Journal of Information Security and Applications 80 (2024) 103679

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Verifiable (*t, n*) Secret Image Sharing Scheme Based on Slim Turtle Shell Matrix

Yijie Lin [a], Chia-Chen Lin [b], Jui-Chuan Liu [a], Chin-Chen Chang [a,*]

[a] Department of Information Engineering and Computer Science, Feng Chia University
[b] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology

ARTICLE INFO

Keywords:
Secret image sharing
(t, n)-SIS
turtle shell
adaptive data hiding capacity
verifiability
message digest

ABSTRACT

As Internet technology advances rapidly, safeguarding confidential messages has become an increasingly pressing concern. Among the various schemes available, secret image sharing (SIS) has gained considerable attention. Particularly, the (*t, n*) secret sharing scheme offers robust protection for hidden secret messages. However, it remains challenging to expand the quality of *t* and *n* while maintaining acceptable image quality for the share images and enhancing security features. To address these challenges, we have developed a variant turtle shell known as the slim turtle shell, which we incorporate into our innovative (*t, n*)-SIS scheme. The slim turtle shell technique not only facilitates the creation of meaningful shares but also ensures that the resulting meaningful shares maintain a high level of visual quality, thanks to its impressive embedding capabilities. Additionally, we have devised two authentication mechanisms to validate the integrity of both shares and secret images. These mechanisms facilitate the detection of tampering during the secret message extraction and verification phase, should the integrity be compromised. Empirical results from our experiments demonstrate an impressive embedding capacity of 1,572,864 bits in our proposed scheme, coupled with an average PSNR (Peak Signal-to-Noise Ratio) of 45.80 dB. Furthermore, our scheme demonstrates superior authentication capabilities when compared to other existing schemes.

1. Introduction

In recent years, the rapid development of the Internet has provided people with convenient means of transmitting and storing information in various formats, such as text, images, and videos. However, as data transfer increases, so does the frequency of malicious attacks by lawbreakers. These attacks can take various forms, including phishing scams, viruses, and hacking attempts. Consequently, it has become increasingly crucial to take steps to safeguard personal and sensitive information. Information security has gained paramount importance in this environment, leading to continuous research and improvement of various data hiding techniques. Several data hiding schemes have been developed, such as least significant bit substitution [1–4], histogram shifting [5,6], exploiting modification direction [7–9], difference expansion [10,11], and magic matrix-based schemes [12–19]. These diverse techniques effectively protect data in different application scenarios and demonstrate excellent performance in multiple aspects.

In 1979, Shamir [20] introduced a secret sharing scheme, which also functions as a cryptographic algorithm. This scheme allows for the partitioning of a secret message into *n* shares that can be distributed among a group of participants. To reconstruct the original secret message, a minimum of *t* shares is needed. Since Shamir's pioneering work, numerous data hiding techniques have emerged, with the embedding of shared content into images becoming a widely employed method for secret image sharing (SIS) [21–24].

SIS schemes have undergone significant evolution over time, driven by diverse performance requirements. Initially, many schemes were based on dual images [25–28]. In 2010, Chang et al. [29] introduced a method leveraging Sudoku matrices and Lagrange polynomials to enhance visual quality. In Chang et al.'s scheme, the average PSNR reaches 44.10 dB and the hiding capacity is 524,288 bits when *t* is set to 2; however, increasing *t* to 3 results in an average PSNR of 44.13 dB and a hiding capacity of 1,048,576 bits. As the need for verifiability grew, authentication mechanisms became crucial, leading to the proposal of various schemes [30–34] to ascertain whether a restored cover image had been tampered with. The aforementioned methods utilized the same cover image. Following that, several schemes [35,36,37] emerged, proposing the utilization of distinct cover images to enhance security.

* Corresponding author
E-mail address: alan3c@gmail.com (C. Chang).

https://doi.org/10.1016/j.jisa.2023.103679

Available online 24 December 2023

For instance, in the scheme developed by Chang et al. [36], they devised a two-layered cheat detection mechanism based on the unique characteristics of the maze matrix they introduced. In addition to the conventional joint cheating detection, their scheme could identify tampered shadows created by a cheater without relying on information from other shadows. However, it should be noted that their scheme falls under the (2,2)-SIS category, and although it maintains a hiding capacity of 1,048, 576 bits which is the same as Chang et al. [29], the average PSNR is limited to 39.88 dB. In 2022, Chen et al. [37] successfully extended their scheme to (2, n)-SIS with a verification feature, thereby upgrading the average PSNR of share images to 49.88 dB. Nonetheless, their scheme exhibits a limited hiding capacity of 786,432 bits.

Based on the aforementioned descriptions, it is evident that maintaining high image quality, extending the (t, n), and providing a verification feature pose significant challenges. To tackle these issues, this paper introduces the utilization of a slim turtle shell technique and a polynomial function with $t-1$ degrees. The integration of our designed slim turtle shell scheme successfully guarantees the selection of the closest coordinates within the qualified area for the given pixel pairs, leading to minimal modifications and distortions. This outcome makes it exceedingly challenging for malicious attackers to distinguish between the cover image and the share image, thereby effectively concealing any hidden information. Additionally, we have devised a verification mechanism that incorporates a hash function and authentication code. This mechanism serves to detect any tampering or removal of pixels within the share images. By leveraging this authentication mechanism, we can identify the specific images and corresponding regions that have been altered. Experimental results firmly validate the exceptional performance of our proposed verifiable(t, n)-SIS scheme. It demonstrates outstanding capabilities in terms of embedding capacity, authentication ability, and preservation of the visual quality of the shares.

The remainder of this paper is structured as follows: Section II discusses related work, including the (t, n) secret sharing scheme and the turtle shell scheme. Section III presents our verifiable (t, n)-SIS scheme. Section IV provides the experimental results. Finally, Section V concludes the paper.

2. Related Work

In this section, we present two essential concepts to establish the necessary background for our article. Firstly, we introduce the (t, n) secret sharing approach, which forms a fundamental basis for our research. Secondly, we delve into the core concept of turtle shell data hiding. These concepts lay the foundation for the subsequent discussions and analysis in our article.

2.1. Secret sharing scheme

In 1979, Shamir [20] introduced the (t, n) secret sharing approach based on Lagrangian interpolation. This approach revolutionized the field by providing a comprehensive and efficient solution for secret sharing. Here, the parameter n represents the total number of participants, and the secret message is divided into n pieces, each piece is distributed to a participant. The parameter t, known as the threshold, signifies the minimum number of participants required to reconstruct the secret message. Collaboration from at least t participants is necessary to recover the original message.

This method not only boasts simplicity in implementation but also offers robust security for safeguarding confidential information. Secret sharing strategies, particularly the threshold approach, have gained significant attention and widespread application. It provides an exceptional level of security assurance, guaranteeing the protection of confidential data even in the event of participant attacks or information leakage. As long as the amount of compromised or leaked information remains below the threshold value, denoted as t, the confidentiality of the data remains intact. Moreover, the value of the threshold t can be adjusted flexibly to strike a balance between security and usability.

As shown in Eq. (1), the secret message can be either set as a_0 or is split into coefficients $a_0, a_1, \ldots, a_{t-1}$ of a polynomial function $f(x)$ with $t-1$ degrees. These coefficients are distributed to t participants and each participant receives a specific point $(x, f(x))$. The prime p is set as 251 when Shamir's scheme is applied to a secret image scenario since the maximum pixel value in a grayscale image is 255, thus, all coefficients are always less than 251. Therefore, the original image needs to be preprocessed to handle pixel values that are over 250. To avoid preprocessing the original image and generating auxiliary information, we apply the secret sharing method over a Galois field GF (2^8) to share different grayscale images and relevant information in our work. To reconstruct the secret message, a collaborative effort of at least $t$participants is required, utilizing the Lagrangian interpolation algorithm to recover the polynomial $f(x)$ and retrieve the original secret message.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} \ (mod\ p). \tag{1}$$

The critical aspect is that a minimum of t participants must collaborate for the Lagrangian interpolation algorithm, Eqs. (2) and (3) can be used to recover the polynomial $f(x)$ and $f(0) = s$ can be calculated, and thus the original secret information s can be retrieved.

$$f(x) = \left(a_0 + \sum_{k=1}^{r-1} a_k x^k \right) \ (mod\ p), \tag{2}$$

$$L(x) = \sum_{j=1}^{r} \left(f(x_j) \times \prod_{\substack{0<k\le r \\ k\neq j}} \frac{x - x_k}{x_j - x_k} \right) \ (mod\ p). \tag{3}$$

2.2. Turtle shell scheme

In 2014, Chang et al. [15] introduced the concept of a turtle shell matrix in the field of data hiding. They constructed a 256 × 256 matrix in advance and adorned it with hexagonal turtle shells. In their defined reference matrix, the adjacent elements within the same row were set to have a difference of 1, while the differences between adjacent elements in a column alternated between 2 and 3. In the reference matrix, there are many non-overlapping turtle shells. Within each turtle shell, there were a total of 8 numbers ranging from 0 to 7. The eight numbers consist of 6 edge numbers and 2 back numbers. By utilizing the pixel pair as the matrix coordinate, an octal number can be found in a turtle shell.

During the embedding process, Chang et al. defined four rules that must be followed in a specific order. Once a rule was satisfied, it indicated that embedding could take place, and subsequent rules no longer needed to be considered. The first rule stated that no changes were required when the value of the coordinate represented by the pixel pair matched the embedded value. The second rule involved locating the coordinate within the turtle shell that matched the embedded value and replacing the pixel pair if the coordinate value corresponded to a back number. The third rule allowed for checking multiple turtle shells to find the nearest coordinates for replacement when the coordinate value corresponded to an edge number. The fourth rule entailed finding the corresponding coordinate within the nearest 3 × 3 block if the coordinate value did not fall within a turtle shell but rather resided on the matrix's edge. To facilitate comprehension, the following examples illustrate the embedding and extraction processes within Chang et al.'s scheme.

Let's consider the original binary secret message as $(100\ 000\ 111)_2$, indicating the associated secret numbers as 4, 0, and 7. Assuming that (3,4), (4,4), and (0,2) represent the three pairs of covered pixels, we will now illustrate the process of embedding each secret number into its respective cover pixel pair in Fig. 1. In the figure, the covered pixels are circled in red, and the matched coordinates are circled in yellow. The solid yellow line represents the final replacement coordinates, while the

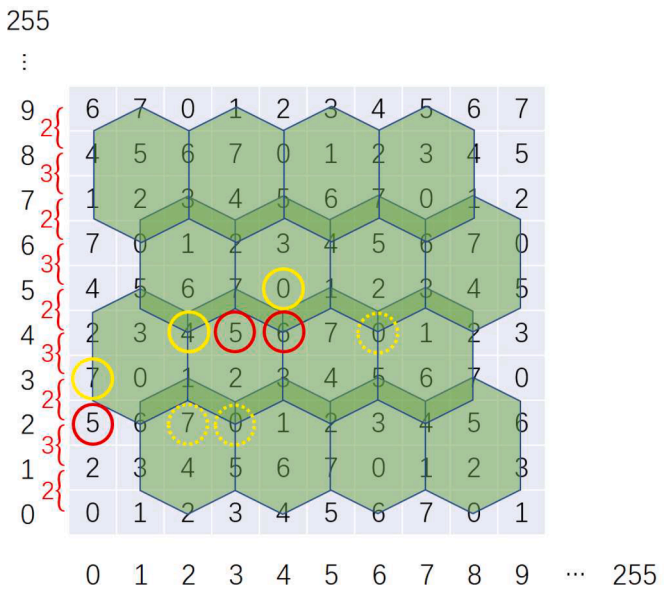


Figure 1. Examples of turtle shell matrix.

dashed line represents discarded matching options.

(1) Embed the secret number 4 into the pixel pair (3, 4)

As the coordinate (3, 4) marked in the red circle falls within the back location of the turtle shell, we locate the coordinate (2, 4) within the turtle shell because its value is 4, which matches the secret number.

(2) Embed the secret number 0 into the pixel pair (4, 4)

As the coordinate (4, 4) is located at the edge of the turtle shell, we look for the neighboring turtle shell and find the coordinate (4, 5), which corresponds to the secret number 0 and has the smallest distance to the original coordinate (4, 4).

(3) Embed the secret number 7 into the pixel pair (0, 2)

The coordinate (0, 2) does not fall within any turtle shell, we locate the nearest coordinate (0, 3) which corresponds to the secret number 7 in the 3 × 3 matrix.

During the extraction process, the receiver can simply map the three coordinates (2, 4), (4, 5), and (0, 3) onto the reference matrix to obtain the decimal values 4, 0, and 7, respectively. By performing a binary conversion, the corresponding secret message $(100\ 000\ 111)_2$ can ultimately be obtained.

3. Proposed Verifiable (*t, n*)-SIS Scheme

In this paper, we present a novel verifiable (*t, n*)-SIS scheme, where *t* is always less than or equal to *n*. Here, (*t*, *n*) indicates that the secret images are divided into *n* pieces and embedded into *n* different cover images, and at least *t* marked images, also called shares, are required to retrieve the hidden secret message(s). Our primary goal is to enhance the capacity for embedding secret images while preserving the visual quality of shares across different cover images. Additionally, we aim to authenticate the integrity of these shares and the hidden secret images. To achieve both verifiability of shares and hidden secret images, we employ two authentication mechanisms. Firstly, we select the two most significant bits (2MSBs) of cover images and utilize the SHA-512 hash function [38] to generate a hash value, which then contributes to the share construction phase. Secondly, we involve the three most significant bits (3MSBs) of secret images in the share construction phase as well. Through experiments, we will demonstrate the successful accomplishment of our objectives by these two authentication mechanisms.

Our proposed verifiable (*t, n*)-SIS scheme comprises three primary phases: (a) slim turtle shell construction phase, (b) share construction phase, and (c) secret message extraction and verification phase, as illustrated in Fig. 2. In this section, we will provide a comprehensive description of each phase.

3.1. Slim turtle shell construction phase

Unlike the traditional turtle shell scheme proposed by Chang et al. in their study [15], we present a novel approach using a slim turtle shell matrix. The purpose of this matrix is to minimize distortion and accommodate a larger number of hidden secret bits during the share construction phase. Illustrated in Fig. 3, our slim turtle shell has a slender and elongated shape, with each row featuring consistent spacing differences of 2, 3, and 3, respectively. This visual representation demonstrates that each slim turtle shell consists of 11 unique numbers ranging from 0 to 10, with the symbol "A" denoting the number 10.

To process each cover image, we first pair adjacent pixels, ensuring that no overlaps occur between pairs. This results in 131,072-pixel pairs for each 512 × 512 pixels cover image. These pixel pairs can then be utilized to represent the coordinates of the slim turtle shell matrix, with the embedded element values corresponding to the respective coordinate values. Algorithm 1 details the steps of the process of slim turtle

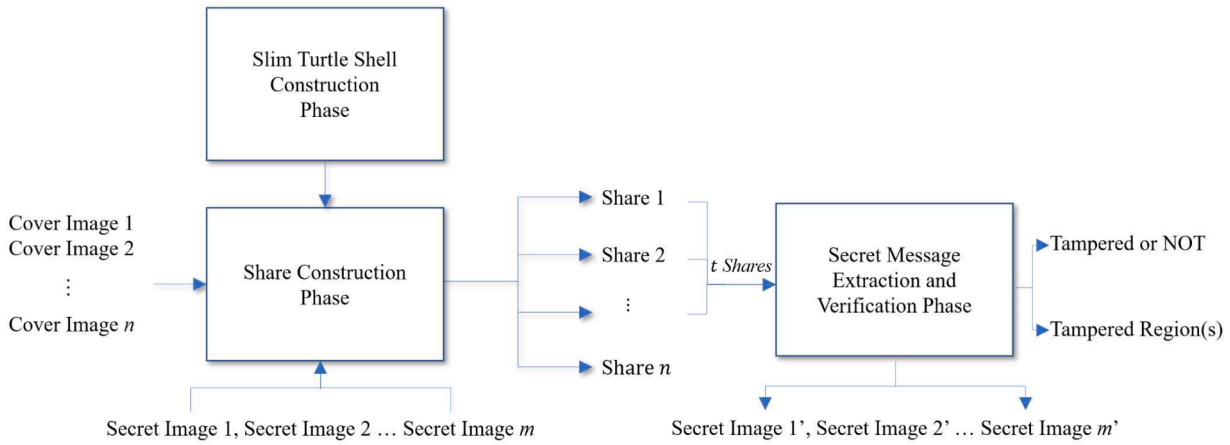


Figure 2. Flowchart of our verifiable (*t, n*)-SIS scheme.

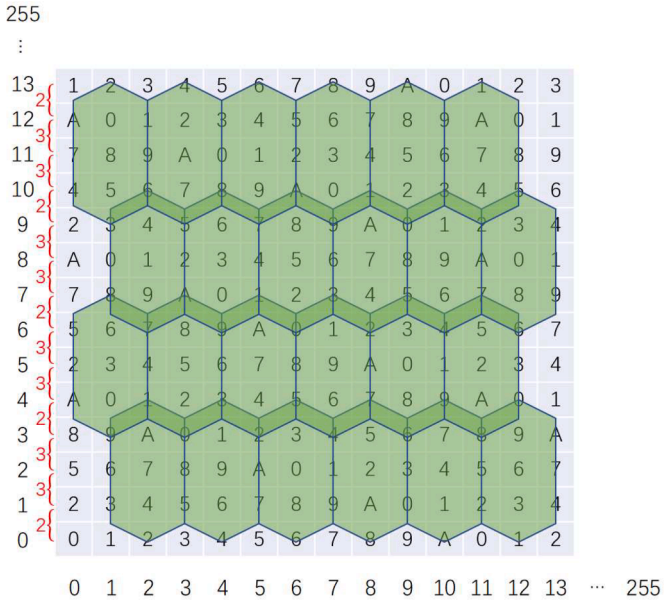


Figure 3. Our proposed slim turtle shell matrix.

Algorithm 1
Slim Turtle Shell Construction Algorithm.

Output: Slim turtle shell matrix

Step 1: Generate a 256 × 256 empty matrix called the slim turtle shell matrix M.
Step 2: Let the point in the lower left corner be the starting point $M(256, 1) = 0$.
Step 3: Generate the coordinate values for the bottom row based on the starting point.
foreach $j \in \{2, 3, \ldots, 256\}$ **do**
$M(256, j) = (M(256, j-1) + 1) mod 11$
Step 4: Generate the coordinate values of the entire matrix according to the bottom row.
foreach $i \in \{255, 254, \ldots, 1\}$ **do**
  foreach $j \in \{1, 2, \ldots, 256\}$ **do**
  if $(i+1) mod 3 = = 1$ **do**
$M(i, j) = (M(i+1, j) + 2) mod 11$
  else
$M(i, j) = (M(i+1, j) + 3) mod 11$
Step 5: Output the slim turtle shell matrix M.

shell construction.

3.2. Share construction phase

Our proposed verifiable (t, n)-SIS scheme utilizes the Secure Hash Algorithm (SHA-512) hashing function to create a 512-bit message digest. By utilizing the SHA-512 hash function, we divide a variable-sized message into non-overlapping blocks of 1024 bits, ultimately resulting in a 512-bit message digest. The purpose of the message digest is to guarantee the integrity of the received message during subsequent verification stages.

As shown in Eqs. (4)-(7), calculate the total length of the binary sequence of MSBs which are sized as $Wc \times Hc \times 2 \times n$ at first, where Wc and Hc are the width and height of the cover image C, respectively, and n is the number of cover images. Second, calculate the total number of pixels in a secret image S. When $t < 4$, only one $Ws \times Hs$-sized secret image S can be carried with our (t, n)-SIS scheme, where Ws and Hs are the width and height of the secret image, respectively. By contrast, two $Ws \times Hs$-sized secret images can be embedded $t \geq 4$ with our (t, n)-SIS scheme. As we can see, as long as the size of a secret image is half of a cover image, Eqs. (4)-(7) will always hold. Take a cover image of size 512 × 512 pixels and a secret image sized 256 × 256 pixels for example. For two secret images, the total number of pixels is equal to 256 × 256 × 2. Third, divide the secret image into non-overlapping 512-pixels divisions and finally get num_{hash} by Eq. (6). It means l_{MSB} needs to be divided into num_{hash} number of non-overlapping binary sequences sized as l_{input}-bits by Eq. (7). Fourth, input each l_{input}-bits binary sequence to the SHA512 hash function and get the $output_j$ where j is ranges from 1 to num_{hash}. Finally, all $output_j$s are formed as $sequence_{hash}$ by Eq. (8) and each bit in the $output_{num_{hash}}$ is denoted as h_i.

$$l_{MSB} = 512 \times 512 \times 2 \times n, \tag{4}$$

$$num_{pixel} = \begin{cases} 256 \times 256, & if\ t < 4 \\ 256 \times 256 \times 2, & otherwise. \end{cases} \tag{5}$$

$$num_{hash} = num_{pixel} \div 512, \tag{6}$$

$$l_{input} = l_{MSB} \div num_{hash}, \tag{7}$$

$$sequence_{hash} = output_1 \parallel output_2 \parallel \ldots \parallel output_{num_{hash}}, \tag{8}$$

$$h_i = output_{num_{hash}}[i]. \tag{9}$$

There are one or two secret images S of size 256 × 256 pixels to be hidden with our scheme depending on if t is less than 4 or vice versa, as defined in Eq. (5). To achieve both objectives of secret image embedding and share construction at the same time, for the given secret image(s) and the $sequence_{hash}$ derived by Eq. (8), we need to transform them into num_{pixel} sets of three references (b_1,b_2,b_3) and one authentication code (AC) that will later serve as $t = 4$ coefficients in Shamir's Lagrange interpolation polynomial during the secret image embedding operation. The detailed transformation operations are described as follows: first, we convert each pixel of the secret image(s) S into a binary stream, and divide each 8-bit secret stream $\{s_1,s_2,\ldots, s_8\}$ into a 3-element groups with each group sized as 2 bits, 3 bits, and 3 bits, as $\{s_1,s_2\}$, $\{s_3,s_4,s_5\}$, $\{s_6,s_7, s_8\}$,respectively. Second, the h value derived from Eq. (9) is contacted with the first 2-bits $\{s_1,s_2\}$ to form $b_{1,num_{pixel}} = \{h,s_1,s_2\}$. For the rest, two 3-bit groups are formed as $b_{2,num_{pixel}} = \{s_3, s_4, s_5\}$, $b_{3,num_{pixel}} = \{s_6, s_7, s_8\}$, respectively. Third, AC designed to verify the integrity of hidden secret image(s) is formed as $AC_{num_{pixel}} = \{s_1, s_2, s_3\}$. The concept of the set of three references (b_1,b_2,b_3) and one AC is demonstrated in Fig. 4.

Once num_{hash} sets of three references (b_1,b_2,b_3) and the ACs are ready, $t = 4$ elements- Lagrange interpolation polynomial proposed by Shamir [20]. In our scheme, t is not limited to 4. The Lagrange interpolation polynomial can be formed for different situations. When $t < 4$, two formulas are needed to handle b_1, b_2, b_3, AC. Yet when $t \geq 4$, only one formula is needed. For the formulas below, x represents the xth cover image ($x \in [1, n]$). When $t < 4$, we need to get the values of $f_1(x)$ and $f_2(x)$ to be embedded into the pixel pairs of the cover images. By contrast, we can use $f_3(x)$ to find values to embed when $t \geq 4$.

When $t = 2$, Eq. (10) and Eq. (11) are employed and three references (b_1,b_2,b_3) and one AC serve as coefficients in $f_1(x)$ and $f_2(x)$ as follows:

$$f_1(x) = b_1 x + b_2 (mod\ 11), \tag{10}$$

$$f_2(x) = b_3 x + AC (mod\ 11 \tag{11}$$

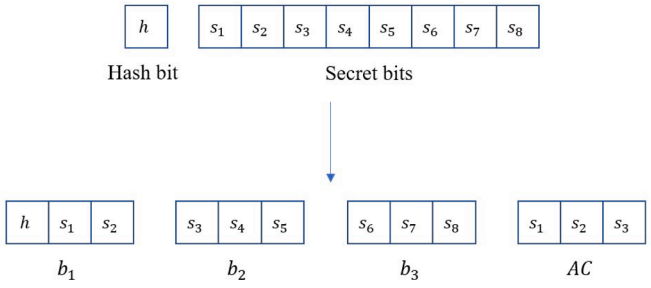


Figure 4. Concept of generating a set of three references (b_1,b_2,b_3) and one AC for later secret image embedding.

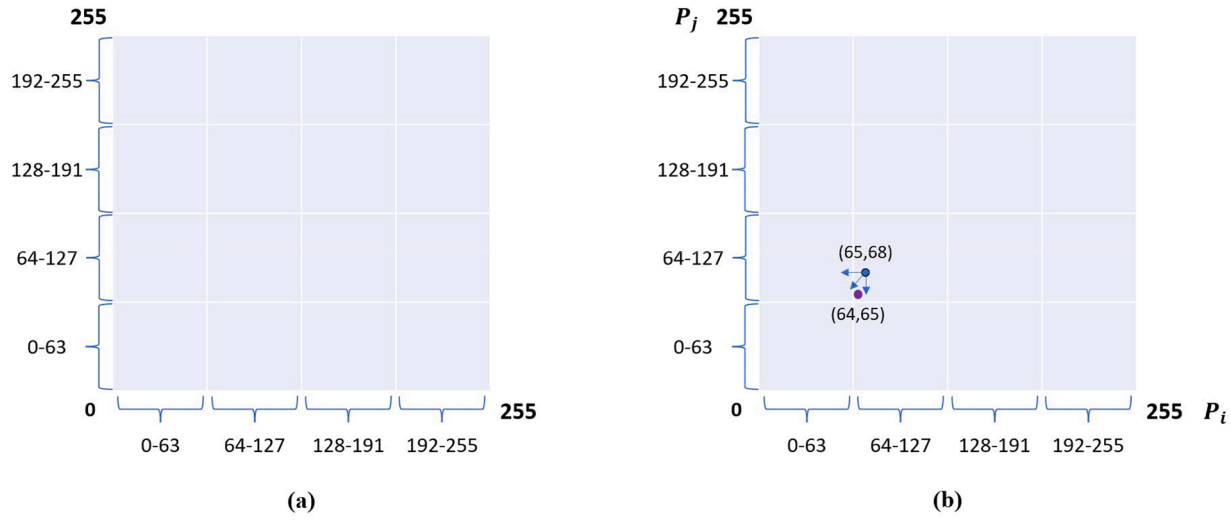


Figure 5. Fixed-Square Pixel-Pair Mapping.

When $t = 3$, Eq. (12) and Eq. (13) are employed, and three references (b_1,b_2,b_3) and one AC serve as coefficients in $f_1(x)$ and $f_2(x)$, and two extra random numbers r_1, r_2 are generated in [0,10].

$$f_1(x) = r_1x^2 + b_1x + b_2 \ (mod\ 11), \tag{12}$$

$$f_2(x) = r_2x^2 + b_3x + AC \ (mod\ 11). \tag{13}$$

When $t = 4$, only one polynomial $f_1(x)$ is required as defined in Eq. (14) three references (b_1,b_2,b_3) and one AC serve as coefficients.

$$f_1(x) = b_1x^3 + b_2x^2 + b_3x + AC \ (mod\ 11). \tag{14}$$

When $t > 4$, since more coefficients are required in $f_1(x)$in Eq. (15), r_1, …, r_{t-4} are random numbers generated in [0,10] beside the three references (b_1,b_2,b_3) and one AC serving as coefficients in $f_1(x)$.

$$f_1(x) = r_1x^{t-1} + \ldots + b_1x^3 + b_2x^2 + b_3x + AC \ (mod\ 11). \tag{15}$$

Based on the above different ts, their corresponding polynomial(s) of order $t - 1$ can be formed and the embedding value for the corresponding $f(x)$ value for the given xth cover image can be obtained. Note that the same value generation for secret image embedding will be conducted $256 \times 256 \times s$ times until all pixels of secret images are processed and s is the number of secret images.

Once all the secret images' pixels are processed, the share construction can begin. Two adjacent pixels $C_x(P_{i,x}, P_{j,x})$ in the xth cover image are transformed into two adjacent pixels $SC_x(P_{i,x}, P_{j,x})$ in the xth share image through our proposed slim turtle shell matrix shown in Fig. 3 while following our Fixed-Square Pixel-Pair Mapping (FSPPM) shown in Fig. 5(a). The FSPPM ensures that the new coordinates and the original ones are restricted to the same square. This restriction guarantees that the recomputed hash values of the selected 2MSBs of pixels in the shares match the hidden hash values derived from the cover images during the secret message extraction and verification phase. For example, in Fig. 5(b), as long as a new coordinate (64,65) falls in the same square as the original coordinate (65,68), the 2MSBs of both horizontal and vertical coordinates will not be changed after share construction.

During the share construction, our (t, n)-SIS combines our slim turtle shell matrix and fixed square-based pixel pairs' mapping principle. Two

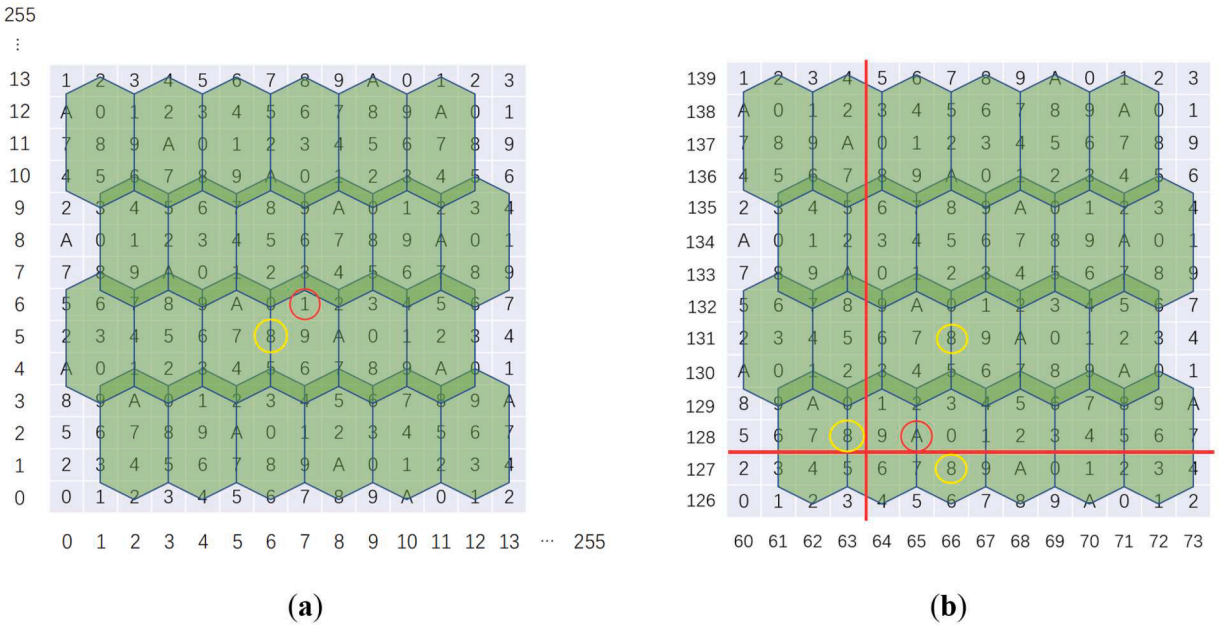


Figure 6. Examples of the combination of our slim turtle shell matrix and FSPPM.

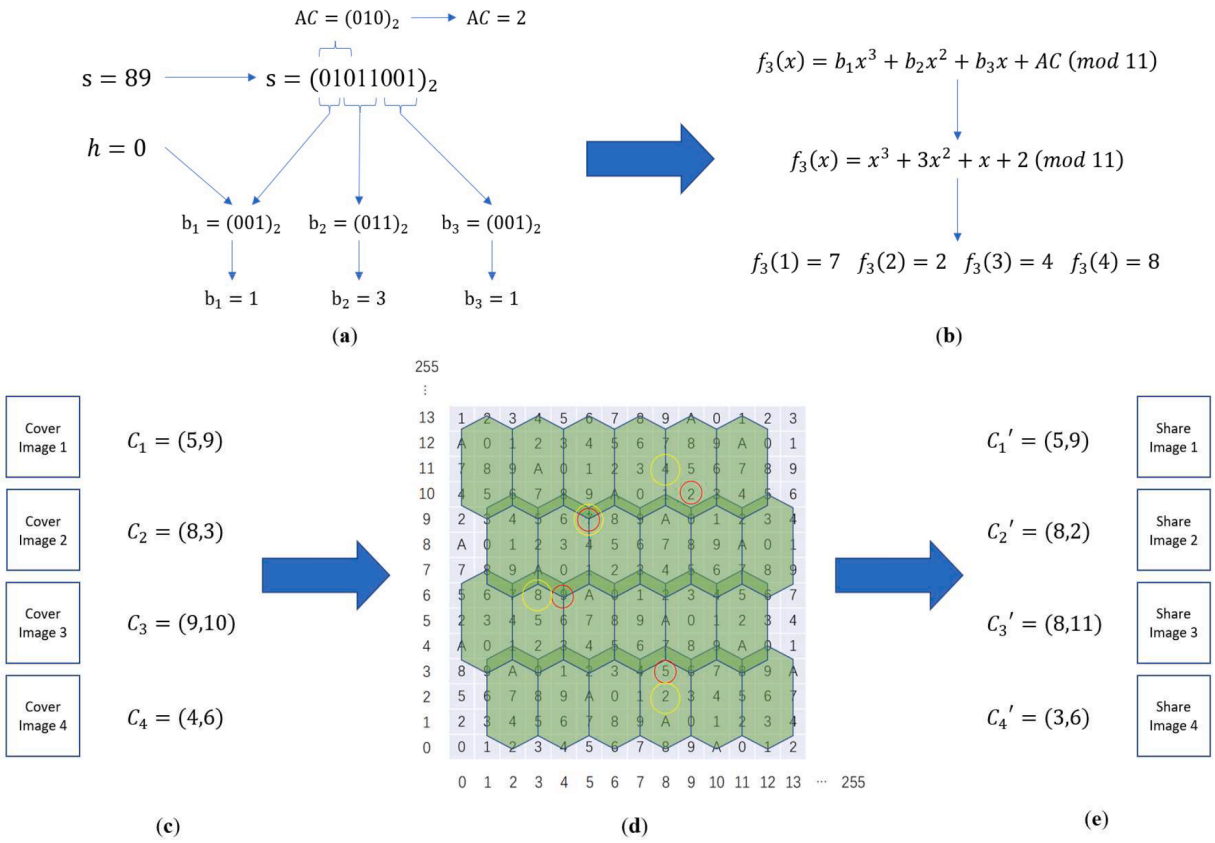


Figure 7. Example of the share construction. (a) Lagrange interpolation polynomial coefficients construction. (b) Lagrange interpolation polynomial equation construction. (c) Cover pixel pairs. (d) Embedding with slim turtle shell matrix. (e) Generation of share pixel pairs.

examples of pixel pairs' mapping are demonstrated in Figs. 6(a) and (b), respectively. In Fig. 6(a), the original pixel pair is marked in a red circle and has coordinates (7,6). The corresponding embedding value $f(x) = 8$ is used to find the closest coordinate. Here, we can see the closet coordinate is (6,5) marked in a yellow circle and its mapping value is 8. The original pixel pair (7,6) and the new pixel pair (6,5) are located in the (0-63) square which follows FSPPM in Fig. 5. Therefore, the original pixel pair (7,6) is successfully transformed into new pixel pair (6,5) to carry embedding value 8. In contrast, in Fig. 6(b), the original pixel pair is marked in a red circle and has coordinates (65,128). The function $f(x) = 8$ indicating the embedding value and is used to find the closest and second closest coordinates. The closest coordinate is (66,127), and the second closest coordinate is (63,128). However, these coordinates are not in the same square as the original coordinates, so they cannot be used. The closest coordinate within the same square is (66,131), as defined in Fig. 5. Therefore, the new pixel pair is (66,131).

To give a clear demonstration of our share construction, an example is depicted in Fig. 7 and a detailed description is given in the following: Let (t, n) be (4,4), meaning there are four different cover images involved and four shares are required to reconstruct the hidden secret message. In our example, each cover image is grayscale and sized as 512 × 512 pixels, while the secret image is 256 × 256 pixels. Two different secret images can be concealed with our (4,4)-SIS scheme. Here, C_x represents the coordinates of the same position in the xth cover image where $x \in [1, n = 4]$, S is the secret pixel value, h is the hash digest of 2MSBs of the n cover images, and AC is the authentication code derived from the corresponding secret pixel. Because $n = 4$ and the cover image is sized 512 × 512 pixels, the 2MSBs binary stream is sized as $512 \times 512 \times 2 \times 4 = 2097152$ bits. Here, we can embed 2 secret images with $256 \times 256 \times 2 = 131072$ pixels in total. During share construction, one hash bit can be simultaneously carried when Shamir's Lagrange interpolation polynomial [20] is utilized to conceal one pixel value of a secret image. Therefore, 131072 AC bits must be pre-generated since there are 131072 secret pixels. In our

Algorithm 2
Sharing Construction Algorithm.

Input: Cover images $C_i (1 \le i \le n)$ with size 512 × 512, 1 or 2 secret image(s) S with size 256 × 256, and the number of shares t.

Output: Share images $C_i' (1 \le i \le n)$ with size 512 × 512.
Step 1: Determine 1 or 2 secret image(s) can be embedded based on the number of shares t.
if $t < 4$ **do**
  The number of secret images is 1.
else
  The number of secret images is 2.
Step 2: Get the 2MSB from each pixel in each cover image, combine them into a sequence, divide them into num_{hash} segments of length l_{input}, input them to the SHA512 hash function, and combine all outputs into a hash sequence $sequence_{hash}$.
Step 3: Correspond each hash bit h of $sequence_{hash}$ to each secret pixel in order to construct polynomial coefficients.
if $t < 4$ **do**
Construct two polynomials $f_1(x)$ and $f_2(x)$ to represent $f(x)$.
else
Construct a polynomial $f_3(x)$ to represent $f(x)$.
foreach $j \in \{1, 2, \ldots, num_{pixelpair}\}$ **do**
$b_{1,j} = \{h, s_{1,j}, s_{2,j}\}$, $b_{2,j} = \{s_{3,j}, s_{4,j}, s_{5,j}\}$, $b_{3,j} = \{s_{6,j}, s_{7,j}, s_{8,j}\}$, $AC_j = \{s_{1,j}, s_{2,j}, s_{3,j}\}$ to construct $f_j(x)$.
Step 4: Modify the coordinates of the corresponding pixel pairs in the cover images by using the slim turtle shell matrix to embed the corresponding $f(x)$ values.
foreach $i \in \{1, 2, \ldots, n\}$ **do**
  foreach $j \in \{1, 2, \ldots, num_{pixelpair}\}$ **do**
    Find the coordinate (x_j', y_j') in the slim turtle shell matrix that is the closest to the coordinate (x_j, y_j) corresponding to the cover image pixel pair $C_i(x_j, y_j)$ and have a coordinate value equal to $f_j(i)$. The coordinates (x_j', y_j') correspond to pixel pair $C_i'(x_j', y_j')$ in the share image.
Step 5: Output n share images $C_i' (1 \le i \le n)$.

proposed scheme, we adopt SHA512 to generate the hash values for 2MSBs of pixels of the 4 cover images, meaning there are $num_{hash} = 256$ binary sequences according to Eq. (6) with lengths of 512 bits. Subsequently, 2MSBs of pixels of the 4 cover images must be divided into num_{hash}=256 and each binary sequence sized as $l_{input} = 8192$ bits according to Eq. (7). Here, we assume the first secret pixel value S is 89 of a secret image and the generated hash digest is (01010111…). Therefore, $h = 0$ and $AC = 2$ is converted from $(010)_2$ which maps to the first bit, second bit and third bit of the binary representation of 89. The binary stream of the secret pixel value S is $\{s_1,s_2,\ldots, s_8\} = (01011001)_2$. After 3-element grouping and padding the h, three groups are obtained as $\{h, s_1,s_2\} = (001)_2$, $\{s_3,s_4,s_5\} = (011)_2$, and $\{s_6, s_7,s_8\} = (001)_2$. After converting them to decimal, we can obtain $b1 = 5$, $b2 = 3$, $b3 = 1$. Using Eq. (12), we construct $f_3(x) = x^3 + 3x^2 + x + 2$ $(mod 11)$ to get $f_3(1) = 7$, $f_3(2) = 2$, $f_3(3) = 4$, and $f_3(4) = 8$. Embedding the equation value into the coordinates $C_1 = (5, 9)$, $C_2 = (8, 3)$, $C_3 = (9, 10)$, and $C_4 = (4, 6)$ through the slim turtle shell matrix and our FSPPM, the corresponding new coordinates $C_1' = (5,9)$, $C_2' = (8,2)$, $C_3' = (8,11)$, and $C_4' = (3,6)$ can be obtained. Finally, using the second cover image as an example, the first and the second pixels 8 and 3 will be changed to 8 and 2 in the second share once the share construction is complete.

To clarify the sharing construction steps, Algorithm 2 illustrates the details.

3.3. Secret message extraction and verification phase

The process of extraction and reconstruction is very simple. We only need to find two neighboring pixels as pairs and map them to the slim turtle shell matrix to obtain the value of $f(x)$. Then, use all $f(x)$ values of the t images to derive the coefficients $b1$, $b2$, $b3$, and AC of the Lagrange interpolation formula used in the share construction phase. As long as the values of $b1$, $b2$, $b3$, and AC are obtained, the corresponding secret pixel is obtained. Once all secret pixels are obtained, the secret images are completely recovered. As for the derived hash bits and ACs, the integrity of the secret images and the original cover images can be verified.

Verification is of utmost importance for secret sharing. By substituting the 2 MSBs of the share image into the SHA-512 hash function and comparing the resulting hash value with the hash value fragments derived during reconstruction, we can easily determine if the cover images have been tampered with. Furthermore, the tampered area can be identified by comparing whether s_1, s_2, s_3 and the hidden authentication code AC are equal.

Fig. 8 is an example of our proposed extraction and verification. Follow the same example demonstrated in Fig. 7. According to the coordinates of the first and second pixels of four different shares $C_1' = (5, 9)$, $C_2' = (8,2)$, $C_3' = (8,11)$, and $C_4' = (3,6)$, we can get their coordinate values through the slim turtle shell matrix, that is, the embedding values derived from $f(x)$ are $f_3(1) = 7$, $f_3(2) = 2$, $f_3(3) = 4$, $f_3(4) = 8$. After obtaining the coordinates of the four equations, the equation $f_3(x) = x^3 + 3x^2 + x + 2(mod 11)$ can be rebuilt. By splitting and calculating the coefficients b_1, b_2, b_3, and AC, the secret message can be extracted and whether it has been tampered with can be verified.

Algorithm 3 steps through the extraction and verification processes for easier understanding.

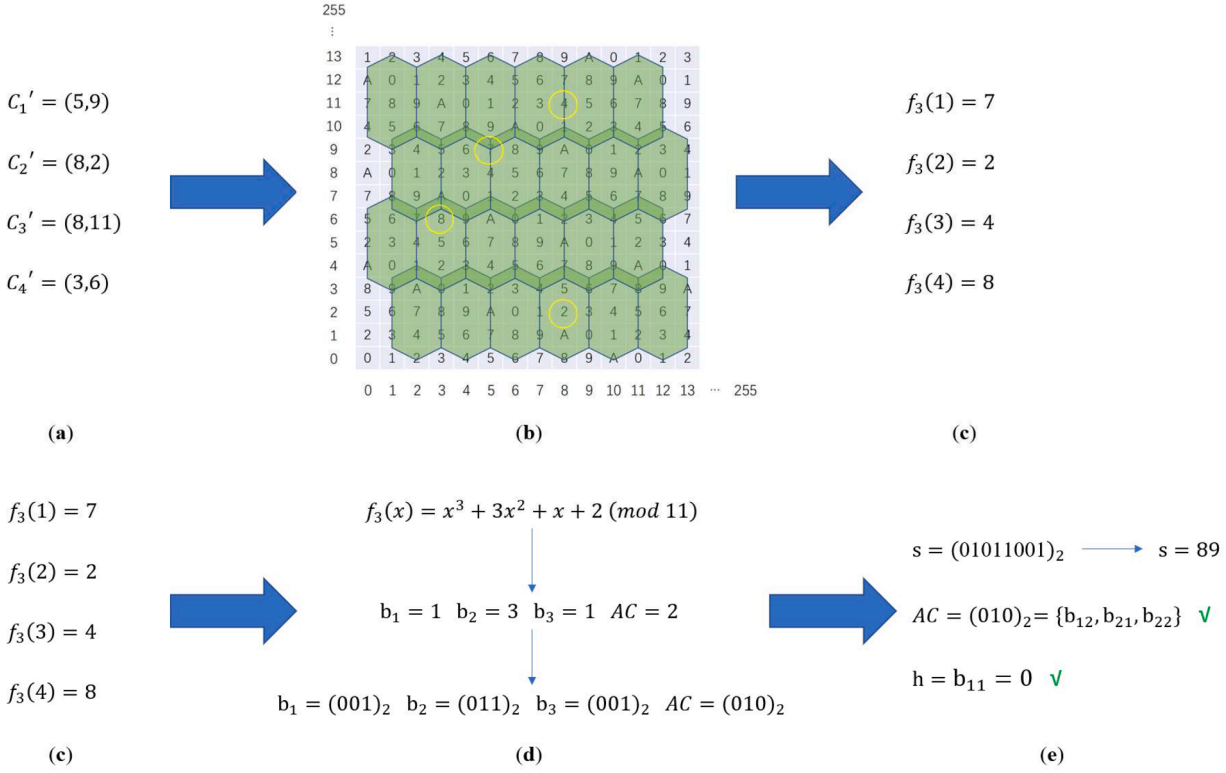


Figure 8. Example of the extraction and verification phase. (a) Four pixel pair shares. (b) Reconstruction with slim turtle shell matrix. (c) Deriving equation results. (d) Lagrange interpolation polynomial coefficients' reconstruction. (e) Extraction of hidden secret message and verification the integrity of cover pixels and secret pixels.

Algorithm 3
Secret Message Extraction and Verification Algorithm.

Input: Share images $C_i'(1 \le i \le t)$ with size 512 × 512.

Output: 1 or 2 secret image(s) S with size 256 × 256.
Step 1: Traverse and process the pixel pairs of the share images.
foreach $i \in \{1, 2, \ldots, n\}$ **do**
  foreach $j \in \{1, 2, \ldots, num_{pixelpair}\}$ **do**
    1.1: Find the coordinate value $f_j(i)$ corresponding to the pixel pair using the slim turtle shell matrix, where $f_j(i) = M(x_j', y_j')$ and (x_j', y_j') is the coordinate corresponding to pixel pair $C_i'(x_j', y_j')$ in the share image.
    1.2: Apply the Lagrangian interpolation algorithm to all coordinates (x_j', y_j') to obtain the coefficients $b_{1,j}$, $b_{2,j}$, $b_{3,j}$ and AC_j of $f_j(x)$.
    1.3: Combine $b_{12,j}$, $b_{13,j}$, $b_{2,j}$ and $b_{3,j}$ to obtain the secret pixel s_j.
    1.4: Verify if AC_j matches the combination of $b_{12,j}$, $b_{13,j}$, $b_{21,j}$. If they match, it indicates that the j-th pixel pair in the image has not been tampered. If they do not match, it suggests the j-th pixel pair has been tampered.
    1.5: Get hash bit $h_j = b_{11,j}$.
Step 2: Combine all secret pixels to reconstruct the secret image(s) and combine all hash bits to reconstruct the $sequence_{hash} = h_1 \parallel h_2 \parallel \ldots \parallel h_{num_{pixel\ pair}}$.
Step 3: Get the 2MSB from each pixel in each share image, combine them into a sequence, divide them into segments of length l_{input}, input them to the SHA512 hash function, compare the output with its corresponding position in $sequence_{hash}$.
if $output \neq sequence_{hash}$(correspondingposition)
    The share image corresponding to the segment has been tampered.
else
    The share image corresponding to the segment has not been tampered.
Combine with Step 2, it allows us to accurately locate which pixel pair in which share image has been tampered.
Step 4:Output the secret image(s) S.

4. Experimental Results

To evaluate the performance of our proposed (t, n)-SIS scheme, our proposed SIS scheme is implemented using MATLAB R2022b on a laptop with a Windows 10 operating system. The laptop's CPU is the 3.20 GHz AMD Ryzen 7 and it has 16 GB of RAM. Several experiments are conducted with four grayscale secret images with a size of 256 × 256 pixels shown in Fig. 9 and eight grayscale cover images with a size of 512 × 512 pixels shown in Fig. 10. In this section, experimental results with different combinations of (t, n) are demonstrated. Moreover, comparisons with some state-of-the-art schemes are also presented to show the advantages of our proposed scheme.

PSNR (peak signal-to-noise ratio, unit is dB), SSIM (structural similarity index), and NC (normalized correlation) are three important metrics used in this paper to evaluate the performance of our proposed scheme. PSNR estimates the difference in visual quality between two different images, which can be obtained with Eq. (16) and Eq. (17). In Eq. (17), m and n represent the size of the image, while i and j represent the corresponding coordinate. I represents the original pixels, while K represents the share pixels in this paper. The smaller the MSE, the lower the degree of distortion and the better the image quality of the generated share. SSIM is also used to measure the difference in visual quality between two images via luminance, contrast, and structure which can be calculated by Eq. (18), where μ represents the mean of the image, σ represents the variance of the image, and C_1 and C_2 are constants added for numerical stability. In Eq. (19), NC represents the degree of correlation between two images, where w and h represent the width and height of the images, respectively. Meanwhile, i and j represent the coordinate, and B and B' represent the pixel values of the two images. Both SSIM and NC are indicators for measuring image similarity and correlation. The similarity between the two images increases as the SSIM value or NC value approaches 1.

$$PSNR = 10 \times log_{10}\frac{255^2}{MSE}, \tag{16}$$

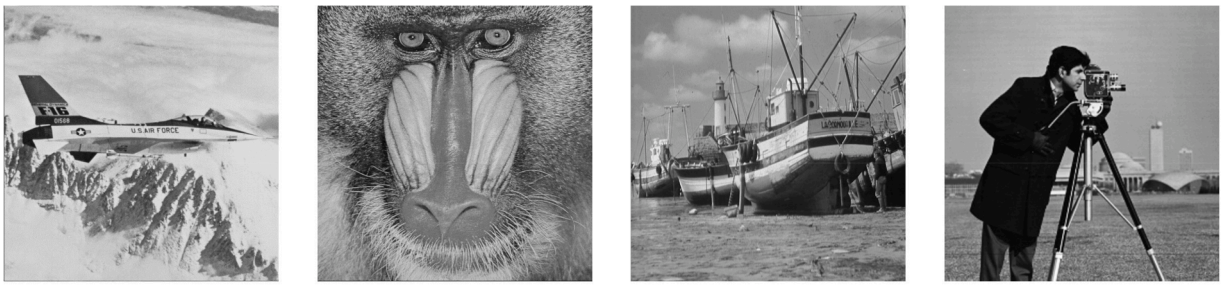
$$MSE = \frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^2, \tag{17}$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{18}$$

$$NC = \frac{\sum_{i=1}^{w}\sum_{j=1}^{h}\overline{(B_{i,j} \oplus B_{i,j}')}}{w \times h}, \tag{19}$$

Table 1 and Table 2 display the experimental results with our proposed SIS under two different scenarios of (t, n). Table 1 shows PSNRs, SSIMs and NCs with (t, n) = (2, 3), in this case, t = 2 is less than 4; therefore, only one secret image sized 256 × 256 pixels is carried according to Eq. (5). In Table 1, we can see that for both SSIMs and NCs, they are always close to 0.99, which means the share images are almost the same as the original cover images. Two different secret images are tested, causing a slight impact on PSNRs, but consistent results are confirmed. Table 2 shows PSNRs, SSIMs and NCs with (t, n) = (4, 8). In this case, two secret images sized 256 × 256 pixels are carried according to Eq. (5). Table 1 shows the average PSNRs of three cover images with different secret images, "Airplane" or "Baboon" are close to 45.8 dB. It is noted that "Airplane" is a relatively smooth image compared with "Baboon." Such results confirm the different secret images' types will not significantly affect the distortion during the share construction phase. In addition, the average visual quality of the shares listed in Table 2 is also higher than 45 dB. A higher PSNR indicates that the signal is less distorted by the hidden data. In this case, the PSNR of the shares is between 45.72 dB and 45.90 dB in Table 2, which is considered to be high quality. It means that the share images are still visually appealing even after the two secret images have been embedded. The evidence presented demonstrates that even when the number of secret images is augmented to two, the visual integrity of our proposed scheme remains mostly unaffected by the quantity of secret images.

From Tables 1 and 2, we can see that for both SSIMs and NCs, they are always close to 0.99, which means the share images are almost the same as the original cover images. Once again, we can see that our scheme under two different scenarios, (2,3) and (4,8), has consistent SSIMs and NCs. Four extracted secret images are also shown in Fig. 11 and their constructed shares with (4,8)-SIS are also listed in Fig. 12.



(a) (b) (c) (d)

Figure 9. Secret images sized 256 × 256 pixels. (**a**) "Airplane". (**b**) "Baboon". (**c**) "Boat". (**d**) "Cameraman".

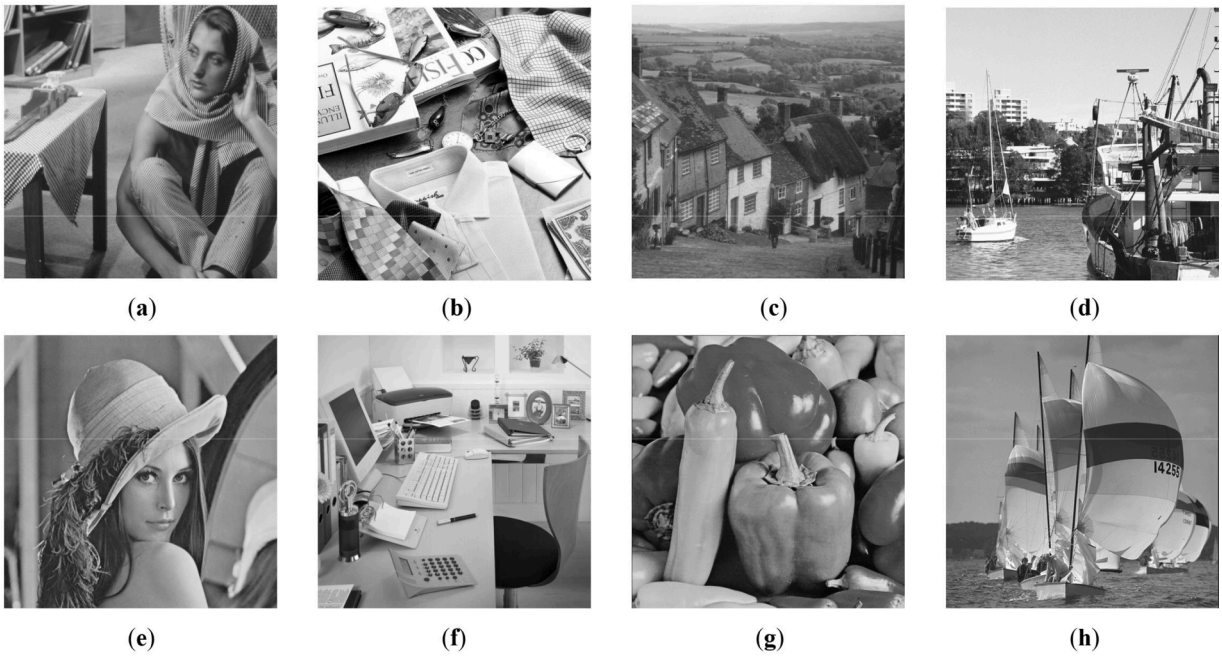


Figure 10. Cover images sized 512 × 512 pixels. (**a**) "Barbara". (**b**) "Fashion". (**c**) "Goldhill". (**d**) "Harbor". (**e**) "Lena". (**f**) "Office". (**g**) "Peppers". (**h**) "Sailboats".

Table 1
Visual quality of three share images when $(t, n) = (2, 3)$ SIS is adopted and single secret image is carried.

Cover Images \ Secret Images		Airplane	Baboon
Barbara	PSNR	45.81	45.80
	SSIM	0.994	0.994
	NC	0.9998	0.9998
Fashion	PSNR	45.88	45.89
	SSIM	0.996	0.996
	NC	0.9999	0.9999
Goldhill	PSNR	45.79	45.77
	SSIM	0.994	0.994
	NC	0.9998	0.9998

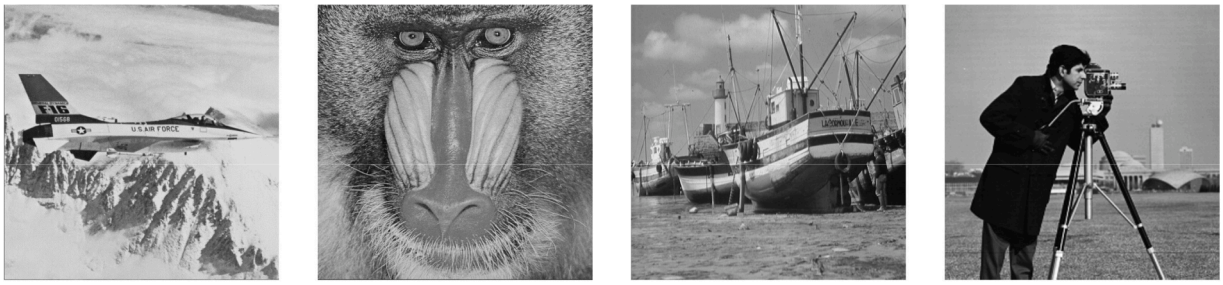
Table 2
Visual quality of eight shares when $(t, n) = (4, 8)$ SIS is adopted and two different secret images are carried.

Cover Images \ Secret Images		Airplane and Baboon	Boat and Cameraman
Barbara	PSNR	45.80	45.80
	SSIM	0.994	0.994
	NC	0.9998	0.9998
Fashion	PSNR	45.90	45.88
	SSIM	0.996	0.996
	NC	0.9999	0.9999
Goldhill	PSNR	45.78	45.76
	SSIM	0.994	0.994
	NC	0.9998	0.9998
Harbor	PSNR	45.85	45.86
	SSIM	0.994	0.994
	NC	0.9999	0.9999
Lena	PSNR	45.76	45.77
	SSIM	0.992	0.992
	NC	0.9998	0.9998
Office	PSNR	45.72	45.75
	SSIM	0.990	0.990
	NC	0.9998	0.9998
Peppers	PSNR	45.80	45.79
	SSIM	0.992	0.992
	NC	0.9998	0.9998
Sail	PSNR	45.77	45.76
	SSIM	0.991	0.991
	NC	0.9997	0.9997

Here, we can see that if there is no attack, the extracted secret images will be completely lossless and remain the same as the original secret images.

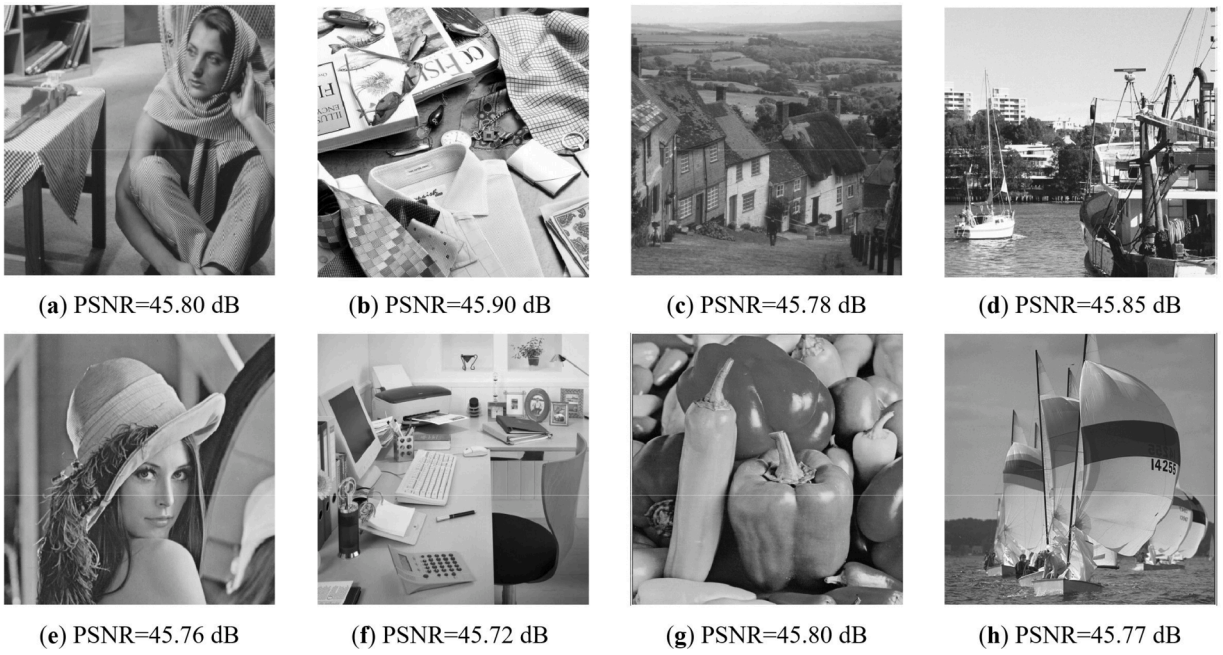
To demonstrate that our SIS scheme performs consistently well on both PSNR and SSIM, we randomly selected 100 images sized 512 × 512 pixels from the BOWS-2 database [39] as cover images, and 2 images sized 512 × 512 pixels as secret images. Fig. 13 and Fig. 14 show the probability distributions of PSNR values and SSIM values for the shares, respectively. As can be seen from the figures, the probability distributions of PSNR and SSIM values are both centered around 45 dB and 0.98, respectively. The bar chart displayed in Fig. 13 reveals that the highest bar corresponds to the range between 45.8 dB and 46.0 dB. This indicates that approximately 50% of our test images maintain an image quality within this specific range. Fig. 14 demonstrates that across the 100 test images as the cover images, the SSIM remains high between the original cover images and their corresponding shares, consistently surpassing 0.984. Notably, 3% of the test images exhibit an SSIM within the range of 0.998 and 1, further emphasizing the exceptionally high similarity between the shares and their corresponding cover images. This indicates that our SIS scheme is able to generate shares with high quality, both in terms of PSNR and SSIM. This makes our SIS scheme a promising candidate for secure image sharing applications.

Table 3 presents a comparison between our proposed scheme and six representative schemes [31,32,33,35,37,40], highlighting our competitive performance across eight features. Notably, Gao et al.'s [32] and Chen et al.'s [37] schemes belong to the SIS category; however, the former is a (2,3)-SIS scheme, while the latter is a (2, n)-SIS scheme. In contrast, our scheme is an (t, n)-SIS scheme, with t not limited to 2. As a result, our proposed (t, n)-SIS scheme exhibits superior scalability compared to Gao et al.' s [32] and Chen et al.'s [37] schemes, as well as outperforming other existing approaches. In Table 3, we set n as 4 for Chen et al.'s scheme, while both t and n are set as 4 and 8, respectively, for our scheme, demonstrating the maximal embedding capacity for both schemes. Schemes of Chen et al. and ours can verify the integrity of the extracted secret image(s). Although our scheme's average PSNR is 45.80 dB, which is lower than the 49.88 dB achieved by Chen et al.'s [37] scheme, it is important to note that our hiding capacity is 1,572,864 bits, which is twice the capacity of Chen et al.'s scheme (786,432 bits). Furthermore, while Chen et al.'s scheme offers a fixed hiding capacity,



(a) PSNR=∞ dB (b) PSNR= ∞ dB (c) PSNR= ∞ dB (d) PSNR= ∞ dB

Figure 11. Reconstructed secret images. (**a**) "Airplane" (PSNR=∞). (**b**) "Baboon" (PSNR=∞). (**c**) "Boat" (PSNR=∞). (**d**) "Cameraman" (PSNR=∞).



(a) PSNR=45.80 dB (b) PSNR=45.90 dB (c) PSNR=45.78 dB (d) PSNR=45.85 dB

(e) PSNR=45.76 dB (f) PSNR=45.72 dB (g) PSNR=45.80 dB (h) PSNR=45.77 dB

Figure 12. Constructed shares. (**a**) "Barbara" (PSNR=45.80). (**b**) "Fashion" (PSNR=45.90 dB. (**c**) "Goldhill" (PSNR=45.78). (**d**) "Harbor" (PSNR=45.85). (**e**) "Lena" (PSNR=45.76). (**f**) "Office" (PSNR=45.72). (**g**) "Peppers" (PSNR=45.80). (**h**) "Sail" (PSNR=45.77).

our scheme provides an adaptive hiding capacity that depends on the value of t. It is also worth mentioning that our proposed scheme embeds the 3MSBs of pixels from the secret images, enabling verification of the integrity of the extracted secret images once the integrity of received shares has been authenticated. Later, in Table 4, we will also prove the verification capability offered by ours outperforms that of Chen et al.'s scheme [37].

For a comprehensive security analysis of our proposed scheme, we employ several metrics, including histogram analysis, NCPR (Number of Changing Pixel Rate), UACI (Unified Averaged Changed Intensity), and entropy. From histogram analysis, the distribution of pixels in an image is depicted. NCPR and UACI are two measurements that evaluate the differences in cover and share images' pixels. NCPR represents the number of changed pixels between the cover image and the shared image, while UACI represents the average changing intensity between the two images. As shown in Eqs. (20)-(22), i and j represents pixel coordinates, respectively. C_1 and C_2 in Eq. (20) represents the cover image and the share image, respectively. If they are equal, then the difference $D = 0$; otherwise, the difference $D = 1$. In Eq. (21), the sum of D values is divided by the size of the image, $M \times N$, to obtain the NPCR value. In Eq. (22), the sum of differences between images C_1 and C_2 is calculated and then divided by $M \times N \times 255$. The value 255 in the formula represents the maximum difference and is equal to the maximum value of 255 minus the minimum value of 0 which results in 255. As illustrated in Eq. (23), where$p(x)$ represents the probability of a certain pixel value x appearing in the image.

$$D(i,j) = \begin{cases} 0, & if\ C_1(i,j) = C_2(i,j), \\ 1, & otherwise \end{cases} \tag{20}$$

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times\ 100\%. \tag{21}$$

$$UACI = \frac{\sum_{i,j} |(C_1(i,j) - C_2(i,j))|}{M \times N \times\ 255} \times\ 100\%. \tag{22}$$

$$H(X) = -\sum [p(x)\ \times log_2(p(x))]. \tag{23}$$

Fig. 15 depicts histograms of cover images and share images. Because the share images carry meaningful content, it is apparent that the histograms of the cover image and a randomly selected share image from the generated set closely resemble each other from Fig. 15. Take Figures 15(a) and (b) as an illustrative example, both are associated with the test image "Barbara". The general pattern observed in the histogram derived from the original image "Barbara" (a) is preserved in the histogram of the share image "Barbara" (b) using our proposed scheme.

By referring to Table 4, it becomes evident that during the share

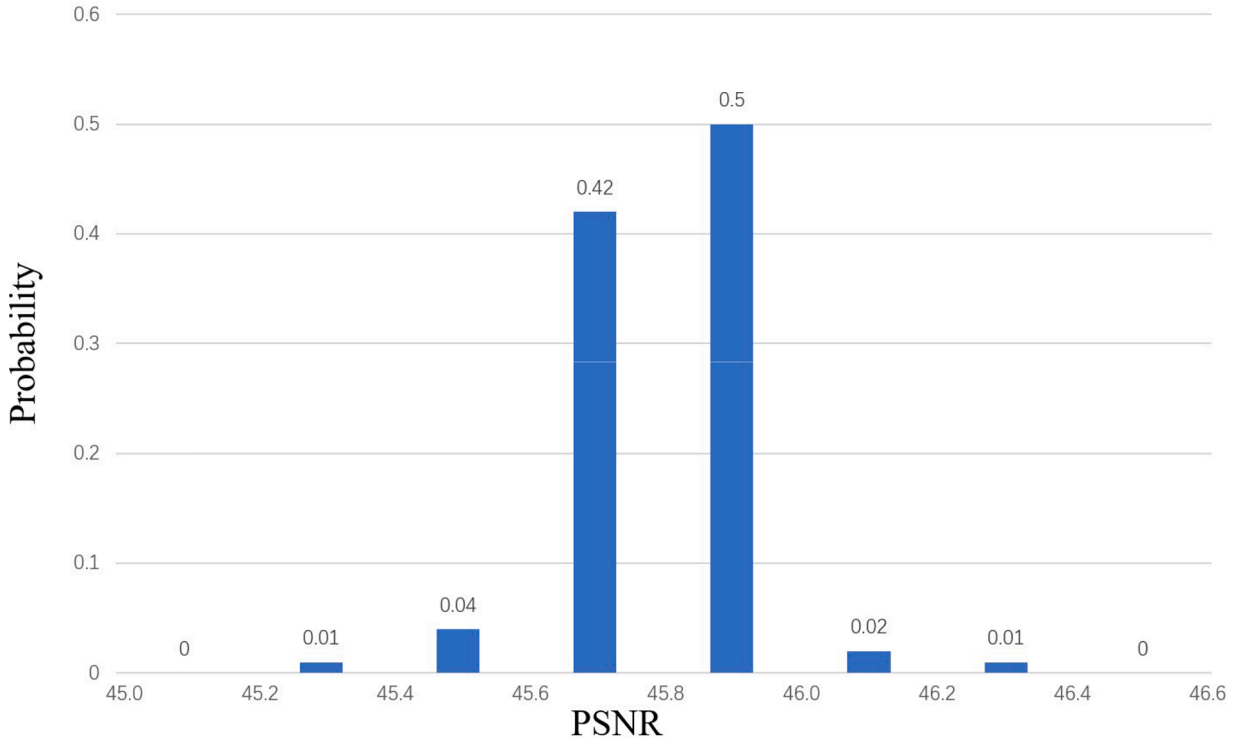


Figure 13. The probability distribution of PSNRs for the shares using 100 random cover images.

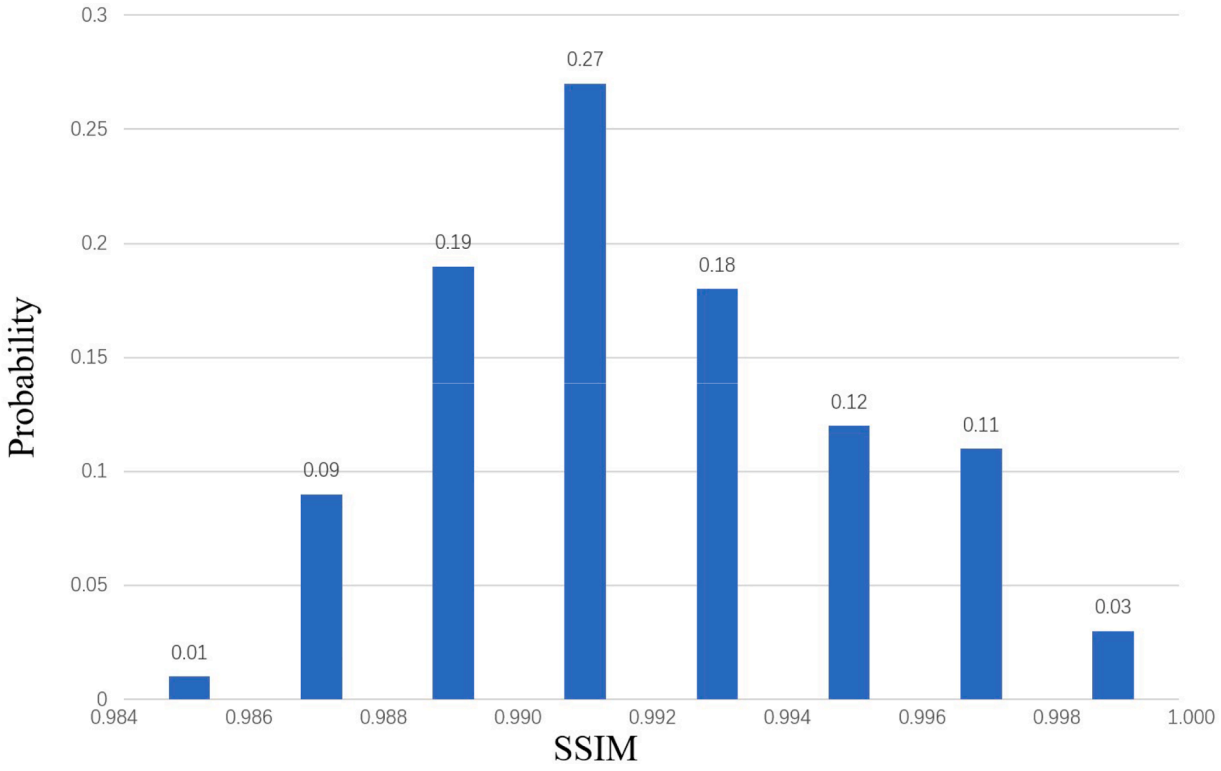


Figure 14. The probability distribution of SSIMs for the shares using 100 random cover images and 2 secret images in database BOWS-2.

construction phase, 70% of the pixels in the cover images are experienced alterations. However, the average intensity of these changes remains around 0.41%. Furthermore, the entropy values of the share images approximate 8 which not only mirrors the entropy of the cover images but also approaches the optimal value of 8. These findings bolster the security performance of the scheme we have proposed.

It is noted that our proposed scheme can be modified and improved if the secret image extraction and verification phase need to be speeded up. Initially, our verification method treated each segment as the input of a hash function and produced many hash function outputs as

Table 3
Comparisons among our proposed scheme and six existing schemes.

Features	Liu et al.'s Scheme [31]	Liu et al.'s Scheme [35]	Gao et al.'s Scheme [33]	Li et al.'s Scheme [40]	Gao et al.'s Scheme [32]	Chen et al.'s Scheme [37]	Proposed Scheme
Meaningful shares	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reversibility of cover images	Yes	-	Yes	Yes	-	Yes	-
Different cover images	-	Yes	-	-	Yes	Yes	Yes
(k, n)-SIS	(2, 2)-SIS	(2, 2)-SIS	(2, 2)-SIS	(3, 3)-SIS	(2, 3)-SIS	(2, n)-SIS	(t, n)-SIS
Fault tolerance	-	-	Yes	-	Yes	Yes	Yes
Average PSNR of shares	48.72 dB	41.72 dB	46.63 dB	49.07 dB	40.41 dB	49.88 dB	45.80 dB
Embedding capacity	524,288 bits	785,525 bits	819,200 bits	624,215 bits	1,310,720 bits	786,432 bits	1,572,864 bits
Integrity check of hidden secret data	-	-	-	-	-	Yes	Yes

Table 4
Average NPCR and UACI between cover images and share images and their respective entropy values.

Images	NPCR	UACI	Entropy of Cover Images	Entropy of Share Images
Barbara	73.01%	0.41%	7.632	7.586
Fashion	72.76%	0.40%	7.740	7.643
Goldhill	73.20%	0.41%	7.478	7.412
Harbor	72.77%	0.41%	7.461	7.386
Lena	73.21%	0.41%	7.446	7.361
Office	72.53%	0.41%	7.605	7.405
Peppers	72.82%	0.41%	7.594	7.532
Sailboat	73.39%	0.41%	7.220	7.140

described in Section III B. Share construction phase. Although this architecture is quite complete, its calculation time is longer than other phases relatively. In order to shorten the calculation time of the hash function, our verification method can perform XOR calculations on all segments in order first obtain a hash value. We can treat it a variant from the proposed scheme. Taking $t = 4$ as an example, the original calculation of 256 hash functions is reduced to one hash function using the variant.

Table 5 presents the execution time for each phase. Notably, the slim turtle shell construction phase exhibits exceptional speed, completing in just 0.0011 seconds. However, the share construction phase demands 362.3578 seconds primarily due to the value of $n = 8$ resulting in a time per image (TPI) of 45.2946 seconds. When $t = 4$ and a single attack is considered, the TPI of secret message extraction and verification with

Table 5
Execution time of each phase when (t, n) = (4, 8).

Phase	(Original scheme) Time (s)	(Variant scheme) Time (s)
Slim Turtle Shell Construction	0.0011	0.0011
Share Construction	362.3578 (TPI: 45.2946)	23.3177 (TPI: 2.9147)
Secret Message Extraction and Verification (Single Attack Area)	97.6888 (TPI: 24.4222)	1.5413 (TPI: 0.3853)
Secret Message Extraction and Verification (Multiple Attack Areas)	98.4476 (TPI: 24.4278)	1.5517 (TPI: 0.3879)

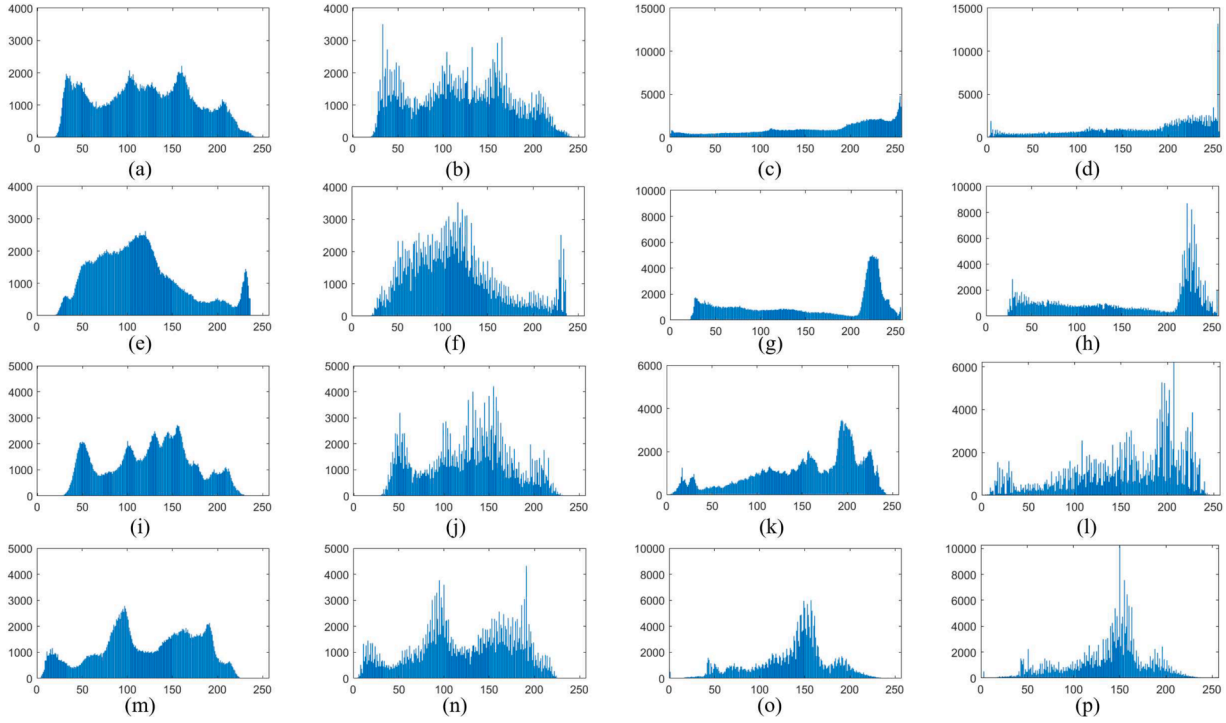


Figure 15. Histogram of cover images and their corresponding share images. (a) Cover image "Barbara". (b) Share image "Barbara". (c) Cover image "Fashion". (d) Share image "Fashion". (e) Cover image "Goldhill". (f) Share image "Goldhill". (g) Cover image "Harbor". (h) Share image "Harbor". (i) Cover image "Lena". (j) Share image "Lena". (k) Cover image "Office". (l) Share image "Office". (m) Cover image "Peppers". (n) Share image "Peppers". (o) Cover image "Sailboat". (p) Share image "Sailboat".

the original scheme can be reduced from 24.2222 seconds to 0.3853 seconds with the variant scheme. More importantly, such computing performance is not affected by different attack sizes. Table 5 illustrates that the secret message extraction and verification phase only requires 0.3879 seconds with our variant scheme even in scenarios where multiple attacks occur. Here, we have to emphasize that 0.3879 seconds is very close to the 0.3853 seconds observed when only a single attack occurs. The experimental findings demonstrate that the required verification time remains nearly constant without being affected by variations in the number or size of the attacks.

Figs. 16(a), (b), and (c) depict fake logos, with (a) being a special image which is a smooth image, (b) a standard image, and (c) uniform noise following the discussions made in [32]. Fig. 16(d) is a share image that has been tampered with a fake logo, whereas (e), (f), and (g) are share images that remain untampered. The reconstructed share image is presented in (h), with the tampered parts marked in black. The found damaged secret image is illustrated in (i), with the tampered part marked in black. Similarly, (j) shows the reconstructed secret image without corrupting in the corresponding position on the share image.

Additionally, we employed the identical counterfeit logo, as depicted in Figs. 16(a), (b), and (c), to showcase outcomes across multiple attack regions. In Fig. 17, (a) and (b) depict the tampered share images, while (c) and (d) show the share images that remain untampered. (e) and (f) represent the reconstructed share images, with tampered parts marked in black. (g) is the reconstructed secret image with tampered parts marked in black, and (h) is the reconstructed, untampered secret image. Based on experimental results shown in Fig. 17, we can conclude that our scheme demonstrates a high degree of accuracy in detecting tampered areas and maintains a reasonable level of fault tolerance.

Eq. (24) shows the Bit Error Rate (BER). Table 6 presents a comparison of the BER values between our proposed scheme and the schemes by Gao et al. [32] and Chen et al. [37] across various image types. Typically, the BER is influenced by the value of t, with Chen et al.'s schemes tested using $t = 2$. In contrast, our proposed scheme allows t to be any number greater than or equal to 2. In the experiment, we have set $t = 4$. As the BER value approaches 1, the verification ability becomes stronger and a value of 1 signifies a completed verification. Our scheme has the capability to completely detect altered parts in theory. However, tampered pixels with values identical to the original pixels may be misclassified as untampered during the detection process and cause the BER value hard to reach 1. Experimental results show that our schemes, whether using the original authentication mechanism or the variant authentication mechanism, exhibit relatively strong verification capabilities. In particular, not only in the case of single attack areas (SAA), but also in the case of multiple attack areas (MAA), we consider two areas of the same size which yields favorable results. Given that

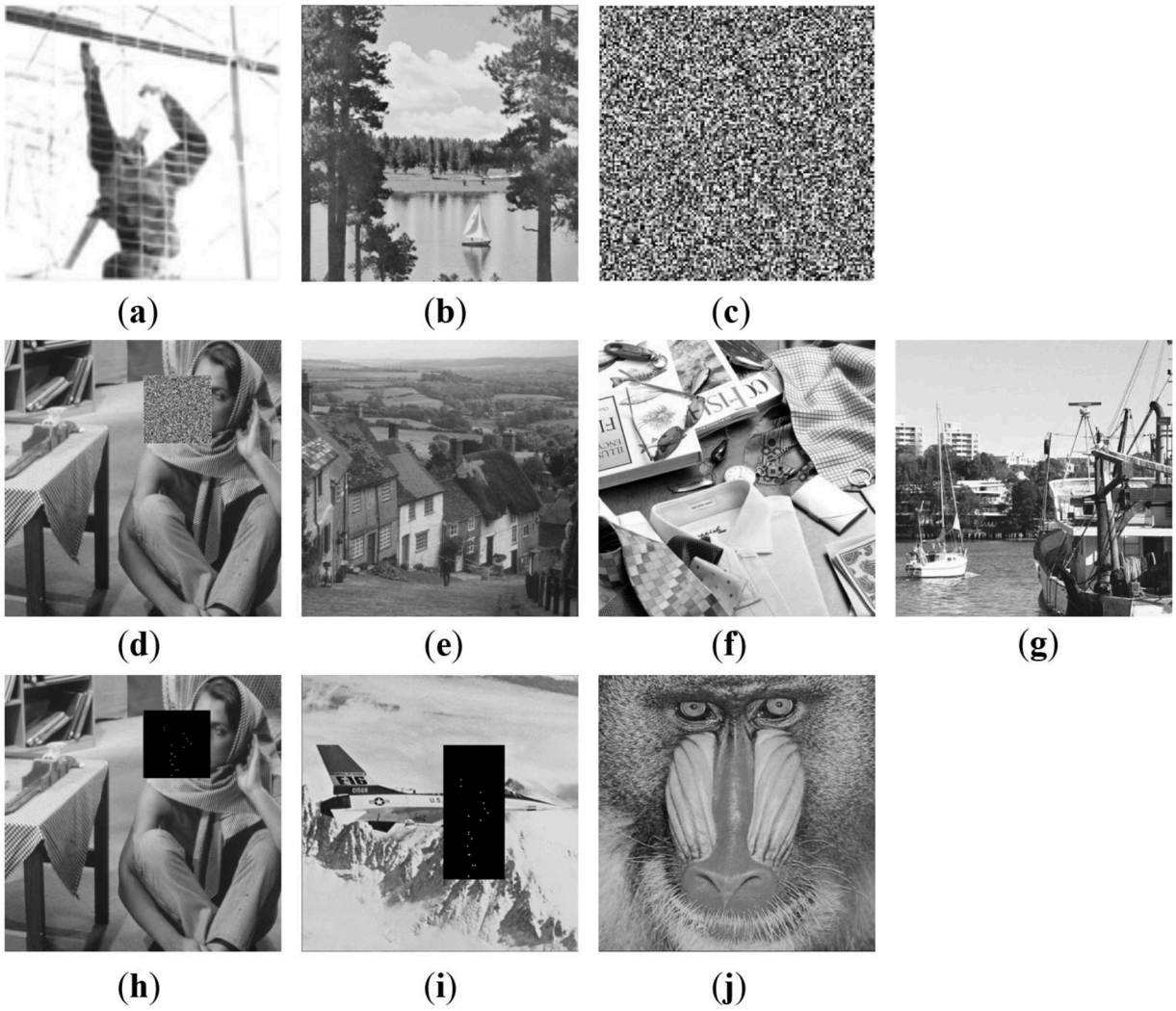


Figure 16. Authentication examples. (single attack area) (**a**) Special image (fake logo). (**b**) Standard image (fake logo). (**c**) Uniform image (fake logo). (**d**) Share image with tampering (fake logo). (**e**)-(**g**) Share image without tampering (real cover). (**h**) Share image with tampered parts marked in black. (**i**) Reconstructed secret image with tampered parts marked in black. (**j**) Reconstructed secret image without any tampering.

Figure 17. Authentication examples. (**a**)-(**b**) Share images with tampering (fake logo). (**c**)-(**d**) Share images without tampering (real cover). (**e**)-(**f**) Share images with tampered parts marked in black. (**g**) Reconstructed secret image with tampered parts marked in black. (**h**) Reconstructed secret image without any tampering.

Table 6
BER value comparison of our scheme and schemes of Gao et al.'s and Chen et al.'

Tempered Image	Gao et al.'s Scheme [32]		Chen et al.'s Scheme [37]	Original (SAA)	Original (MAA)	Variant (SAA)	Variant (MAA)
t	2	2	2	4	4	4	4
Special image	0.684	0.942	0.997	0.999	0.999	0.999	0.999
Standard image	0.084	0.834	0.995	0.998	0.997	0.998	0.997
Uniform noise	0.097	0.836	0.995	0.997	0.995	0.997	0.995

attack detection relies on AC rather than hash values, the Bit Error Rates (BERs) shown in Table 6 remain consistent whether using the original scheme or the variant scheme.

$$BER = \frac{the\ number\ of\ truly\ detected\ pixels}{the\ number\ of\ fake\ pixels} \times\ 100\% \tag{24}$$

5. Conclusion

In this paper, an innovative secret sharing scheme, referred to as the slim turtle shell scheme, is proposed to safeguard secret messages. In this scheme, secret messages are allocated to n share images, and a minimum of t share images is required to extract the concealed secret message. To enhance the visual quality of the hidden secret images, we employ a new slim turtle shell scheme that offers a substantial embedding capacity while maintaining good visual quality. Moreover, we introduce two authentication mechanisms based on SHA-512 and the 3MSBs of secret images, respectively. These mechanisms utilize FSPPM concealed within a polynomial function of $t-1$ degrees during the share construction phase. As a result, the integrity of hidden image(s) and cover images can be successfully verified during the secret message extraction and verification phase, ultimately enhancing the security of our scheme. The experimental results validate that our proposed scheme exhibits a high payload and satisfactory visual quality, with probability distributions of PSNR and SSIM values centered around 45 dB and 0.98, respectively. Additionally, it demonstrates excellent authentication capability, with an average Bit Error Rate (BER) of up to 0.998.

Authorship contributions

Category 1

Conception and design of study: Chin-Chen Chang, Chia-Chen Lin, Yijie Lin; analysis and/or interpretation of data: Chin-Chen Chang, Chia-Chen Lin, Yijie Lin.

Category 2

Drafting the manuscript: Yijie Lin; revising the manuscript critically for important intellectual content:Chin-Chen Chang, Chia-Chen Lin, Jui-Chuan Liu.

Category 3

Approval of the version of the manuscript to be published:Chin-Chen Chang, Yijie Lin, Chia-Chen Lin, Jui-Chuan Liu.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to data.

References

[1] Chan C K, Cheng L M. Hiding data in images by simple LSB substitution. Pattern Recognition 2004;37(3):469–74.

[2] Mielikainen J. LSB matching revisited. IEEE Signal Processing Letters 2006;13(5):285–7.
[3] Wang R Z, Lin C F, Lin J C. Image hiding by optimal LSB substitution and genetic algorithm. Pattern Recognition 2001;34(3):671–83.
[4] Ker A D. Improved detection of LSB steganography in grayscale images. Information Hiding: 6th International Workshop, IH 2004, Toronto, Canada, May 23-25, 2004, Revised Selected Papers 6. Springer Berlin Heidelberg; 2005. p. 97–115.
[5] Ni Z, Shi Y Q, Ansari N, et al. Reversible data hiding. IEEE Transactions on Circuits and Systems for Video Technology 2006;16(3):354–62.
[6] Qin C, Chang C C, Huang Y H, et al. An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism. IEEE Transactions on Circuits and Systems for Video Technology 2012;23(7):1109–18.
[7] Zhang X, Wang S. Efficient steganographic embedding by exploiting modification direction. IEEE Communications Letters 2006;10(11):781–3.
[8] Kim H J, Kim C, Choi Y, et al. Improved modification direction methods. Computers & Mathematics with Applications 2010;60(2):319–25.
[9] Liu Y, Chang C C, Huang P C, et al. Efficient information hiding based on theory of numbers. Symmetry 2018;10(1):19.
[10] Tian J. Reversible data embedding using a difference expansion. IEEE Transactions on Circuits and Systems for Video Technology 2003;13(8):890–6.
[11] Alattar A M. Reversible watermark using the difference expansion of a generalized integer transform. IEEE Transactions on Image Processing 2004;13(8):1147–56.
[12] Chang C C, Chou Y C, Kieu TD. An information hiding scheme using Sudoku. In: 2008 3rd international conference on innovative computing information and control. IEEE; 2008. p. 17. -17.
[13] Hong W, Chen T S, Shiu CW. A minimal Euclidean distance searching technique for Sudoku steganography. In: 2008 International Symposium on Information Science and Engineering. 1. IEEE; 2008. p. 515–8.
[14] Kieu D, Wang Z H, Chang C C, Li MC. A Sudoku based wet paper hiding scheme. Int. J. Smart Home 2009;3:1–11.
[15] Chang C C, Liu Y, Nguyen TS. A novel turtle shell based scheme for data hiding. In: 2014 tenth international conference on intelligent information hiding and multimedia signal processing. IEEE; 2014. p. 89–93.
[16] Liu Y, Chang C C, Nguyen TS. High capacity turtle shell-based data hiding. IET Image Processing 2016;10(2):130–7.
[17] Jin Q, Li Z, Chang C C, et al. Minimizing Turtle-Shell Matrix Based Stego Image Distortion Using Particle Swarm Optimization. Int. J. Netw. Secur. 2017;19(1):154–62.
[18] Liu L, Chang C C, Wang A. Data hiding based on extended turtle shell matrix construction method. Multimedia Tools and Applications 2017;76:12233–50.
[19] Xie X Z, Lin C C, Chang CC. Data hiding based on a two-layer turtle shell matrix. Symmetry 2018;10(2):47.
[20] Shamir A. How to share a secret. Communications of the ACM 1979;22(11):612–3.
[21] Fang W P. Non-expansion visual secret sharing in reversible style. International Journal of Computer Science and Network Security 2009;9(2):204–8.
[22] Tsai D S, Chen T, Horng G. On generating meaningful shares in visual secret sharing scheme. The Imaging Science Journal 2008;56(1):49–55.
[23] Shyu S J, Chen MC. Optimum pixel expansions for threshold visual secret sharing schemes. IEEE Transactions on Information Forensics and Security 2011;6(3):960–9.
[24] Harn L, Xia Z, Hsu C, et al. Secret sharing with secure secret reconstruction. Information Sciences 2020;519:1–8.
[25] Chang C C, Kieu T D, Chou Y C. In: Reversible data hiding scheme using two steganographic images. TENCON 2007-2007 IEEE Region 10 Conference. IEEE; 2007. p. 1–4.
[26] Chang C C, Chou Y C, Kieu TD. Information hiding in dual images with reversibility. In: 2009 Third International Conference on Multimedia and Ubiquitous Engineering. IEEE; 2009. p. 145–52.
[27] Qin C, Chang C C, Hsu TJ. Reversible data hiding scheme based on exploiting modification direction with two steganographic images. Multimedia Tools and Applications 2015;74:5861–72.
[28] Lee C F, Huang YL. Reversible data hiding scheme based on dual stegano-images using orientation combinations. Telecommunication Systems 2013;52:2237–47.
[29] Chang C C, Lin P Y, Wang Z H, et al. A sudoku-based secret image sharing scheme with reversibility. J. Commun. 2010;5(1):5–12.
[30] Yan X, Lu Y, Liu L, et al. Reversible image secret sharing. IEEE Transactions on Information Forensics and Security 2020;15:3848–58.
[31] Liu Y, Chang CC. A turtle shell-based visual secret sharing scheme with reversibility and authentication. Multimedia Tools and Applications 2018;77:25295–310.
[32] Gao K, Horng J H, Chang CC. An authenticatable (2, 3) secret sharing scheme using meaningful share images based on hybrid fractal matrix. IEEE Access 2021;9:50112–25.
[33] Gao K, Horng J H, Liu Y, et al. A reversible secret image sharing scheme based on stick insect matrix. IEEE Access 2020;8:130405–16.
[34] Lin J Y, Horng J H, Chang CC. A novel (2, 3)-threshold reversible secret image sharing scheme based on optimized crystal-lattice matrix. Symmetry 2021;13(11):2063.
[35] Liu Y, Chang C C, Huang P C. Security protection using two different image shadows with authentication. Math. Biosci. Eng 2019;16(4):1914–32.
[36] Chang C C, Horng J H, Shih C S, et al. A maze matrix-based secret image sharing scheme with cheater detection. Sensors 2020;20(13):3802.
[37] Chen Y H, Lee J Y, Chiang M H, et al. Verifiable (2,n)Image Secret Sharing Scheme Using Sudoku Matrix. Symmetry 2022;14(7):1445.
[38] NIST. SHA-256, SHA-384, SHA-512. Washington D.C.: NIST, US Department of Commerce, Draft; 2000.
[39] Bas P, Furon T. Image database of BOWS-2, 20; 2017. Accessed: JunAvailable, http://bows2.ec-lille.fr/.
[40] Li X S, Chang C C, He M X, et al. A lightweight authenticable visual secret sharing scheme based on turtle shell structure matrix. Multimedia Tools and Applications 2020;79:453–76.