multimedia Tools and Applications*, vol. 74, no. 11, pp. 3823-3842, June 2015. Article (CrossRef Link)

[22] D. Ou and W. Sun, "High Payload Image Steganography with Minimum Distortion Based on Absolute Moment Block Truncation Coding," *Multimedia Tools and Applications*, vol. 74, no. 21, pp. 9117-9139, Nov. 2015. Article (CrossRef Link)

[23] Y. H. Huang, C. C. Chang and Y. H. Chen, "Hybrid secret hiding schemes based on absolute moment block truncation coding," *Multimedia Tools and Applications*, vol. 76, no. 5, pp. 6159–6174, March 2017. Article (CrossRef Link)

[24] A. Malik, G. Sikka and H. K. Verma, "A High Payload Data Hiding Scheme Based on Modified AMBTC technique," *Multimedia Tools and Applications*, vol. 76, no. 12, pp. 14151-14167, June 2017. Article (CrossRef Link)

[25] R. Kumar, D. S. Kim, K. H. Jung, " Enhanced AMBTC based data hiding method using hamming distance and pixel value differencing," *Journal of Information Security and Applications*, vol. 47, pp. 94-103, August 2019. Article (CrossRef Link)

Chin-Feng Lee received her Ph.D. degree in Computer Science and Information Engineering in 1998 from National Chung Cheng University in Taiwan. She is currently a professor in the Department of Information Management at Chaoyang University of Technology, Taiwan. Her research interests include steganography, image processing, and data mining.

Chin-Chen Chang received the B.Sc. degree in applied mathematics and the M.Sc. degree in computer and decision sciences from National Tsing Hua University, and the Ph.D. degree in computer engineering from National Chiao Tung University. He was with the National Chung Cheng University from 1989 to 2005. He is currently Chair Professor with the Department of Information Engineering and Computer Science, Feng Chia University. Prior to joining Feng Chia University, he was an Associate Professor with Chiao Tung University, a Professor with National Chung Hsing University, and Chair Professor with National Chung Cheng University. He had also been a Visiting Researcher with Tokyo University, Japan, and a Visiting Scientist with Kyoto University, Japan. During his service in Chung Cheng, he served as the Chairman of the Institute of Computer Science and Information Engineering, Dean of College of Engineering, Provost, and then the Acting President of Chung Cheng University and the Director of Advisory Office in Ministry of Education, Taiwan. He has won many research awards and honorary positions by and in prestigious organizations both nationally and internationally. He is currently a Fellow of IEE, U.K. Since his early years of career development, he consecutively won the Outstanding Talent in Information Sciences of the R.O.C., Acer Dragon Award of the Ten Most Outstanding Talents, the Outstanding Scholar Award of the R.O.C., the Outstanding Engineering Professor Award of the R.O.C., the Distinguished Research Awards of National Science Council of the R.O.C., the Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, and so on. On numerous occasions, he was invited to serve as a Visiting Professor, Chair Professor, Honorary Professor, Honorary Director, Honorary Chairman, Distinguished Alumnus, Distinguished Researcher, and a Research Fellow by universities and research institutes. His current research interests include database design, computer cryptography, image compression, and data structures.

Guan-Long Li is currently pursuing the M.S. degree with Feng-Chia University. His research interests include data hiding and information security.