Matrix-Based Secret Sharing for Reversible Data Hiding in Encrypted Images

Zhongyun Hua, *Member, IEEE*, Yanxiang Wang, Shuang Yi, Yifeng Zheng, Xingyu Liu, Yongyong Chen, and Xinpeng Zhang, *Member, IEEE*

Abstract—Traditional schemes for reversible data hiding in encrypted images (RDH-EI) focus on one data hider and cannot resist the single point of failure. Besides, the image security is determined by one party, rather than multiple parties. Thus, it is valuable to design RDH-EI schemes with multiple data hiders for stronger security. In this article, we propose a multiple data hiders-based RDH-EI scheme using a new secret sharing technique. First, we devise an (r, n)-threshold $(r \leq n)$ matrix-based secret sharing (MSS) using matrix theory, and theoretically verify its efficacy and security properties. Then, using the MSS, we propose an (r, n)-threshold RDH-EI scheme called MSS-RDHEI. The content owner encrypts an image to be n encrypted images using the MSS with an encryption key, and outsources these encrypted images to n data hiders. Each data hider can embed some data, e.g., copyright and identification information, into the encrypted image for the purposes of storage, management, or other processing, and these data can also be losslessly extracted. An authorized receiver can recover the confidential image from r encrypted images. By designing, our MSS-RDHEI scheme can withstand $n - r$ points of failure. Experimental results show that it ensures the image content confidentiality and achieves a much larger embedding capacity than state-of-the-art schemes.

Index Terms—Encrypted image, information processing in encrypted domain, multiple data-hiders, reversible data hiding, secret sharing

1 Introduction

Data hiding in images is an active research topic in multimedia security area [1], [2], [3]. Reversible data hiding (RDH) is a data hiding technique that allows user to embed additional messages into cover media [4], [5]. In the last two decades, many RDH schemes have been developed using different techniques such as the difference expansion [6], histogram shifting [7], and prediction error expansion [8]. Recently, due to the fast development of cloud computing, RDH in encrypted images (RDH-EI) has attracted increasing attention. Different from the steganography [3] that hides data into plain-image, RDH-EI techniques have recently emerged to hide secret data into encrypted image and thus can protect both the cover image and secret data [2]. For instance, consider an application of remote medical consultation as shown in Fig. 1. The IoT device generates medical images which are to be outsourced to the cloud for storage and management. For privacy protection, the generated images are first encrypted to be encrypted images by the IoT gateway (e.g., operated by medical center), and then outsourced to the cloud server. The cloud server can embed some data, e.g., time stamps, remarks and copyright data, into the encrypted image for authentication management, copyright protection, or other processing [9], [10], [11], while the embedded data can be completely extracted. The authorized hospital, doctor or patient can recover the medical image using the secret key.

The first RDH-EI scheme was proposed by Puech et al. in 2008 and it uses the Advanced Encryption Standard (AES) to encrypt the original image [12]. Afterwords, many RDH-EI schemes using other encryption strategies were proposed [13]. These early RDH-EI schemes have weaknesses in embedding capacity. To improve this, many new RDH-EI schemes have been developed so far [14], [15], [16], [17] and these works can be divided into two categories: reserving room before encryption (RRBE) [18], [19], [20] and vacating room after encryption (VRAE) [9], [21], [22], [23]. The RRBE strategy vacates room for data embedding before encryption and the room reservation is performed by image owner. For example, Yin et al. proposed an RRBE-based RDH-EI scheme by compressing the prediction errors of pixels in plain-images and the scheme has a high embedding capacity [2]. This strategy can fully exploit the pixel correlations in the plain-image and thus obtain relatively large embedding capacity [24], [25]. However, since the room reservation is performed by the image

- *Zhongyun Hua, Yanxiang Wang, Yifeng Zheng, Xingyu Liu, and Yongyong Chen are with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong 518055, China. E-mail: {huazyum, wyxhitsz, liuxyuh, YongyongChen.cn}@gmail.com, yifeng.zheng@hit.edu.cn.*
- *Shuang Yi is with the Engineering Research Center of Forensic Science, Chongqing Education Committee, College of Criminal Investigation, Southwest University of Political Science and Law, Chongqing 401120, China. E-mail: yishuang@swupl.edu.cn.*
- *Xinpeng Zhang is with the School of Computer Science, Fudan University, Shanghai 200433, China. E-mail: zhangxinpeng@fudan.edu.cn.*

Manuscript received 21 September 2021; revised 26 July 2022; accepted 8 October 2022. Date of publication 1 November 2022; date of current version 1 September 2023.
This work was supported in part by the National Natural Science Foundation of China under Grants 62002301, 62071142, 62106063, and 61936214, in part by Guangdong Basic and Applied Basic Research Foundation under Grants 2021A1515011406, 2021A1515110027 and 2022A1515010819, in part by the Shenzhen College Stability Support Plan under Grants GXWD20201230155427003-20200824210638001 and GXWD20201230155427003-20200824113231001, in part by the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant KJQN202200303, and in part by the University Innovation Research Group project of Chongqing (Evidence Science Technology Innovation and Application) under Grant CXQT21033.
(Corresponding author: Zhongyun Hua.)
Digital Object Identifier no. 10.1109/TDSC.2022.3218570

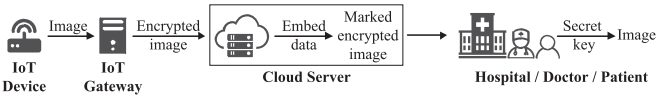

Fig. 1. Secure storage and management of medical images in remote medical consultation.

owner, it may cause heavy computation cost to the image owner, especially for some performance-limited terminal devices. On the other hand, the VRAE strategy directly encrypts the original image. The data are embedded into the encrypted image and the room vacation is performed by data hider. This strategy can greatly reduce the computation cost to the image owner, and is more suitable for cloud services, in which the data hider is the cloud server that has large computation resource. But its embedding capacity is usually smaller than the RRBE strategy.

To retain the pixel redundancy for data embedding in the encrypted domain, most VRAE-based RDH-EI schemes encrypt images using some lightweight encryption strategies such as exclusive-or (XOR) operation [26], [27], and combination of block permutation and block co-modulation [28], [29] (block co-XOR [30], [31]). However, the authors in [32] pointed out that the XOR stream cipher is insecure under ciphertext-only-attack, and the combination of block permutation and block co-modulation is also insecure. To address the security issue, some VRAE-based RDH-EI schemes using homomorphic encryption have been developed [33], [34], [35]. They encrypt the original images using some homomorphic encryption algorithms and the additional data are embedded into the encrypted images using homomorphic property. Since the homomorphic encryption algorithms follow the strict security requirements, the encrypted images can achieve a high security. However, these homomorphic encryption algorithms encrypt data by heavy cryptographic operations, and this causes significant computation cost and data expansion.

These traditional RDH-EI schemes all focus on one data hider and cannot resist the single point of failure. Besides, the image security is determined by one party, rather than multiple parties. To improve the security, recently some VRAE-based RDH-EI schemes using secret sharing techniques have been proposed [36], [37], [38], [39] and some of them can send encrypted images to multiple data hiders. In 2018, Wu et al. designed an RDH-EI scheme using secret sharing technique [36]. It encrypts an original image into multiple encrypted images and sends each encrypted image to one data hider. With a pre-set number of marked encrypted images and the key, a receiver can recover the original image. Afterwards, to improve the embedding capacity, Chen et al. designed an RDH-EI scheme with multiple data hiders using secret sharing technique as well [38]. However, some schemes process each pixel independently and there isn't any random number involved, resulting in limited security [37]. Besides, these schemes don't make full use of pixel redundancy and thus may still result in a relatively low embedding capacity.

In this paper, we propose a new RDH-EI scheme with multiple data hiders using a new secret sharing technique. It is a VRAE-based RDH-EI scheme. First, we devise a new (r, n)-threshold matrix-based secret sharing (MSS) scheme using matrix theory. The MSS can encrypt an original image into n encrypted images and the security is dependent on the secret sharing structure. Using the MSS, we propose an (r, n)-threshold $(r \leq n)$ RDH-EI scheme called MSS-RDHEI, in which a block error mixture encoding (BEME) method is developed to encode images in the encrypted domain. Different from existing RDH-EI schemes [36], [37], [38] that independently share pixels using polynomial operations, the MSS-RDHEI shares pixels using matrix operations with cipher-feedback mechanism. It allows content owner to encrypt an original image to n encrypted images for n data hiders. Each data hider can embed some additional data, e.g., copyright and identification information, into the encrypted image for the purposes of storage, management, or other processing, and the encrypted image with data embedded is called marked encrypted image. Since the developed BEME can fully exploit the high data redundancy, a large room can be vacated for data embedding. An authorized receiver can extract the embedded data of each marked encrypted image, and recover the confidential image from r marked encrypted images using related secret keys. By designing, our MSS-RDHEI scheme can withstand $n - r$ points of failure, and well balance the embedding capacity and image security. Performance evaluations demonstrate that it can protect the image content confidentiality while achieving a much larger embedding capacity than state-of-the-art schemes.

The remainder of the paper is organized as follows. Section 2 introduces the MSS and analyzes its security. Section 3 presents the MSS-RDHEI scheme and Section 4 simulates it and discusses its properties. Section 5 evaluates the performance of the MSS-RDHEI scheme and compares it with existing schemes. Section 6 concludes the paper.

2 MSS

In this section, we devise a new (r, n)-threshold $(r \leq n)$ matrix-based secret sharing (MSS) scheme using matrix theory and the cipher-feedback mechanism of AES.

2.1 Construction of MSS

First, we present Theorem 1 [40] to indicate the sufficient condition of matrix inverse operation in the finite field.

Theorem 1. *For an $r \times r$ square matrix $\mathbf{A}$ and a prime number F, the matrix multiplication*

$$\mathbf{m} = \mathbf{A} \times \mathbf{n} \bmod F$$

is reversible and its inversed operation is

$$\mathbf{n} = \mathbf{A^{-1}} \times \mathbf{m} \bmod F$$

if $det(\mathbf{A})$ is coprime with F.

From Theorem 1, we can deduce that the key point of (r, n)-threshold MSS is to construct an $n \times r$ coefficient matrix, in which the determinant of its any $r \times r$ sub-matrix is coprime with a prime F. Inspired by this, we propose a matrix generation method described as follows.

- *Step 1:* Generate n positive integers $q_1, q_2, \ldots, q_n$ by a pseudorandom number generator (PRNG), where q_k $(1 \leq k \leq n)$ is smaller than a prime number F.

- *Step 2:* Generate n different positive integers $p_1, p_2, \ldots, p_n$ using the PRNG, where p_k $(1 \le k \le n)$ is smaller than F.
- *Step 3:* Initialize an $n \times r$ matrix and its first column consists of the n integers in q_k.
- *Step 4:* The elements in the 2-nd to r-th columns of the matrix are generated by $x(i,j) = x(i,j-1) \times (p_i + j - 2)$.

Algorithm 1 presents the pseudocode of generating the coefficient matrix $\mathbf{X}$ using $2n$ random integers generated by a PRNG.

Algorithm 1. Generation of the Coefficient Matrix

Input: Random integer sequences $\mathbf{p}$ and $\mathbf{q}$, where $\mathbf{p} = \{p_k\}_{k=1}^{n}$ with n different elements and $\mathbf{q} = \{q_k\}_{k=1}^{n}$. Besides, $0 < q_k, p_k < F$, and F is a prime number.

```
Initialize an n × r matrix X ∈ R^{n×r}.
Set the first column of X using {q_1, q_2, ..., q_n}.
for j = 2 to r do
    for i = 1 to n do
        x(i, j) = x(i, j − 1) × (p_i + j − 2).
    end for
end for
```

Output: Coefficient matrix $\mathbf{X}$.

Then we propose Lemma 1 to state that the determinant of any $r \times r$ sub-matrix of the coefficient matrix generated by Algorithm 1 is coprime with F.

Lemma 1. *For the $n \times r$ matrix generated by Algorithm 1, the determinant of any $r \times r$ sub-matrix is coprime with F.*

Proof. Assume that the i_1-th, i_2-th, $\cdots$, i_r-th rows are selected from the $n \times r$ matrix $\mathbf{X}$ generated by Algorithm 1. Then an $r \times r$ sub-matrix $\mathbf{X}_r$ can be constructed as

$$\mathbf{X}_r = \begin{bmatrix} q_{i_1} & q_{i_1}p_{i_1} & \cdots & q_{i_1}p_{i_1}\cdots(p_{i_1}+r-2) \\ q_{i_2} & q_{i_2}p_{i_2} & \cdots & q_{i_2}p_{i_2}\cdots(p_{i_2}+r-2) \\ \vdots & \vdots & \ddots & \vdots \\ q_{i_r} & q_{i_r}p_{i_r} & \cdots & q_{i_r}p_{i_r}\cdots(p_{i_r}+r-2) \end{bmatrix}.$$

Now we calculate its determinant. First, extract all the q_k from every row, and we can get the determinant of $\mathbf{X}_r$ as

$$det(\mathbf{X}_r) = q_{i_1}q_{i_2}\cdots q_{i_r} \times \begin{vmatrix} 1 & p_{i_1} & \cdots & p_{i_1}(p_{i_1}+1)\cdots(p_{i_1}+r-2) \\ 1 & p_{i_2} & \cdots & p_{i_2}(p_{i_2}+1)\cdots(p_{i_2}+r-2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & p_{i_r} & \cdots & p_{i_r}(p_{i_r}+1)\cdots(p_{i_r}+r-2) \end{vmatrix}.$$

Then, for the j-th $(2 \le j \le r)$ column, subtract the multiplication of its previous column with $(p_{i_1} + j - 2)$, the determinant becomes

$$det(\mathbf{X}_r) = q_{i_1}q_{i_2}\cdots q_{i_r} \times \begin{vmatrix} 1 & 0 & \cdots & 0 \\ 1 & p_{i_2}-p_{i_1} & \cdots & p_{i_2}(p_{i_2}+1)\cdots(p_{i_2}+r-3)(p_{i_2}-p_{i_1}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & p_{i_r}-p_{i_1} & \cdots & p_{i_r}(p_{i_r}+1)\cdots(p_{i_r}+r-3)(p_{i_r}-p_{i_1}) \end{vmatrix}.$$

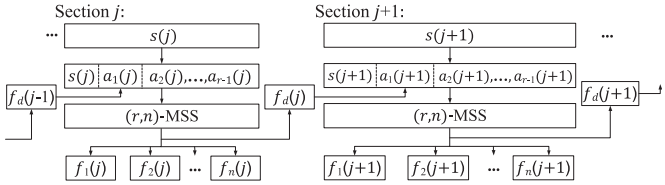

Fig. 2. Schematic structure of the proposed MSS.

Using the Laplace expansion along the first row to the matrix, we obtain that

$$det(\mathbf{X}_r) = q_{i_1}q_{i_2}\cdots q_{i_r} \times \begin{vmatrix} p_{i_2}-p_{i_1} & \cdots & p_{i_2}(p_{i_2}+1)\cdots(p_{i_2}+r-3)(p_{i_2}-p_{i_1}) \\ p_{i_3}-p_{i_1} & \cdots & p_{i_3}(p_{i_3}+1)\cdots(p_{i_3}+r-3)(p_{i_3}-p_{i_1}) \\ \vdots & \ddots & \vdots \\ p_{i_r}-p_{i_1} & \cdots & p_{i_r}(p_{i_r}+1)\cdots(p_{i_r}+r-3)(p_{i_r}-p_{i_1}) \end{vmatrix}.$$

It can be observed that all entries in the h-th $(1 \le h \le r-1)$ row have a factor of $(p_{i_{h+1}} - p_{i_1})$. After extracting these factors, one can obtain that

$$det(\mathbf{X}_r) = \prod_{k=1}^{r} q_{i_k} \prod_{2 \le k \le r} (p_{i_k} - p_{i_1}) \times \begin{vmatrix} 1 & p_{i_2} & \cdots & p_{i_2}(p_{i_2}+1)\cdots(p_{i_2}+r-3) \\ 1 & p_{i_3} & \cdots & p_{i_3}(p_{i_3}+1)\cdots(p_{i_3}+r-3) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & p_{i_r} & \cdots & p_{i_r}(p_{i_r}+1)\cdots(p_{i_r}+r-3) \end{vmatrix}.$$

Repeat the above subtraction, expansion, and factor extraction steps, one can eventually obtain that

$$\begin{aligned} det(\mathbf{X}_r) &= \prod_{k=1}^{r} q_{i_k} \prod_{2 \le k \le r} (p_{i_k} - p_{i_1}) \cdots \prod_{r-1 \le k \le r} (p_{i_k} - p_{i_{r-2}}) \\ &\quad \times \begin{vmatrix} 1 & p_{i_{r-1}} \\ 1 & p_{i_r} \end{vmatrix}. \\ &= \prod_{k=1}^{r} q_{i_k} \prod_{1 \le j < k \le r} (p_{i_k} - p_{i_j}). \end{aligned}$$

According to Algorithm 1, all the parameters satisfy that $0 < q_k, p_k < F, (k = 1, 2, \ldots, n)$ and all p_k are different from each other. Since F is a prime number, all factors of $det(\mathbf{X}_r)$ are coprime with F. Thus, $det(\mathbf{X}_r)$ is coprime with F. Because the i_1-th, i_2-th, $\cdots$, i_r-th rows are any r rows of $\mathbf{X}$, the determinant of any $r \times r$ submatrix is coprime with F. □

Using the coefficient matrix generated by Algorithm 1 and a prime number F, an (r,n)-threshold MSS scheme can be designed and Fig. 2 shows its structure. As can be seen, the vector for the j-th sharing operation is $\mathbf{a} = [s(j), a_1(j), \ldots, a_{r-1}(j)]^T$, in which $s(j)$ is the j-th element of the secret $\mathbf{S}$, $a_2(j), \ldots, a_{r-1}(j)$ are the $r-2$ random values for the current section, and $a_1(j) = f_d(j-1)$ is randomly selected from the previous sharing result $(d \in \{1, 2 \cdots, n\})$ and for the first section $j = 1$, $f_d(0)$ can be any random integer. The generated $\mathbf{f}(j) = [f_1(j), f_2(j), \ldots, f_n(j)]^T$ contains the j-th element of the n share results. With this cipher-feedback mechanism, a tiny change in the secret can cause complete difference in the sharing results even if the $a_2, \ldots, a_{r-1}$ are the same. The

detailed operation of the (r, n)-threshold MSS scheme is described as follows. (1) Generate an $n \times r$ coefficient matrix $\mathbf{X}$ using Algorithm 1; (2) The sharing operation for the j-th element of the secret $\mathbf{S}$ is performed as

$$\mathbf{f}(j) = \mathbf{X} \times \mathbf{a} \bmod F. \tag{1}$$

Then a party P_i holds the share (i, f_i), where i is the identity.

When r shares $(i_1, f_{i_1}), (i_2, f_{i_2}), \ldots, (i_r, f_{i_r})$ have been collected, a same $n \times r$ coefficient matrix $\mathbf{X}$ is first generated using Algorithm 1 with the same random values. Then an $r \times r$ matrix $\mathbf{X}_r$ is constructed using the i_1-th, i_2-th, $\cdots$, i_r-th rows of $\mathbf{X}$. According to Lemma 1, $det(\mathbf{X}_r)$ and F are coprime. According to Theorem 1, the element of the j-th section, $s(j)$, can be recovered by

$$\mathbf{a} = \mathbf{X}_r^{-1} \times \mathbf{f}^{(r)}(j) \bmod F, \tag{2}$$

where $\mathbf{f}^{(r)}(j) = [f_{i_1}(j), f_{i_2}(j), \ldots, f_{i_r}(j)]^{\mathrm{T}}$ is the j-th elements of the r shares.

2.2 Security Analysis

We now analyze the security of the proposed MSS, and discuss its property. First, we give the definition of secrecy of an (r, n)-threshold secret sharing scheme as Definition 1 [41].

Definition 1. *The secrecy of an (r, n)-threshold secret sharing is defined as that any t $(t < r)$ shares with their identities can't reveal anything about the original secret [41].*

From the aspect of information theory, Definition 1 can be described as that an (r, n)-threshold secret sharing over F is secure if for all $s \in \mathbb{Z}_F$, we have $P[S = s] = P[S = s | S_{i_1} = f_{i_1}, S_{i_2} = f_{i_2}, \ldots, S_{i_t} = f_{i_t}]$, where $t < r$ and $f_{i_1}, f_{i_2}, \ldots, f_{i_t}$ are the shares of parties $i_1, i_2, \ldots, i_t$.

The proposed MSS scheme shares secret using matrix multiplication. However, under some conditions, it can also be written as an $(r-1)$-degree polynomial, which has similar property with the Shamir's secret sharing.

Lemma 2. *The MSS scheme over F is secure if $q_1 = q_2 = \cdots = q_n$ and the distributions of a_k $(1 \le k \le r-1)$ are random.*

Proof. Suppose that $q_1 = q_2 = \cdots = q_n = q$. Then an $(r-1)$-degree polynomial can be constructed for the sharing operation as

$$f(X) = \left(s \times q + \sum_{k=1}^{r-1} (a_k \times q \times X \cdots (X + k - 1)) \right) \bmod F. \tag{3}$$

When any r shares with their identities have been collected (e.g. $f_{i_1}, f_{i_2}, \ldots, f_{i_r}$), one can reconstruct the $(r-1)$-degree polynomial f using the Lagrange interpolation as

$$f(X) = \sum_{j=1}^{r} \left(f_{i_j} \times \prod_{0 < k \le r k \ne j} \frac{X - p_{i_k}}{p_{i_j} - p_{i_k}} \right) \bmod F, \tag{4}$$

where $X \in \{p_1, p_2, \ldots, p_n\}$.

Since the distributions of a_k $(1 \le k \le r-1)$ are random, $f(X)$ is a random polynomial that passes the point $(0, s \times q)$. In the finite field $\mathbb{Z}_F$, the total number of $(r-1)$-degree polynomials is F^r. According to the Lagrange interpolation, the number of $(r-1)$-degree polynomials that pass through t different points is F^{r-t} (see Corollary 3.10 on Page 57 of [42]). Thus, the numbers of $(r-1)$-degree polynomials that pass through $t+1$ different points $\{(0, s \times q), (p_{i_1}, f_{i_1}), (p_{i_2}, f_{i_2}), \ldots, (p_{i_t}, f_{i_t})\}$, t different points $\{(p_{i_1}, f_{i_1}), (p_{i_2}, f_{i_2}), \ldots, (p_{i_t}, f_{i_t})\}$, and one point $(0, s \times q)$ are $F^{r-(t+1)}$, F^{r-t} and F^{r-1}, respectively. Then we can obtain

$$\begin{aligned} &P[S_{i_1} = f_{i_1}, S_{i_2} = f_{i_2}, \ldots, S_{i_t} = f_{i_t}] \\ &= P[f(p_{i_1}) = f_{i_1}, f(p_{i_2}) = f_{i_2}, \ldots, f(p_{i_t}) = f_{i_t}] \\ &= \frac{\#\text{polynomials s.t. } f(p_{i_1}) = f_{i_1}, \ldots, f(p_{i_t}) = f_{i_t}}{\text{Total number of polynomials}} \\ &= \frac{F^{r-t}}{F^r} = F^{-t}. \end{aligned} \tag{5}$$

The constant item $s \times q$ can be obtained as $f(0) = s \times q \bmod F$. Since q is coprime with F, the operation $f(0) = s \times q \bmod F$ is reversible and thus the secret s can be obtained when $X = 0$. So $P[f(0) = sq]$ is identical to the probability $P[S = s]$. Then

$$\begin{aligned} &P[S = s, S_{i_1} = f_{i_1}, S_{i_2} = f_{i_2}, \ldots, S_{i_t} = f_{i_t}] \\ &= P[S = s] \frac{P[f(0) = sq, f(p_{i_1}) = f_{i_1}, \ldots, f(p_{i_t}) = f_{i_t}]}{P[f(0) = sq]} \\ &= P[S = s] \frac{\frac{\#\text{polynomials s.t. } f(0)=sq, f(p_{i_1})=f_{i_1}, \ldots, f(p_{i_t})=f_{i_t}}{\text{Total number of polynomials}}}{\frac{\#\text{polynomials s.t. } f(0)=sq}{\text{Total number of polynomials}}} \\ &= P[S = s] \frac{F^{r-(t+1)}}{F^{r-1}} \\ &= P[S = s] F^{-t}. \end{aligned} \tag{6}$$

From Eqs. (5) and (6), we can obtain that

$$\begin{aligned} &P[S = s | S_{i_1} = f_{i_1}, S_{i_2} = f_{i_2}, \ldots, S_{i_t} = f_{i_t}] \\ &= \frac{P[S = s, S_{i_1} = f_{i_1}, S_{i_2} = f_{i_2}, \ldots, S_{i_t} = f_{i_t}]}{P[S_{i_1} = f_{i_1}, S_{i_2} = f_{i_2}, \ldots, S_{i_t} = f_{i_t}]} \\ &= \frac{P[S = s] F^{-t}}{F^{-t}} = P[S = s]. \end{aligned} \tag{7}$$

Thus, nothing about the original secret can be obtained from the t shares with their identities. Then Definition 1 is satisfied and this completes the proof. □

From Lemma 2, the (r, n)-threshold MSS scheme can share a secret S to be n shares. To recover the original secret, at least r shares are required. Assume that the secret S has l elements and an attacker has collected t $(t < r)$ shares. Each section in the original secret contains one element and generates one element for every share. Each element in the shares has F possibilities. Thus, when one has t shares, the possibility of correctly recovering one section is $(1/F)^{r-t}$. Then the probability of correctly recovering the original secret is $(1/F)^{(r-t)\times l}$.

2.3 Comparisons With Prior Secret Sharing Schemes

Some secret sharing schemes based on matrix have been developed and used in image applications. We compare our MSS with these secret sharing schemes from the aspects of

TABLE 1
Comparisons of Different Matrix-Based Secret Sharing Schemes on Image Applications

Methods	Efficacy guarantee	Image format	Lossless recovery	Security guarantee
Yu et al. [43]	No	Any	Yes	No
Ding et al. [44]	No	Any	Yes	Yes
Liu et al. [45]	Yes	Binary	No	Yes
Cheng et al. [46]	Yes	Binary	Yes	Yes
Chen et al. [47]	Yes	Any	Yes	No
Bao et al. [48]	Yes	Any	Yes	No
Our MSS	Yes	Any	Yes	Yes

efficacy guarantee, image format, recovery quality and security, and Table 1 lists the results. For the scheme of Yu et al. [43], it shares an image by multiplying image pixel with a coefficient matrix, and the coefficient matrix is constructed using the linearly independent property of vectors. However, the success of this method depends on probability and does not have theory to guarantee its efficacy. For the (r, n)-threshold scheme, construct an $n \times r$ coefficient matrix $\mathbf{K}$ and check whether $\mathbf{K}$ satisfies the two conditions. 1) Any r row vectors of $\mathbf{K}$ are linearly independent; 2) The determinant of any $r \times r$ submatrix of $\mathbf{K}$ is coprime with 256. If any of the two conditions is unsatisfied, the coefficient matrix construction is unsuccessful and one should repeat the construction process until a coefficient matrix satisfying the two conditions is obtained. In addition, its security hasn't been analyzed and cannot be guaranteed.

The scheme of Ding et al. [44] is very similar with that of Yu et al. [43]. The difference is that the modular coefficient in the scheme of Yu et al. [43] is 256, however, it is a prime integer in the scheme of Ding et al. [44]. Its success also relies on experiment trying and does not have theory to guarantee its efficacy. But it can satisfy the security requirement, because its modular coefficient is a prime integer.

The schemes of Liu et al. [45] and Cheng et al. [46] are visual secret sharing schemes, which can only work on binary images. When using these schemes to share a greyscale image, one should first decompose the image into bit planes, share these bit planes one-by-one, and finally combine the shared bit planes to obtain a greyscale image. Besides, the scheme of Liu et al. [45] is a lossy method.

For the schemes of Chen et al. [47] and Bao et al. [48], the sharing process is to divide an image into several parts according to a generated matrix with elements 0 and 1. These schemes have efficacy guarantee, can be applied to any image format, and can completely recover the original image. However, they directly divide an image into several parts without encrypting image pixels. To protect image confidentiality, they should be combined with a secure stream cipher [48].

For our proposed MSS design, its efficacy and security have theoretical guarantee. Its efficacy property has been theoretically analyzed in the last paragraph of Section 2.1 and its security property has been proved in Lemma 2. It can be applied to image with any format and the sharing process is lossless. In addition, it can preserve some pixel redundancy in the generated secret shares while simultaneously protecting image confidentiality, which will be discussed in Section 4.2.

3 Proposed MSS-RDHEI

In this section, we introduce a new (r, n)-threshold $(r \leq n)$ RDH-EI scheme using the MSS called MSS-RDHEI and Fig. 3 shows its structure. The content owner can encrypt the original image block-by-block to generate n encrypted images. After sending the n encrypted images to n data hiders, a block error mixture encoding (BEME) method is proposed to encode the encrypted images and then each data hider can independently embed additional data into one encrypted image to obtain marked encrypted image. Finally, a receiver who has collected r marked encrypted images and the corresponding keys can losslessly recover the original image and embedded messages.

3.1 Content Owner

The content owner encrypts the original image to generate n encrypted images using an encryption key. The encryption is performed block-by-block. Suppose that the original image $\mathbf{I}$ is of size $M \times N$. First, divide the image into non-overlapping blocks with block size $B \times B$. Then, perform a block-based scrambling to $\mathbf{I}$ to obtain the scrambled image $\mathbf{I}'$. The scrambling operation is based on the encryption key. Afterwards, directly encrypt the scrambled image $\mathbf{I}'$ using the (r, n)-threshold MSS to obtain n image shares. Finally, a post-processing operation is done to these encrypted images to process the overflow information and generate n final encrypted images.

A simple block-based scrambling operation is used here. Since each block in the scrambling operation should have the same size, then $\lfloor (M \times N)/(B \times B) \rfloor$ image blocks are scrambled. Specifically, the scrambled image is obtained by first generating a random number sequence using the encryption key, and then sorting the random number sequence to get an index vector, and finally scrambling the image blocks using the index vector.

3.1.1 (r, n)-Threshold MSS

For an (r, n)-threshold MSS scheme, an $n \times r$ coefficient matrix $\mathbf{X}$ is required in each sharing operation and it is generated using Algorithm 1 with the encryption key. Any pseudorandom number generator (PRNG) can be used to generate the pseudorandom integers for the coefficient matrix $\mathbf{X}$. Here, the enhanced 2D-SLM map presented in [49] is used to generate these pseudorandom integers, since it has a high efficiency. Its definition is shown as

$$\begin{cases} x_{n+1} &= \sin(\pi \hat{a}(\sin(\pi y_n) + \hat{b}) x_n (1 - x_n)) \\ y_{n+1} &= \sin(\pi \hat{a}(\sin(\pi x_{n+1}) + \hat{b}) y_n (1 - y_n)), \end{cases} \tag{8}$$

where $\hat{a}$ and $\hat{b}$ are two control parameters, and the x_n and y_n are two iteration variables. The encryption key is to generate the initial state of the enhanced 2D-SLM map. The prime number in the MSS-RDHEI is set as $F = 257$. Then, n random integers $z_1, z_2, \ldots, z_n$ are generated by

$$z_i = (\lfloor (x_i + y_i) \times 2^{32} \rfloor \bmod 256) + 1, \tag{9}$$

where $\lfloor x \rfloor$ indicates the largest integer that is smaller than or equal to x. Then parameter $\mathbf{p}$ can be obtained as $\mathbf{p} = \{z_1, z_2, \ldots, z_n\}$. Note that when any two elements of $\mathbf{p}$ have

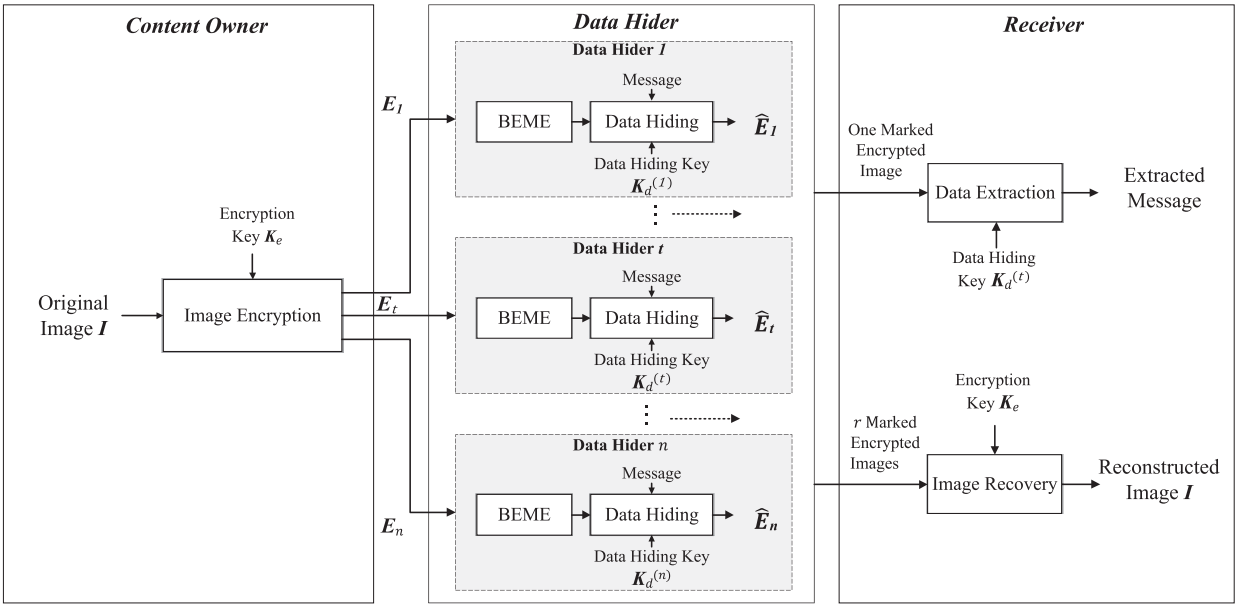

Fig. 3. Overview of the proposed MSS-RDHEI.

the same value, generate another random number using the above Eq. (9) until all the elements in $\mathbf{p}$ are different. Using the parameters $\mathbf{p}$, $\mathbf{q} = \mathbf{1_{n\times 1}}$ and prime number $F = 257$, an $n \times r$ coefficient matrix $\mathbf{X}$ can be generated using Algorithm 1.

The image sharing is performed block-by-block. To preserve redundancy for data hiding in encrypted domain, the coefficient matrix $\mathbf{X}$ for sharing the pixels of each block is the same. Then a total number of $\lceil (M \times N)/(B \times B) \rceil$ coefficient matrix $\mathbf{X}$s are generated using the encryption key. For an image block $\mathbf{IB}$, it is encrypted into n shares with the same size using the constructed coefficient matrix $\mathbf{X}$. For the j-th pixel $\mathbf{IB}(j)$, the (r, n)-threshold sharing operation is $\mathbf{f} = \mathbf{X} \times \mathbf{a} \bmod 257$, which is expressed as

$$\begin{bmatrix} f_1(j) \\ f_2(j) \\ \vdots \\ f_n(j) \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1r} \\ x_{21} & x_{22} & \cdots & x_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nr} \end{bmatrix} \times \begin{bmatrix} \mathbf{IB}(j) \\ a_1(j) \\ \vdots \\ a_{r-1}(j) \end{bmatrix} \bmod 257, \tag{10}$$

where $a_2, a_3, \ldots, a_{r-1}$ are $r-2$ random integers, and $a_1(j) = f_d(j-1)$ is a randomly selected shared result from the previous sharing. When sharing the first section, $f_d(0)$ can be set as any random integer. Notice that the $a_1, a_2, \ldots, a_{r-1}$ should also be the same when sharing all pixels of each block. Note that only the pixels with value 256 are required to be post-processed when setting the modular coefficient as 257. However, the pixels with values 251, 252, 253, 254 and 255 should be pre-processed when setting the modular coefficient as 251. Thus, the modular coefficient in our scheme is set as 257, rather than 251.

After sharing all the $\lceil (M \times N)/(B \times B) \rceil$ blocks, n image shares can be generated. Because the modular coefficient $F = 257$, the elements with value 256 may exist in these image shares. Thus, to get 8-bit grayscale encrypted images, these elements with value 256 should be specially processed.

3.1.2 Post-Processing

Before sending the encrypted images to data hiders, these pixels with value 256 should be processed. Besides, the block size B, parameter r and identity of each encrypted image should be stored in the fixed positions of the encrypted images such that both the data hider and receiver can extract these information from the encrypted images and marked encrypted images. To solve these problems, a post-processing operation is designed as follows.

- *Step 1:* Process the pixels with value 256. For a pixel with value 256, if it is the first pixel, change its value to 255. Otherwise, change its value to be the same as its previous pixel. A bitmap $\mathbf{BM}$ is generated to record the locations of these overflow pixels by:

$$\mathbf{BM} = \begin{cases} \mathbf{BM}||1, & \text{for } \mathbf{E}_t(i,j) = 256; \\ \mathbf{BM}||0, & \text{otherwise.} \end{cases} \tag{11}$$

  where $\mathbf{E}_t(i,j)$ is a pixel of the encrypted image $\mathbf{E}_t$ before post-processing. Since only a few pixels are overflow pixels, the $\mathbf{BM}$ is quite sparse and we compress it using the arithmetic coding. Then the number of '0's in the $\mathbf{BM}$ and the length of the compressed bitmap $\mathbf{CBM}$ should be stored. They are encoded using $\lceil \log_2(MN) \rceil$ bits and $\lceil \log_2(MN) \rceil$ bits, respectively.
- *Step 2:* Three 8-bit streams are used to encode the block size B, parameter r and identity of each encrypted image. We embed these $2 \times \lceil \log_2(MN) \rceil + 24$ bits into the last $2 \times \lceil \log_2(MN) \rceil + 24$ pixels of the encrypted image by least significant bit (LSB) replacement. The original $2 \times \lceil \log_2(MN) \rceil + 24$ LSBs are extracted in advance and combined with the compressed bitmap $\mathbf{CBM}$ to construct the overflow information $\mathbf{O}$.
- *Step 3:* For each encrypted image, embed its $\mathbf{O}$ into it by a simple conventional difference expansion-based

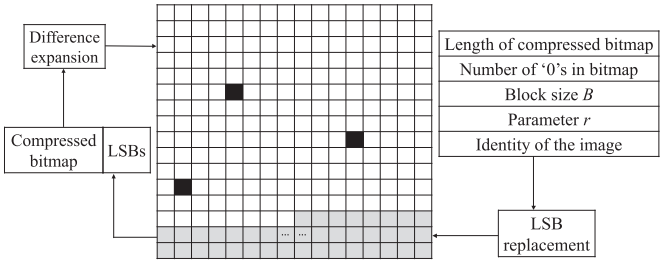

Fig. 4. Illustration of the post-processing.

reversible data hiding [6]. Finally, the n final encrypted images can be generated.

Fig. 4 shows the illustration of the post-processing for each encrypted image. The black pixels indicate the pixels with value 256. First, find their positions to generate a bitmap and change their values within range [0,255]. Then, these side information are embedded into the last pixels by LSB replacement and the original LSBs of these last pixels are combined with the compressed bitmap. Finally, the compressed bitmap and LSBs are embedded to the front pixels of the encrypted image using difference expansion-based reversible data hiding.

Since the pixels with value 256 are only a small proportion of the whole image, about $1/257$ in an encrypted image, the compressed bitmap **CBM** is very short. Besides, when using the same coefficient matrix **X** and parameters in the MSS, the data redundancy in each block can be well preserved. The post-processing has less computation cost and only cause few effects to the embedding performance of the encrypted images. After the post-processing, all pixels in the n encrypted images are constrained within the range $[0, 255]$. Thus, each encrypted image can be saved as an 8-bit grayscale image and sent to a data hider.

3.2 Data Hider

When a data hider receives an encrypted image, he/she can embed additional data into the encrypted image to obtain the marked encrypted image. The data hider first vacates embedding room, and then embeds data into the vacated room. To improve the embedding capacity, a new block error mixture encoding (BEME) method is developed to encode image block. First, the pixel errors of each block are calculated. To better utilize the pixel correlation, the BEME method first divides all the pixel errors into different classes, and then mixedly encodes the bits of pixel errors and their classes.

3.2.1 Errors Calculation

The median edge detector (MED) [50] is an efficient predictor and an improved MED is used here. For an encrypted image, first extract the LSBs of the last 24 pixels to obtain the block size B, parameter r and identity of encrypted image. For an 8-bit image block **IB** with size $B \times B$, the prediction value of its pixel $p(i,j)$ using the improved MED is calculated as

$$\tilde{p}(i,j) = \begin{cases} a, & \text{for } i = 1, j \neq 1; \\ b, & \text{for } i \neq 1, j = 1; \\ \max(b,a), & \text{for } c \leq \min(b,a); \\ \min(b,a), & \text{for } c \geq \max(b,a); \\ b+a-c, & \text{Otherwise} \end{cases} \tag{12}$$

where $a = p(i,j-1)$, $b = p(i-1,j)$ and $c = p(i-1,j-1)$, and they are the adjacent pixels of the current pixel $p(i,j)$ in the left, upper and upper-left directions. Then the error $e(i,j)$ is calculated as

$$e(i,j) = p(i,j) - \tilde{p}(i,j). \tag{13}$$

Then the image block **IB** can be represented by the first pixel and the $B \times B - 1$ pixel errors.

According to the range of $|e(i,j)|$, all the prediction errors can be divided into different classes. Since the $e(i,j)$ may be positive or negative, its sign should be stored and a sign sequence **SN** is used to store the signs of all the prediction errors, which is generated as

$$\mathbf{SN} = \begin{cases} \mathbf{SN}||1, & \text{for } e(i,j) < 0; \\ \mathbf{SN}||0, & \text{for } e(i,j) > 0; \\ \mathbf{SN}, & \text{otherwise.} \end{cases} \tag{14}$$

The sign sequence **SN** has smaller size than the original image block, because the sign of a prediction error doesn't need to be stored if $e(i,j) = 0$. We use $\lceil \log_2(MN) \rceil$ bits to encode the length of the **SN** for all the image blocks and put it at the front of the **SN** of all the image blocks.

For an 8-bit grayscale image, its absolute prediction errors $|e(i,j)|$s are within the range [0,256). After obtaining the prediction errors, one possible strategy is to directly compress the 8 bit planes of these prediction errors. However, this strategy doesn't separate the encodable bits from the un-encodable bits, and thus cannot obtain a high compression efficiency. To obtain a high compression efficiency, we separate these encodable bits and un-encodable bits, and only encode the encodable bits.

3.2.2 BEME

Recently, some prediction error encoding methods have been developed according to the range of the prediction errors [51], [52], [53]. These methods separate a pixel or its prediction error into l most significant bits and $(8-l)$ least significant bits according to the range of the prediction error e, and then encode the l most significant bits to reverse room for data embedding, while remaining the $(8-l)$ least significant bits. The l in the method [51] is calculate as

$$l = \begin{cases} 7, & e = 0; \\ 7-n, & |e| = 2^{n-1}; \\ 8-n, & 2^{n-1} < |e| < 2^n. \end{cases} \quad (1 \leq n \leq 7) \tag{15}$$

The methods in [52], [53] have the same dividing operation and the l is calculated as

$$l = \begin{cases} 8, & e = 0; \\ 7-n, & 2^{n-1} \leq |e| < 2^n. \end{cases} \quad (1 \leq n \leq 7) \tag{16}$$

The encoding method can reserve more room for data embedding if a larger l is obtained from a same prediction error. These previous methods cannot sufficiently separate the encodable bits and un-encodable bits. Here, we propose a new method, called BEME, in which the l is calculated as

$$l = \begin{cases} 8, & e = 0; \\ 9-n, & 2^{n-1} \leq |e| < 2^n. \end{cases} \quad (1 \leq n \leq 8) \tag{17}$$

TABLE 2
Classifications of the Absolute Prediction Errors $|e_{i,j}|$

Class	$\lvert e_{i,j}\rvert$	Class code	Reserved bit(s)
1	$\lvert e_{i,j}\rvert = 0$	0000 0000	*
2	$\lvert e_{i,j}\rvert = 1$	0000 0001	*
3	$\lvert e_{i,j}\rvert \in [2,4)$	0000 001	e_1
4	$\lvert e_{i,j}\rvert \in [4,8)$	0000 01	e_2e_1
5	$\lvert e_{i,j}\rvert \in [8,16)$	0000 1	$e_3e_2e_1$
6	$\lvert e_{i,j}\rvert \in [16,32)$	0001	$e_4e_3e_2e_1$
7	$\lvert e_{i,j}\rvert \in [32,64)$	001	$e_5\ e_4e_3e_2e_1$
8	$\lvert e_{i,j}\rvert \in [64,128)$	01	$e_6e_5\ e_4e_3e_2e_1$
9	$\lvert e_{i,j}\rvert \in [128,256)$	1	$e_7e_6e_5\ c_4e_3e_2e_1$

It is obvious that our method can obtain a larger l than the methods in [51], [52], [53] for a same prediction error. Besides, when $|e| \in [128, 255]$, our BEME method can obtain $l = 1$, however, the methods in [51], [52], [53] cannot work and thus $l = 0$.

In our BEME method, all the $|e(i,j)|$s can be classified into nine classes and Table 2 displays the classification details. For each prediction error $|e(i,j)|$ in an 8-bit gray-scale image block, it can be represented as an 8-bit stream '$e_8e_7e_6e_5\ e_4e_3e_2e_1$,' in which '$e_8$' is the most significant bit and 'e_1' is the least significant bit. As can be seen from Table 2, the class of prediction error only tells the l-most significant bit(s) of a pixel. Thus, the $(8-l)$-least significant bit (s) of the prediction errors should be reserved. Then each prediction error can be represented by eight bits, including the class code of $|e(i,j)|$ and its reserved bit(s). All these bits are represented by eight error class sequences $\{\mathbf{EC}_8, \mathbf{EC}_7, \ldots, \mathbf{EC}_1\}$ and a reserved bit sequence **RB**, which are generated as follows.

- *Step 1:* For the class plane index $l = 8$, when a prediction error satisfies that $|e(i,j)| \in [2^{l-1}, 2^l)$, append '1' to the sequence $\mathbf{EC}_l$. When $|e(i,j)| < 2^{l-1}$, append '0' to the sequence $\mathbf{EC}_l$. Otherwise, keep the sequence $\mathbf{EC}_l$ unchanged.
- *Step 2:* Repeat *Step 1* for $l = 8, 7, \ldots, 1$ to obtain the eight sequences $\{\mathbf{EC}_8, \mathbf{EC}_7, \ldots, \mathbf{EC}_1\}$.
- *Step 3:* Store all the reserved bits in **RB** by

$$\mathbf{RB} = \begin{cases} \mathbf{RB}, & \text{for } |e(i,j)| \le 1; \\ \mathbf{RB}||e_1, & \text{for } |e(i,j)| \in [2,4); \\ \mathbf{RB}||e_2e_1, & \text{for } |e(i,j)| \in [4,8); \\ \mathbf{RB}||e_3e_2e_1, & \text{for } |e(i,j)| \in [8,16); \\ \mathbf{RB}||e_4e_3e_2e_1, & \text{for } |e(i,j)| \in [16,32); \\ \mathbf{RB}||e_5e_4e_3e_2e_1, & \text{for } |e(i,j)| \in [32,64); \\ \mathbf{RB}||e_6e_5e_4e_3e_2e_1, & \text{for } |e(i,j)| \in [64,128); \\ \mathbf{RB}||e_7e_6e_5e_4e_3e_2e_1, & \text{for } |e(i,j)| \in [128,256). \end{cases} \tag{18}$$

Using the sign sequence **SN**, eight error class sequences $\{\mathbf{EC}_8, \mathbf{EC}_7, \cdots, \mathbf{EC}_1\}$, and the reserved bit sequence **RB**, one can completely recover the original image block except for the first pixel.

First, the class code for each pixel is recovered. Specifically, by scanning $\mathbf{EC}_8, \mathbf{EC}_7, \ldots, \mathbf{EC}_1$ sequentially, we can get the class code for each pixel. For each pixel, if a '0' is found in $\mathbf{EC}_8$, we get an element '0' and go on to retrieve an element from $\mathbf{EC}_7$. If the retrieved element is '0,' we get a stream '00' and go on to retrieve an element from the $\mathbf{EC}_6$. Repeat this until an element '1' is retrieved in $\mathbf{EC}_i$. Then the class code of the current pixel is obtained and the class code of the next pixel is also generated beginning from the $\mathbf{EC}_8$. For example, if the element '1' is retrieved in the $\mathbf{EC}_6$, the class code for the current pixel is '001' and its prediction error is $|e(i,j)| \in [32, 64)$.

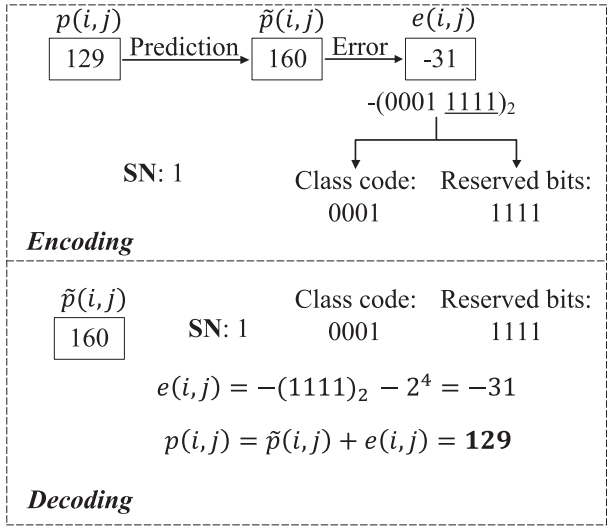

Fig. 5. An illustrative example of the BEME.

After obtaining the class code, the prediction value $\tilde{p}(i,j)$ of the pixel is calculated. Using the class code and prediction value $\tilde{p}(i,j)$, the decoding process can be classified as the following situations.

- For $|e(i,j)| = 0$, the pixel value $p(i,j) = \tilde{p}(i,j)$.
- For $|e(i,j)| \in [2^l, 2^{l+1})$ $(l = 0, 1, \ldots, 7)$, l bits are retrieved from the reserved bit sequence **RB** and these l bits are the l-least significant bits of the prediction error $(e_l \cdots e_1)_2$. The sign of the current $e(i,j)$ is retrieved from sign sequence **SN**. If the sign is 0, which indicates that the $e(i,j)$ is positive, the prediction error can be obtained as $e(i,j) = (e_l \cdots e_1)_2 + 2^l$. If the sign is 1, which indicates that the $e(i,j)$ is negative, the prediction error can be obtained as $e(i,j) = -(e_l \cdots e_1)_2 - 2^l$. Finally, the pixel value $p(i,j) = \tilde{p}(i,j) + e(i,j)$.

To better demonstrate the pixel decoding process, we provide a numeral example and shows the details in Fig. 5. Suppose that $p(i,j) = 129$ and its prediction value is $\tilde{p}(i,j) = 160$. Then its prediction error is $e(i,j) = -31$, namely $-(00011111)_2$. Thus, the current sign $\mathbf{SN}(k) = 1$, the class code is '0001' and the 4-least significant bits '1111' of the prediction error are reserved. In the decoding phase, we first calculate its prediction value $\tilde{p}(i,j) = 160$, and obtain the current sign $\mathbf{SN}(k) = 1$ and the class code of its prediction error '0001'. According to Table 2, we can get that its prediction error $|e(i,j)| \in [2^4, 2^5)$, and the 4-least significant bits of the prediction error are retried as $(1111)_2$. Then the prediction error $e(i,j) = -(1111)_2 - 2^4 = -31$. Then we calculate the pixel value

$$p(i,j) = \tilde{p}(i,j) + e(i,j) = 160 - 31 = 129.$$

After encoding all the $\lceil (M \times N)/(B \times B) \rceil$ blocks, the eight error class sequences $\{\mathbf{EC}_8, \mathbf{EC}_7, \cdots, \mathbf{EC}_1\}$ and the reserved bit sequence **RB** for each block can be generated. Combine the error class sequences of all image blocks to obtain eight error class sequences $\{\mathbf{C}_8, \mathbf{C}_7, \cdots, \mathbf{C}_1\}$ for the whole image. Then combine the reserved bit sequence of all image blocks to obtain the whole reserved bits **RB**. Since a natural image has high pixel correlations, the elements in $\{\mathbf{C}_8, \mathbf{C}_7, \ldots, \mathbf{C}_1\}$ are sparse. Thus, we compress these eight sequences using the arithmetic coding to obtain $\{\mathbf{CC}_8, \mathbf{CC}_7, \ldots, \mathbf{CC}_1\}$.

To totally recover the original bit sequences, the side information **SI** should be stored and it contains the numbers of '0' in $\mathbf{C}_i$, the lengths of $\mathbf{CC}_i$, the length of **RB**, and the sign sequence **SN** of all the image blocks. The numbers of '0' in the eight sequences $\mathbf{C}_i$ are represented using eight integers $\{L_8, L_7, \ldots, L_1\}$, and each integer can be encoded using $\lceil \log_2(MN) \rceil$ bits. The lengths of the eight compressed sequences $\mathbf{CC}_i$ are represented by eight integers $\{LC_8, LC_7, \ldots, LC_1\}$, and each integer is encoded using $\lceil \log_2(MN) \rceil$ bits. The length of the reserved bit sequence **RB** is represented by L_B and it is encoded using $\lceil \log_2(8MN) \rceil = 3 + \lceil \log_2(MN)) \rceil$ bits.

Finally, the block size B, parameter r and identity of encrypted image are embedded into the last 24 pixels by LSB replacement. The **SI**, eight compressed error class sequences $\{\mathbf{CC}_8, \mathbf{CC}_7, \ldots, \mathbf{CC}_1\}$ and reserved bit sequence **RB** should be embedded into each image block except for the first pixel.

3.2.3 Data Hiding

The embedding space is the pixels of all the image block except for the first pixel. From the side information **SI** of the whole image, the data hider can find out the position of the first available space, and can thus embed the messages into these space. To enhance the security level, the additional data to be embedded are first encrypted using an existing cryptographic algorithm (e.g. AES) with a data hiding key $\boldsymbol{K}_d$.

3.3 Receiver

A receiver with r marked encrypted images can reconstruct the original image. According to whether the receiver has the encryption key used in the image encryption phase and the data hiding key used in the data embedding phase, he/she can reconstruct the original image and the embedded data, respectively.

3.3.1 Data Extraction With Data Hiding Key

For each marked encrypted image, if the receiver has the related data hiding key, he/she can extract the embedded data. The receiver should also first extract the block size B from the LSB of the last 8 pixels, and then find out the position of the first available space in the marked encrypted image. After finding out this position, he/she can extract all the embedded data from every block except for the first pixel. Finally, the additional data can be obtained by decrypting the extracted data using the corresponding data hiding key.

3.3.2 Image Reconstruction With Encryption Key

A receiver collecting r marked encrypted images and the encryption key can recover the original image using the following steps.

- *Step 1:* For each marked encrypted image, obtain the block size B, parameter r and the identity of the current marked encrypted image from the LSB of the last 24 pixels. Then for each block, all the pixels except for the first pixel are extracted, and then extract the side information **SI**, eight compressed sequences $\{\mathbf{CC}_8, \mathbf{CC}_7, \ldots, \mathbf{CC}_1\}$ and the reserved bit sequence **RB** for the encrypted image.
- *Step 2:* From the side information **SI**, we can extract $\{L_8, L_7, \ldots, L_1\}$, $\{LC_8, LC_7, \ldots, LC_1\}$, L_B, and the sign sequence **SN**.
- *Step 3:* From the $\{L_8, L_7, \ldots, L_1\}$ and $\{\mathbf{CC}_8, \mathbf{CC}_7, \ldots, \mathbf{CC}_1\}$, the eight error class sequences $\{\mathbf{C}_8, \mathbf{C}_7, \ldots, \mathbf{C}_1\}$ can be obtained using the decoding process of arithmetic coding. The $\{\mathbf{C}_8, \mathbf{C}_7, \ldots, \mathbf{C}_1\}$ contain the eight error class sequences $\{\mathbf{EC}_8, \mathbf{EC}_7, \cdots, \mathbf{EC}_1\}$ of all the image blocks, while the **RB** contains the reserved bits **RB** of all the image blocks.
- *Step 4:* Recover the encrypted image $\mathbf{E}_t$ by performing the inversed BEME to each image block using the sign sequence **SN**, error class sequences $\{\mathbf{EC}_8, \mathbf{EC}_7, \ldots, \mathbf{EC}_1\}$ and reserved bit sequence **RB**.
- *Step 5:* Suppose that the identities of the r collected marked encrypted images are $i_1, i_2, \ldots, i_r$. Perform *Step 1* and *Step 4* to the r marked encrypted images to obtain the r encrypted image $\mathbf{E}_{i_1}, \mathbf{E}_{i_2}, \ldots, \mathbf{E}_{i_r}$. Then perform the inversed operation of the post-processing in Section 3.1.2 to recover the original r encrypted images that may exist value 256.
- *Step 6:* Generate the same coefficient matrix **X** using Algorithm 1 with the encryption key for each image block. Then an $r \times r$ matrix $\mathbf{X}_r$ is constructed using the i_1-th, i_2-th, $\cdots$, i_r-th rows of **X**. The inverse of matrix is

$$\mathbf{X}_r^{-1} = \frac{\mathbf{X}_r^*}{det(\mathbf{X}_r)} \bmod 257, \tag{19}$$

  where $\mathbf{X}_r^*$ is the adjoint matrix of $\mathbf{X}_r$.
- *Step 7:* For the k-th image block of the scrambled image, its j-th pixel can be recovered using the j-th pixels of the k-th image block of the r encrypted images, namely $f_{i_1}(j), f_{i_2}(j), \ldots, f_{i_r}(j)$. The reconstruction process is presented as

$$\begin{bmatrix} \mathbf{IB}(j) \\ a_1(j) \\ \vdots \\ a_{r-1}(j) \end{bmatrix} = \mathbf{X_r^{-1}} \times \begin{bmatrix} f_{i_1}(j) \\ f_{i_2}(j) \\ \vdots \\ f_{i_r}(j) \end{bmatrix} \bmod 257, \tag{20}$$

  Once all pixels have been recovered, the k-th image block can be obtained. After all the image blocks have been recovered, the scrambled image $\mathbf{I}'$ is obtained.
- *Step 8:* Perform the inversed scrambling operation using the same encryption key, and the original image **I** can be recovered.

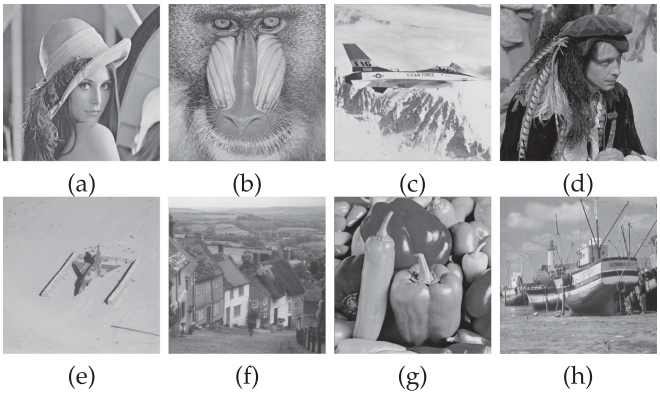

Fig. 6. Eight test images: (a) *Lena*; (b) *Baboon*; (c) *Jetplane*; (d) *Man*; (e) *Airplane*; (f) *Goldhill*; (g) *Peppers*; (h) *Boat*.

When a receiver has r marked encrypted images and the data hiding and encryption keys, he/she can simultaneously recover the embedded data and original image.

4 Simulation Results

This section simulates the MSS-RDHEI and analyzes the data hiding performance in the encrypted images. Eight standard grayscale images with size 512×512 are used as the test images and they are shown in Fig. 6. The embedded data are randomly generated binary bits and encrypted by AES before embedding.

4.1 Experimental Results

These are three parameters in our proposed (r, n)-threshold MSS-RDHEI scheme, namely the parameters r, n and block size B. The n and r indicate the total number of encrypted images and required number of encrypted images to recover the original image. The block size B determines the visual effect of the encrypted images and the data embedding performance of the encrypted images.

Since the settings of r and n have few influences to the performance of each encrypted image, our experiment investigates the MSS-RDHEI scheme under different block sizes. Figs. 7, 8, and 9 show the simulation results of the (3,3)-threshold MSS-RDHEI scheme under block size $B = 2$, $B = 4$ and $B = 8$, respectively. As can be seen, the MSS-RDHEI scheme can encrypt meaningful images into unrecognized encrypted images. Since the encryption is block-by-block, some data redundancy still exists within each block. Thus, the visual effect is better with smaller block size, which can be seen from the encrypted results in Figs. 7, 8, and 9. However, even if block artifact exists when the block size is large, all the pixels in the encrypted images can still be distributed uniformly, which can be seen from their histograms. Besides, Table 3 lists the Shannon entropies of these encrypted images in Figs. 7, 8, and 9. As can be seen, all the encrypted images encrypted by our MSS-RDHEI with different block sizes can achieve very large Shannon entries, which are very close the theoretical maximum value 8. This further indicates the uniform distribution of these encrypted images. When embedding additional encrypted data into these encrypted images, the obtained marked encrypted images have good visual effects.

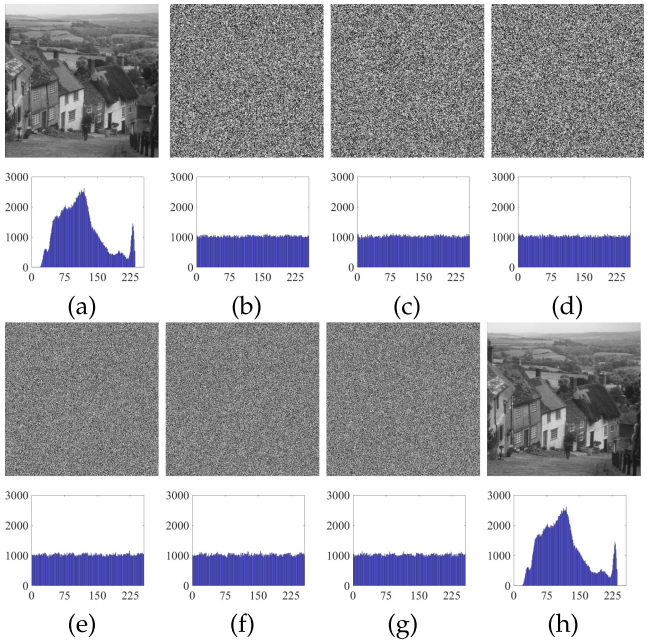

Fig. 7. Simulation results of the (3,3)-threshold MSS-RDHEI with block size $B = 2$: (a) original image *Goldhill*; (b)-(d) three encrypted images; (e)-(g) three marked encrypted images with data embedded; (h) reconstructed image *Goldhill* from the marked encrypted images (e), (f), and (g).

4.2 Redundancy Preservation Within Block

First, we analyze that some data redundancy remains within each block in the encrypted images when the sharing parameters are kept same. When sharing an image, we use $\mathbf{q} = \mathbf{1_{n\times 1}}$ to generate the coefficient matrix $\mathbf{X}$. Then the sharing operation for the j-th pixel of the k-th block $\mathbf{IB}(j)$ described in Eq. (10) becomes

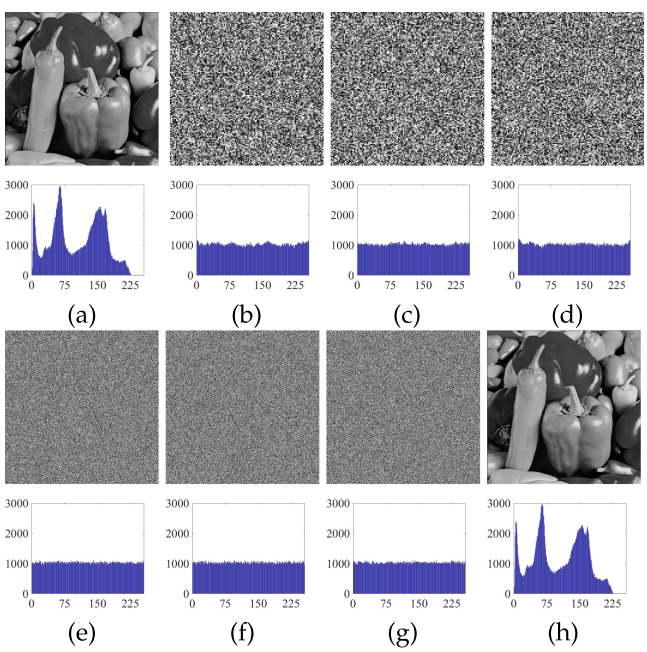

Fig. 8. Simulation results of the (3,3)-threshold MSS-RDHEI with block size $B = 4$: (a) original image *Peppers*; (b)-(d) three encrypted images; (e)-(g) three marked encrypted images with data embedded; (h) reconstructed image *Peppers* from the marked encrypted images (e), (f), and (g).

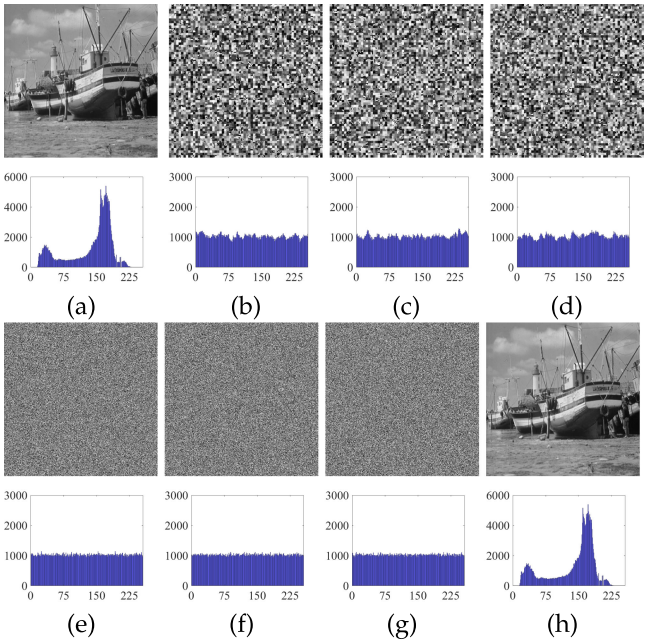

Fig. 9. Simulation results of the (3,3)-threshold MSS-RDHEI with block size $B = 8$: (a) original image *Boat*; (b)-(d) three encrypted images; (e)-(g) three marked encrypted images with data embedded; (h) reconstructed image *Boat* from the marked encrypted images (e), (f), and (g).

$$\begin{bmatrix} f_1(j) \\ f_2(j) \\ \vdots \\ f_n(j) \end{bmatrix} = \begin{bmatrix} 1 & x_{12} & \cdots & x_{1r} \\ 1 & x_{22} & \cdots & x_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n2} & \cdots & x_{nr} \end{bmatrix} \times \begin{bmatrix} \mathbf{IB}(j) \\ a_1 \\ \vdots \\ a_{r-1} \end{bmatrix} \bmod 257. \quad (21)$$

Suppose that $A_i = \sum_{p=2}^{r}(x_{ip} \times a_{p-1}) \bmod 257$. Then the generated shares can be rewritten as

$$f_i(j) = A_i + \mathbf{IB}(j) \bmod 257, \quad (22)$$

where $i = 1, 2, \ldots, n$. For two pixels $\mathbf{IB}(j_1)$ and $\mathbf{IB}(j_2)$ of the k-th block, their i-th shares using the same parameters are expressed as

$$\begin{cases} f_i(j_1) = \mathbf{IB}(j_1) + A_i - \alpha \times 257, \\ f_i(j_2) = \mathbf{IB}(j_2) + A_i - \beta \times 257, \end{cases} \quad (23)$$

where α and β are the two integers. Since $f_i(j_1), A_i, \mathbf{IB}(j_1) \in [0, 256]$, $\alpha = 0$ or 1. Similarity, it is also that $\beta = 0$ or 1. Then, the difference between $f_i(j_1)$ and $f_i(j_2)$ is given by:

$$\Delta f_i = \begin{cases} \mathbf{IB}(j_1) - \mathbf{IB}(j_2), & \text{for } \alpha = \beta; \\ \mathbf{IB}(j_1) - \mathbf{IB}(j_2) - 257, & \text{for } \alpha = 1, \beta = 0; \\ \mathbf{IB}(j_1) - \mathbf{IB}(j_2) + 257, & \text{for } \alpha = 0, \beta = 1. \end{cases} \quad (24)$$

Owing to the high pixel correlation within original image blocks, $\alpha = \beta$ is highly possible. When $\alpha = \beta$, $\Delta f_i = \mathbf{IB}(j_1) - \mathbf{IB}(j_2)$, which means that the difference of the two pixels in the i-th share is equal to the difference of the two pixels in the original image. Then the high pixel correlation within each original image block can be well-preserved and our BEME method can vacate room for data embedding using the pixel redundancy caused by the pixel correlation. For each generated share, the pixel redundancy only exists within each block and the pixels of different blocks have no pixel redundancy, since the sharing parameters for different blocks are completely different.

TABLE 3
Shannon Entropies of the Encrypted Images in Figs. 7, 8 and 9

Images	B	Original image	Encrypted image 1	Encrypted image 2	Encrypted image 3
Goldhill	2	7.4777	7.9993	7.9992	7.9991
Peppers	4	7.5715	7.9986	7.9983	7.9985
Boat	8	7.1237	7.9960	7.9952	7.9972

Our MSS-RDHEI scheme can well balance the trade-off between image confidentiality and embedding capacity. The space of the brute-force attack indeed reduces, due to the inner-pixel redundancy in each block. However, with small block size, the encrypted images can still achieve a large space for brute-force attack, which is detailed discussed in Section 5.3. Besides, we encrypt the pixels from different blocks using different and random parameters. Then the obtained results are random and do not retain any information of the original pixels. We have theoretically discussed and proved this security property in Lemma 2 of Section 2.2. Thus, our MSS-RDHEI scheme can well protect the image contents, compared with existing VRAE-based RDH-EI schemes using lightweight encryption strategies.

4.3 Data Embedding Performance

The data embedding performance of the encrypted image is another important indicator of an RDH-EI scheme. In our MSS-RDHEI scheme, the embedding performance is determined by the developed BEME encoding method and block size. This section investigates the embedding performance of our MSS-RDHEI scheme with different block sizes.

The effective embedding capacity (EC) indicates the maximum bit number that can be embedded into a cover image by a data hider. It can be evaluated using the embedding rate as bits per pixel (*bpp*), which is calculated as

$$\text{ER} = \frac{\text{Total embedding capacity - Payload}}{\text{Total number of pixels}}. \quad (25)$$

The total embeddable capacity is the capacity that vacated by the BEME method in the encrypted domain, which indicates that the total number of image bits minus the left bits after BEME. The payload includes the side information **SI** of the encrypted image, and 24 bits to encode the parameters B, r and the identity of the encrypted image.

An RDH-EI scheme is expected to have a larger embedding rate so that a cover image can be embedded more additional data. There are three parameters in our (r, n)-threshold MSS-RDHEI scheme, namely the parameters r, n and the block size B. The proposed MSS encrypts an image to be n encrypted images with the same size. Then the parameters r and n have slight influences to the EC. Since the proposed BEME method encodes the encrypted image block-by-block, the EC is greatly affected by the block size B. Our proposed BEME method can make full use of the pixel redundancy, and it thus can achieve a high encoding efficiency and thus

TABLE 4
Embedding Rates of the (3,3)-Threshold MSS-RDHEI for Eight Test Images of Size 512×512 Under Different Block Sizes B

Images	Encrypted images	Total EC	Payload	ER (*bpp*)	Total EC	Payload	ER (*bpp*)	Total EC	Payload	ER (*bpp*)
			$B = 2$			$B = 4$			$B = 8$	
Lena	1^{st}	746775	176622	2.1750	957709	220205	2.8134	1020096	230610	3.0117
	2^{nd}	748146	176598	2.1803	958234	220192	2.8154	1022174	230550	3.0198
	3^{rd}	746518	176634	2.1739	957885	220282	2.8137	1023792	230608	3.0258
Baboon	1^{st}	384841	190977	0.7395	502041	238371	1.0058	538411	250188	1.0995
	2^{nd}	386083	190990	0.7442	501496	238363	1.0038	536863	250236	1.0934
	3^{rd}	386266	190987	0.7449	503176	238425	1.0099	539559	250253	1.1036
Jetplane	1^{st}	799240	166082	2.4153	1034904	206182	3.1613	1103380	215693	3.3863
	2^{nd}	800937	166047	2.4219	1034759	206073	3.1612	1109645	215441	3.4111
	3^{rd}	801422	166083	2.4236	1036994	206159	3.1694	1107678	215445	3.4036
Man	1^{st}	607812	181633	1.6257	782459	226772	2.1198	835798	237972	2.2805
	2^{nd}	608084	181666	1.6267	785402	226794	2.1309	834430	238007	2.2752
	3^{rd}	607726	181685	1.6252	783473	226752	2.1237	832607	238017	2.2682
Airplane	1^{st}	1001166	122145	3.3532	1257350	153179	4.2121	1328004	160631	4.4532
	2^{nd}	993267	122666	3.3211	1258010	152996	4.2153	1316669	161767	4.4056
	3^{rd}	997740	122214	3.3399	1253769	153338	4.1978	1323549	160938	4.4350
Goldhill	1^{st}	665444	183413	1.8388	864846	228262	2.4284	926126	239518	2.6192
	2^{nd}	667345	183412	1.8461	862966	228392	2.4207	925922	239445	2.6187
	3^{rd}	666275	183440	1.8419	863298	228299	2.4223	924804	239425	2.6145
Peppers	1^{st}	707477	181883	2.0050	898190	227453	2.5587	952015	238843	2.7205
	2^{nd}	707833	181887	2.0063	901830	227463	2.5725	950474	238770	2.7149
	3^{rd}	708506	181908	2.0088	901566	227506	2.5713	954882	238854	2.7314
Boat	1^{st}	711913	178186	2.0360	925809	221902	2.6852	990160	232620	2.8898
	2^{nd}	712325	178157	2.0377	925166	221806	2.6831	993051	232676	2.9006
	3^{rd}	713524	178093	2.0425	923992	221901	2.6783	992337	232673	2.8979

vacates a large embedding capacity. Table 4 lists the embedding rates of the (3,3)-threshold MSS-RDHEI scheme for different images under block size $B = 2$, $B = 4$ and $B = 8$, respectively. It is obvious that the embedding rates significantly improve with the increment of block size B. Besides, an image with higher smoothness usually has higher encoding efficiency, which means a larger embedding rate. Obviously, the image *Airplane* has much larger embedding rates than other images with the same block size. Fig. 10 shows the embedding rates of the eight test images with the increment of block size. When the block size is small, the embedding rates quickly increase with the increment of the block size. However, when the block size increases to be large, the embedding rates stay almost the same. One has great flexibility to select a proper block size according to different situations. It is obvious that different images have different embedding rates under the same block size. This is because the embedding capacity is vacated by compressing the pixel redundancy in each block. An image can achieve larger embedding rates if it has more pixel redundancy. Usually, a smooth image has more pixel redundancy. Thus, the smoothness of an image highly determines its final embedding rates. We only list the embedding rates of our MSS-RDHEI scheme under (3,3)-threshold. This is because the settings of r and n cause slight change to the embedding rates. With different (r, n)-thresholds, our MSS-RDHEI can achieve almost the same embedding rates.

5 Performance Analysis

This section analyzes the performance of our MSS-RDHEI scheme and compares it with other secret sharing-based RDH-EI schemes.

5.1 Data Expansion

In an RDH-EI scheme, data expansion happens when the total size of the marked encrypted image is larger than that of the original image [36]. The expansion rate is introduced

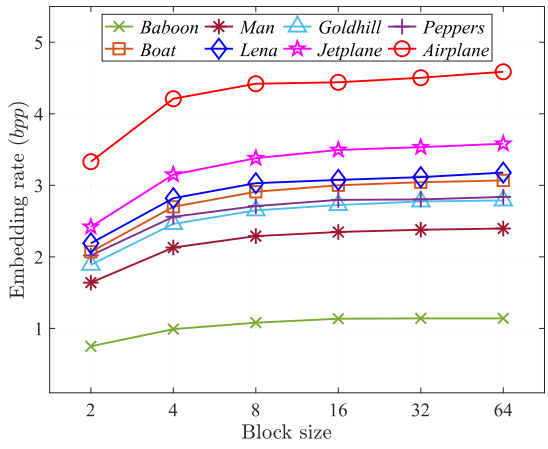

Fig. 10. Embedding rates of eight test images under different block sizes.

TABLE 5
Data Expansion Rates of the Homomorphic Encryption-Based and Secret Sharing-Based RDH-EI Schemes

Methods	Total	Each data hider
Ke et al. [33]	256	256
Chen et al. [34]	128	128
Li et al. [35]	128	128
Wu et al. [36]	n	1
Chen et al. [37]	n	1
Chen et al. [38]	n	1
Qin et al. [39]	n	1
MSS-RDHEI	n	1

to quantitatively measure the data expansion and it is defined as the ratio between the total bits of the marked encrypted image and the total bits of the original image. Here, we only compare the data expansion of homomorphic encryption-based and secret sharing-based RDH-EI schemes, because these two types of schemes are both vacating room after encryption (VRAE) methods. Most traditional VRAE methods do not cause data expansion. However, they have limited security level [32], [54], since they usually use some light-weight encryption methods to keep data redundant in the encrypted domain.

Our (r, n)-threshold MSS-RDHEI encrypts an original image into n encrypted images and each one owns the same size with the original image. These encrypted images are individually sent to n different data hiders for data embedding. Table 5 lists the data expansion comparison of homomorphic encryption-based and secret sharing-based RDH-EI schemes. It can be seen that data expansion occurs seriously in these schemes that apply homomorphic encryption [33], [34], [35]. Compared to the homomorphic encryption-based schemes, the secret sharing-based schemes in [36], [37], [38], [39] and the proposed MSS-RDHEI scheme have much less expansion rates and their expansion rates are acceptable. For all the secret sharing-based schemes, their expansion rate is n for the whole scheme. However, for each data hider, the expansion rate is 1 and there is no data expansion. Note that for the scheme in [37], there is only one encrypted image, which means that $n = 1$.

5.2 Embedding Capacity

Since the used MED predictor has high performance and the developed BEME is a high-efficiency encoding method, our MSS-RDHEI can achieve very high embedding rates. Firstly, we design experiments to test the effect of our used MED predictor. Specifically, we compare the embedding rates of our MSS-RDHEI scheme using different predictors including the MED predictor used in our scheme, gradient-adjusted prediction (GAP) [31], chess-board prediction (CBP) (also called rhombus predictor) [31], local difference prediction (LDP) [53], L predictor (LP) [53] and adaptive L predictor (ALP) [53]. Table 6 shows the test results of (2,2)-threshold under different images with block size $B = 4$. We use the same experiment settings for all the predictors to provide a fair comparison. It shows that our MSS-RDHEI scheme has the smallest embedding rates using the ALP, because a coefficient should be stored for each block in the ALP. Our MSS-RDHEI can achieve the largest embedding rates when using the MED predictor. This is because the prediction operations are performed block-by-block, and the MED predictor needs less edge pixels, compared to other predictors.

TABLE 6
Embedding Rates of Our MSS-RDHEI Scheme Using Different Predictors With (2,2)-Threshold

Images		Predictors					
		GAP	CBP	LDP	LP	ALP	MED
Lena	1^{st}	2.7164	2.6439	1.9324	2.7439	1.6492	**2.8093**
	2^{nd}	2.7319	2.6700	1.9529	2.7642	1.6605	**2.8275**
Baboon	1^{st}	0.9500	0.8194	0.4711	0.9069	0.1536	**1.0094**
	2^{nd}	0.9495	0.8206	0.4680	0.9086	0.1479	**1.0093**
Jetplane	1^{st}	3.0029	2.8869	2.1336	3.0017	1.8287	**3.1591**
	2^{nd}	3.0062	2.8977	2.1408	3.0098	1.8308	**3.1648**
Man	1^{st}	2.0392	1.9099	1.3053	2.0381	1.1163	**2.1249**
	2^{nd}	2.0450	1.9252	1.3154	2.0514	1.1195	**2.1331**
Airplane	1^{st}	4.1162	3.7573	3.7226	3.8736	2.5842	**4.2097**
	2^{nd}	4.1254	3.7733	3.7324	3.8858	2.5933	**4.2176**
Goldhill	1^{st}	2.2769	2.1354	1.5435	2.2736	1.2782	**2.4250**
	2^{nd}	2.2761	2.1351	1.5476	2.2746	1.2853	**2.4259**
Peppers	1^{st}	2.5498	2.5220	1.9052	2.5304	1.5098	**2.5738**
	2^{nd}	2.5437	2.5107	1.8950	2.5225	1.5163	**2.5644**
Boat	1^{st}	2.5172	2.3484	1.7621	2.5185	1.5018	**2.6785**
	2^{nd}	2.5191	2.3555	1.7663	2.5230	1.5139	**2.6822**

Then we test the effect of our developed BEME by comparing the embedding rates of our MSS-RDHEI scheme using different encoding methods including our developed BEME method, the Yu et al. [51] and Mohammadi et al. [52], [53] encoding methods. Table 7 shows the test results of (2,2)-threshold under different images. To provide a fair comparison, we use the same experiment settings for all the encoding methods. It shows that our MSS-RDHEI can achieve the largest embedding rates when using our BEME method. This is because our BEME method can better utilize the data redundancy of the prediction errors, which has been theoretically discussed in Section 3.2.2.

Finally, we compare our MSS-RDHEI scheme with existing secret sharing-based RDH-EI schemes. The block size in our MSS-RDHEI scheme is set as 4. The embedding rate of the scheme of Chen et al. [38] is $7/n$. Thus, its embedding rate reduces when n increases and is not applicable when $n > 7$. The scheme of Chen et al. [37] is not applicable when r is odd. For the scheme of Wu et al. [36], it contains two methods with different embedding rates. We use the larger one as the competing method. The scheme of Qin et al. [39] also contains two methods and the one with larger embedding rate is used as the competing method as well. Fig. 11 shows the embedding rates of different secret sharing-based RDH-EI schemes with different settings of r and n. The schemes of Chen et al. [37], Wu et al. [36], Qin et al. [39] and our MSS-RDHEI can achieve almost the same embedding rates with different settings of r and n. Since the embedding rate of the scheme of Chen et al. [38] is $7/n$, which is determined only by the parameter n. It can

TABLE 7
Embedding Rates of Our MSS-RDHEI Scheme Using Different Encoding Methods With (2,2)-Threshold

Images	Yu et al. [51]		Mohammadi et al. [52], [53]		Our BEME	
	1^{st}	2^{nd}	1^{st}	2^{nd}	1^{st}	2^{nd}
Lena	2.4745	2.4826	2.4101	2.4123	**2.8119**	**2.8204**
Baboon	0.7931	0.7932	0.8415	0.8457	**1.0082**	**1.0088**
Jetplane	2.6575	2.6578	2.7421	2.7486	**3.1616**	**3.1619**
Man	1.7772	1.7822	1.7267	1.7304	**2.1221**	**2.1275**
Airplane	3.7034	3.6817	3.2913	3.2826	**4.2151**	**4.1957**
Goldhill	2.0898	2.1052	1.9293	1.9377	**2.4218**	**2.4372**
Peppers	2.2536	2.2463	2.1665	2.1563	**2.5758**	**2.5690**
Boat	2.3162	2.3092	2.2653	2.2641	**2.6792**	**2.6722**

achieve the largest embedding rates for most images when $n = 2$. With the increment of n, our MSS-RDHEI scheme can achieve the largest embedding rates than other secret sharing based-methods for most images. Thus, our MSS-RDHEI scheme shows a good performance in embedding capacity.

5.3 Security Evaluation

Since the embedded data in many RDH-EI schemes are encrypted using some encryption standards such as the AES and DES before being embedded, they can well protect the embedded data. However, most existing RDH-EI schemes cannot well protect the original images. In our MSS-RDHEI scheme, the image is encrypted by the MSS scheme, whose security is dependent on the proposed MSS scheme. As analyzed in Section 2.2, the MSS scheme is secure. Thus, the image contents can be well protected. In this section, we further statistically evaluate the security of the MSS-RDHEI from several aspects of protecting image.

When sharing an image, the pixels are encrypted block-by-block and each block uses the same sharing parameters. Thus, our scheme can preserve some pixel redundancy of

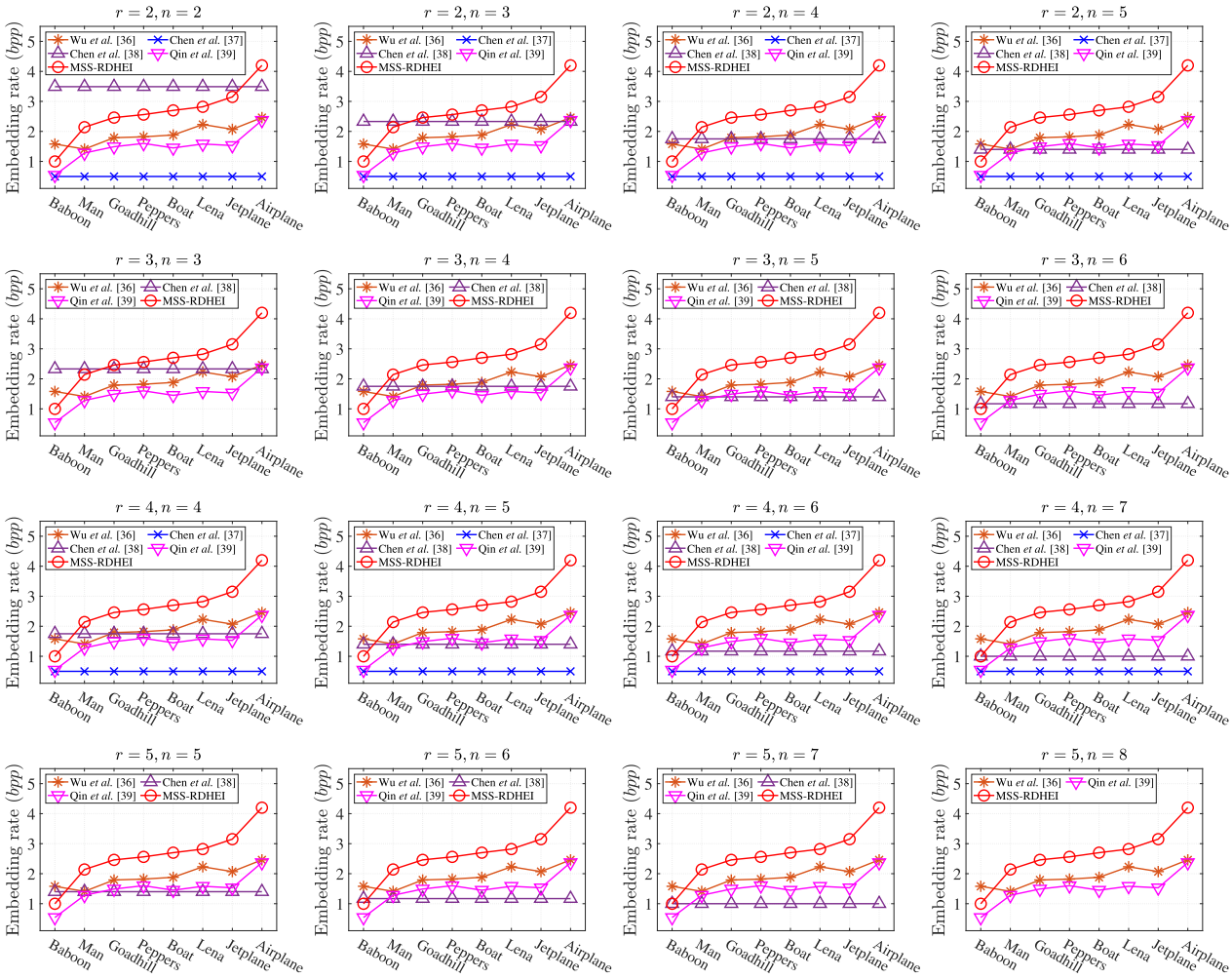

Fig. 11. Embedding rates of different secret sharing-based RDH-EI schemes with different (r, n)-thresholds.

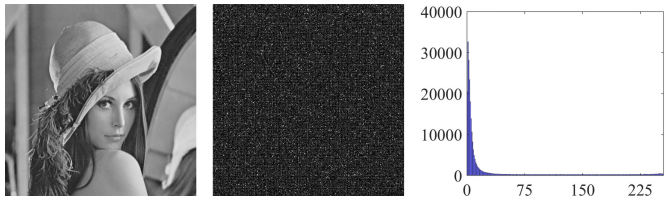

Fig. 12. Prediction errors of the encrypted image using MED predictor: (a) the image *Lena*; (b) prediction errors of the encrypted image; (c) histogram of the prediction errors.

the original image within each block, as discussed in Section 4.2. However, the sharing parameters for different blocks are completely different and random. When embedding data into each share, we keep the first pixel of each block unchanged and calculate the prediction errors of the rest pixels using the first pixel of the related block. Fig. 12 shows the distribution of the prediction errors. Note that for all the prediction error-based RDH-EI schemes (e.g., those schemes in [21], [28], [29], [31], [39]), their prediction errors aren't distributed uniformly. This is because to vacate room for data embedding, all the prediction errors are generated using some predictors and have very small values. However, in our scheme, the first pixel of each block is encrypted using different and random parameters. When sharing two pixels using different and random parameters, the results are random and do not retain any information of the two pixels. We have theoretically discussed and proved this security property in Lemma 2 of Section 2.2. Thus, the first pixels of all the blocks are protected, and the prediction errors are generated from the first pixel of each block. Suppose that the original image size is $M \times N$ and the block size is $B \times B$. Since the security of a block is dependent on its first pixel and the sharing result is within [0,256], the space of the sharing operation is $257^{\lceil (M\times N)/(B\times B)\rceil}$. Besides, a block-based scrambling operation is performed to the original image. Then the space of the block-based scrambling is $\lfloor (M \times N)/(B \times B)\rfloor!$. Then the whole space for the brute-force analysis is $257^{\lceil (M\times N)/(B\times B)\rceil} \times \lfloor (M \times N)/(B \times B)\rfloor!$.

The proposed MSS scheme used in the MSS-RDHEI is a non-deterministic and randomized encryption strategy. Encrypting an image several times using a same encryption key, the generated encrypted images are totally different. This is because some random numbers $a_1, \ldots, a_{r-1}$ shown in Eq. (1) are used and these random numbers are different in each encryption. Fig. 13 shows the encrypted results of image *Lena* by (3,3)-threshold MSS for block size $B = 4$ using a same encryption key in two executions. As can be seen, the encrypted images generated in the two executions using the same key are completely different (see Figs. 13c, 13f, and 13i). A non-deterministic and randomized encryption system has ability to resist many potential attacks [55].

We also statistically evaluate the performance of the scheme for differential attack using the number of pixel change rate (NPCR) and the uniform average change intensity (UACI) [56]. The NPCR and UACI measure how the difference in the plaintext can spread to the ciphertext. According to the discussions in [56], an encryption algorithm is expected to have a NPCR score 99.609% and a UACI score 33.464%. Table 8 shows the NPCR and UACI scores of the MSS-RDHEI scheme for different test images at (6,6)-threshold and block size $B = 4$. As can be seen, the NPCR and UACI scores of our MSS-RDHEI scheme are close to 99.609% and 33.464%, respectively. Thus, our MSS-RDHEI scheme shows good performance in terms of resisting the differential attack.

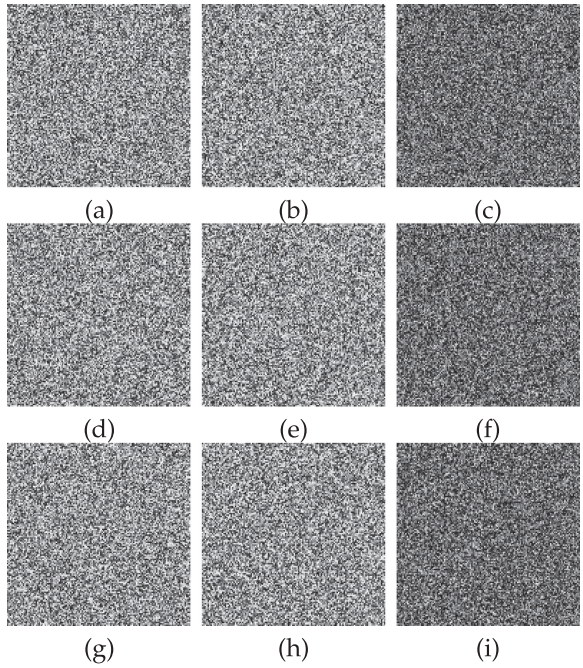

Fig. 13. Encrypted results of image *Lena* by (3,3)-threshold MSS using a same encryption key in two executions: (a) the first encrypted image in the first execution; (b) the first encrypted image in the second execution; (c) difference of (a) and (b); (d) the second encrypted image in the first execution; (e) the second encrypted image in the second execution; (f) difference of (d) and (e); (g) the third encrypted image in the first execution; (h) the third encrypted image in the second execution; (i) difference of (g) and (h).

5.4 Comparisons With Prior Works

Finally, we compare the features of different RDH-EI schemes from the aspects of the separability, recovery quality and security, and Table 9 lists the results. It can be seen that the schemes in [35], [37], [57] aren't separable, which means that the embedded data can only be extracted after the marked encrypted images have been decrypted. The other schemes are separable, which means that the embedded data extraction is independent of the image decryption. The scheme in [57] is a lossy method, which means that the original images cannot be recovered completely. The other schemes are lossless. For the RDH-EI schemes in [28], [57], to keep data redundant in the encrypted domain, they usually use some lightweight encryption methods such as block permutation, co-modulation and XOR with a fixed encryption key. These encryption strategies can only achieve limited security strengths [32], [54]. The homomorphic encryption-based schemes in [33], [34], [35] can achieve a high security level. However, they have quite high computation cost and large data expansion, due to the heavy cryptogtraphic operations. The secret sharing-based schemes have no data expansion for each data hider. The scheme of Chen et al. [37] can also achieve a lightweight security with

TABLE 8
NPCR and UACI Scores for Different Images of MSS-RDHEI At (6,6)-Threshold

Images	NPCR (%)						UACI (%)					
	$\mathbf{E}_1$	$\mathbf{E}_2$	$\mathbf{E}_3$	$\mathbf{E}_4$	$\mathbf{E}_5$	$\mathbf{E}_6$	$\mathbf{E}_1$	$\mathbf{E}_2$	$\mathbf{E}_3$	$\mathbf{E}_4$	$\mathbf{E}_5$	$\mathbf{E}_6$
Lena	99.6056	99.6254	99.5960	99.6029	99.5948	99.6177	33.4733	33.3219	33.3340	33.4790	33.4496	33.4230
Baboon	99.6632	99.5502	99.6334	99.5720	99.5937	99.6239	33.8773	33.9168	33.5803	33.7230	33.7047	33.9104
Jetplane	99.5918	99.5869	99.6044	99.6414	99.5693	99.5190	33.3723	33.5853	33.7585	33.3498	33.5182	33.5510
Man	99.6361	99.6445	99.6078	99.5201	99.5483	99.5468	33.5609	33.5807	33.7237	33.4500	33.5760	33.5812
Airplane	99.5972	99.6502	99.6544	99.7036	99.6254	99.4877	33.4122	33.3602	33.2067	33.6453	33.6460	33.8331
Goldhill	99.6738	99.5808	99.6113	99.6574	99.5876	99.4980	33.6399	33.4713	33.5909	33.6248	33.7411	33.1490
Peppers	99.6819	99.5071	99.6349	99.6628	99.6532	99.5155	33.6179	33.5917	33.6862	33.4712	33.2420	33.4126
Boat	99.6235	99.5617	99.5235	99.6239	99.5861	99.5781	33.8525	33.3841	33.4208	33.3950	33.5618	33.4403

TABLE 9
Comparisons of Different VRAE-Based RDH-EI Schemes

Methods	Encryption strategies	Separable	Lossless recovery	Security	Without data expansion
Yi et al. [28]	Block permutation & Co-modulation	Yes	Yes	limited	Yes
Bhardwaj et al. [57]	XOR	No	No	limited	Yes
Ke et al. [33]	Homomorphic Encryption	Yes	Yes	High	No
Chen et al. [34]	Homomorphic encryption	Yes	Yes	High	No
Li et al. [35]	Homomorphic encryption	No	Yes	High	No
Wu et al. [36]	Secret sharing	Yes	Yes	High	Yes
Chen et al. [37]	Secret sharing	No	Yes	limited	Yes
Chen et al. [38]	Secret sharing	Yes	Yes	High	Yes
Qin et al. [39]	Secret sharing	Yes	Yes	High	Yes
MSS-RDHEI	Secret sharing	Yes	Yes	High	Yes

The data expansion is to each data hider.

a proper setting of parameter t, according to the discussions in the original literature [37]. This is because all the sharing parameters are generated from the encryption key and no random integers are used in the encryption process. The other secret sharing-based RDH-EI schemes in [36], [38], [39] and our MSS-RDHEI scheme can well ensure the image content confidentiality.

6 Conclusion

In this paper, we propose a new VRAE-based RDH-EI scheme using a new secret sharing technique with multiple data hiders. First, we devised an (r, n)-threshold matrix-based secret sharing (MSS) technique using the matrix theory and the cipher-feedback mechanism. It can securely encrypt an original image into several encrypted images using matrix multiplication. Using the MSS, we further proposed an (r, n)-threshold $(r \leq n)$ RDH-EI scheme called MSS-RDHEI, in which a block error mixture encoding (BEME) method with high efficiency is developed to encode encrypted images. The MSS-RDHEI allows a content owner to encrypt an original image to be n encrypted images, and then outsource these n encrypted images to n data hiders. Each data hider can embed some additional data, e.g., copyright and identification information, into the encrypted image for the purposes of storage, management, or other processing, and the encrypted image with data embedded is called marked encrypted image. Since the developed BEME can make full use of pixel redundancy in the encrypted images, a large room can be vacated for data embedding. A receiver can extract the embedded data of a marked encrypted image, or reconstruct the original image from r marked encrypted images using related secret keys. Experiment results show that the MSS-RDHEI can well protect the image contents and achieves a much larger embedding capacity than existing secret sharing-based schemes.

References

[1] H. Yao, F. Mao, C. Qin, and Z. Tang, "Dual-JPEG-image reversible data hiding," *Inf. Sci.*, vol. 563, pp. 130–149, Jul. 2021.

[2] Z. Yin, Y. Peng, and Y. Xiang, "Reversible data hiding in encrypted images based on pixel prediction and bit-plane compression," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 992–1002, Mar./Apr. 2022.

[3] W. Tang, B. Li, M. Barni, J. Li, and J. Huang, "An automatic cost learning framework for image steganography using deep reinforcement learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 952–967, 2021.

[4] Y. Du, Z. Yin, and X. Zhang, "High capacity lossless data hiding in JPEG bitstream based on general VLC mapping," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1420–1433,Mar./Apr. 2022.

[5] F. Peng, Z.-X. Lin, X. Zhang, and M. Long, "Reversible data hiding in encrypted 2D vector graphics based on reversible mapping model for real numbers," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 9, pp. 2400–2411, Sep. 2019.

[6] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transa. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[7] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.

[8] T. Zhang, X. Li, W. Qi, and Z. Guo, "Location-based PVO and adaptive pairwise modification for efficient reversible data hiding," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2306–2319, 2020.

[9] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, and Y. Y. Tang, "Secure reversible image data hiding over encrypted domain via key modulation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 3, pp. 441–452, Mar. 2016.
[10] R. Wang, G. Wu, Q. Wang, L. Yuan, Z. Zhang, and G. Miao, "Reversible data hiding in encrypted images using median edge detector and two's complement," *Symmetry*, vol. 13, no. 6, May 2021, Art. no. 921.
[11] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted JPEG bitstreams," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1055–1067, Nov./Dec. 2018.
[12] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," in *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, Bellingham, WA, USA: International Society for Optics and Photonics, Mar. 2008.
[13] X. Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, Apr. 2011.
[14] P. Puteaux and W. Puech, "A recursive reversible data hiding in encrypted images method with a very high payload," *IEEE Trans. Multimedia*, vol. 23, pp. 636–650, 2021.
[15] Y.-C. Chen, C.-W. Shiu, and G. Horng, "Encrypted signal-based reversible data hiding with public key cryptosystem," *J. Vis. Commun. Image Representation*, vol. 25, no. 5, pp. 1164–1170, Jul. 2014.
[16] P. Puteaux and W. Puech, "An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 7, pp. 1670–1681, Jul. 2018.
[17] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public-key cryptography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.
[18] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 3, pp. 553–562, Mar. 2013.
[19] W. Zhang, K. Ma, and N. Yu, "Reversibility improved data hiding in encrypted images," *Signal Process.*, vol. 94, pp. 118–127, Jan. 2014.
[20] Y. Qiu, Z. Qian, H. Zeng, X. Lin, and X. Zhang, "Reversible data hiding in encrypted images using adaptive reversible integer transformation," *Signal Process.*, vol. 167, Feb. 2020, Art. no. 107288.
[21] Y. Wang and W. He, "High capacity reversible data hiding in encrypted image based on adaptive MSB prediction," *IEEE Trans. Multimedia*, vol. 24, pp. 1288–1298, 2021, doi: 10.1109/TMM.2021.3062699.
[22] X. Liao, K. Li, and J. Yin, "Separable data hiding in encrypted image based on compressive sensing and discrete fourier transform," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 20 739–20 753, Oct. 2017.
[23] H. Ge, Y. Chen, Z. Qian, and J. Wang, "A high capacity multi-level approach for reversible data hiding in encrypted images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2285–2295, Aug. 2019.
[24] S. Yi and Y. Zhou, "Adaptive code embedding for reversible data hiding in encrypted images," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 4322–4326.
[25] Z. Yin, Y. Xiang, and X. Zhang, "Reversible data hiding in encrypted images based on multi-MSB prediction and huffman coding," *IEEE Trans. Multimedia*, vol. 22, no. 4, pp. 874–884, Apr. 2020.
[26] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 826–832, Apr. 2012.
[27] Z. Qian and X. Zhang, "Reversible data hiding in encrypted images with distributed source encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 636–646, Apr. 2016.
[28] S. Yi and Y. Zhou, "Separable and reversible data hiding in encrypted images using parametric binary tree labeling," *IEEE Trans. Multimedia*, vol. 21, no. 1, pp. 51–64, Jan. 2019.
[29] Y. Qiu, Q. Ying, Y. Yang, H. Zeng, S. Li, and Z. Qian, "High-capacity framework for reversible data hiding in encrypted image using pixel prediction and entropy encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 5874–5887, Sep. 2022, doi: 10.1109/TCSVT.2022.3163905, 2022.
[30] K. Gao, J.-H. Horng, and C.-C. Chang, "High-capacity reversible data hiding in encrypted images based on adaptive block encoding," *J. Vis. Commun. Image Representation*, vol. 84, Apr. 2022, Art. no. 103481.
[31] F. Huang, J. Huang, and Y.-Q. Shi, "New framework for reversible data hiding in encrypted domain," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 12, pp. 2777–2789, Dec. 2016.
[32] L. Qu, F. Chen, S. Zhang, and H. He, "Cryptanalysis of reversible data hiding in encrypted images by block permutation and co-modulation," *IEEE Trans. Multimedia*, vol. 24, pp. 2924–2937, 2021, doi: 10.1109/TMM.2021.3090588.
[33] Y. Ke, M.-Q. Zhang, J. Liu, T.-T. Su, and X.-Y. Yang, "Fully homomorphic encryption encapsulated difference expansion for reversible data hiding in encrypted domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2353–2365, Aug. 2020.
[34] B. Chen, X. Wu, W. Lu, and H. Ren, "Reversible data hiding in encrypted images with additive and multiplicative public-key homomorphism," *Signal Process.*, vol. 164, pp. 48–57, Nov. 2019.
[35] M. Li and Y. Li, "Histogram shifting in encrypted images with public key cryptosystem for reversible data hiding," *Signal Process.*, vol. 130, pp. 190–196, Jan. 2017.
[36] X. Wu, J. Weng, and W. Yan, "Adopting secret sharing for reversible data hiding in encrypted images," *Signal Process.*, vol. 143, pp. 269–281, Feb. 2018.
[37] Y.-C. Chen, T.-H. Hung, S.-H. Hsieh, and C.-W. Shiu, "A new reversible data hiding in encrypted image based on multi-secret sharing and lightweight cryptographic algorithms," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 12, pp. 3332–3343, Dec. 2019.
[38] B. Chen, W. Lu, J. Huang, J. Weng, and Y. Zhou, "Secret sharing based reversible data hiding in encrypted images with multiple data-hiders," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 978–991, Mar./Apr. 2022.
[39] C. Qin, C. Jiang, Q. Mo, H. Yao, and C.-C. Chang, "Reversible data hiding in encrypted image via secret sharing based on GF (p) and GF (2^8)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 1928–1941, 2022, doi: 10.1109/TCSVT.2021.3091319.
[40] W. Trappe, *Introduction to Cryptography With Coding Theory*. Noida, UP, India: Pearson Education India, 2006.
[41] S. Goldwasser and M. Bellare, *Lecture notes on cryptography*, Cambridge, MA, USA: MIT Press, 1996.
[42] M. Rosulek, *The Joy of Cryptography*. Corvallis, OR, USA: Oregon State Univ., 2021.
[43] L. Yu, L. Liu, Z. Xia, X. Yan, and Y. Lu, "Lossless and efficient secret image sharing based on matrix theory modulo 256," *Mathematics*, vol. 8, no. 6, Jun. 2020, Art. no. 1018.
[44] W. Ding, K. Liu, X. Yan, H. Wang, L. Liu, and Q. Gong, "An image secret sharing method based on matrix theory," *Symmetry*, vol. 10, no. 10, Oct. 2018, Art. no. 530.
[45] Z. Liu, G. Zhu, Y.-G. Wang, J. Yang, and S. Kwong, "A novel (t, s, k, n)-threshold visual secret sharing scheme based on access structure partition," ACM Transactions on Multimedia Computing, *Commun. Appl.*, vol. 16, no. 4, pp. 1–21, Nov. 2020.
[46] Y. Cheng, Z. Fu, and B. Yu, "Improved visual secret sharing scheme for QR code applications," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2393–2403, Sep. 2018.
[47] C.-C. Chen, C.-S. Lin, and J.-Z. Chen, "Boolean-based (k, n, m) multi-secret image sharing," *Axioms*, vol. 11, no. 5, Apr. 2022, Art. no. 197.
[48] L. Bao, S. Yi, and Y. Zhou, "Combination of sharing matrix and image encryption for lossless -secret image sharing," *IEEE Trans. Image Process.*, vol. 26, no. 12, pp. 5618–5631, Dec. 2017.
[49] Z. Hua, Y. Zhou, and B. Bao, "Two-dimensional sine chaotification system with hardware implementation," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 887–897, Feb. 2020.
[50] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.
[51] C. Yu, X. Zhang, X. Zhang, G. Li, and Z. Tang, "Reversible data hiding with hierarchical embedding for encrypted images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 2, pp. 451–466, Feb. 2022.
[52] A. Mohammadi, M. Nakhkash, and M. A. Akhaee, "A high-capacity reversible data hiding in encrypted images employing local difference predictor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2366–2376, Aug. 2020.

[53] A. Mohammadi, "A general framework for reversible data hiding in encrypted images by reserving room before encryption," 2021, *arXiv:2103.15240*.
[54] F. Khelifi, "On the security of a stream cipher in reversible data hiding schemes operating in the encrypted domain," *Signal Process.*, vol. 143, pp. 336–345, Feb. 2018.
[55] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Boca Raton, FL, USA: CRC Press, 2020.
[56] Y. Wu et al., "NPCR and UACI randomness tests for image encryption," *Cyber Journals: Multidisciplinary Journals Sci. Technol. J. Sel. Areas Telecommun.*, vol. 1, no. 2, pp. 31–38, Apr. 2011.
[57] R. Bhardwaj and A. Aggarwal, "An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem," *Pattern Recognit. Lett.*, vol. 139, pp. 60–68, Nov. 2020.

Zhongyun Hua (Member, IEEE) received the BS degree in software engineering from Chongqing University, Chongqing, China, in 2011, and the MS and PhD degrees in software engineering from the University of Macau, Macau, China, in 2013 and 2016, respectively. He is currently an associate professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen, China. His research interests include chaotic system, chaos-based applications, and multimedia security. He has published more than sixty papers on the subject, receiving more than 3900 citations.

Yanxiang Wang received the BS degree in software engineering from Tongji University, Shanghai, China, in 2015. He is currently working toward the MS degree in computer science and technology from the Harbin Institute of Technology, Shenzhen, China. His research interests include reverible data hiding and secret sharing.

Shuang Yi received the BS degree in software engineering from Chongqing University, Chongqing, China, in 2011, and the PhD degree in software engineering from the University of Macau, Macau, China, in 2018. She is currently a lecture with the College of Criminal Investigation, Southwest University of Political Science and Law, Chongqing. Her research interests include data hiding, secret sharing, and image processing and multimedia security.

Yifeng Zheng received the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2019. He is an assistant professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He worked as a postdoc with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia and City University of Hong Kong. His work has appeared in prestigious venues such as ESORICS, DSN, ACM AsiaCCS, IEEE INFOCOM, IEEE ICDCS, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, and *IEEE Transactions on Services Computing*. He received the Best Paper Award in the European Symposium on Research in Computer Security (ESORICS) 2021. His current research interests include focused on security and privacy related to cloud computing, IoT, machine learning, and multimedia.

Xingyu Liu received the BS degree in computer science and technology from the Harbin Institute of Technology, Shenzhen, China, in 2022. She is currently working toward the MS degree in computer science and technology from the Harbin Institute of Technology. Her research interests include reverible data hiding and secret sharing.

Yongyong Chen received the BS and MS degrees from the Shandong University of Science and Technology, Qingdao, China, in 2014 and 2017, respectively, and the PhD degree from the University of Macau, Macau, in 2020. He is currently an assistant professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He has published more than 30 research papers in top-tier journals and conferences, including *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Circuits and Systems for Video Technology*, *IEEE Transactions on Geoscience and Remote Sensing*, *IEEE Transactions on Computational Imaging*, *IEEE Journal of Selected Topics in Signal Processing*, *Pattern Recognition* and *ACM Transactions on Multimedia*. His research interests include image processing, data mining, and computer vision.

Xinpeng Zhang (Member, IEEE) received the BS degree in computational mathematics from Jilin University, China, in 1995, and the ME and PhD degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. Since 2004, he was with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. He is also with the faculty of the School of Computer Science, Fudan University. He was with The State University of New York at Binghamton as a visiting scholar from 2010 to 2011, and also with Konstanz University as an experienced Researcher, sponsored by the Alexander von Humboldt Foundation from 2011 to 2012. His research interests include multimedia security, AI security, and image processing. He has published more than 300 papers in these areas. He was an associate editor of the *IEEE Transactions in Information Forensics and Security* from 2014 to 2017.