


## Article

# Lossless Reversible Data Hiding in Encrypted Image for Multiple Data Hiders Based on Pixel Value Order and Secret Sharing

Haoyang Yu <sup>1,2</sup>, Junwei Zhang <sup>3</sup> , Zixiao Xiang <sup>2</sup>, Biao Liu <sup>2</sup> and Huamin Feng <sup>2,\*</sup><sup>1</sup> School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China<sup>2</sup> Beijing Electronic Science and Technology Institute, Beijing 100070, China<sup>3</sup> School of Cyber Engineering, Xidian University, Xi'an 710071, China

\* Correspondence: fenghm@besti.edu.cn

**Abstract:** Reversible data hiding in encrypted images (RDH-EI) is instrumental in image privacy protection and data embedding. However, conventional RDH-EI models, involving image providers, data hiders, and receivers, limit the number of data hiders to one, which restricts its applicability in scenarios requiring multiple data embedders. Therefore, the need for an RDH-EI accommodating multiple data hiders, especially for copyright protection, has become crucial. Addressing this, we introduce the application of Pixel Value Order (PVO) technology to encrypted reversible data hiding, combined with the secret image sharing (SIS) scheme. This creates a novel scheme, PVO, Chaotic System, Secret Sharing-based Reversible Data Hiding in Encrypted Image (PCSRDH-EI), which satisfies the  $(k, n)$  threshold property. An image is partitioned into  $N$  shadow images, and reconstruction is feasible when at least  $k$  shadow images are available. This method enables separate data extraction and image decryption. Our scheme combines stream encryption, based on chaotic systems, with secret sharing, underpinned by the Chinese remainder theorem (CRT), ensuring secure secret sharing. Empirical tests show that PCSRDH-EI can reach a maximum embedding rate of 5.706 bpp, outperforming the state-of-the-art and demonstrating superior encryption effects.

**Keywords:** secret sharing; secure multi-party computing; reversible data hiding in encrypted domain



**Citation:** Yu, H.; Zhang, J.; Xiang, Z.; Liu, B.; Feng, H. Lossless Reversible Data Hiding in Encrypted Image for Multiple Data Hiders Based on Pixel Value Order and Secret Sharing. *Sensors* **2023**, *23*, 4865. <https://doi.org/10.3390/s23104865>

Academic Editors: Shaoen Wu, Periklis Chatzimisios, Jinbo Xiong and Mahmoud Daneshmand

Received: 30 March 2023

Revised: 11 May 2023

Accepted: 15 May 2023

Published: 18 May 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Reversible data hiding (RDH) has evolved into a compelling methodology, facilitating the embedding of confidential data within various forms of media. This spans sectors such as military, medical, national governance, and copyright-protected content [1]. Traditional RDH research, conducted in the realm of plaintext, primarily revolves around three data embedding techniques: histogram shifting [2], difference expansion [3], and lossless compression [4]. The overarching objective of these methodologies is to augment the embedding rate while enhancing the visual quality of the carrier images.

In summary, Reversible Data Hiding (RDH) technology is a robust tool for reversibly embedding confidential data without damaging the original carrier. Its applications are expansive, and it proves particularly beneficial in sensitive sectors where data security is imperative. However, for content-sensitive scenarios, it is necessary to employ encryption techniques to develop an Encrypted Domain Reversible Data Hiding (RDH-EI) method, suitable for information-sensitive situations. These encryption techniques secure image content by transforming the original image into an unintelligible version using an encryption key.

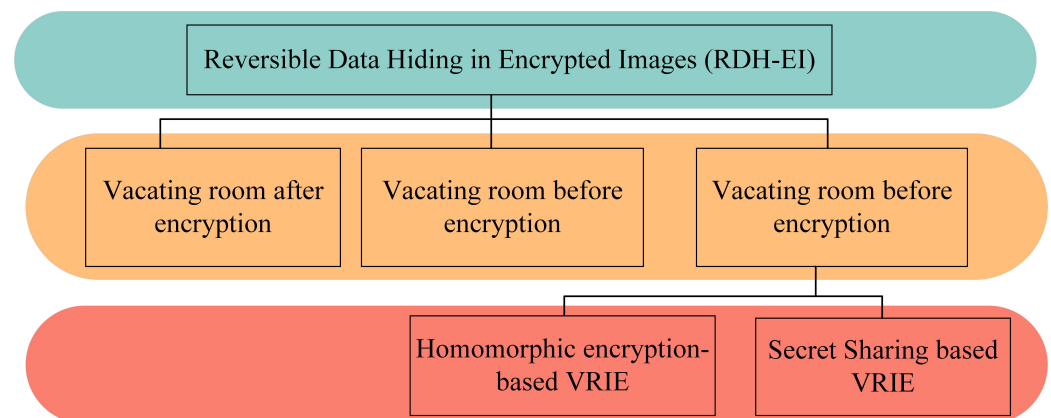
RDH in encrypted images (RDH-EI) is especially advantageous in sensitive fields where data security is paramount. Despite its limitation in multiple data hider scenarios, the inherent reversibility of RDH technology makes it an optimal choice for situations

demanding zero tolerance for image loss. An encrypted domain RDH-EI method, suitable for content-sensitive scenarios, can be developed using encryption techniques.

The traditional RDH-EI model restricts the number of data providers and data hiders to a single entity, limiting its applicability in multiple data hider scenarios. However, the crucial feature of reversibility permits perfect reconstruction of the original carrier during covert data extraction. This attribute renders RDH technology an ideal choice for scenarios necessitating zero tolerance for image loss, such as satellite imagery in the military, government images in judicial determinations, and medical imaging in healthcare.

RDH-EI is a subset of Reversible Data Hiding in Encrypted Domain (RDH-ED) [5], effectively resolves the issue of embedding and extracting confidential data from encrypted images. In this technique, the encrypted image data serves as the carrier. The data can be embedded into the encrypted image without causing any pixel loss in the carrier image during extraction.

As shown in Figure 1, according to the differences in secret data embedding models, we divide RDH-EI into three categories.



**Figure 1.** Literature review of RDH-EI, in which three categories of reversible data hiding with encryption methods can be identified: Vacating Room After Encryption (VRAE), Vacating Room Before Encryption (VRBE), and Vacating Room In Encryption (VRIE).

- (a) Vacating Room After Encryption (VRAE). In the framework of VRAE, Puech et al. [6] first proposed the method of vacating room after AES encryption, in which additional data can only be extracted according to the local standard deviation of the image before decryption of the token image; subsequently, Zhang et al. [5] vacated space by flipping the three lowest significant bits after encryption based on stream ciphers, and secret data can only be extracted using the fluctuation function defined by the local characteristics of the image after decryption of the token image; in order to achieve the separability of the algorithm, Zhang et al. [7] first proposed a separable scheme by losslessly compressing the ciphertext image to generate redundancy, but its embedding capacity is low.
- (b) Vacating Room Before Encryption (VRBE). The VRBE mode is used to generate redundancy by using image correlation or other pre-processing operations before image encryption, and its implementation methods are mainly based on lossless compression [8], pixel prediction [9–11], and frequency domain transformation [12].
- (c) Vacating Room In Encryption (VRIE). (1) Homomorphic encryption-based VRIE. Zhang et al. [13] proposed the RDH-EI scheme based on VRIE by quantizing the encrypted domain after LWE encryption and using the redundancy generated by ciphertext expansion to embed secret data; Huang et al. [14] used prediction error to free up redundant space during stream cipher encryption to enhance the embedding capacity. In recent years, Chen et al. [15] proposed an algorithm based on Paillier public key encryption, which uses the homomorphic property to embed information in the ciphertext domain, and the decrypted plaintext still maintains the relevant properties

of the embedded information, but there is partial pixel overflow after embedding the information; Wu et al. [16] improved the method of the literature [15] by solving the overflow problem, and the security of the homomorphic encryption algorithm relies on long keys, which increases the computational overhead and brings about severe data scaling. (2) Secret Sharing-based VRIE. Secret sharing techniques are widely applied in edge computing [17], data sharing [18], and outsourcing computing [19]. Thien and Lin [20] first proposed the concept of secret image sharing in 2002, which is based on the idea of secret sharing (SS) proposed by Shamir [21] in 1979. This technique can effectively solve the data extension problem of the RDH-EI algorithm based on homomorphic encryption, and Wu et al. [22] first proposed the RDH-ED algorithm based on secret sharing encryption, which splits the carrier image into multiple shadow images of the same size as the carrier image with secret sharing encryption, and then embeds the secret data into the shadow images by means of difference expansion and prediction error histogram translation. The problems of high encryption overhead and serious data expansion are effectively solved. As an important multi-party secure cryptosystem, this method uses a threshold function to address important data shares multiplied by different shares stored at different users. Chen et al. [23] further reduced the time complexity by embedding secret data into a pair of pixels using the additive homomorphism property of multiple secret sharing and combining the difference expansion. Ke et al. [24] proposed a separable RDH-ED based on the Chinese residue theorem separable RDH-ED, which achieves separability by combining two embedding methods. In 2022, Xiong et al. [25] proposed an RDH-EI scheme, which uses Asmuth-Bloom's secret sharing scheme based on the CRT to divide the pixels into several secret share subgraphs, but the embedding efficiency (0.5 bpp) of this method still has room for optimization.

To solve the problems of low embedding efficiency, low embedding capacity, complex auxiliary information for embedding, and the limited number of embedding parties in RDH-EI, we propose a lossless RDH-EI method for multiple data hiders based on PVO and secret sharing. Compared with the existing RDH-EI based on secret sharing, we put forward a new method for the first time, which combines an Arnold cat map and a logistic equation [26] with a CRT-based secret sharing scheme and adds PVO technology, which improves the embedding rate and encryption efficiency while improving security. In addition, unlike other RDH-EI that requires auxiliary information and guiding images, the proposed method in this paper does not need to send guiding images. However, the data hider can automatically convert the shadow image into the guide image by using the data embedding key, which can significantly improve the embedding efficiency.

**Main contributions of this paper.** This paper describes the main contributions of a lossless RDH-EI (Reversible Data Hiding in Encrypted Images) method for multiple data hiders, based on PVO (Pixel Value Order) and secret sharing. The following are the key contributions of the proposed method:

**Novel PCSRDH-EI Method:** This paper introduces an innovative lossless RDH-EI method, which allows for multiple data hiders to embed data in encrypted images without any loss. This method is based on the PVO chaotic system and secret sharing, achieving a maximum embedding rate and surpassing existing methods, as well as significantly improving encryption efficiency and effect.

**Enhanced Security with Combined Techniques:** Combining stream encryption and secret sharing technology, this paper ensures a secure environment for data hiders to embed their data without fear of leaks. Additionally, the proposed scheme guarantees lossless data embedding and extraction, preserving the original image digital assets and enabling full recovery.

**No Extra Guide Parameters Required:** The proposed scheme does not require any extra guide parameters. By embedding/extracting the key, the data hider can transform the shadow image into an guide image that does not expose the original content information, thereby improving the usability of the scheme.

Organization of this paper. We divide our paper into five sections. Section 1 describes the research area and the existing schemes' shortcomings and details the paper's main contributions. Section 2 presents the specific details of the proposed scheme. In Section 3, we give a precise analysis and demonstration of our scheme. Section 4 presents the scheme's effectiveness and a detailed comparison with the state-of-the-art encryption techniques. In Section 5, we give a summary and outlook of our work.

## 2. PCSRDH-EI Method

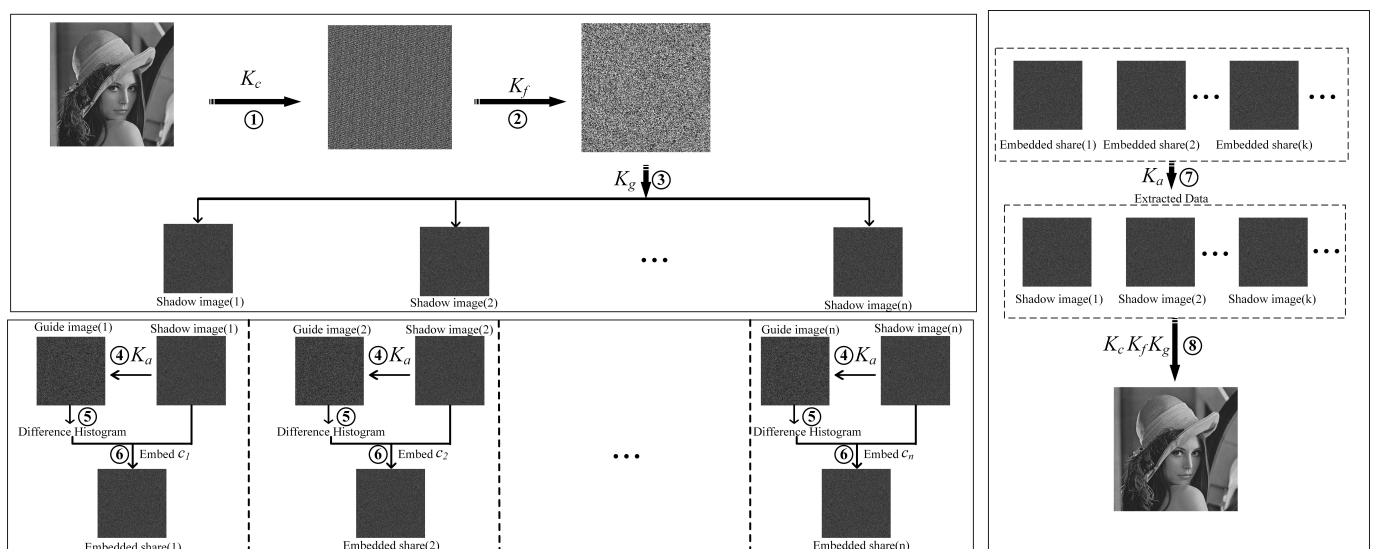
In this section, we provide a detailed description of our proposed method. As depicted in Figure 2, our model involves three key participants: an image owner, a minimum of three data hidiers, and at least one receiver. The primary stages of our method encompass image encryption, shadow image creation, data embedding, data extraction, and carrier image recovery.

During the Image Encryption phase, the image owner divides the image into blocks and implements coarse-grained disruption encryption between these blocks. They further apply fine-grained encryption algorithms and the PVO algorithm within the blocks to enhance the security of the encrypted image.

In the Shadow Image Generation phase, the image owner partitions the encrypted image into numerous shadow images and distributes them to a predetermined number of data hidiers. This measure ensures that no individual data hider can access the entire image, thereby augmenting the security of the scheme.

During the Data Embedding stage, each data hider employs the embedding secret key to produce the embedding guide map and embeds the confidential data within their respective shadow images. This procedure ensures the dispersion of the embedded data across multiple shadow images, which further bolsters the scheme's security.

In the Embedded Data Extraction and Image Recovery phase, the receiver employs the secret extraction key to retrieve the embedded data from the shadow images. Upon amassing a minimum of 'k' secret shares, the receiver can flawlessly recover the carrier image by integrating these shares with the encryption key.



**Figure 2.** The framework of our scheme. The scenario of our method has one image owner, at least three data hidiers, and at least one receiver. The main steps are ① coarse-grained encryption, ② fine-grained encryption, ③ shadow image generation, ④ guide image generation, ⑤ difference generation, ⑥ data embedding, ⑦ embedded data extraction, and ⑧ carrier image recovery.

### 2.1. Image Encryption

The image owner splits a carrier image of size  $H \times W$  into non-overlapping sub-blocks of size  $2 \times 2$  and the number  $\lceil H/2 \times W/2 \rceil$ , where  $H$  is the height of the image and  $W$  is the width of the image.  $i_{h,w}$  is the pixel value at location  $(h, w)$ , where  $I_{h,w} \in [0, 255], 1 \leq h \leq H, 1 \leq w \leq W$ . Each block has a number in order, numbered in the range  $1, 2, \dots, \lceil H/2 \times W/2 \rceil$ . This section divides encryption into two main steps: coarse-grained and fine-grained. The encryption unit of coarse-grained encryption is a  $2 \times 2$  non-overlapping sub-block, the encryption algorithm is Arnold's cat map, and the target of fine-grained encryption is the four pixels inside the non-overlapping sub-block, which is implemented as Algorithm 1.

---

#### Algorithm 1 Image encryption.

---

Input: The original image  $I(H \times W)$ , coarse-grained encryption key  $K_c \leftarrow (p, q)$ , fine-grained encryption key  $K_f \leftarrow (\eta, x_0, \forall)$ .

Output: The encrypted image  $I'$

1. Initialize  $h \leftarrow 0, w \leftarrow 0, s \leftarrow 0, t \leftarrow 0$
  2. Use  $K_c$  to encrypt the original image  $I$  by Equation (2) and obtain  $I^*$
  3. Use  $K_f$  to generate the fine-grained encryption stream  $\sigma$  by Equation (3)
  4. while  $h \leq H$  do
  5.     While  $w \leq W$  do
  6.          $I'_{h,w} = I^*_{h,w} + (\sigma_{s,t} \bmod 256) + 256$
  7.          $w \leftarrow w + 1$
  8.          $s = \lfloor h/2 \rfloor, t = \lfloor w/2 \rfloor$
  9.     end while
  10.  $h \leftarrow h + 1$
  11.  $s = \lfloor h/2 \rfloor, t = \lfloor w/2 \rfloor$
  12. end while
  13. Return  $I'$
- 

Secret key definition. The implementation of encryption or decryption methods, as described in the work by [27], requires both the sender and the recipient of the image to possess a secret key  $SK$ . This unique key consists of two parts: a coarse-grained encryption key denoted as  $K_c \leftarrow (p, q)$  and a fine-grained encryption key denoted as  $K_f \leftarrow (\eta, x_0, \forall)$ . This secret key is generated from 65 hexadecimal digits (260 bits), ranging from  $P_1$  to  $P_{52}$ , and used to compute the initial conditions and control parameters of the chaotic mapping using the expression indicated below:

$$\begin{aligned}
 K_{c1} &= (P_1, P_2, \dots, P_{13})_{10} \\
 K_{c2} &= (P_{14}, P_{15}, \dots, P_{26})_{10} \\
 K_{f1} &= \frac{(P_{27}, P_{38}, \dots, P_{39})_{10}}{2^{52} + 1} \times 10^{-8} \\
 K_{f2} &= \frac{(P_{40}, P_{41}, \dots, P_{52})_{10}}{2^{52} + 1} \\
 K_{f3} &= (P_{53}, P_{54}, \dots, P_{65})_{10}
 \end{aligned} \tag{1}$$

In which  $K_{c1}, K_{c2}, K_{f3}$  belong to  $(0, 2^{52})$ , and  $K_{f1}, K_{f2}$  belong to  $(0, 1)$ .

Coarse-grained encryption. Before the image owner splits the image into shadow images, the image needs to be scrambled to ensure its security. Since the fine-grained encryption only needs to satisfy the randomness to generate the scrambling table, and the test sets used in this paper are square matrix, i.e.,  $H = W$ , this paper uses a two-dimensional Arnold's cat map to generate the scrambling table. For the case of  $H \neq W$ , other similar random number sorting algorithms can be used to generate the scrambling table, which

this paper will not discuss. The image owner uses the coarse-grained encryption key  $K_c \leftarrow (p, q)$  as the secret seed key and performs

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} \leftarrow \begin{bmatrix} 1 & p \\ q & pq + 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N \quad (2)$$

where  $(x_n, y_n)$  is the position coordinates of the image block in the original image,  $(x_{n+1}, y_{n+1})$  is the transformed position, mod is the modulus operation, and  $N$  is the size of the image; the image must be square. Otherwise, it does not have the condition of the Arnold transform, for rectangular images can also be converted into a square way for topological reasoning. Each coordinate has a unique new coordinate corresponding to it after permutation, which means that each image block has a unique position to mark the position after permutation, and the image encryption procedure is formulated by

$1 \leq t \leq \lceil H/2 \times W/2 \rceil$ , which is the  $t$ -th encrypted image block.  $I$  is the carrier image and  $I^*$  is the coarse-grained encrypted image.

**Fine-grained encryption.** In the proposed scheme, there is no restriction on the specific scrambling method, and we take the enlarged 1D logical mapping as an example. 1D logical mapping only needs to save the seed key, the enlargement factor, and the length of the generated 1D stream, which significantly saves the storage space and the efficiency of crucial transmission. The image owner uses the fine-grained encryption key  $K_f \leftarrow (\eta, x_0, \forall)$  to generate a random stream by

$$x_{r+1} = \eta x_r (1 - x_r) \quad (3)$$

Furthermore, the generated sequence  $x = (x_1, x_2, \dots, x_{\lceil H/2 \times W/2 \rceil})$  is enlarged by a factor of  $\forall$  and rounded down, to obtain the size  $\lceil H/2 \times W/2 \rceil$ , and perform the following encryption operation:

$$I'_{h,w} = I^*_{h,w} + (\sigma_{s,t} \bmod 256) + 256 \quad (4)$$

where  $I'_{h,w}$  is the result of the image owner's encryption using the key  $K_f$ , where  $i$  and  $j$  denote rows and columns, respectively, where  $h = 1, 2, \dots, H, w = 1, 2, \dots, W$ .  $I_{h,w}$  and  $I'_{h,w}$  are the ciphertext and plaintext at  $(h, w)$ , respectively.

Moreover,  $T_{f_{s,t}}$  is the key corresponding to the  $(s, t)$ th image block in  $T_f$ , where  $s = \lceil h/2 \rceil, t = \lceil w/2 \rceil$ , at the block level, the image owner uses fine-grained encryption by

$$I' \left( I^{(1)}, I^{(2)}, \dots, I^{(\lceil H/2 \times W/2 \rceil)} \right) = \text{Enc}_{\text{fine}} \left( I^*, K_f \right) \quad (5)$$

where  $I^*$  is the block-level data after coarse-grained encryption,  $I'$  is the block-level data after coarse-grained encryption, and  $1 \leq t \leq \lceil H/2 \times W/2 \rceil$  is the  $t$ -th encrypted image block.

## 2.2. Shadow Image Generation

Since the encryption algorithm in this paper employs a permutation algorithm between blocks, there is no scrambling algorithm for pixels between blocks. The RDH algorithm based on histogram shifting (HS) can meet the user's needs. When the data hider receives the encrypted share, the secure multi-party embedding is performed under the guidance of the dealer. The data distributor can collect encrypted secret data and initiate the embedding phase in a practical application with multiple users and messages. By involving data distributors, communication between multiple data hidens can be avoided. Data hiding based on histogram embedding can be performed on encrypted domains, achieving the high visual quality of the tagged images. Watermark embedding is divided into three main steps. Algorithm 2 demonstrates the detailed algorithmic process in the form of pseudo-code.

**Algorithm 2** Shadow image generation.

Input: The encrypted image  $I'$ , share generation key  $K_g = (\vartheta, Mb)$ , number of shadow images  $n$

Output: The shadow images  $\tau$ , position correspondence table  $\omega$

1. Initialize  $h \leftarrow 0, w \leftarrow 0$
2. while  $h \leq H/2$  do
3.     While  $w \leq W/2$  do
4.          $\delta_1 \leftarrow x[h, w], \delta_2 \leftarrow x[h + 1, w], \delta_3 \leftarrow x[h, w + 1], \delta_4 \leftarrow x[h + 1, w + 1]$
5.         Record the original position  $[\delta_1, \delta_2, \delta_3, \delta_4]$  in  $\omega[h, w]$
6.         Sort  $x[h, w]$  by Equation (6) to get  $\{\delta'_1, \delta'_2, \delta'_3, \delta'_4\}$  and  $x'[h, w]$
7.         Computes  $d_{\max} = \delta'_4 - \delta'_3, d_{\min} = \delta'_1 - \delta'_2$
8.          $w \leftarrow w + 1, t = \lceil w/2 \rceil$
9.     end while
10.  $h \leftarrow h + 1, s = \lceil h/2 \rceil$
11. end while
12. while  $h \leq H$  do
13.     While  $w \leq W$  do
14.         While  $i \leq n$  do
15.              $\tau_i(h, w) = \vartheta(x(h, w)) \bmod Mb^{(i)}$
16.              $i \leftarrow i + 1$
17.         end while
18.          $w \leftarrow w + 1$
19.     end while
20.      $h \leftarrow h + 1$
21. end while
22. Return  $\tau$

## 2.2.1. Pixel Value Order

Intra-block ascending mapping. For a block of  $2 \times 2$  pixels  $\delta = (\delta_1, \delta_2, \delta_3, \delta_4)$ , the image owner computes the new block of pixels utilizing an ascending ordering algorithm:

$$(\delta'_1, \delta'_2, \delta'_3, \delta'_4) \leftarrow \text{PVO}(\delta_1, \delta_2, \delta_3, \delta_4) \quad (6)$$

where the set  $(\delta'_1, \delta'_2, \delta'_3, \delta'_4)$  to the set  $(\delta_1, \delta_2, \delta_3, \delta_4)$  is a full projection and satisfies

$$\delta'_1 \leq \delta'_2 \leq \delta'_3 \leq \delta'_4 \quad (7)$$

Therefore, the two largest values  $\delta'_4$  and  $\delta'_3$  can predict the value of the other with the value of one, and  $\delta_1$  and  $\delta'_2$  can also predict the value of the other with the value of one, and their prediction differences are

$$\begin{cases} d_{\max} = \delta'_4 - \delta'_3 \\ d_{\min} = \delta'_1 - \delta'_2 \end{cases} \quad (8)$$

$d_{\max}$  and  $d_{\min}$  are the maximum and minimum prediction errors in the image chunks, respectively. For the result of  $I'$  after the fine-grained encryption of the image, and after repeating the same PosCon-operation, we can obtain  $d_{\max}$  and  $d_{\min}$  for all chunks of the whole image, and calculate the histograms of the maximum prediction error and minimum prediction error accordingly.

In addition, the image owner uses 00,01,10,11 to denote the original pixel positions (1,2,3,4) matched by the scrambled pixels, respectively, and saves the original position correspondence table  $\omega$  corresponding to the scrambled image pixels.

### 2.2.2. Key Generation

Shadow recovery key generation. For the secret sharing scheme with a  $(k, n)$  threshold, we choose a set of modules  $Mb = \{Mb^{(1)}, Mb^{(2)}, \dots, Mb^{(n)}\}$ , where the elements in  $Mb$  satisfy  $\{128 < Mb^{(1)} < Mb^{(2)} < \dots < Mb^{(n)} \leq 251 < Mp\}$ , satisfying the following conditions:

$$\gcd(Mb^{(i)}, Mb^{(j)}) = 1, i \neq j \quad (9)$$

$$\gcd(Mb^{(i)}, Mp) = 1, i = 1, 2, \dots, n \quad (10)$$

$$Mp = \prod_{i=1}^k Mb^{(i)} \quad (11)$$

In addition, we set the secret key  $\vartheta$  to satisfy the following relation:

$$\vartheta \in \mathbb{Z}_{(\prod_{i=1}^k Mb^{(i)})(2^8-1)} \quad (12)$$

$$\gcd(Mb^{(i)}, \vartheta) = 1, i = 1, 2, \dots, n \quad (13)$$

Then, the shadow image is generated by  $K_g \leftarrow (\vartheta, Mb)$ .

Embedding key generation. The randomly chosen integer  $\vartheta$  is the secret key, and using the equivalence property of congruence since  $\vartheta$  is the amplification parameter, the following two calculations have equivalence conditions under  $\text{mod } b$ :

$$x\vartheta \text{ mod } b \Leftrightarrow x(\vartheta \text{ mod } b) \text{ mod } b \quad (14)$$

Therefore, firstly,  $\vartheta$  is converted into the residue form of  $a_i = \vartheta \text{ mod } Mb_i$ , and the secret parameter  $\vartheta$  can be guaranteed not to be revealed without exposing a certain number of  $Mb_i$ . The generated key is  $K_a \leftarrow (Mb, a, \Delta, \chi)$ ,  $\Delta$  is the threshold parameter which determines the embedding payload, pixels exceeding the threshold  $\Delta$  are panned, and those within the threshold  $\Delta$  are embedded. The size of  $\Delta$  will affect the visual quality of the labeled image. For example, when increasing, the embedding payload will increase, but the visual quality will decrease.

### 2.2.3. Shadow Image Generation

Unlike the scheme proposed by Xiong et al. [25] which only adds large random numbers (only addition), the secret sharing scheme used in this paper uses the method of adding perturbations first and then scaling up (a combination of addition and multiplication) to improve security. For a single pixel value,  $x(h, w)$ ,  $h = 1, 2, \dots, H$ ,  $w = 1, 2, \dots, W$ , an image of size  $H \times W$ , and the encryption key  $K_g = (\vartheta, Mb)$ , let us calculate

$$\begin{aligned} \vartheta(x(h, w)) &= \tau_1(h, w) \text{ mod } Mb^{(1)} \\ \vartheta(x(h, w)) &= \tau_2(h, w) \text{ mod } Mb^{(2)} \\ &\dots \\ \vartheta(x(h, w)) &= \tau_n(h, w) \text{ mod } Mb^{(n)} \end{aligned} \quad (15)$$

where  $\tau_1(h, w)$ ,  $\tau_2(h, w)$ ,  $\dots$ ,  $\tau_n(h, w)$  represent the secret shares split by the pixel values at position  $(h, w)$  in the image, and the  $n$  secret shadow images after splitting  $(\tau^{(1)}, \tau^{(2)}, \dots, \tau^{(n)})$  are sent to the corresponding data embedders  $DH^{(i)}$  for  $i = 1, 2, \dots, n$ .

### 2.3. Data Embedding

Since the same noise is added to each image sub-block during fine-grained encryption, the differences between pixels do not change, which means that additional information



can be embedded in the DH embedding data using homomorphic embedding. A module homomorphic embedding approach is adopted in this paper, in which any shadow image after image segmentation is used as a carrier. The additive homomorphic property of secret sharing is taken advantage of so that the data hider can embed all extra data into the shadow image independently. This algorithm can extract the extra data rapidly from the newly generated shadow image. It is convenient for multiple users to tag, manage, and retrieve secret texts independently too improve the embedding efficiency, i.e., to embed as much data as possible with fewer modifications. Importantly, each data hider owns an independent shadow image and can produce the embedding instruction and data embedding images independently. Consequently, the embedding capacity scales linearly with the number of data hidens, such that the overall capacity is obtained by multiplying the capacity of a single image by the number of hidens involved. Algorithm 3 demonstrates the detailed algorithmic process in the form of pseudo-code.

---

**Algorithm 3** Distributed data embedding.
 

---

Input: The secret share of the  $i$ -th embedding party  $\tau_i$ , data embedding key  $K_a \leftarrow (Mb, a, \Delta, \chi)$ , data to be embedded  $c_i \leftarrow \{c_{i_0}, c_{i_1}\}$ .

Output: The  $i$ -th embedded shadow image  $\mathbb{S}_i$

1. Initialize  $h \leftarrow 0, w \leftarrow 0$
  - 2, while  $h \leq H/2$  do
  3.     While  $w \leq W/2$  do
  4.          $\delta_1 \leftarrow \tau_i'[2h, 2w], \delta_2 \leftarrow \tau_i'[2h + 1, 2w]$
  5.          $\delta_3 \leftarrow \tau_i'[2h, 2w + 1], \delta_4 \leftarrow \tau_i'[2h + 1, 2w + 1]$
  6.          $\delta_j = \delta_j a^{(i)} \bmod Mb^{(i)}, j = 1, 2, 3, 4$
  7.         Compute  $d_{\max}[h, w], d_{\min}[h, w]$  by Equations (17) and (18)
  8.         Embed  $c_{i_1}[h, w]$  into  $\delta_4'$  by Equation (19) and  $c_{i_0}[h, w]$  into  $\delta_1'$  by Equation (20)
  9.          $\mathbb{S}_i[2h, 2w] \leftarrow \delta_1', \mathbb{S}_i[2h + 1, 2w] \leftarrow \delta_2$
  10.         $\mathbb{S}_i[2h, 2w + 1] \leftarrow \delta_3, \mathbb{S}_i[2h + 1, 2w + 1] \leftarrow \delta_4'$
  11.         $w \leftarrow w + 1$
  12.     end while
  13.  $h \leftarrow h + 1$
  14. end while
  15. Return  $\mathbb{S}_i$
- 

Upon receiving the image to be embedded  $\tau^{(i)}$ ,  $DH^{(i)}$  divides it into non-overlapping  $2 \times 2$  sub-blocks. We denote the block at position  $(h, w)$  as  $\{\delta_1[h, w], \delta_2[h, w], \delta_3[h, w], \delta_4[h, w]\}$  (for ease of writing, we abbreviate it as  $\{\delta_1, \delta_2, \delta_3, \delta_4\}$ ) and embed  $c_i[h, w] \leftarrow \{c_{i_0}, c_{i_1}\}$ . Specifically,  $c_{i_0}[h, w]$  is embedded into  $\delta_1$  while  $c_{i_1}[h, w]$  is embedded into  $\delta_4$ . To obtain a difference histogram that encodes the image difference information, we employ the secret extraction key  $K_a$  as follows:

$$\delta_j = \delta_j a^{(i)} \bmod Mb^{(i)}, j = 1, 2, 3, 4 \quad (16)$$

Prediction error recovery. For a single  $2 \times 2$  pixel block, the four elements within the block perform sum and mode operations using the same cryptographic secret key. This means that the difference between any two pixels within the block should satisfy the ascending ordering, and thus the true prediction error is recovered in the following manner:

$$d_{\max}[h, w] = \begin{cases} \delta_4 - \delta_3, & \text{if } \delta_4 - \delta_3 \geq 0 \\ \delta_4 - \delta_3 + Mb^{(i)}, & \text{if } \delta_4 - \delta_3 < 0 \end{cases} \quad (17)$$

$$d_{\min}[h, w] = \begin{cases} \delta_1 - \delta_2, & \text{if } \delta_1 - \delta_2 \leq 0 \\ \delta_1 - \delta_2 - Mb^{(i)}, & \text{if } \delta_1 - \delta_2 > 0 \end{cases} \quad (18)$$

The difference histograms are mainly concentrated around the value of 0, and the prediction error statistics of ciphertext images are the same as those of plaintext images,

so the prediction error statistics of ciphertext images retain the statistical characteristics of plaintext. The prediction error histogram of a raw image is generally Laplacian, so we can use the histogram shifting technique to hide the data in the ciphertext image.

**Histogram embedding.** We perform the translation embedding in two prediction error histograms of the ciphertext image. It is worth noting that, in addition to the 0 value, several histograms of its nearby values are also high, so we utilize multiple histogram embeddings to increase the embedding capacity. In addition, due to the unique nature of module homomorphic secret sharing in this paper, the selected module bases are less than 252. Therefore, the data overflow before secret sharing will not affect the embedding flags and results after secret sharing in the all-position embeddable histogram translation embedding method. The image owner can use an additional auxiliary information table to cooperate with data embedding and extraction. Therefore, the histogram panning embedding method based on secret sharing can effectively solve the problem of possible overflow and underflow during the data panning process.

We set the storage threshold at  $\Delta$ , pixels exceeding the threshold  $\Delta$  are shifted, those within the threshold  $\Delta$  are embedded, and according to the prediction error  $d_{\max}$ , the data  $c_1$  are embedded by shifting the maximum value  $\delta_4$ , the maximum value of possible embedding is  $\chi$ , and the data embedding algorithm  $\text{embed}(\tau)$  is executed, so the data embedding is performed by

$$\delta'_4 \leftarrow \begin{cases} \delta_4 + c_1[h, w]\Delta & \text{if } 0 \leq d_{\max}[h, w] \leq \Delta \\ \delta_4 + \chi\Delta & \text{else} \end{cases} \quad (19)$$

According to the equation prediction error  $d_{\min}$ , embedding the data  $c_0$  by shifting the minimum  $\delta_1$ , the data embedding is performed by

$$\delta'_1 \leftarrow \begin{cases} \delta_1 - c_0[h, w]\Delta & \text{if } -\Delta \leq d_{\min}^{(i)} \leq 0 \\ \delta_1 - \chi\Delta & \text{else} \end{cases} \quad (20)$$

The embedding algorithm is executed on all image blocks of the image to obtain the labeled image  $\mathbb{S}^{(i)} = \text{embed}(\tau^{(i)})$ , and send it to the corresponding receiver<sup>(i)</sup>,  $i = 1, 2, \dots, n$ .

#### 2.4. Data Extraction

When the secret data are embedded into the encrypted shadow share, the receiver can collect the tagged encrypted shadow images from different data hidere. Upon receiving the extraction secret key  $K_a^{(i)}$  that corresponds to the  $Share^{(i)}$  of the  $i$ -th shadow image, the  $i$ -th receiver can recover the embedded data individually. At position  $(h, w)$  of  $\mathbb{S}^{(i)}$ , the block is denoted as  $\{\delta_1[h, w], \delta_2[h, w], \delta_3[h, w], \delta_4[h, w]\}$  (for ease of writing, we abbreviate it as  $\{\delta_1, \delta_2, \delta_3, \delta_4\}$ ). We extract  $w[h, w] \leftarrow \{c'_0[h, w], c'_1[h, w]\}$ , wherein  $c'_0[h, w]$  is extracted into  $\delta_1$ , and  $c'_1[h, w]$  is extracted into  $\delta_4$ . Moreover, we can obtain  $d_{\max}[h, w]$  and  $d_{\min}[h, w]$  using the same method as in Equations (16)–(18).

Iterating through all the storage blocks of the block, we obtain the computational prediction error  $d_{\max}[h, w]$  and  $d_{\min}[h, w]$ . From the nature of the  $2 \times 2$  pixel-based block, it is known that there are two positions within the block where the data can be embedded, and the possible embedded data are  $\tilde{w}_0$  by Equation (21) and  $\tilde{w}_1$  by Equation (22).

$$\tilde{w}_0[h, w] = \begin{cases} \left\lfloor \frac{d_{\max}^{(i)}(h, w)}{\Delta} \right\rfloor & \text{if } \Delta < d_{\max}^{(i)}(h, w) \leq \chi\Delta \text{ and } d_{\min}^{(i)}(h, w)/\Delta \neq 0 \\ \left( \frac{d_{\max}^{(i)}(h, w)}{\Delta} \right) - 1 & \text{if } \Delta < d_{\max}^{(i)}(h, w) \leq \chi\Delta \text{ and } d_{\min}^{(i)}(h, w)/\Delta = 0 \\ 0 & \text{if } 0 < d_{\max}^{(i)}(h, w) \leq \Delta \\ NULL & \text{if } \chi\Delta < d_{\max}^{(i)}(h, w) \end{cases} \quad (21)$$

$$\tilde{w}_1[h, w] = \begin{cases} \left\lfloor \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \right\rfloor & \text{if } \Delta < \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \leq \chi\Delta \text{ and } \frac{d_{\min}^{(i)}(h, w)}{\Delta} \neq 0 \\ \left\lfloor \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \right\rfloor - 1 & \text{if } \Delta < \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \leq \chi\Delta \text{ and } \frac{d_{\min}^{(i)}(h, w)}{\Delta} = 0 \\ 0 & \text{if } 0 < \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \leq \Delta \\ NULL & \text{if } \chi\Delta < \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \end{cases} \quad (22)$$

Iterate over all blocks and recover all embedded data. Algorithm 4 demonstrates the detailed algorithmic process in the form of pseudo-code.

---

**Algorithm 4** Data extraction.

Input: The  $i$ -th embedded shadow image  $S_i$ , data extraction key  $K_a \leftarrow (Mb, a, \Delta, \chi)$ .

Output: The extracted data  $c'$

1. Initialize  $h \leftarrow 0, w \leftarrow 0$
  - 2, while  $h \leq H/2$  do
  3.     While  $w \leq W/2$  do
  4.          $\delta_1 \leftarrow S'_i[2h, 2w], \delta_2 \leftarrow S'_i[2h + 1, 2w]$
  5.          $\delta_3 \leftarrow S'_i[2h, 2w + 1], \delta_4 \leftarrow S'_i[2h + 1, 2w + 1]$
  6.          $\delta_j = \delta_j a^{(i)} \bmod Mb^{(i)}, j = 1, 2, 3, 4$
  7.         Compute  $d_{\max}[h, w], d_{\min}[h, w]$  by Equations (17) and (18)
  8.         Extract  $c'_1[h, w]$  from  $\delta'_4$  by Equation (21)
  9.         Extract  $c'_0[h, w]$  from  $\delta'_1$  by Equation (22)
  10.         $w \leftarrow w + 1$
  11.     end while
  12.  $h \leftarrow h + 1$
  13. end while
  14. Return  $c'$
- 

### 2.5. Image Recovery

Algorithm 5 demonstrates the detailed algorithmic process of image recovery in the form of pseudo-code. After receiving the secret shares of  $k$ , the image owner first uses the inverse theorem based on the residual theorem of the Chinese Remainder to obtain the fine-grained encrypted digital image data. For the share of the  $h$ th row and  $w$ th column of the image  $S_1(h, w), S_2(h, w), \dots, S_k(h, w), S_4^{(i)}(h, w)$  and  $S_1^{(i)}(h, w)$  can be recovered by

$$\hat{S}_4^{(i)}(h, w) = \begin{cases} S_4^{(i)}(h, w) - \left\lfloor \left| \frac{d_{\max}^{(i)}(h, w)}{\Delta} \right| \right\rfloor \Delta \bmod Mb_i & \text{if } \tilde{w}_0 = \left\lfloor \left| \frac{d_{\max}^{(i)}(h, w)}{\Delta} \right| \right\rfloor \\ S_4^{(i)}(h, w) - d_{\max}^{(i)}(h, w) + \Delta \bmod Mb_i & \text{if } \tilde{w}_0 = \left( \left\lfloor \left| \frac{d_{\max}^{(i)}(h, w)}{\Delta} \right| \right\rfloor - 1 \right) \\ S_4^{(i)}(h, w) \bmod Mb_i & \text{if } \tilde{w}_0 = 0 \\ S_4^{(i)}(h, w) - \Delta\chi \bmod Mb_i & \text{if } \tilde{w}_0 = NULL \end{cases} \quad (23)$$

$$\hat{S}_1^{(i)}(h, w) = \begin{cases} S_1^{(i)}(h, w) + \left\lfloor \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \right\rfloor \Delta \bmod Mb_i & \text{if } \tilde{w}_1 = \left\lfloor \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \right\rfloor \\ S_1^{(i)}(h, w) - d_{\min}^{(i)}(h, w) - \Delta & \text{if } \tilde{w}_1 = \left( \left\lfloor \left| \frac{d_{\min}^{(i)}(h, w)}{\Delta} \right| \right\rfloor - 1 \right) \\ S_1^{(i)}(h, w) \bmod Mb_i & \text{if } \tilde{w}_1 = 0 \\ S_1^{(i)}(h, w) + \Delta\chi \bmod Mb_i & \text{if } \tilde{w}_1 = NULL \end{cases} \quad (24)$$

The secret can be recovered by executing the following secret recovery algorithm:

$$\begin{aligned}\tilde{I}(h, w) &= \sum_{i=1}^k M_i \mathbb{S}_i(h, w) M_i^{-1} \bmod Mp \\ M_i &= Mp / Mb_i \\ M_i M_i^{-1} &= 1 \bmod Mp\end{aligned}\quad (25)$$

---

**Algorithm 5** Image recovery.

---

Input: The  $i$ -th embedded shadow image  $\mathbb{S}_i$ , data extraction key  $K_a \leftarrow (Mb, a, \Delta, \chi)$ , decryption key  $K_c$  and  $K_f$ , position correspondence table  $\omega$ .

Output: Recovered image

1. Initialize  $h \leftarrow 0, w \leftarrow 0$
  2. while  $h \leq H/2$  do
  3.     While  $w \leq W/2$  do
  4.          $\delta_1 \leftarrow \mathbb{S}'_i[2h, 2w], \delta_2 \leftarrow \mathbb{S}'_i[2h + 1, 2w]$
  5.          $\delta_3 \leftarrow \mathbb{S}'_i[2h, 2w + 1], \delta_4 \leftarrow \mathbb{S}'_i[2h + 1, 2w + 1]$
  6.          $\delta_j = \delta_j a^{(i)} \bmod Mb^{(i)}, j = 1, 2, 3, 4$
  7.         Compute  $d_{\max}[h, w], d_{\min}[h, w]$  by Equations (17) and (18)
  8.         Recover  $\hat{\mathbb{S}}_4^{(i)}[h, w]$  by Equation (23) and  $\hat{\mathbb{S}}_1^{(i)}[h, w]$  by Equation (24)
  9.         Recover  $\hat{\mathbb{S}}[h, w]$  by Equation (25)
  10.         $w \leftarrow w + 1$
  11.     end while
  12.  $h \leftarrow h + 1$
  13. end while
  14. Recovery  $I$  by Equation (26)–(28)
  15. Return  $I$
- 

After recovering the secret sub share, a fine-grained decryption algorithm is executed, and it is decrypted by

$$I^*(h, w) = \tilde{I}(h, w) - 256 - (T_{f,s,t} \bmod 256) \quad (26)$$

where  $\tilde{I}(h, w)$  is the decryption result obtained by the image owner using the secret recovery algorithm,  $I_{i,j}$  and  $I'_{i,j}$  are the  $(h, w)$  ciphertext and plaintext, respectively.  $T_{f,s,t}$  is the key corresponding to the  $(s, t)$  image block in  $T_f$ , where  $s = \lceil h/2 \rceil, t = \lceil w/2 \rceil$ . After finishing the coarse-grained decryption, the IO executes the coarse-grained decryption algorithm using the key  $K_c$  and finishes the decryption to obtain the lossless carrier image  $I$ :

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} pq + 1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N \quad (27)$$

$$I(I^{(1)}, I^{(2)}, \dots, I^{(\lceil H/2 \times W/2 \rceil)}) = \text{Dec}_{\text{coarse}}(I^*, K_c) \quad (28)$$

The method proposed in this paper has significant social impact in fields such as military imagery, remote healthcare, and forensic investigation. These fields demand the assurance of data security and integrity, which the proposed method can improve and ensure. As a multi-party secure cryptographic system, it can enhance data disaster recovery through secret sharing properties and have a positive effect on data security.

### 3. Demonstration and Analysis

#### 3.1. Method Demonstration

In the last section, we introduced the specific details of our proposed protocol. To better understand the proposed method, in this chapter, we show the whole method in distri-

butions. That is, the image encryption example diagram, secret split example diagram, data embedding example diagram, and data extraction example diagram; additionally, the extraction of feature images is integrated into the overall framework.

We will give a concrete example of the protocol and give an analysis and security proof of the protocol. We will take four  $2 \times 2$  image blocks as an example and set the parameters of  $(k, n)$  secret sharing as 4 and 5.

**Image encryption.** As shown in Figure 3, let an image with 16 original pixels be  $I$ . The image is divided into four  $2 \times 2$  dot blocks  $(I^{(1)}, I^{(2)}, I^{(3)}, I^{(4)})$ , and pixels of the same color are grouped together in the image encryption example diagram:

$$\begin{cases} I^{(1)} = (I_1^{(1)}, I_2^{(1)}, I_3^{(1)}, I_4^{(1)}) = (159, 152, 151, 158) \\ I^{(2)} = (I_1^{(2)}, I_2^{(2)}, I_3^{(2)}, I_4^{(2)}) = (152, 152, 154, 153) \\ I^{(3)} = (I_1^{(3)}, I_2^{(3)}, I_3^{(3)}, I_4^{(3)}) = (151, 158, 153, 160) \\ I^{(4)} = (I_1^{(4)}, I_2^{(4)}, I_3^{(4)}, I_4^{(4)}) = (156, 156, 155, 156) \end{cases} \quad (29)$$

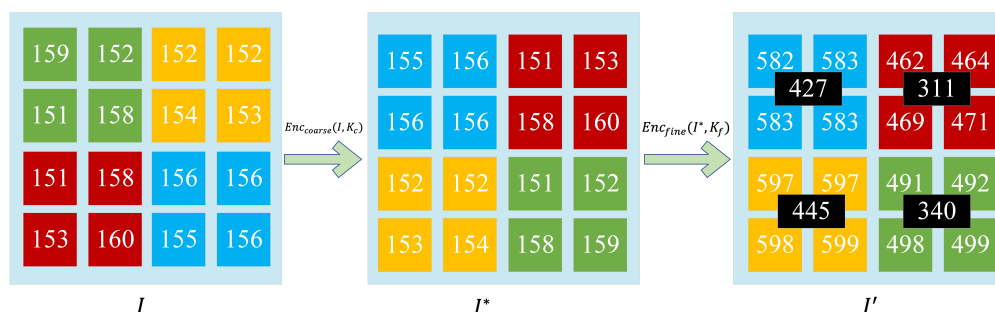
The image owner uses the coarse-grained Arnold image permutation algorithm  $Enc_{coarse}(I, K_c)$  to obtain  $I^*$ :

$$\begin{cases} I^{*(1)} = (I_1^{*(1)}, I_2^{*(1)}, I_3^{*(1)}, I_4^{*(1)}) = (155, 156, 156, 156) \\ I^{*(2)} = (I_1^{*(2)}, I_2^{*(2)}, I_3^{*(2)}, I_4^{*(2)}) = (151, 153, 158, 160) \\ I^{*(3)} = (I_1^{*(3)}, I_2^{*(3)}, I_3^{*(3)}, I_4^{*(3)}) = (152, 152, 153, 154) \\ I^{*(4)} = (I_1^{*(4)}, I_2^{*(4)}, I_3^{*(4)}, I_4^{*(4)}) = (151, 152, 158, 159) \end{cases} \quad (30)$$

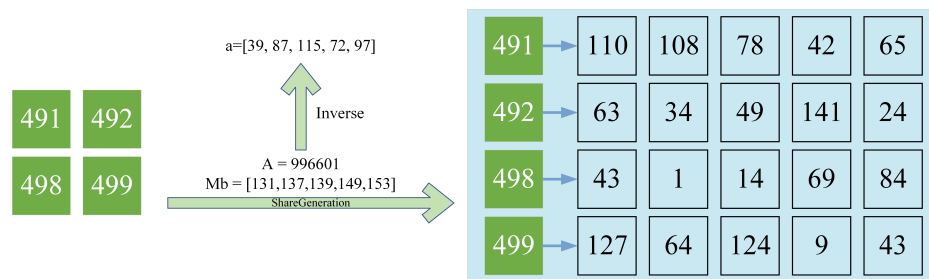
The image owner uses the fine-grained image scrambling algorithm  $Enc_{fine}(I^*, K_f)$ , where the encryption key  $K_f = 427, 311, 455, 340$ , and obtains  $I'$ :

$$\begin{cases} I'^{(1)} = (I_1'^{(1)}, I_2'^{(1)}, I_3'^{(1)}, I_4'^{(1)}) = (582, 583, 583, 583) \\ I'^{(2)} = (I_1'^{(2)}, I_2'^{(2)}, I_3'^{(2)}, I_4'^{(2)}) = (462, 464, 459, 471) \\ I'^{(3)} = (I_1'^{(3)}, I_2'^{(3)}, I_3'^{(3)}, I_4'^{(3)}) = (597, 597, 598, 599) \\ I'^{(4)} = (I_1'^{(4)}, I_2'^{(4)}, I_3'^{(4)}, I_4'^{(4)}) = (491, 492, 498, 499) \end{cases} \quad (31)$$

Shadow image generation: As shown in Figure 4, the image owner randomly picks the module  $Mb$  and the random mapping parameter  $A = 996, 601$ , as well as the secret key  $K_a \leftarrow (Mb, a, \Delta\chi)$  for the module  $Mb$  inverse of the random parameter  $A$ :



**Figure 3.** Example of image encryption. The size of the demo image is  $4 \times 4$ , the coarse-grained encryption secret key is  $(11, 10, 01, 00)$ , and the fine-grained encryption key is  $(427, 311, 455, 340)$ .



**Figure 4.** Example of shadow images generation. The random number is 996,601, the module base is (131, 137, 139, 149, 153), and the corresponding inverse is (39, 87, 115, 72, 97).

$$Mb = (Mb_1, Mb_2, Mb_3, Mb_4, Mb_5) = (131, 137, 139, 149, 153) \tag{32}$$

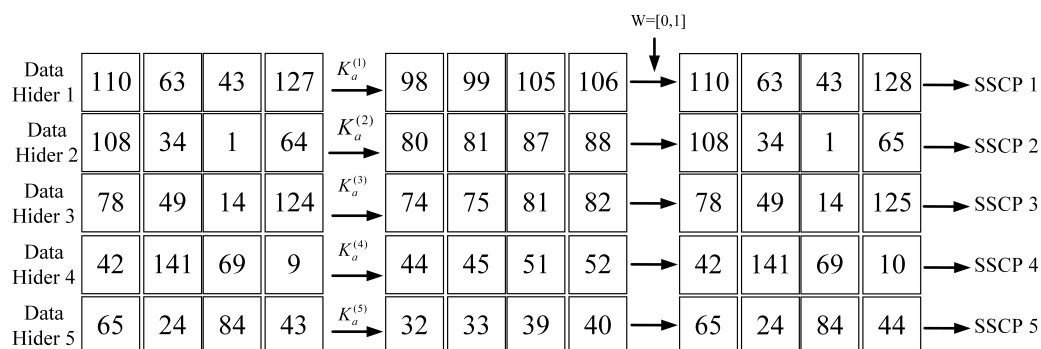
$$a = A^{-1} \text{ mod } Mb = (a^{(1)}, a^{(2)}, a^{(3)}, a^{(4)}, a^{(5)}) = (39, 87, 115, 72, 97) \tag{33}$$

The image owner sends the split secret data  $I'$  and the embedding secret key  $K_a$  to the corresponding data hider using a trusted channel.  $\tilde{I}^{(1\sim5)}$  is the secret share map of the difference information obtained by  $DH^{(1\sim5)}$  with  $K_a^{(1\sim5)}$ :

$$\begin{cases} \tilde{I}^{(1)} = (I_1^{(1)}, I_1^{(2)}, I_1^{(3)}, I_1^{(4)})a^{(1)} \text{ mod } Mb^{(1)} = (98, 99, 105, 106) \\ \tilde{I}^{(2)} = (I_2^{(1)}, I_2^{(2)}, I_2^{(3)}, I_2^{(4)})a^{(2)} \text{ mod } Mb^{(2)} = (80, 81, 87, 88) \\ \tilde{I}^{(3)} = (I_3^{(1)}, I_3^{(2)}, I_3^{(3)}, I_3^{(4)})a^{(3)} \text{ mod } Mb^{(3)} = (74, 75, 81, 82) \\ \tilde{I}^{(4)} = (I_4^{(1)}, I_4^{(2)}, I_4^{(3)}, I_4^{(4)})a^{(4)} \text{ mod } Mb^{(4)} = (44, 45, 51, 52) \\ \tilde{I}^{(5)} = (I_5^{(1)}, I_5^{(2)}, I_5^{(3)}, I_5^{(4)})a^{(5)} \text{ mod } Mb^{(5)} = (32, 33, 39, 40) \end{cases} \tag{34}$$

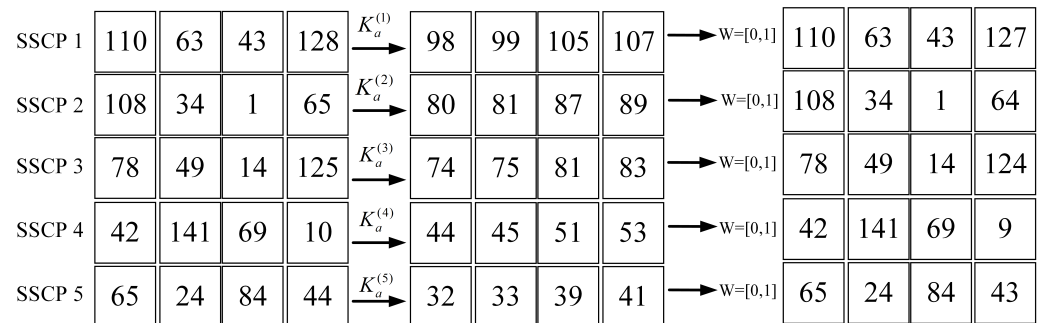
As shown in Figure 5, we can see that  $|d_{\max}^{(1\sim5)}| = |d_{\min}^{(1\sim5)}| = 1$ ; both are less than or equal to  $\Delta$ . Therefore, the data hider performs the data embedding operation for both the highest and lowest bits to obtain the shadow image share after embedding the data and sends the secret share to the corresponding SSCP:

$$\begin{cases} \text{Share}^{(1)} = (110 - 0 \times \Delta, 63, 43, 127 + 1 \times \Delta) = (110, 63, 43, 128) \\ \text{Share}^{(2)} = (108 - 0 \times \Delta, 34, 1, 64 + 1 \times \Delta) = (108, 34, 1, 65) \\ \text{Share}^{(3)} = (78 - 0 \times \Delta, 49, 14, 124 + 1 \times \Delta) = (78, 49, 14, 125) \\ \text{Share}^{(4)} = (42 - 0 \times \Delta, 141, 69, 9 + 1 \times \Delta) = (42, 141, 69, 10) \\ \text{Share}^{(5)} = (65 - 0 \times \Delta, 24, 84, 43 + 1 \times \Delta) = (65, 24, 84, 44) \end{cases} \tag{35}$$



**Figure 5.** Example of data hiding phase. The secret data to be embedded are (0,1), and the data hider first converts the shadow image into a guide image. Subsequently, it performs an embedding operation on the shadow image based on the difference between the pixels of the guide image.

As shown in Figure 6, receiver  $(1\sim 5)$  recovers the secret share map that represents the image difference information and obtains  $W = [0, 1]$  after receiving the embedding data recovery request using  $K_a^{(1\sim 5)}$ . The shadow image recovery and image decryption are the reverse of the above steps and will not be repeated.



**Figure 6.** Example of data extraction phase. Receiver first converts the embedded shadow image into a guide image. Subsequently, it performs an data extraction operation on the shadow image based on the difference between the pixels of the guide image, and obtains the secret data (0,1).

### 3.2. Method Analysis

**Definition 1.** The coarse-fine-grain encryption method combined with PVO performs the encryption and decryption of the image correctly.

**Proof.** For an image  $I$ , let us suppose the initial position of pixel points is  $(x_n, y_n)$ ; according to the coarse-grained encryption and decryption method is Equations (2) and (27), there is

$$\begin{bmatrix} pq+1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} \bmod N = \begin{bmatrix} pq+1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N \quad (36)$$

and there is

$$\begin{bmatrix} pq+1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (37)$$

It can be easily seen that

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \quad (38)$$

Thus, the coarse-grained encryption is lossless and reversible.

Moreover, the fine-grained encryption uses a one-dimensional logical chaotic system to generate a fixed sequence with the same initial value  $x_0$  and parameter  $\mu$ . Therefore, the random numbers added to the image pixel blocks can be recovered by generating the same sequence, so fine-grained encryption is lossless and reversible.

Furthermore,  $(\delta_1, \delta_2, \delta_3, \delta_4) \leftarrow PVO(\delta_1, \delta_2, \delta_3, \delta_4)$ , where the set  $(\delta_1, \delta_2, \delta_3, \delta_4)$  and where the set  $(\delta_1, \delta_2, \delta_3, \delta_4)$  to the set  $(\delta_1, \delta_2, \delta_3, \delta_4)$  is a full projection. The position of the original pixel can be restored losslessly according to the position relationship table if the position relationship table is saved with 8 bits. The position of the original pixel can be restored losslessly according to the position relationship table. Therefore, the correctness of the encryption and decryption combined with the coarse-fine-grained encryption method of PVO is proved.  $\square$

**Definition 2.** This secret-sharing method can correctly split the encrypted image into multiple shadow images and recover the shadow image into the carrier image. The message hider embeds the message, and the receiver retrieves the embedded secret without affecting the lossless recovery of the carrier image.

**Proof.** Let the encryption key of the secret sharing algorithm be  $K_g = (\vartheta, Mb)$ , the original data are  $x$ , and the image owner performs the secret splitting algorithm with Equation (15), where  $b_1, b_2, b_3, \dots, b_k$  are two mutually prime positive integers, and in the embedding phase of the data hider, there are

$$\begin{cases} \tilde{r}_1 = r_1 + \Delta w \pmod{Mb_1} \\ \tilde{r}_2 = r_2 + \Delta w \pmod{Mb_1} \\ \vdots \\ \tilde{r}_k = r_k + \Delta w \pmod{Mb_k} \end{cases} \tag{39}$$

There must be a unique solution for the embedded data, i.e.,

$$T = x\vartheta + w \equiv B_1B_1^{-1}(r_1 + w) + \dots + B_kB_k^{-1}(r_k + w) \pmod{B} \tag{40}$$

In case the image owner needs to recover the carrier image, there is

$$x = \frac{B_1B_1^{-1}(r_1 + \Delta w) + \dots + B_kB_k^{-1}(r_k + \Delta w) \pmod{B} - \Delta w}{\vartheta}$$

Thus, the splitting and recovery of secret sharing are non-destructive and reversible. Thus, definition 2 is proved.  $\square$

**Definition 3.** The proposed scheme in this paper exhibits a threshold property of  $(k, n)$ , which implies that  $s$  colluding parties with less than  $k$  information embedders would still fail to reconstruct the images encrypted using PVO's coarse-fine-grained encryption method.

**Proof.** We take the example of each data hider having a single shadow image, assuming there are  $n$  data hidens. Let  $K_g = (\vartheta, Mb)$  be the encryption key of the secret sharing algorithm, and let  $I(h, w) \in [0, 255]$  be the original pixel values. To ensure that the split shadow images still take the form of images, with a maximum single-channel pixel value of 255, the elements of  $M_b$  must satisfy the following:

$$\begin{aligned} & \{128 < Mb^{(1)} < Mb^{(2)} < \dots < Mb^{(n)} \leq 251 < Mp\} \\ & \gcd(Mb^{(i)}, Mb^{(j)}) = 1, i \neq j \\ & \gcd(Mb^{(j)}, Mp) = 1, i = 1, 2, \dots, n \\ & \gcd(Mb^{(i)}, \vartheta) = 1, i = 1, 2, \dots, n \end{aligned} \tag{41}$$

Assuming the image owner sets a  $(k, n)$  threshold during the key generation stage,  $x\vartheta$  must satisfy the following relationship to achieve the threshold property:

$$\prod_{i=1}^{k-1} Mb^{(i)} < x\vartheta < \prod_{i=1}^k Mb^{(i)} \tag{42}$$

However, the size of the pixel values  $x$  changes after fine encryption. Therefore, different random  $\vartheta$  must be set to achieve a dynamically compatible threshold scheme when setting different values of  $k$ :

$$\prod_{i=1}^{k-1} Mb^{(i)} / x < \vartheta < \prod_{i=1}^k Mb^{(i)} / x \tag{43}$$

This allows for  $(k, n)$  threshold secret sharing. Thus, definition 3 is proved.  $\square$



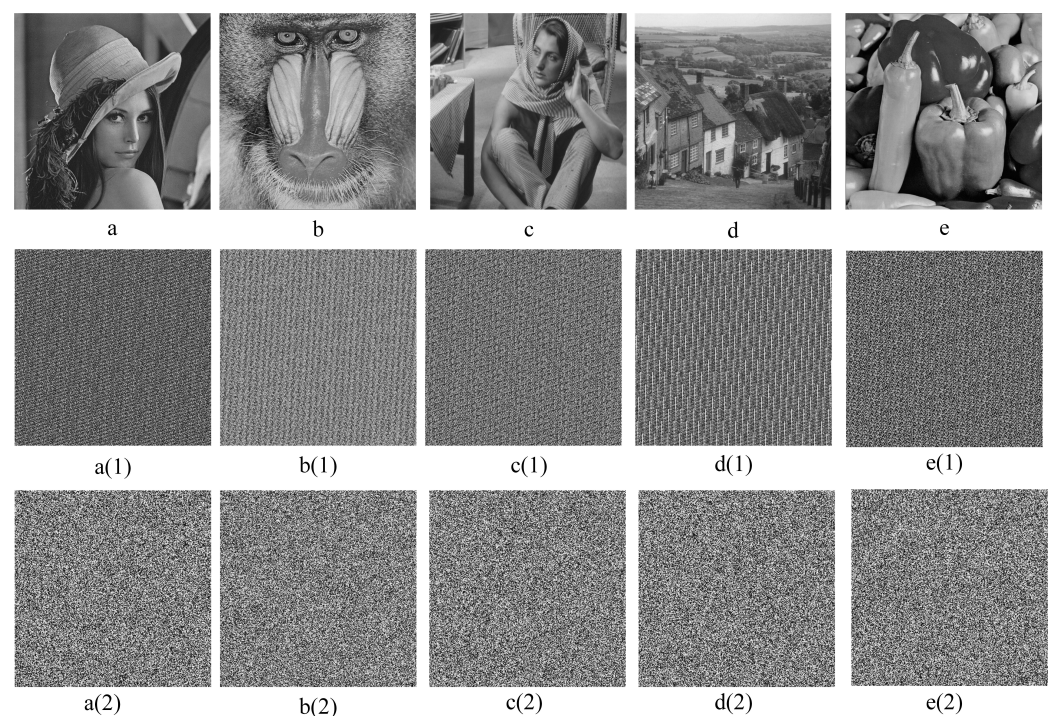
#### 4. Experiments and Numerical Results

The six test images, including “Lena”, “Baboon”, “Barbara”, “Goldhill”, and “Peppers”, are from the dataset used for the experiments. All test images are grayscale images with a size of  $512 \times 512$ . The experimental environment was as follows: host configuration CPU AMD 5800X, memory 32 GB, operating system Windows 11, programming language python 3.9, and MATLAB 2022b.

##### 4.1. Performance of Image Encryption

According to the guideline [28], we conduct multiple experiments to ensure the security of image encryption by testing image randomness, similarity metrics, graphic correlation, and noise robustness.

**Image randomness.** The intruder shares two secret images at the same time. To the naked eye, the shadow image must look like noise. In addition, the shadow images can be evaluated in a histogram; the more uniform the histogram distribution, the more secure the proposed scheme is. As shown in Figure 7, the first line is the encryption effect of Lena, Baboon, Barbara, Goldhill, and Peppers, and the second line is the encryption result after coarse-grained encryption, which exhibits a strong regularity among pixels, but it is visually impossible to tell what the original image is. The third line is the encryption result after fine-grained encryption. From the line, we can see that after the fine-grained encryption, the statistical features of the gray pixels of the carrier image can be effectively concealed.



**Figure 7.** The visual effect of image encryption. The first row (a–e) is the original carrier image, the second (a(1)–e(1)) is the image after coarse-grained encryption, and the third (a(2)–e(2)) is the image after fine-grained encryption.

In addition, we propose two evaluation metrics for the similarity metric, which can be measured by the peak signal-to-noise ratio (PSNR) shown in Equation (44) and the

structural similarity (SSIM) shown in Equations (45) and (46). PSNR evaluates the similarity of images and MSE denotes mean square error:

$$\begin{cases} PSNR = 10 \times \log_{10} \left( \frac{255^2}{MSE} \right) dB \\ MSE = \frac{1}{W \times H} \sum_{i=1}^W \sum_{j=1}^H [S'(i, j) - S(i, j)]^2 \end{cases} \quad (44)$$

$$SSIM(S, S') = \frac{(2\mu_S \mu_{S'} + c_1)(2\sigma_{SS'} + c_2)}{(\mu_S^2 + \mu_{S'}^2 + c_1)(\sigma_S^2 + \sigma_{S'}^2 + c_2)} \quad (45)$$

$$\begin{cases} c_1 = (0.01 \times L)^2, c_2 = (0.03 \times L)^2, \\ \{x, y \mid S, S' \text{ and } x \neq y\}, \{L \mid 0 \leq L \leq 255\}, \\ \mu_x = \frac{1}{W \times H} \sum_{i=1}^W \sum_{j=1}^H x(i, j), \sigma_x = \sqrt{\sigma_x^2} \\ \sigma_x^2 = \frac{1}{W \times H - 1} \sum_{i=1}^W \sum_{j=1}^H (x(i, j) - \mu_x)^2, \\ \sigma_{xy} = \frac{1}{W \times H - 1} \sum_{i=1}^W \sum_{j=1}^H (x(i, j) - \mu_x)(y(i, j) - \mu_y) \end{cases} \quad (46)$$

Furthermore, we use the one-dimensional grayscale entropy of an image to the amount of average information in an image. The one-dimensional entropy indicates the amount of information contained in the aggregated features of grayscale distribution in the image so that  $p_i$  denotes the proportion of pixels with a grayscale value  $i$  in the image. The one-dimensional grayscale entropy of a grayscale image is defined as, for an image, the closer the information entropy is to 8, the more confusing the image is. As can be seen from Table 1, the information entropy of the images after fine-grained encryption can reach 7.999 information entropy, indicating that the proposed encryption effect can meet the security requirements. As shown in Figure 8, the histogram also shows that the statistical properties of the original pixels are effectively masked after encryption, which has statistically solid properties before fine-grained encryption:

$$H = \sum_{i=0}^{255} p_i \log p_i \quad (47)$$

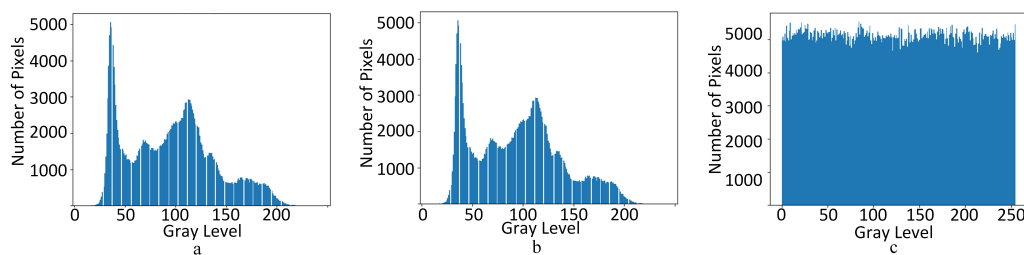


Figure 8. The gray histogram of Lena. From left to right are the original carrier image (a), the coarse-grained encrypted image (b), and the fine-grained encrypted image (c).

Table 1. PSNR, entropy, and SSIM for the encrypted image.

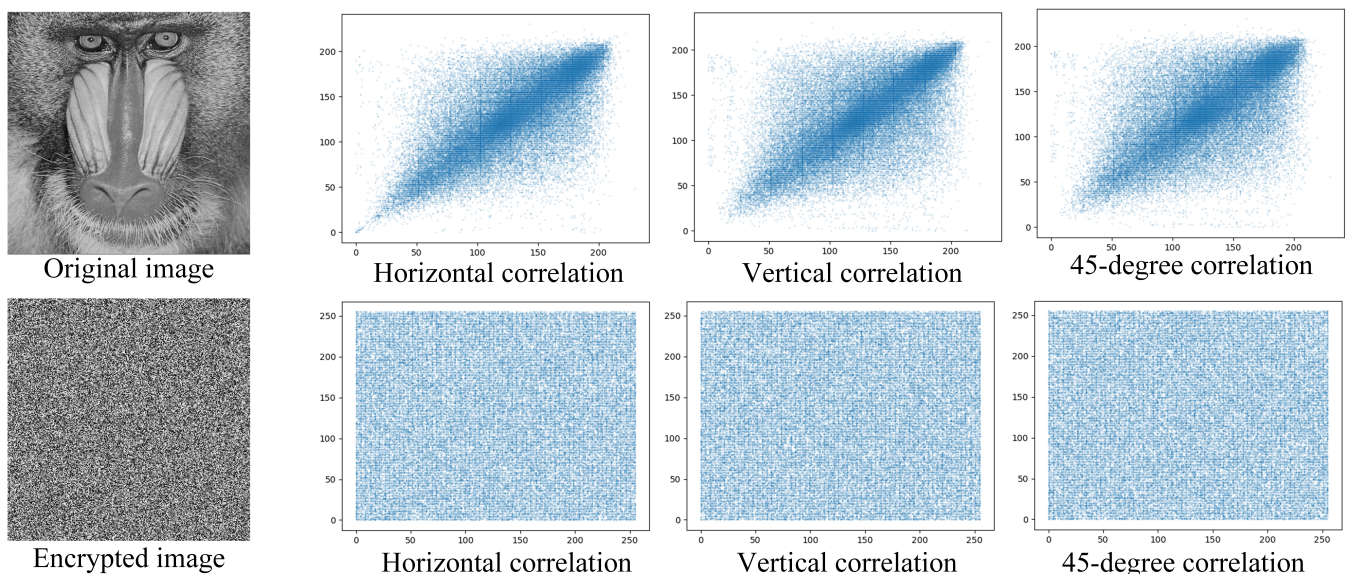
Test Images	Coarse-Grained Encrypted Image			Coarse-Grained Encrypted Image			Restored Image		
	PSNR (dB)	Entropy	SSIM	PSNR (dB)	Entropy	SSIM	PSNR (dB)	Entropy	SSIM
Lena	12.272	7.234	0.030	8.913	7.999	0.011	∞	7.234	0
Baboon	12.614	7.358	0.032	9.526	7.999	0.011	∞	7.358	0
Barbara	11.666	7.466	0.029	9.104	7.999	0.012	∞	7.466	0
Goldhill	11.28	7.478	0.0327	9.040	7.999	0.011	∞	7.4777	0
Peppers	9.942	7.571	0.0182	8.430	7.999	0.012	∞	7.571	0

**Graphic Correlation.** Pixel values in plain images are highly similar to their neighbors, whether located horizontally, vertically, or diagonally. This strong correlation poses a risk for statistical attacks, necessitating efficient cryptosystems to create non-correlated

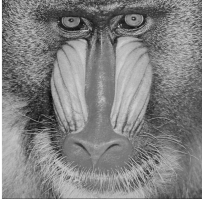
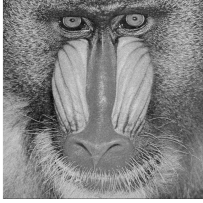
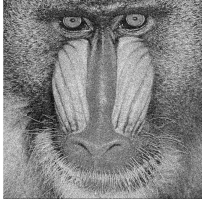
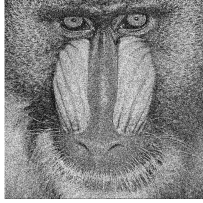
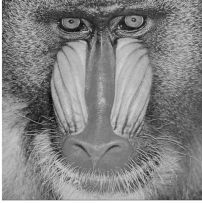
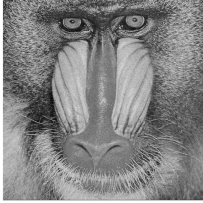
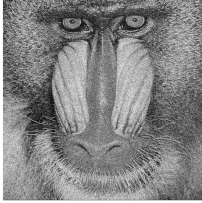
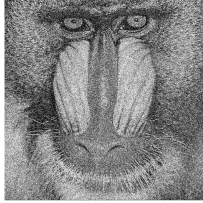
encrypted images. Graphic correlation is a visual inspection of an image's pixel correlation, with the horizontal axis indicating pixel intensity and the vertical axis showing neighbor pixel values.

As shown in Figure 9, we conducted thorough experiments to validate the correlation between pixels. We performed a block-by-block analysis of the graphic correlation of both the original image and the image encrypted with our fine-grained encryption system. The results revealed a significant similarity between the pixel values of the original image and its neighboring pixels, creating a severe vulnerability to statistical attacks. Additionally, the correlation pattern over the 45-degree line in the original image was observed to be strong. The closer the pattern was to this line, the higher the pixel correlation. Nevertheless, we observed that our fine-grained encryption system generated an encrypted image. A visual inspection of its graphic correlation revealed strong non-correlation in the horizontal, vertical, and 45-degree directions. This is due to most neighboring pixels possessing different intensity values.

**Noise Robustness.** The transmission channel may cause encrypted images to suffer from noise attacks or interference, making it difficult to recover the plain image. Therefore, evaluating noise robustness is necessary for chaos-based image cryptosystems. Our selected algorithms employ substitution–permutation networks, stream encryption, and one-per-one pixel encryption, providing certain advantages against noise. With these principles and bulk image data, the original plain image can still be reconstructed with high visual quality even during the encryption process. To further enhance the noise robustness of our method, we have adopted an error correction algorithm. In this algorithm, if the first pixel value of a  $2 \times 2$  image block is greater than the second pixel value, the second pixel value is assigned to the first pixel. Similarly, if the third pixel value is greater than the fourth pixel value, the fourth pixel value is assigned to the third pixel. We test the visual effects of our proposed algorithm under different levels of noise (1%, 10%, 30%, 50%) to demonstrate its ability to resist noise attacks, as shown in Figure 10. The test results indicate that our proposed algorithm with error correction capability has a stronger resistance against noise attacks than the recovery based on pure chaotic systems, as indicated by the MES results.

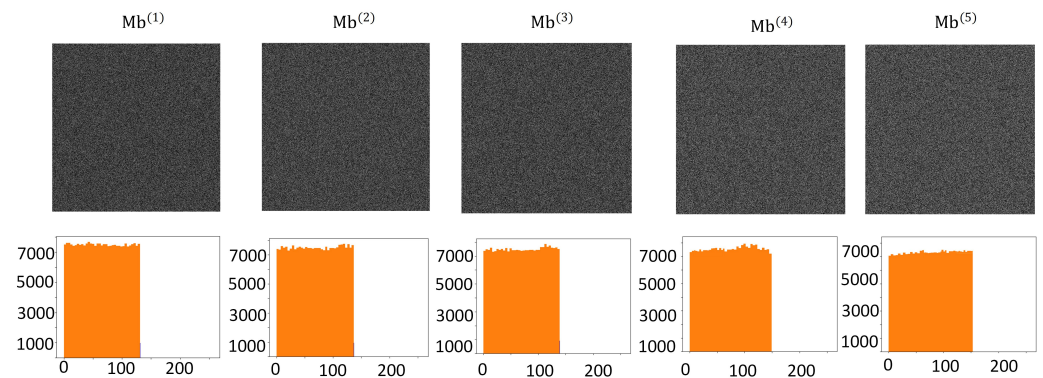


**Figure 9.** Results of the graphic correlation test in the horizontal, vertical, and 45-degree directions.

Noise rate	1%	10%	30%	50%
Chaotic based				
MSE	0.47	57.00	96.01	103.52
Our scheme				
MSE	0.42	52.18	93.51	98.48

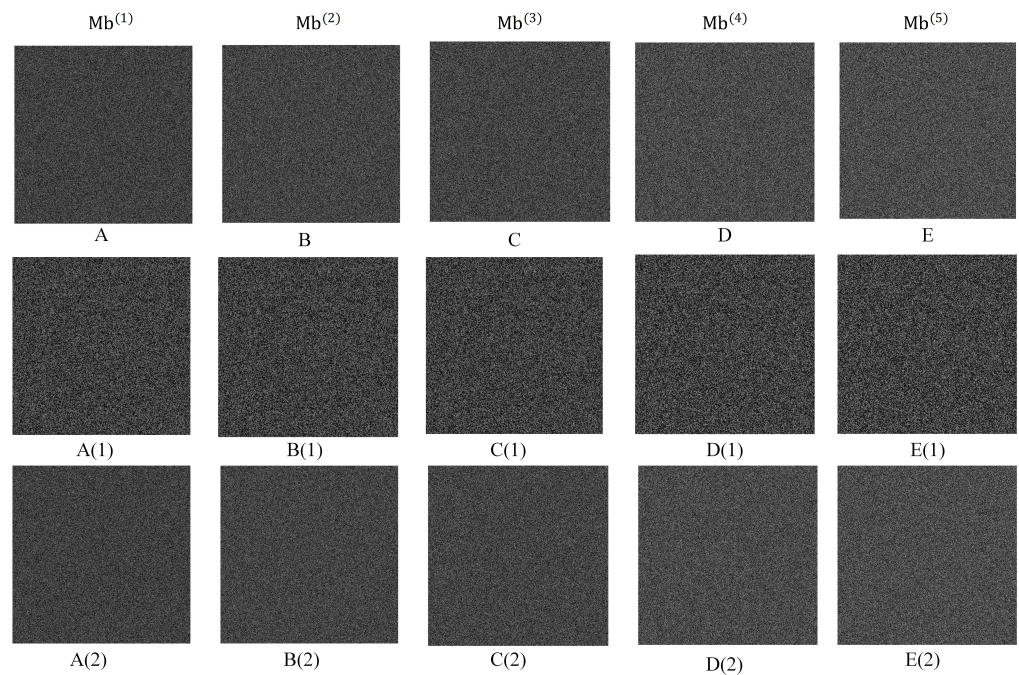
**Figure 10.** Noise robustness: The visual effects under different noise rates (1%, 10%, 30%, 50%) are shown through the MES results, which demonstrate the stronger ability of the proposed algorithm to resist noise attacks.

**The randomness of shadow images.** Assuming that an attacker can share two shadow images simultaneously, both must be indistinguishable noise classes. In addition, shadow images can be evaluated in histograms; the more uniform the distribution of the histograms, the safer the proposed scheme. As can be seen from Figure 11, the histogram distribution is approximately the same under different module bases, and no valid information can be distinguished from the shadow image.



**Figure 11.** Comparison of statistical histograms of shadow images under different modules.

Figure 12 shows the shadow image under different module bases and the data obtained by embedding the secret key. It can be seen that the shadow image is sufficiently able to show the correlation between pixels. For the convenience of the display, the pixel box in the upper right corner corresponds to the pixel value of the first  $2 \times 2$  sub-block in the upper leftmost corner. It can be seen that each data hider recovers a different guide image, but all generate the same different histogram.



**Figure 12.** Visual display of the shadow images, guide images, and embedded images. The first row (A–E) is the shadow image, the second row (A(1)–E(1)) is the embedded guide image, and the third row (A(2)–E(2)) is the shadow image after embedding the secret message statistical histograms of shadow images under different modulus bases.

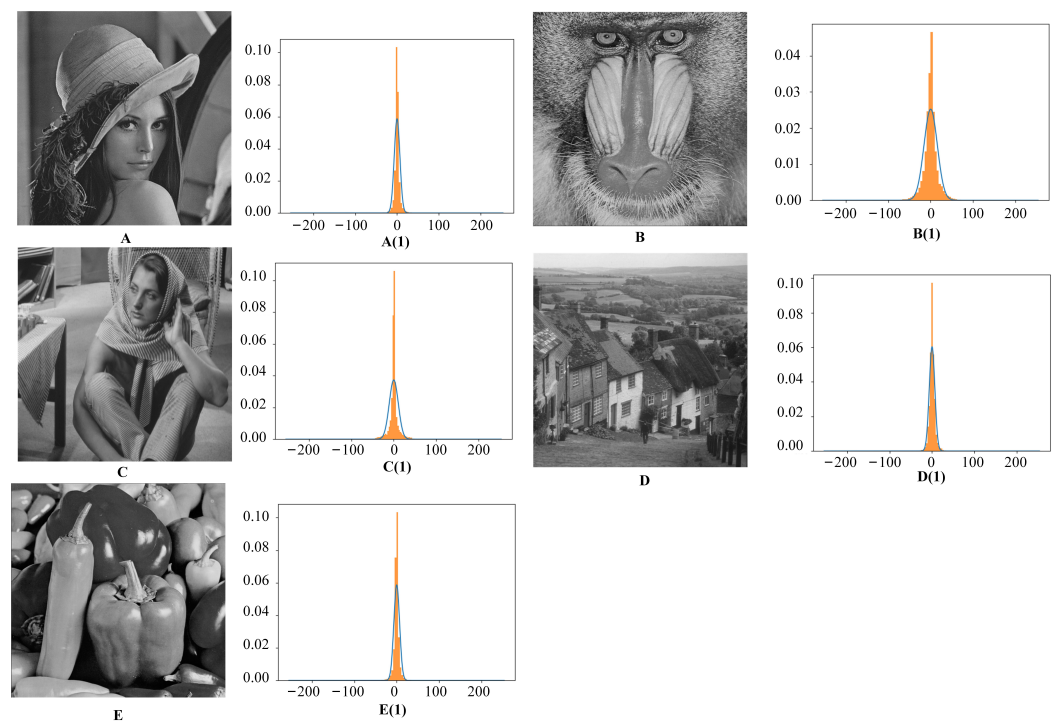
#### 4.2. Comparison with the State of the Art

To better measure our proposed method, we introduce an embedding rate formula. In order to show the embedding strategy more clearly, the embedding rate used to evaluate the embedding capacity of the proposed scheme and other related schemes is calculated as follows:

$$ER(bpp) = \frac{\text{Total number of embedded bits of the image}}{H \times W} \quad (48)$$

It can be seen that the embedding capacity is related to the embedding step size and embedding size. The larger the step size, the larger the embedding capacity, and the more pronounced the influence on the visual effect.

The statistical histograms of the ciphertext prediction errors for five images are shown in Figure 13. Since the plaintext prediction error and ciphertext have the same statistical error, we can use the histogram transfer formula Equations (17) and (18) to hide the data in the ciphertext images. In addition, the prediction error histograms of the ciphertext images of “Lena”, “Goldhill”, and “Peppers” are mainly around the value of 0, while the prediction difference histograms of Baboon and Barbara are far from the value of 0 of the three images mentioned above. It can also be seen from the table that Baboon and Barbara are different from “Lena”, “Goldhill”, and “Peppers” in terms of having lower embedding rates.



**Figure 13.** Comparison of pixel difference histograms of different images. Images (A–E) represent the original images, while image (A(1)–E(1)) correspond to the difference histograms of the encrypted images.

Table 2 shows the theoretical embedding rate/actual embedding rate. We conducted tests on the actual embedding rate by employing Lena plots and set the three-out-of-four threshold. It is known from the algorithm proposed in this paper that, for each block, the pixel corresponding to the maximum value and the pixel corresponding to the minimum value of the four-pixel blocks can perform embedding, i.e., half of the pixels can perform embedding after deciding whether the value in the pixel can be embedded in the data according to different thresholds. In addition, the size of the embedding amount should be less than the threshold value, and this paper uses the embedding amount of bits 1, 2, 3 corresponding to the threshold bits 1, 2, 3, 4, respectively. The table shows that the embedding rate is obtained at 1.038, 2.316, and 5.226, depending on how the embedding amount and threshold bits are selected. It can be seen that the proposed method in this paper is linear in the growth of the embedding amount. This growth is in line with the increase in the corresponding embedding amount and the number of threshold bits.

**Table 2.** Embedding rate of proposed scheme.

Embedding Amount	Theoretical Embedding Rate				Actual Embedding Rate			
	1 bit	2 bit	3 bit	4 bit	1 bit	2 bit	3 bit	4 bit
1 bit	0.5	1.5	1.5	1.5	0.346	1.038	1.305	1.419
2 bit	-	3	3	3	-	2.316	2.613	2.838
3 bit	-	-	6	6	-	-	5.226	5.679

We compare the embedding rate of the proposed method with several state-of-the-art methods. As shown in Table 3, the embedding rate of the proposed method is significantly higher than those of the prior art methods. Furthermore, the embedding rate of the method varies with the different histograms of the image. The closer the value in the difference histogram is to zero, the higher the embedding rate is. Images with smooth textures can

obtain higher embedding rates, while images with complex textures have lower embedding rates, such as the image Baboon.

**Table 3.** Embedding rate among the proposed scheme and state-of-the-art schemes.

Image Name	[10]	[11]	[23]	[29]	[30]	Ours
Lena	0.973	1.575	0.5	3.5	5.3	<b>5.676</b>
Baboon	0.872	0.728	0.5	3.5	<b>5.3</b>	4.776
Barbara	0.96	1.265	0.5	3.5	5.3	<b>5.367</b>
Goldhill	0.963	1.542	0.5	3.5	5.3	<b>5.583</b>
Peppers	0.976	1.545	0.5	3.5	5.3	<b>5.703</b>
Average	0.959	1.331	0.5	3.5	5.3	<b>5.421</b>

This paper and other methods exploit the correlation of natural images, such as MSB prediction [10,11], and differential extension [23], all of which have embedding rates that vary with the redundancy of the images. In addition, the method proposed in this paper does not require data compression. However, it performs data embedding by recovering histograms that retain pixel difference information through the adaptive generation of bootstrap maps that automatically generate embedding secret keys, from which high embedding rates are obtained.

In the scheme comparison, as shown in Table 4, all the compared algorithms use the Vacating room approach to achieve an embedding efficiency higher than 0.5 bpp. Among them, [10,11,23] use the Vacating room approach before encryption to increase the embedding capacity. In addition, the proposed method in this paper uses a combination of stream encryption and secret sharing based on chaos, which is more secure than other state-of-the-art algorithms, single stream encryption [10,11], and single secret sharing [23]. In addition, this paper innovatively proposed a way of generating guidance graphs by shadow images, which reduces the scheme's complexity and the variety of information transfer.

**Table 4.** Feature comparison among the proposed scheme and state-of-the-art schemes.

Scheme	Separable	Vacating Room	Encryption Strategy	Multi-Data Hider	Availability of Guidance Information
[10]	Yes	Before	Stream cipher	No	Yes
[11]	Yes	Before	Stream cipher	No	Yes
[23]	No	Before	Secret sharing	No	Yes
Ours	Yes	Before	Stream cipher+ Secret sharing	Yes	No

## 5. Conclusions

This study addresses several key issues, including reversible information hiding in encrypted domains in various data-embedding scenarios, improved encryption effectiveness, and enhanced information-embedding capability. We propose a lossless RDH-EI method based on PVO and secret sharing for multiple data hidere, which addressed the problem of secure and efficient data embedding in encrypted images. The method ensures lossless embedding and extraction, and the original image's digital assets can be fully recovered. The proposed PVO-based secret sharing scheme achieves a high embedding rate and is significantly better than existing encryption techniques in terms of efficiency and effect. The combined stream encryption with the secret sharing approach significantly improves the security of the method, ensuring that the data hidere can embed their data securely without any fear of data leakage. Moreover, the proposed scheme does not require any extra guide parameters, which enhances its usability.

**Author Contributions:** Conceptualization, H.Y. and H.F.; methodology, H.Y. and J.Z.; writing—original draft preparation, H.Y.; writing—review and editing, H.Y. and H.F.; supervision, B.L. and Z.X.; project administration, Z.X.; funding acquisition, H.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is partially supported by NSFC-General Technology Fundamental Research Joint Fund under grant U1836215, National Nature Science Foundation of China under grant 62102040, BUPT Excellent Ph.D. Students Foundation (CX2021124), the “High-precision” Discipline Construction Project of Beijing Universities (No.20210071Z0403).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ke, Y.; Zhang, M.Q.; Liu, J.; Su, T.T.; Yang, X.Y. Fully homomorphic encryption encapsulated difference expansion for reversible data hiding in encrypted domain. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2353–2365. [[CrossRef](#)]
2. Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
3. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [[CrossRef](#)]
4. Celik, M.U.; Sharma, G.; Tekalp, A.M.; Saber, E. Lossless generalized-LSB data embedding. *IEEE Trans. Image Process.* **2005**, *14*, 253–266. [[CrossRef](#)] [[PubMed](#)]
5. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [[CrossRef](#)]
6. Puech, W.; Chaumont, M.; Strauss, O. A reversible data hiding method for encrypted images. In Proceedings of the Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, San Jose, CA, USA, 27–31 January 2008; Volume 6819, pp. 534–542.
7. Zhang, X. Separable reversible data hiding in encrypted image. *IEEE Trans. Inf. Forensics Secur.* **2011**, *7*, 826–832. [[CrossRef](#)]
8. Chen, C.C.; Chang, C.C.; Chen, K. High-capacity reversible data hiding in encrypted image based on Huffman coding and differences of high nibbles of pixels. *J. Vis. Commun. Image Represent.* **2021**, *76*, 103060. [[CrossRef](#)]
9. Thodi, D.M.; Rodriguez, J.J. Prediction-error based reversible watermarking. In Proceedings of the 2004 International Conference on Image Processing, ICIP’04, Singapore, 24–27 October 2004; Volume 3, pp. 1549–1552.
10. Puteaux, P.; Puech, W. An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1670–1681. [[CrossRef](#)]
11. Wu, H.T.; Yang, Z.; Cheung, Y.M.; Xu, L.; Tang, S. High-capacity reversible data hiding in encrypted images by bit plane partition and MSB prediction. *IEEE Access* **2019**, *7*, 62361–62371. [[CrossRef](#)]
12. Westfeld, A. F5—A steganographic algorithm. In Proceedings of the International Workshop on Information Hiding, Pittsburgh, PA, USA, 25–27 April 2001; pp. 289–302.
13. Ke, Y.; Zhang, M.Q.; Liu, J.; Su, T.T.; Yang, X.Y. A multilevel reversible data hiding scheme in encrypted domain based on LWE. *J. Vis. Commun. Image Represent.* **2018**, *54*, 133–144. [[CrossRef](#)]
14. Huang, D.; Wang, J. High-capacity reversible data hiding in encrypted image based on specific encryption process. *Signal Process. Image Commun.* **2020**, *80*, 115632. [[CrossRef](#)]
15. Chen, Y.C.; Shiu, C.W.; Horng, G. Encrypted signal-based reversible data hiding with public key cryptosystem. *J. Vis. Commun. Image Represent.* **2014**, *25*, 1164–1170. [[CrossRef](#)]
16. Wu, X.; Chen, B.; Weng, J. Reversible data hiding for encrypted signals by homomorphic encryption and signal energy transfer. *J. Vis. Commun. Image Represent.* **2016**, *41*, 58–64. [[CrossRef](#)]
17. Xiong, J.; Bi, R.; Zhao, M.; Guo, J.; Yang, Q. Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles. *IEEE Wirel. Commun.* **2020**, *27*, 24–30. [[CrossRef](#)]
18. Zhou, Z.; Tian, Y.; Xiong, J.; Ma, J.; Peng, C. Blockchain-enabled Secure and Trusted Federated Data Sharing in IIoT. *IEEE Trans. Ind. Inform.* **2022**, *19*, 6669–6681. [[CrossRef](#)]
19. Wang, L.; Tian, Y.; Xiong, J. Achieving reliable and anti-collusive outsourcing computation and verification based on blockchain in 5G-enabled IoT. *Digit. Commun. Netw.* **2022**, *8*, 644–653. [[CrossRef](#)]
20. Thien, C.C.; Lin, J.C. Secret image sharing. *Comput. Graph.* **2002**, *26*, 765–770. [[CrossRef](#)]
21. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [[CrossRef](#)]
22. Wu, X.; Weng, J.; Yan, W. Adopting secret sharing for reversible data hiding in encrypted images. *Signal Process.* **2018**, *143*, 269–281. [[CrossRef](#)]
23. Chen, Y.C.; Hung, T.H.; Hsieh, S.H.; Shiu, C.W. A new reversible data hiding in encrypted image based on multi-secret sharing and lightweight cryptographic algorithms. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 3332–3343. [[CrossRef](#)]
24. Ke, Y.; Zhang, M.; Zhang, X.; Liu, J.; Su, T.; Yang, X. A reversible data hiding scheme in encrypted domain for secret image sharing based on Chinese remainder theorem. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 2469–2481. [[CrossRef](#)]
25. Xiong, L.; Han, X.; Yang, C.N.; Xia, Z. RDH-DES: Reversible Data Hiding over Distributed Encrypted-Image Servers based on Secret Sharing. *ACM Trans. Multim. Comput. Commun. Appl.* **2022**, *19*, 1–19. [[CrossRef](#)]



26. Musanna, F.; Kumar, S. Generating visually coherent encrypted images with reversible data hiding in wavelet domain by fusing chaos and pairing function. *Comput. Commun.* **2020**, *162*, 12–30. [[CrossRef](#)] [[PubMed](#)]
27. Murillo-Escobar, M.; Cruz-Hernández, C.; Cardoza-Avenidaño, L.; Murillo-Escobar, D.; López-Gutiérrez, R. Multibiosignal chaotic encryption scheme based on spread spectrum and global diffusion process for e-health. *Biomed. Signal Process. Control.* **2022**, *78*, 104001. [[CrossRef](#)]
28. Murillo-Escobar, M.A.; Meranza-Castillón, M.O.; López-Gutiérrez, R.M.; Cruz-Hernández, C. Suggested integral analysis for chaos-based image cryptosystems. *Entropy* **2019**, *21*, 815. [[CrossRef](#)]
29. Chen, B.; Lu, W.; Huang, J.; Weng, J.; Zhou, Y. Secret sharing based reversible data hiding in encrypted images with multiple data-hiders. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 978–991. [[CrossRef](#)]
30. Weng, C.Y.; Yang, C.H. Reversible Data Hiding in Encrypted Image Using Multiple Data-Hiders Sharing Algorithm. *Entropy* **2023**, *25*, 209. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.