symmetry

MDPI

Article

Data Hiding Based on a Two-Layer Turtle Shell Matrix

Xiao-Zhu Xie [1,2], Chia-Chen Lin [3,*] and Chin-Chen Chang [2]

1 Engineering Research Center for Software Testing and Evaluation of Fujian Province, Xiamen University of Technology, Xiamen 361024, China; xz4xxz@gmail.com
2 Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan; alan3c@gmail.com
3 Department of Computer Science and Information Management, Providence University, Taichung 40724, Taiwan
* Correspondence: mhlin3@pu.edu.tw

Received: 30 December 2017; Accepted: 9 February 2018; Published: 13 February 2018

Abstract: Data hiding is a technology that embeds data into a cover carrier in an imperceptible way while still allowing the hidden data to be extracted accurately from the stego-carrier, which is one important branch of computer science and has drawn attention of scholars in the last decade. Turtle shell-based (TSB) schemes have become popular in recent years due to their higher embedding capacity (EC) and better visual quality of the stego-image than most of the none magic matrices based (MMB) schemes. This paper proposes a two-layer turtle shell matrix-based (TTSMB) scheme for data hiding, in which an extra attribute presented by a 4-ary digit is assigned to each element of the turtle shell matrix with symmetrical distribution. Therefore, compared with the original TSB scheme, two more bits are embedded into each pixel pair to obtain a higher EC up to 2.5 bits per pixel (bpp). The experimental results reveal that under the condition of the same visual quality, the EC of the proposed scheme outperforms state-of-the-art data hiding schemes.

Keywords: data hiding; turtle shell; two-layer; enhanced visual quality; improved hiding capacity

1. Introduction

Data hiding is a technology that can imperceptibly embed secret data into the carrier (e.g., text, sound, images, videos) to obtain the stego-carrier. After being transmitted in the communication channel, the receiver can extract the secret data accurately from the stego-carrier using the extraction algorithm. Due to the imperceptibility characteristic, stego-carriers can avoid attracting malicious attackers during transmission, which in some terms is superior to cryptographic encryption, another communications security technology. In practical applications, data hiding is widely used in the fields of medical images, copyright protection, military secrets, forensic evidence and anonymous communications [1,2]. Meanwhile, with the rapid development and the wide use of digital images, data hiding in images has attracted increasing attention from researchers in recent decades [3]. In addition, some researchers further studied this field and proposed several reversible data hiding methods in which the cover image can be recovered losslessly after the secret data was extracted [4–6]. The performance of data hiding technology is measured mainly by embedding capacity (EC) and the image quality after embedding.

The existing data hiding techniques are conducted mainly in three domains, i.e., the frequency domain, the compression domain, and the spatial domain. In the frequency domain, the cover image is first transformed to the discrete cosine transform (DCT) coefficients [4,7] or the discrete wavelet transformation (DWT) coefficients [8,9], in which the secret data is embedded. Since only a small quantity of the coefficients can be used for embedding, the embedding capacity cannot meet the requirements for most of the applications in practicality. In the compression domain, the cover image is first compressed to vacate space for embedding. Several researchers have proposed data hiding schemes based on vector quantization (VQ) compression [10–13], which can achieve larger embedding capacities; however, this capacity is at the expense of the distortion of the cover image.

In terms of the spatial domain, schemes can be roughly categorized into three types: the least significant bit (LSB) substitution [14–16], the exploiting modification direction (EMD) [17–19], and the magic matrix based (MMB) schemes [20–26]. Among them, LSB is the most commonly used scheme and was first proposed by Bender et al. in 1996 [3]. The original LSB scheme directly replaces the LSBs of each pixel with the secret bits. Later on, Wang et al. [14] proposed an optimal LSB substitution to improve the image quality and developed a genetic algorithm to solve the huge computation problem. The algorithm of LSB replacement is quite simple; however, the hiding data can be easily detected [16]. In order to avoid being attacked by malicious users, Mielikainen [15] improved the LSB matching method and pairwisely embedded data by modifying their parity. The embedding capacities of [14,15] are both 1 bit per pixel (bpp). Zhang and Wang [17] fully exploited the directions of modification so as to embed one secret $(2n+1)$-ary digit into a vector with n pixels by changing at most one LSB of one pixel. The method, called EMD, can achieve a large embedding capacity with less distortion. In 2010, Kim et al. [18] proposed two methods called 2-EMD and EMD-2, which can achieve a larger capacity than the original EMD method presented in [13] with similar distortion. Particularly in EMD-2, Kim et al., embedded one $(2w+1)$-ary digit into a vector of n pixels by flipping at most LSBs of 2 pixels, where $w = 4$ when $n = 2$, $w = 8 + 5(n-3)$ when $n > 2$. Moreover, the two methods can be generalized to n-EMD and EMD-n, where $n > 1$. Later on, Liu et al. [19] further improved the embedding capacity by combining the EMD and the Chinese remainder theorem.

Unlike the aforementioned methods, several novel schemes based on MMB have been proposed in the past few years. In 2008, Chang et al. [20] proposed a novel data hiding scheme using Sudoku, which considers a pixel pair as the coordinate of a Sudoku matrix to specify the value to embed one 9-ary digit into each pixel pair. The embedding capacity is 1.5 bpp. Hong et al. [21] improved the method in [20] by searching embedding positions based on the nearest Euclidean distance to achieve higher image quality. To enhance the security, a Sudoku-based wet paper hiding scheme was presented by Wang et al. [22]. In 2014, Chang et al. [23] put forward a novel TSB scheme, in which a secret 8-ary digit is embedded into each pixel pair with the guidance of the turtle shell. The scheme in [23] can maintain a large embedding capacity (1.5 bpp) with less distortion. In 2016, Liu et al. [24] produced an extra location table to achieve a larger embedding capacity of 2 bpp. Then, in 2017, for better image quality and the security of the hidden data, Jim et al. [25] proposed a new method using particle swarm optimization to minimize the distortion. In the same year, Liu et al. [26] extended the turtle shell matrix to different matrix models to meet diverse requirements of embedding capacity and image quality. The scheme in [26] can maintain a good image quality with an average peak signal-to-noise ratio (PSNR) of 41.87 when the embedding capacity is high, up to 2.5 bpp.

In this paper, we introduce an extra attribute of a 4-ary digit, namely type, to each element of the turtle shell matrix, and then proposed a two-layer turtle shell matrix based (TTSMB) scheme. The first layer is a type of 4-ary digit represented by two bits, and the second layer is the value of an 8-ary digit represented by three bits. Similar to TSB schemes, each pixel pair, considered the coordinate of the turtle shell matrix, can specify only one element in matrix. Therefore, for each pixel pair, five bits can be embedded by the two-layer turtle shell matrix, and the embedding capacity can reach 2.5 bpp. Meanwhile, the experimental results proved that the stego-image kept a high visual quality.

In Section 2, the related work of the original TSB scheme proposed by Chang et al. is introduced. Section 3 gives a detailed depiction of the proposed scheme, followed by an analysis of experimental results in Section 4. Finally, conclusions are made in Section 5.

2. Chang et al.'s Scheme

Chang et al. [23] first put forward the concept of the turtle shell matrix in the field of data hiding. The turtle shell matrix is a magic matrix on which you arbitrarily draw a turtle shell that contains eight digits from 0 to 7. Based on the characteristics, a pixel pair, which is considered the coordinate of the matrix, can specify an 8-ary digit. The following section gives a brief review of Chang et al.'s scheme.

2.1. Construction of the Turtle Shell Matrix

The turtle shell is defined as a hexagon containing eight different digits from 0 to 7, including two digits inside the hexagon, called back digits, and six digits on the edges of the hexagon, called the edge digits. As shown in Figure 1, a turtle shell matrix sized 256 $\times$ 256 is constructed for data embedding. The construction rules are as follows: first, the values of elements in the same row change according to the gradient ascent/descent with a magnitude of "1"; second, the values of elements in the same column change according to the gradient ascent/descent with an alternate magnitude of "2" and "3".

Once the data hider constructs a turtle shell matrix, he/she can embed the secret data into the cover image to obtain the stego-image according to the matrix. With the stego-image and the matrix, the receiver can accurately extract the secret data. Since the construction of the turtle shell matrix follows certain rules, there is no need to send the whole matrix to the receiver. Instead, only the value with the coordinate (0, 0) and the construction rules need to be shared with the receiver. For easy explanation, the shared information is called the construction information. With the construction information, the receiver can construct the exact matrix. The embedding and extraction procedures are depicted in detail in the following section.

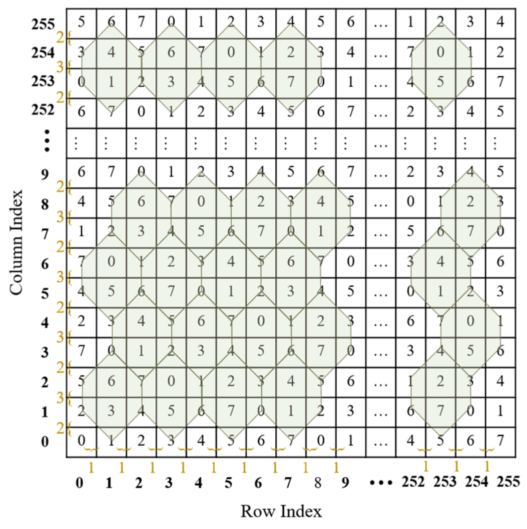

Figure 1. An example of a turtle shell matrix.

2.2. Data Embedding and Data Extraction

Assume that a secret binary stream S is embedded into a cover image I with size of $W \times H$ according to the turtle shell matrix $M = [m(i,j)]_{256\times256}$. The embedding procedure of Chang et al.'s method is described as follows:

Step 1: Convert the secret binary stream S into a sequence of octal digits $So = \{s_1, s_2, \cdots, s_n\}$, where n is the number of octal digits.

Step 2: Divide the cover image I into non-overlapping pixel pairs (p_k, p_{k+1}), where $k \in \{1, 3, \cdots, \mathrm{W} \times \mathrm{H} - 1\}$. Consider (p_k, p_{k+1}) as the coordinate of matrix M to specify the value $m(p_k, p_{k+1})$.

Step 3: Embed one octal digit $s_t (1 \leq t \leq n)$ into each pixel pair. The algorithm can be categorized into two cases.

Case 1: $m(p_k, p_{k+1}) = s_t$, which means that the current pixel pair exactly corresponds to the secret octal digit; therefore, do nothing and go to Step 4.

Case 2: $m(p_k, p_{k+1}) \neq s_t$, which means the current pixel pair cannot correspond to the secret octal digit; therefore, find the closest $m(p_r, p_c)$ that equals to s_t, and then replace (p_k, p_{k+1}) with (p_r, p_c) in the cover image to embed the octal digit s_t. Find the closest $m(p_r, p_c)$ according to the following rules:

Rule 1: If $m(p_k, p_{k+1})$ is a back digit, then find $m(p_r, p_c)$ in the current turtle shell.

Rule 2: If $m(p_k, p_{k+1})$ is an edge digit, then find the closest $m(p_r, p_c)$ in the involved turtle shells.

Rule 3: If $m(p_k, p_{k+1})$ is not included in any turtle shell, then find the closest $m(p_r, p_c)$ in the 3×3 sub-block where $m(p_k, p_{k+1})$ is located.

Step 4: Repeat Step 3 to embed the next octal digit into the follow-up pixel pair until all the secret data is embedded.

Finally, the stego-image I' is obtained.

At the procedure of the data extraction, with the construction information, the receiver constructs the turtle shell matrix $M = [m(i, j)]_{256 \times 256}$. According to stego-image I' and the turtle shell matrix, the secret data can be easily extracted by mapping the stego pixel pairs to the elements of the turtle shell matrix, the values of which are exactly the secret data.

2.3. Example of Data Embedding and Data Extraction

To better understand the process, an example of the embedding and extraction procedure of Chang et al.'s scheme is given in this section. Table 1 gives different cases of data embedding.

Case 1: $m(p_k, p_{k+1}) = s_t$

The to-be-embedded binary data 100 are first converted into an octal digit 4, the value of which is just equal to the value specified by the original pixel pair, viz., $m(2, 4)$ in Figure 2. Then, embed the secret data by doing nothing, since the stego pixel pair still is (2,4).

Case 2:. $m(p_k, p_{k+1}) \neq s_t$

Rule 1: The element that the original pixel pair (2,7) specifies, $m(2, 7)$, is a back digit. The to-be-embedded binary data 110 is converted to an octal digit 6. Find the satisfactory element in the turtle shell that the original pixel pair located, which is $m(2, 8)$. Then, replace the original pixel pair with the stego pixel pair (2,8) in the cover image.

Rule 2: The element that the original pixel pair (5,3) specifies, $m(5, 3)$, is an edge digit. Find the closest satisfactory pair in the involved turtle shells, which is $m(6, 4)$, to embed the converted octal digital 0, as shown in Figure 2. Thus, replace the original pixel pair with the stego pixel pair (6,4) in the cover image.

Rule 3: The element that the original pixel pair (2,0) specifies, $m(2, 0)$, is not involved in any turtle shell. Find the closest satisfactory pair in the 3×3 sub-block, which is $m(3, 1)$ to carry the converted octal digit 5. Thus, replace the original pixel pair with the stego pixel pair (3,1) in the cover image.

On the receiver side, the turtle shell matrix (as shown in Figure 2) is first constructed according to the construction rules. Then, the stego-image is divided into non-overlapping pixel pairs. Take the stego pixel pair (2,4) for example. The element value '4' is extracted by mapping the coordinate (2,4) in the turtle shell matrix. Finally, the value '4' is converted into binary data '100', which is the secret data.

Table 1. Cases of embedding using the turtle shell in Figure 2.

Cases		Original Pixel Pair	To-Be-Embedded Data	Stego Pixel Pair
Case 1: $m(p_k, p_{k+1}) = s_t$	Rule 0	(2,4)	$(100)_2 \rightarrow (4)_8$	(2,4)
Case 2: $m(p_k, p_{k+1}) \neq s_t$	Rule 1	(2,7)	$(110)_2 \rightarrow (6)_8$	(2,8)
	Rule 2	(5,3)	$(000)_2 \rightarrow (0)_8$	(6,4)
	Rule 3	(2,0)	$(101)_2 \rightarrow (5)_8$	(3,1)

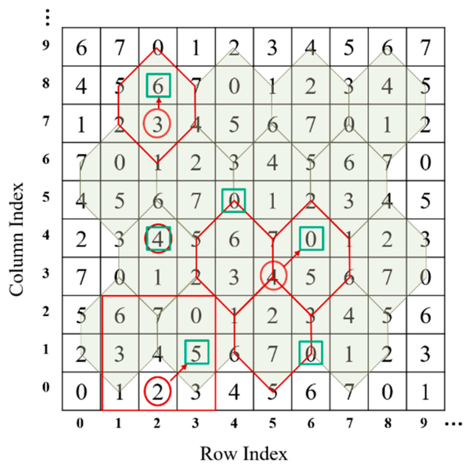

Figure 2. Example of embedding.

3. Proposed Scheme

3.1. Construction of the Two-Layer Matrix

Observing the turtle shell matrix $M = [m(i,j)]_{256\times256}$, we can say that, to some extent, it is composed with turtle shells sized 127 × 127. In the proposed scheme, we assign a 4-ary digit, referred to as type, to each turtle shell, as shown in Figure 3. For ease of description, the type matrix of the turtle shell is denoted as $T = [t(x,y)]_{127\times127}$. The type matrix is constructed following the rules: first, the values of type in the same row change according to the gradient ascent/descent with a magnitude of "1"; second, the values of the elements in the same column change according to the gradient ascent/descent with an alternate magnitude of "2" and "1". Figure 3 shows an example of a type matrix for turtle shell.

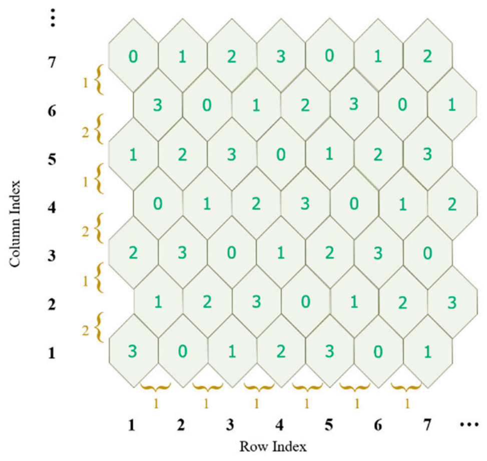

Figure 3. An example of type matrix for turtle-shell.

To go further, we assign an extra attribute, type, to each $m(i,j)$, which is denoted as $t_m(i,j)$. The algorithm for calculating $t_m(i,j)$ can be described as follows: conduct an exclusive operation on all involved turtle shell types. If there are no involved turtle shells, then set a value of '4'. The algorithm can be summarized into four cases, as shown in Figure 4.

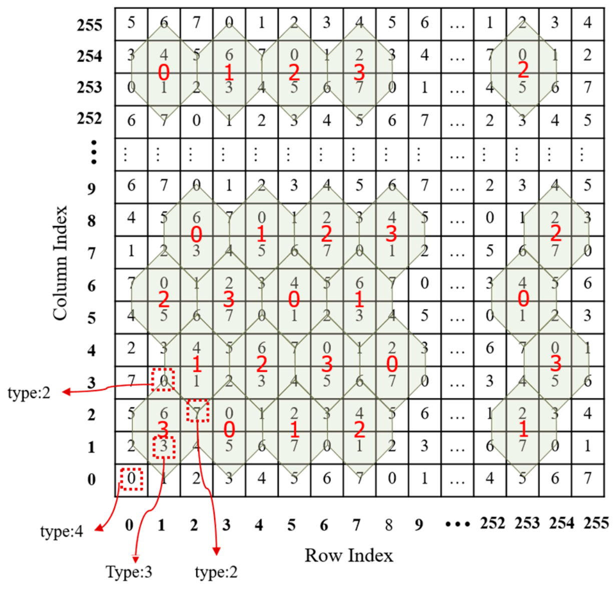

Figure 4. Four cases of calculating the type of $m(i, j)$.

Case 1: No involved turtle shells, e.g., $m(0, 0)$, and the type of such element is set as 4, e.g., $t_m(0, 0) = 4$.

Case 2: Only one turtle shell involved, e.g., $m(1, 1)$. The type of such element is set as the type of the turtle shell, e.g., $t_m(1, 1) = 3$.

Case 3: Two turtle shells involved, e.g., $m(1, 3)$. Then, the type of such element is the result of an operation exclusive of binary representations of the corresponding types of two turtle shells, e.g., $t_m(1, 3) = 1 \oplus 3 = 2$.

Case 4: Three turtle shells involved, e.g., $m(2, 2)$. The type of such element is the result of an operation exclusive of binary representations of the corresponding types of three turtle shells, e.g., $t_m(2, 2) = 1 \oplus 3 \oplus 0 = 2$.

According to the calculation algorithm, each $m(i, j) \in \{0, 1, \cdots, 7\}$ in the turtle shell matrix M is assigned to a corresponding type $t_m(i, j) \in \{0, 1, 2, 3\}$. In other words, a two-layer turtle shell matrix is generated. The first layer is the value matrix of elements, viz., $M = [m(i, j)]_{256\times256}$. The second layer is the type matrix of elements, viz., $T_m = [t_m(i, j)]_{256\times256}$. Figure 5 shows part of a two-layer turtle shell matrix. Digits in black are the values of elements, and digits in red are the values of a corresponding type. The statistical results confirm that all combinations of type and value, in other words, the $8 * 4$ mappings ($\{0, 1, \cdots, 7\} \to \{0, 1, 2, 3\}$), can be found in any parallelogram district consisting of nine turtle shells. Turtle shells in blue in Figure 5 show an example. In this case, we can exploit the type matrix $T_m = [t_m(i, j)]_{256\times256}$ as well as $M = [m(i, j)]_{256\times256}$ to hide data. Consequently, a TTSMB data hiding scheme is proposed in this paper.

On the data hider side, he/she constructs a two-layer turtle shell matrix, and then embeds every five bits into each pixel pair. As in the TSB scheme, the construction information is shared with the receiver. With the construction information, the receiver reconstructs the two-layer matrix and extracts the secret data from the stego-image using the matrix. The difference is that the construction information in TTSMB includes extra information of the second layer. Since the construction of the second layer matrix (i.e., the type matrix) can be calculated according to the above-mentioned algorithm, the construction information of the second layer includes the value with the coordinate (0,0) in $T = [t(x, y)]_{127\times127}$, the construction rules of T, and the algorithm for generating type matrix T_m. The procedures for data embedding and extraction are described in detail in the following sections.

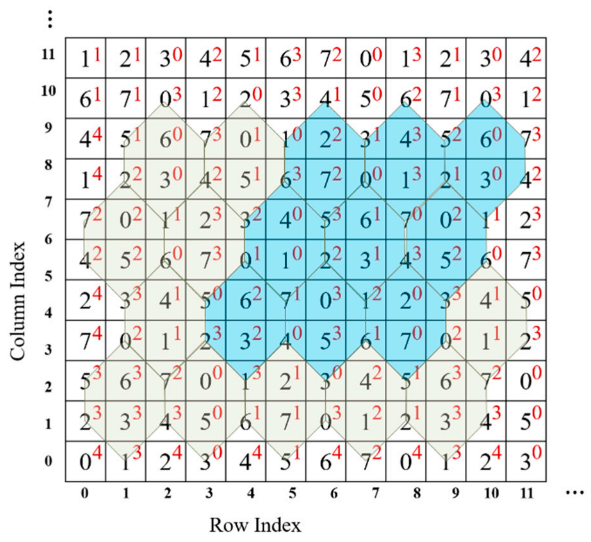
Figure 5. Part of a two-layer turtle shell matrix.

3.2. Data Embedding and Data Extraction Procedures

Here, two procedures are described, respectively, one is data embedding procedure and the other is data extraction procedure to present the core concept of our proposed scheme.

Data embedding procedure

Figure 6 shows the flowchart of data embedding. Firstly, the cover image is divided into non-overlapping pixel pairs and the binary secret stream is divided into sub-streams with five bits, which are further converted into 4-ary digits and octal digits. Then, embed a 4-ary digit and an octal digit into each pixel pair using the constructed two-layer matrix. Lastly, we get the stego-image after all the secret data are embedded. The detail is described as below.

Input: A cover image I sized $W \times H$, the binary secret stream S with length L.
Output: A stego-image I'.
Step 1: Generate a two-layer turtle shell matrix. First, construct a turtle shell matrix $M = [m(i,j)]_{256\times256}$ according to the rules described in Section 2. Second, a corresponding type matrix $T_m = [t_m(i,j)]_{256\times256}$ is calculated according to the algorithm depicted in the Section 3.1.
Step 2: Divide the cover image I into non-overlapping pixel pairs (p_k, p_{k+1}), where $k \in \{1, 3, \cdots, \mathrm{W}\times\mathrm{H}-1\}$. Consider (p_k, p_{k+1}) as the coordinate of matrix M to specify the value $m(p_k, p_{k+1})$ with the corresponding type $t_m(p_k, p_{k+1})$.
Step 3: Divide S into sub-streams s_r with five bits, where $r \in \left\{1, 2, \cdots, \left\lceil \frac{L}{5} \right\rceil\right\}$. For each sub-stream, covert the first two bits into a 4-ary digit d_{rf} and the last three bits into an octal digit d_{rl}.
Step 4: Embed each sub-stream s_r into one pixel pair (p_k, p_{k+1}) according to the following rule: Find the closest element $m(u,v)$, where $m(u,v) = d_{rl}$ and $t_m(u,v) = d_{rf}$, by spiral scanning from the $m(p_k, p_{k+1})$, as shown in Figure 7. Replace (p_k, p_{k+1}) with (u,v) in the cover image to embed the sub-stream s_r consisting of d_{rf} and d_{rl}.
Step 5: Repeat Step 4 until all the sub-streams are embedded.

Finally, the stego-image I' is obtained.

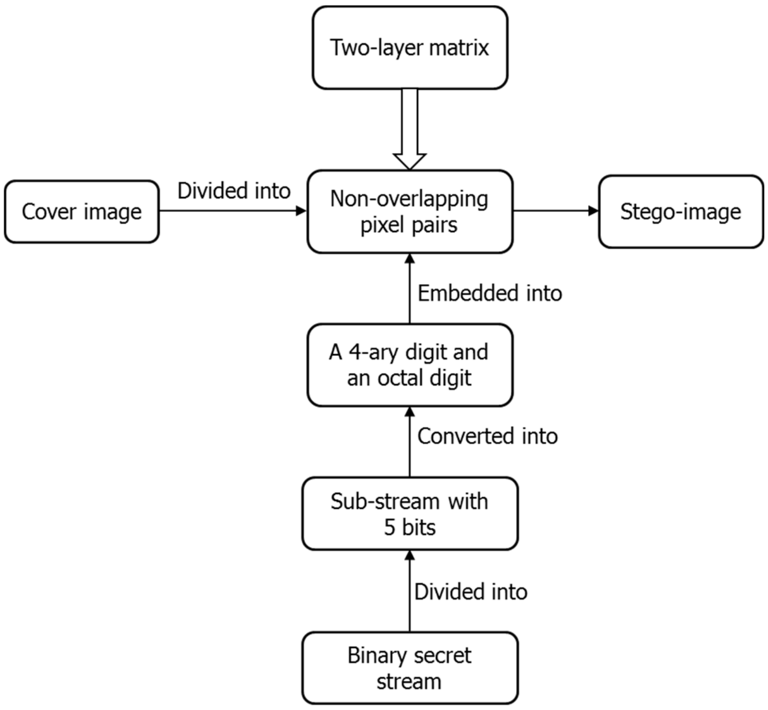

Figure 6. Flowchart of data embedding.

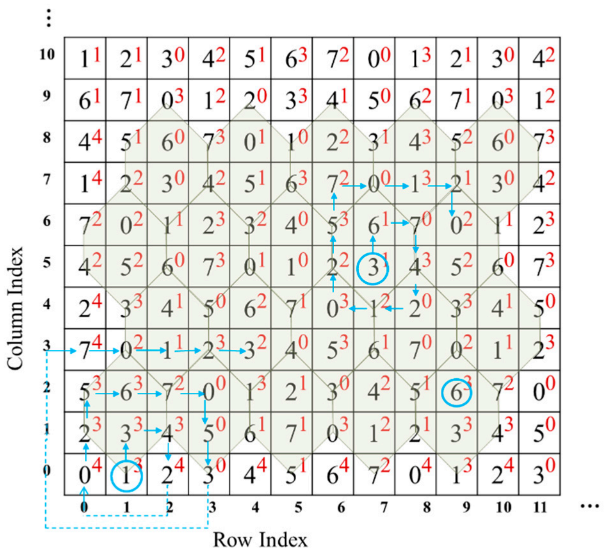

Figure 7. Spiral scanning for finding the closest element.

Data extraction procedure

Input: A stego-image I' sized $W \times H$, the construction information.
Output: The secret binary stream S.
Step 1: Reconstruct the two-layer turtle shell matrix ($M = [m(i,j)]_{256\times256}$ and $T_m = [t_m(i,j)]_{256\times256}$) using the shared construction information in the same way as described in the data embedding procedure.
Step 2: Divide the stego-image I' into non-overlapping pixel pairs (p'_k, p'_{k+1}), where $k \in \{1, 3, \cdots, \mathrm{W} \times \mathrm{H} - 1\}$.

Step 3: Extract the hidden secret data by mapping each pixel pair $\left(p'_k, p'_{k+1}\right)$ to the two-layer turtle shell matrix (M and T_m), that is $t_m\left(p'_k, p'_{k+1}\right)$ and $m\left(p'_k, p'_{k+1}\right)$, which are in 4-ary and octal format, respectively.
Step 4: Convert the values of these two digits (i.e., $t_m\left(p'_k, p'_{k+1}\right)$ and $m\left(p'_k, p'_{k+1}\right)$) into binary bits, and then combine them to form the sub-stream of secret data.
Step 5: Finally, combine all the sub-streams to form the secret binary stream S when all the sub-streams are extracted.

3.3. Example of Data Embedding and Data Extraction

For better understanding, an example of TTSMB data embedding and data extraction is given in this section. Table 2 gives an example of data embedding. The corresponding two-layer turtle shell matrix is shown in Figure 7. For the first original pixel pair (9,2), the to-be-embedded bit stream is 11,110. The bit stream 11,110 is firstly converted to two digits, 3 and 6. Since $m(9,2) = 6$ and $t_m(9,2) = 3$ (as shown in Figure 7), there is no need to change the original pixel pair, which itself just specifies the secret data. Thus, the stego pixel pair still is (9,2). For the second original pixel pair (7,5), the to-be-embedded bit stream is 10,000, which is converted to 2 and 0. Find the closest satisfactory element by spiral scanning the surrounding elements, i.e., (9,6) where $m(9,6) = 0$ and $t_m(9,6) = 2$. Then, change the original pixel pair to (9,6) in the cover image. Conduct the same operation on the third original pixel pair. The only special circumstance that needs to be mentioned is that, when scanning to the edge of the matrix, extend outward to form a spiral, as shown at the bottom-left corner in Figure 7.

On the receiver side, the two-layer turtle shell (as shown in Figure 7) is first constructed using the shared construction information. Then, the stego-image is divided into non-overlapping pixel pairs, from which every five bits can be extracted. Take the stego pixel pair (9,2), for example. The values of the mapping elements $t_m(9,2)$ and $m(9,2)$, viz., '3' and '6', are extracted. Then, '3' and '6' are converted into binary data '11' and '110', respectively. Finally, we get the combined secret data '11,110'.

Table 2. Example of data embedding using the two-layer turtle shell matrix in Figure 7.

Original Pixel Pair	Secret Bit Stream	Stego Pixel Pair
(9,2)	$(11)_2$ $(110)_2$ (is converted to) $\rightarrow$ $(3)_4$ $(6)_8$	(9,2)
(7,5)	$(10)_2$ $(000)_2$ (is converted to) $\rightarrow$ $(2)_4$ $(0)_8$	(9,6)
(1,0)	$(10)_2$ $(011)_2$ (is converted to) $\rightarrow$ $(2)_4$ $(3)_8$	(4,3)

4. Experimental Results

Experiments were conducted to verify the performance of the proposed scheme. Six test grayscale images, 'Lena', 'Airplane', 'Boat', 'Baboon', 'Peppers', and 'Sailboat', were used from the University of Southern California-Signal and Image Processing Institute (USC-SIPI) image database [27]. The sizes of the six test images were 512 × 512 shown in Figure 8. The binary secret stream S was randomly generated. In addition, Figure 9a–f are the corresponding stego-images with maximum embedding capacity (EC) of 2.5 bpp. The results show that, for all the test images, the stego-images remain high quality with a PSNR not lower than 47 dB. For a cover image sized $W \times H$, EC and PSNR can be calculated by Equations (1) and (2), respectively. The higher *PSNR* indicates the better image quality:

$$\mathrm{EC} = \frac{N_S}{W \times H}(\mathrm{bpp}), \tag{1}$$

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right), \tag{2}$$

where N_s is the number of embedding bits, and the mean square error (MSE) is defined as Equation (3):

$$\text{MSE} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \left(I(i,j) - I'(i,j) \right)^2. \tag{3}$$

Herein, $I(i,j)$ and $I'(i,j)$ are the original pixel values and the stego pixel values, respectively.

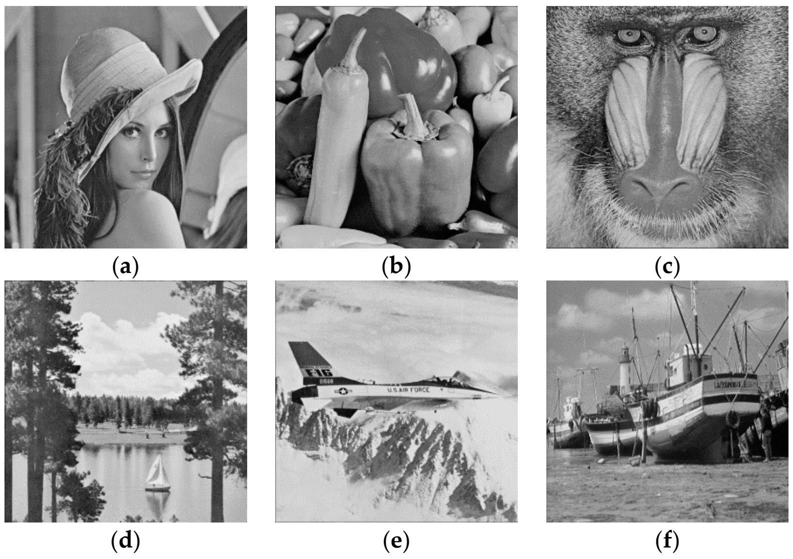
Figure 8. Six original cover images: (**a**) Lena; (**b**) Peppers; (**c**) Baboon; (**d**) Sailboat; (**e**) Airplane; (**f**) Boat.

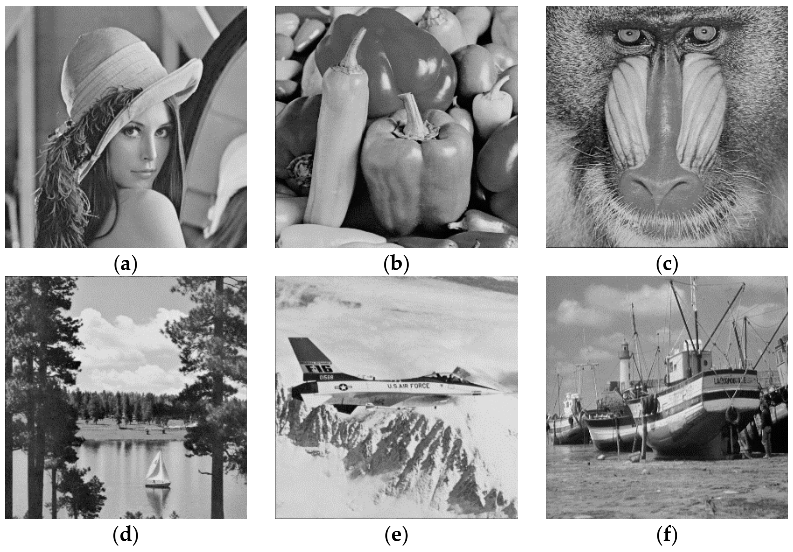
Figure 9. Six stego-images with maximum embedding capacity of 2.5 bpp. (**a**) Stego Lena with PSNR = 47.13; (**b**) Stego Peppers with PSNR = 47.13; (**c**) Stego Baboon with PSNR = 47.09; (**d**) Stego Sailboat with PSNR = 47.12; (**e**) Stego Airplane with PSNR = 47.11; (**f**) Stego Boat with PSNR = 47.15.

Comparisons with four previous TSB schemes are made in Table 3. It can be observed that the proposed scheme obviously outperforms the other four in terms of the embedding capacity and image quality. To be more detailed, the embedding capacity of the proposed scheme (i.e., 2.5 bpp) is much higher than that of [23] (i.e., 1.5 bpp), just with a slight decrease of PSNR. The image quality of the proposed scheme is better than that of [24,25], even with a higher embedding capacity. That is, with an embedding capacity of 2 bpp, the average PSNRs are 45.55 dB and 45.57 dB for [24,25], respectively. With an embedding capacity of 2.5 bpp, the average PSNR is 47.12 dB in the proposed scheme. Though [26] can achieve a high embedding capacity as can the proposed scheme, viz. 2.5 bpp, there is a remarkable improvement of the image quality in the proposed scheme, with an average PSNR of 47.12 dB versus 41.87 dB.

From Table 3, we can also find that, for all the five TSB based schemes including the proposed scheme, in the condition of a certain EC, the PSNRs of all the test images keep steady on the whole. The reason for the phenomenon is that the changing magnitude of pixels during the embedding procedure is determined by the constructed turtle shell matrix and the embedding algorithm, in other words, the image quality of stego-image is irrelevant to the cover image. Therefore, we can draw a remark: PSNR of the proposed scheme always keeps around 47.12 dB and higher than the other four TSB schemes, regardless of the content of test images.

Table 3. Comparison of PSNR (dB) and EC (bpp) with four other TSB schemes.

Test Images	**[23]**		**[24]**		**[25]**		**[26]**		**The Proposed**	
	EC	**PSNR**	**EC**	**PSNR**	**EC**	**PSNR**	**EC**	**PSNR**	**EC**	**PSNR**
Lena	1.5	49.42	2	45.55	2	45.57	2.5	41.87	2.5	47.13
Peppers	1.5	49.40	2	45.54	2	45.56	2.5	41.90	2.5	47.13
Baboon	1.5	49.39	2	45.55	2	45.57	2.5	41.85	2.5	47.09
Sailboat	1.5	49.40	2	45.55	2	45.58	2.5	41.89	2.5	47.12
Airplane	1.5	49.39	2	45.58	2	45.57	2.5	41.85	2.5	47.11
Boat	1.5	49.40	2	45.54	2	45.58	2.5	41.88	2.5	47.15
Average	1.5	49.40	2	45.55	2	45.57	2.5	41.87	2.5	47.12

Comparisons between other non-TSB schemes are made in this paper as well. As shown in Table 4, the embedding capacities of other non-TSB schemes are no more than 1.6 bpp (i.e., 1 bpp for EMD [17], 1.37 bpp for EMD-2 [18], 1.5 bpp for [20], 1.585 bpp for [21]), which is much lower than that of the proposed scheme (i.e., 2.5 bpp). Moreover, the image quality of the proposed scheme outperforms that of [20,21], even with the larger embedding capacity (i.e., 1.6 bpp for the proposed scheme vs. 1.5 bpp and 1.585 bpp for [20,21], respectively). From both Tables 3 and 4, we can also infer that TSB schemes perform better than EMB- and Sudoku-based schemes in terms of image quality and embedding capacity, which boils down to the property that eight digits (0–7) can be found within a turtle shell, result in the maximum changing magnitude is "2" during the embedding procedure.

Table 4. Comparison of PSNR (dB) and EC (bpp) between other non-TSB schemes.

Test Images	**EMD [17]**		**EMD-2 [18]**		**[20]**		**[21]**		**The Proposed**	
	EC	**PSNR**	**EC**	**PSNR**	**EC**	**PSNR**	**EC**	**PSNR**	**EC**	**PSNR**
Lena	1	52.12	1.37	50.86	1.5	44.97	1.585	48.68	1.6	49.06
Peppers	1	52.11	1.37	50.78	1.5	44.67	1.585	48.67	1.6	49.06
Baboon	1	52.11	1.37	50.69	1.5	44.68	1.585	48.66	1.6	49.04
Airplane	1	52.11	1.37	50.79	1.5	45.02	1.585	48.68	1.6	49.05
Average	1	52.11	1.37	50.78	1.5	44.84	1.585	48.67	1.6	49.05

In addition to PSNR, structural similarity Index (SSIM) is also used as image quality evaluation criteria. PSNR measures the difference between two images based on the estimated absolute errors, and SSIM considers image perceptual degradation in structural information. The SSIM index is calculated on the blocks of an image. Assuming there are two blocks, x and y, of common size, $N \times N$, the SSIM index can be calculated according to Equation (4), and $0 \leq SSIM \leq 1$. The larger the SSIM index is, the higher the similarity is. If $SSIM = 1$, then the two images are the same.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{4}$$

with:

μ_x, μ_y are the average of x and y, respectively;
σ_x^2, σ_y^2 are the variance of x and y, respectively;
σ_{xy} is the covariance of x and y;
$c_1 = (k_1L)^2, c_2 = (k_2L)^2$ are two variables to stabilize the division with weak denominator;
L is the dynamic range of the pixel-values; and
$k_1 = 0.01$ and $k_2 = 0.03$ by default.

Table 5 shows a comparison of SSIM between the proposed scheme and [26]. It is obvious to see the absolute advantage of the proposed scheme; that is, for all the test images, the SSIM of the proposed scheme is higher than that of [26] when embedding the maximum capacity of secret data (2.5 bpp). Additionally, an average SSIM of 0.03 improvement is obtained.

Table 5. Comparison of SSIM with the EC of 2.5 bpp.

Test Image	Lena	Baboon	Peppers	Airplane	Sailboat	Boat	Average
[26]	0.940	0.964	0.943	0.940	0.951	0.952	0.948
The proposed	0.973	0.991	0.974	0.970	0.980	0.981	0.978

5. Conclusions

Observing that turtle shell matrix $M = [m(i,j)]_{256\times256}$ is composed with turtle shells sized 127×127, a type of 4-ary digit is assigned to each turtle shell. To go further, an extra attribute, type, is assigned to each element of the turtle shell matrix by conducting an exclusive operation on all involved turtle shell types. Therefore, a two-layer turtle shell-based scheme is proposed in this paper. The first layer is a 4-ary digit type represented by 2 bits, the second layer is the value of the 8-ary digit represented by three bits. The statistical results show that 4×8 combination cases can be found in any parallelogram district consisting of nine turtle shells. As a result, for each pixel pair, five bits can be embedded. The experimental results verify that the proposed scheme outperforms the state-of-the-art data hiding schemes in terms of embedding capacity while maintaining a high image quality.

Acknowledgments: This work is supported by the Natural Science Foundation of China under Grant 61503316, the Natural Science Foundation of Fujian Province under Grant 2016J01326, and Open Fund of Engineering Research Center for Software Testing and Evaluation of Fujian Province.

Author Contributions: Chin-Chen Chang conceived and designed the experiments; Xiao-Zhu Xie performed the experiments and wrote the paper; and Chia-Chen Lin analyzed the data.

Conflicts of Interest: The authors declare no conflict of interest. Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Petitcolas, F.A.P.; Anderson, R.J.; Kuhn, M.G. Information hiding—A Survey. *Proc. IEEE* **1999**, *87*, 1062–1078. [CrossRef]
2. Sreejith, R.; Senthil, S. A novel tree based method for data hiding and integrity in medical images. In Proceedings of the International Conference on Electrical, Instrumentation and Communication Engineering, Karur, India, 21–28 April 2017. [CrossRef]

3. Bender, W.; Gruhl, D.; Morimoto, N.; Lu, A. Techniques for data hiding. *IBM Syst. J.* **1996**, *35*, 313–336. [CrossRef]
4. Chang, C.C.; Lin, C.C.; Tseng, C.S.; Tai, W.L. Reversible hiding in DCT-based compressed images. *Inf. Sci.* **2007**, *177*, 2768–2786. [CrossRef]
5. Lu, T.C.; Leng, H.S. Reversible dual-image-based hiding scheme using block folding technique. *Symmetry* **2017**, *9*, 223. [CrossRef]
6. Liu, W.L.; Leng, H.S.; Huang, C.K.; Chen, D.C. A block-based division reversible data hiding method in encrypted images. *Symmetry* **2017**, *9*, 308. [CrossRef]
7. Lin, C.C.; Shiu, P.F. High capacity data hiding scheme for DCT-based images. *J. Inf. Hiding Multimedia Signal Process.* **2010**, *1*, 220–240.
8. Abdelwahab, A.A.; Hassan, L.A. A discrete wavelet transform based technique for image data hiding. In Proceedings of the 25th National Radio Science Conference, Tanta, Egypt, 18–20 March 2008; pp. 1–9. [CrossRef]
9. Liu, H.; Liu, J.; Huang, J.; Huang, D.; Shi, Y.Q. A robust DWT-based blind data hiding algorithm. In Proceedings of the IEEE International Symposium on Circuits and Systems, Phoenix-Scottsdale, AZ, USA, 26–29 May 2002; Volume 2, pp. 672–675. [CrossRef]
10. Chang, C.C.; Wu, W.C. Hiding secret data adaptively in vector quantization index tables. *IEE Proc. Vis. Image Signal Process.* **2006**, *153*, 589–597. [CrossRef]
11. Hu, Y.C. High capacity image hiding scheme based on vector quantization. *Pattern Recognit.* **2006**, *39*, 1715–1724. [CrossRef]
12. Chang, C.C.; Kieu, T.D.; Wu, W.C. A loss-less data embedding technique by joint neighboring coding. *Pattern Recognit.* **2009**, *42*, 1597–1603. [CrossRef]
13. Pizzolante, R.; Carpentieri, B.; Castiglione, A.; De Maio, G. The AVQ algorithm: Watermarking and compression performances. In Proceedings of the International Conference on Intelligent Networking and Collaborative Systems, Fukuoka, Japan, 30 November–2 December 2011; pp. 698–702. [CrossRef]
14. Wang, R.Z.; Lin, C.F.; Lin, J.C. Image hiding by optimal LSB substitution and genetic algorithm. *Pattern Recognit.* **2001**, *34*, 671–683. [CrossRef]
15. Mielikainen, J. LSB matching revisited. *IEEE Signal Process. Lett.* **2006**, *13*, 285–287. [CrossRef]
16. Ker, A.D. Improved detection of LSB steganography in grayscale images. *Proc. Inf. Hiding Workshop* **2004**, *3200*, 97–115. [CrossRef]
17. Zhang, X.; Wang, S. Efficient steganographic embedding by exploiting modification direction. *IEEE Commun. Lett.* **2006**, *10*, 781–783. [CrossRef]
18. Kim, H.J.; Kim, C.; Choi, Y. Improved modification direction methods. *Comput. Math. Appl.* **2010**, *60*, 319–325. [CrossRef]
19. Liu, Y.; Chang, C.C.; Huang, P.C.; Hsu, C.Y. Efficient information hiding based on theory of numbers. *Symmetry* **2018**, *10*, 19. [CrossRef]
20. Chang, C.C.; Chou, Y.C.; Kieu, T.D. An information hiding scheme using Sudoku. In Proceedings of the Third International Conference on Innovative Computing Information and Control, Dalian, China, 18–20 June 2008; pp. 17–22. [CrossRef]
21. Hong, W.; Chen, T.S.; Shiu, C.W. A minimal Euclidean distance searching technique for Sudoku steganography. In Proceedings of the International Symposium on Information Science and Engineering, Shanghai, China, 20–22 December 2008; pp. 515–518. [CrossRef]
22. Wang, Z.H.; Chang, C.C.; Li, M.C. A Sudoku based wet paper hiding scheme. *Int. J. Smart Home* **2009**, *3*, 1–11.
23. Chang, C.C.; Liu, Y.; Nguyen, T.S. A novel turtle shell based scheme for data hiding. In Proceedings of the 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Kitakyushu, Japan, 27–29 August 2014; pp. 89–93. [CrossRef]
24. Liu, Y.; Chang, C.C.; Nguyen, T.S. High capacity turtle shell-based data hiding. *IET Image Process.* **2016**, *10*, 130–137. [CrossRef]
25. Jin, Q.; Li, Z.; Chang, C.C. Minimizing turtle-shell matrix based stego image distortion using particle swarm optimization. *Int. J. Netw. Secur.* **2017**, *19*, 154–162. [CrossRef]
26. Liu, L.; Chang, C.C.; Wang, A. Data hiding based on extended turtle shell matrix construction method. *Multimedia Tools Appl.* **2017**, *76*, 12233–12250. [CrossRef]

27. USC-SIPI Image Database. Available online: http://sipi.usc.edu/database/ (accessed on 3 May 2017).