EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Adaptive and separable multiary reversible data hiding in encryption domain

Mingji Yu[1], Yuchen Liu[1], Hu Sun[1], Heng Yao[1*] and Tong Qiao[2]

Abstract

To ensure the security of digital information, information needs to be stored and processed in the encryption domain during the cloud storage process, during which the user does not allow the cloud provider (CP) to access the image contents, and the CP must embed additional information in the image. Therefore, reversible data hiding in an encryption domain (RDHEI) has emerged. This paper presents an adaptive and separable multiary RDHEI method. First, the image is divided into two parts that consist of reference pixels and non-reference pixels. Next, we compute the prediction errors of the non-reference pixels, followed by a replacement of the original non-reference values with the computed prediction errors. Then, the entire image is encrypted with a specially designed stream cipher strategy so that the CP can embed additional information by modifying the prediction errors without knowing the original image. In addition, although our method vacates space before encryption, the CP can adaptively control the reserved space, unlike traditional RDHEI schemes which also vacate space before encryption. Because our method is separable, the embedded message can be extracted from both encryption and plaintext domains accurately. The experimental results demonstrate the efficacy of the proposed method.

Keywords: Reversible data hiding, encryption domain, stream cipher, prediction error histogram

1 Introduction

Data hiding technology can achieve the purposes of secret transmission and content authentication by embedding secret information into the host [1–4]. As a significant branch of data hiding, reversible data hiding (RDH) is an algorithm that embeds additional data into a specific payload, such as image, video, or audio signals, and recovers the additional data and the payload without any loss. Traditional RDH can be roughly divided into four categories: (1) lossless compression (LC) [5], in which the space for data hiding is generated by LC of the image, (2) histogram shifting (HS) [6, 7], in which the embedding space is generated by shifting the gray-scale histogram of the image, (3) difference expansion (DE) [8] [9], in which the differences between nearby pixels are expanded to reserve space, and (4) prediction error expansion (PEE) [10–14], in which the space for embedding is generated by shifting the prediction error histogram (PEH).

All the methods mentioned above were applied in the plaintext domain. However, with the rapid development of Internet technology and cloud storage applications, a new requirement for RDH has emerged. On the one hand, many images must be uploaded to a cloud server, followed by message embedding and other post-processing procedures; on the other hand, the content of the images should not be accessible by the cloud provider (CP), especially in the case of critical and private images, such as medical images that are private, and military images that relate to national security. In order to deal with this paradox, RDH has inevitably emerged in the encryption domain. While original images must be protected and recovered losslessly, we expect to achieve the highest embedding capacity. In order to obtain an optimal balance of security and embedding capacity, several studies have been carried out. The existing RDH in encrypted images (RDHEI) can be mainly divided into two categories: methods based on reserving

* Correspondence: hyao@usst.edu.cn
[1]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
Full list of author information is available at the end of the article

room before encryption (RRBE), and methods based on vacating space after encryption (VRAE).

For an RRBE-based method, Ma et al. [15] proposed an RDHEI method that reserved a room for hiding by embedding some pixels in the original image using a traditional RDH algorithm. Zhang et al. [16] proposed calculating the prediction error to replace the image pixels, followed by an encryption with a traditional encryption algorithm so the data hider can embed the additional data by modifying the prediction error. Cao et al. [17] provided a method that implemented a patch-level sparse representation technique to create vacated space for data hiding. Xu and Wang [18] proposed a scheme to enable a traditional PEE strategy in the plaintext domain to be effective in the encryption domain. Yi and Zhou [19] provided a binary-block embedding strategy to improve embedding capacity and visual quality in marked decrypted images. Li et al. [20] proposed a separable RDH scheme, in which encryption quality was improved by combing the permutation and stream cipher during the encryption process, and the embedding rate (ER) was increased by replacing pixels with their corresponding prediction errors.

For a VRAE-based method, Qian et al. [21] proposed a method based on progressive recovery that consisted of three agents, including the content owner, the data hider, and the recipient. In [22], the use of HS with a public-key cryptosystem was implemented to realize RDHEI. Agrawal and Kumar [23] presented a method based on additive modulo 256 and utilized a property of the mean to hide data. Singh and Raman [24] proposed an RDHEI scheme based on a Chinese remainder theorem (CRT)-based sharing scheme to transmit media information over cloud architecture and prove its original ownership. Xiao et al. [25] proposed a separable RDHEI method by utilizing pixel value ordering to embed secret data in each block in an additive homomorphic encrypted image. Liu and Pun [26] presented a redundant space transfer (RST) scheme to create redundant space for data hiding in which traditional RDH technology could be applied. Qin et al. [27] presented an RDHEI method in which owners used an analog stream cipher and block permutation to encrypt blocks of original images that did not overlap, and the data hider could classify blocks and use data hiding keys when embedding. Khelifi et al. [28] used a special encryption method for original images, and then compressed the encryption image to a bit series to create space for data hiding. Wu et al. [29] presented a method applied in homomorphic encrypted images so that some of the hidden data could be extracted in the encryption domain, and the rest could be extracted in the plaintext domain. Fu et al. [30] encrypted a host image via block scrambling and a stream cipher, followed by an adaptive compression of the most significant bit (MSB) layer to vacate space for data hiding. Qin et al. [31] scrambled an image with three different levels and vacated the embedding space by using sparse matrix coding to compress the least significant bit (LSB) of the encrypted image.

From the researches mentioned above, RDHEI has developed to a certain degree. However, unlike traditional RDH in the plaintext domain, there are still many lines of research to follow. Motivated by Xu's work [18], which was devoted to applying the traditional PEE method in the encryption domain, in this work, we propose a method for improving encryption quality and the ER. Specifically, compared with [18], the proposed method offers the following three improvements: (1) in [18], the vacated space for data hiding was determined by the image owner; in our work, the image owner needs only to encrypt the image, while the work of creating space is conducted by the data hider, thereby reducing the workload. (2) A new encryption progress is proposed so that the encrypted image is less noticeable and better able to resist attacks from hackers. (3) A new method is presented to increase the ER by shifting the PEH based on the unique characteristics of the encryption domain.

This article is organized as follows: Section 2 describes the proposed method in detail, Section 3 presents the experimental results of the proposed method, and finally, Section 4 concludes this paper.

2 Methods/experimental

In this section, we propose a separable RDHEI scheme, which mainly consists of prediction and replacement, encryption, data hiding, data extraction, and image recovery. Specifically, the image owner uses our method to encrypt the image and vacate space for data hiding during a subsequent procedure, which is conducted by the CP. The CP then uses the reserved room to embed data. If he does not have the encryption keys, he can hide data but he will not access the original content, so the original owner can keep his images safe and the data hider can guarantee his rights by embedding additional information. A flow diagram of encryption and data hiding is briefly shown in Fig. 1.

2.1 Prediction and replacement

Suppose that the original image $\mathbf{X}$ is an 8-bit gray-scale image with a size of $M \times N$, and its pixels are denoted as $X(i, j)$, $1 \le i \le M$, $1 \le j \le N$. First, we divide all the pixels using a checkerboard pattern into two sets, defined as the reference and non-reference pixels. In detail, the pixels in odd rows and odd columns are selected as reference pixels, and the other pixels are non-reference pixels. Figure 2 shows the division between reference and non-reference pixels, which are denoted in gray and white, respectively. To predict one non-reference pixel, we tend to use the four adjacent reference pixels for the sake of accuracy. Depending on whether there are enough reference pixels around the non-reference pixels,

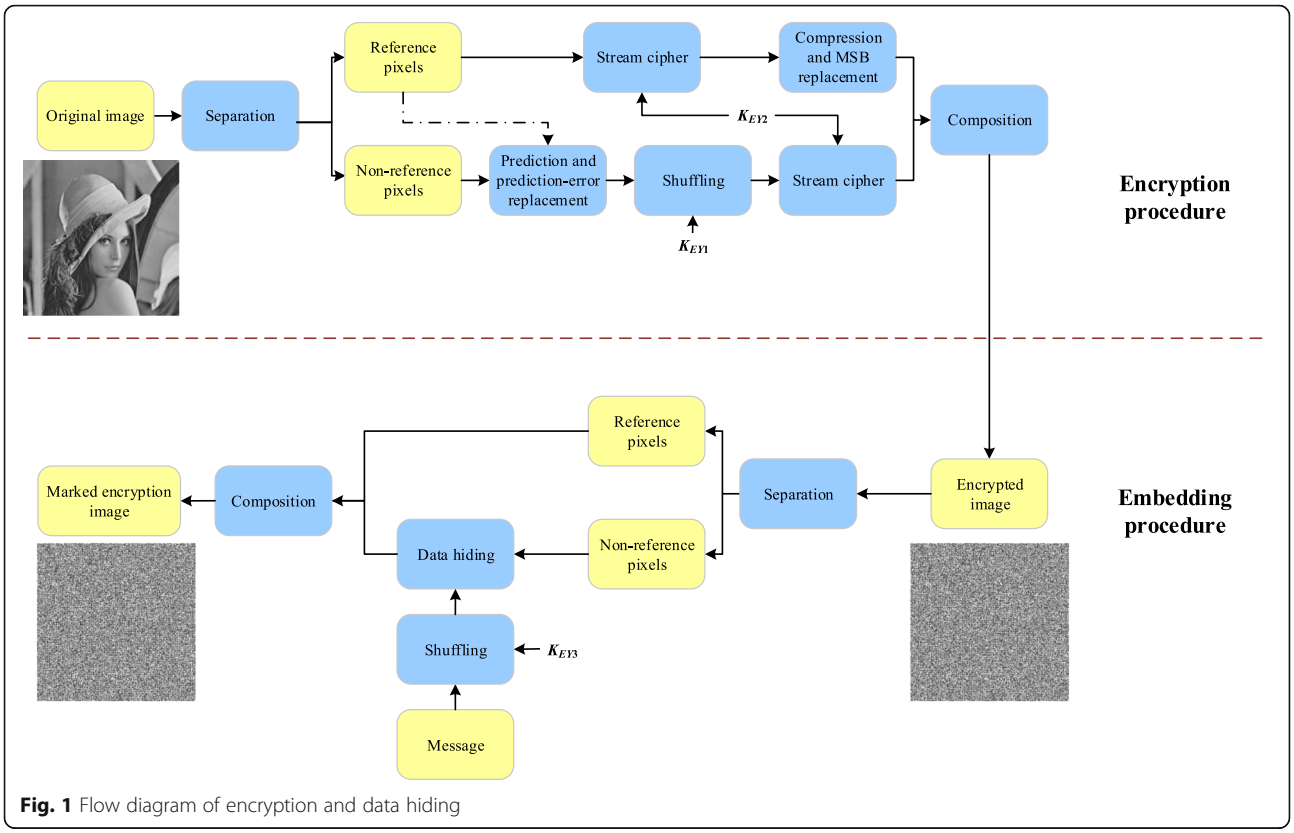


Fig. 1 Flow diagram of encryption and data hiding

all non-reference pixels are then further divided into two sets. For some non-reference pixels, there is a sufficient number of reference pixels around them, and we denote them as Δ. For other non-reference pixels, due to the lack of adjacent reference pixels, we use the prediction results of their adjacent non-reference pixels as references, and we denote these pixels as Ω.

The prediction process is executed following an order that predicts the pixels belonging to Δ first, followed by the prediction of pixels belonging to Ω. Their corresponding prediction values $P(i,j)$ can be computed as

$$P(i,j) = \text{round}([X(i-1,j-1) + X(i+1,j-1) + X(i+1,j+1) + X(i-1,j+1)]/4), \text{if } X(i,j) \in \Delta, \quad (1)$$

and

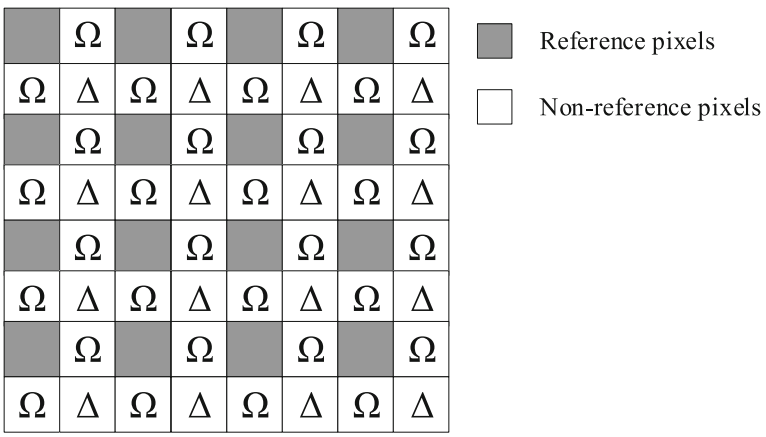
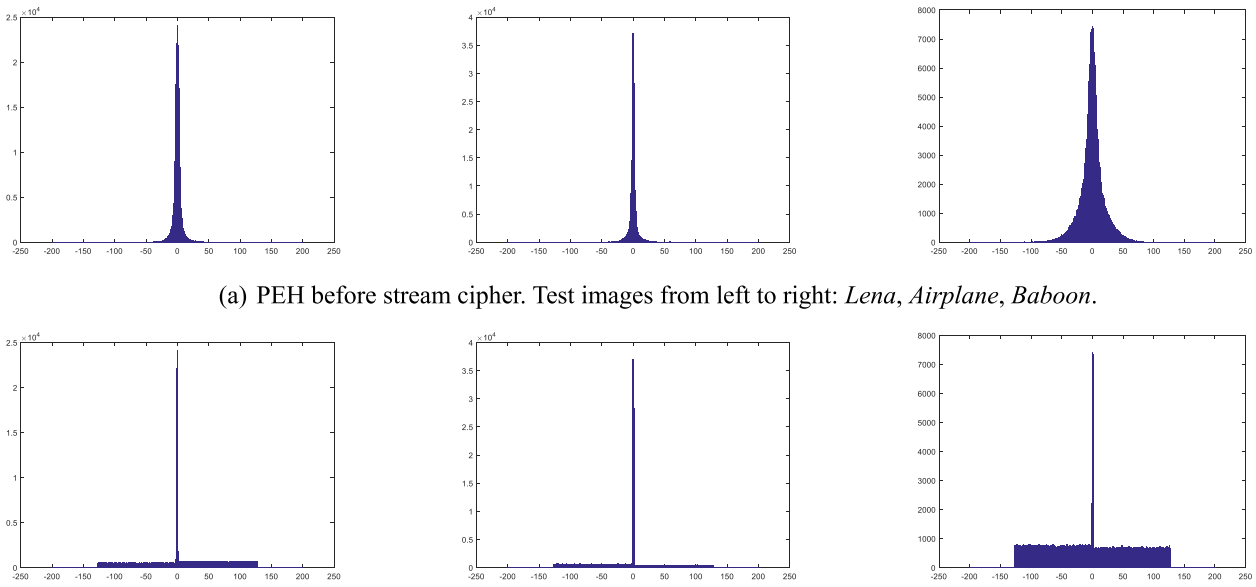


Fig. 2 Division of pixels for prediction



(a) PEH before stream cipher. Test images from left to right: *Lena*, *Airplane*, *Baboon*.

(b) PEH after stream cipher. Test images from left to right: *Lena*, *Airplane*, *Baboon*.

Fig. 3 PEH before and after stream cipher

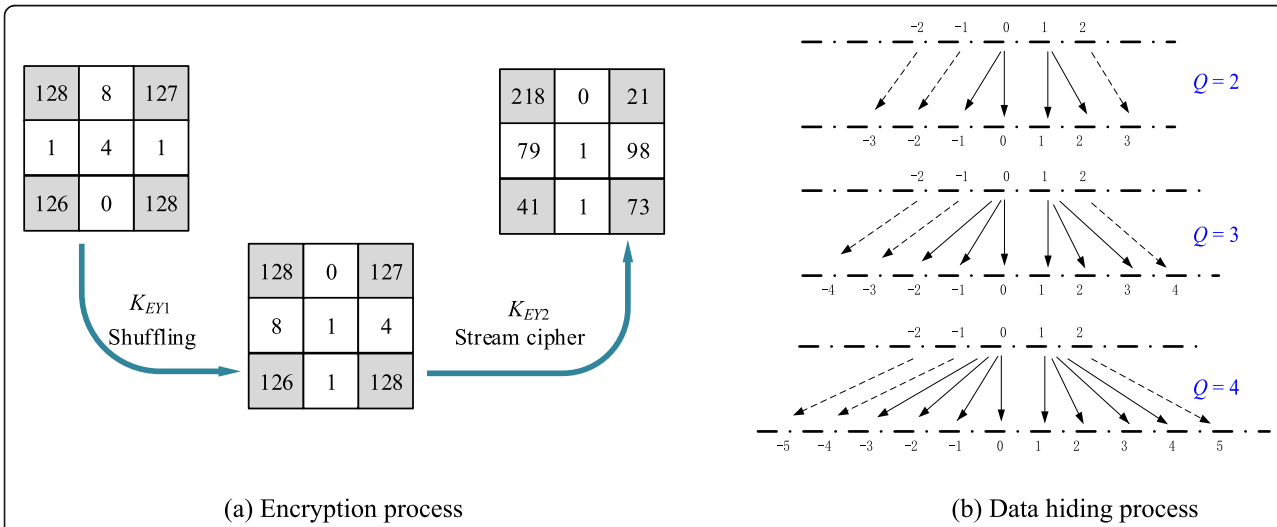
$$P(i,j) = \text{round}([X(i-1,j) + X(i+1,j) + P(i,j-1) + P(i,j+1)]/4), \text{if } X(i,j) \in \Omega, \quad (2)$$

Then, we compute the prediction error $E(i,j)$ as

$$E(i,j) = X(i,j) - P(i,j), \text{if } X(i,j) \in (\Delta \cup \Omega). \quad (3)$$

Note that in theory, the value range of $E(i,j)$ is from − 255 to 255, which requires a nine-bit number to represent. However, according to our observation, to correctly show an image, $E(i,j)$ generally ranges from − 127 to 127. Thus, in our work, we continue to use 8 bits to represent the prediction error. Here, we only use the least significant 7 bits to show the absolute value of errors, and the MSB to indicate the sign, i.e., "0" and "1" indicate positive and negative signs, respectively. Because the prediction error represented in our work only varies from − 127 to 127, prediction errors out of range of the set are modified as

$$E'(i,j) = \begin{cases} E(i,j) - 127, & \text{if } E(i,j) > 127, \\ E(i,j) + 127, & \text{if } E(i,j) < -127, \\ E(i,j), & \text{otherwise.} \end{cases} \quad (4)$$



(a) Encryption process

(b) Data hiding process

Fig. 4 Examples of encryption and data hiding

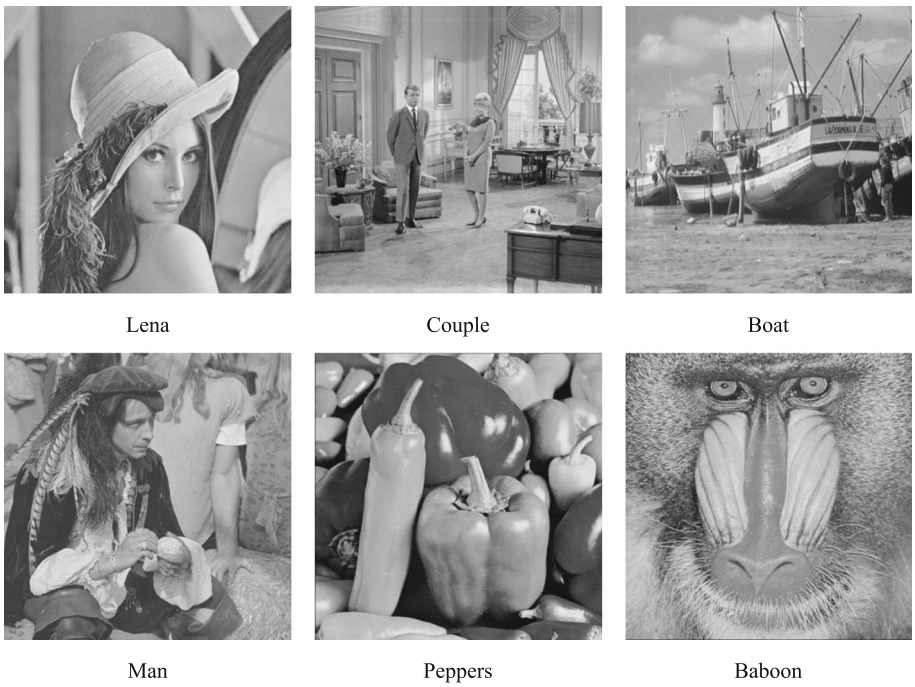


Fig. 5 Six standard test images

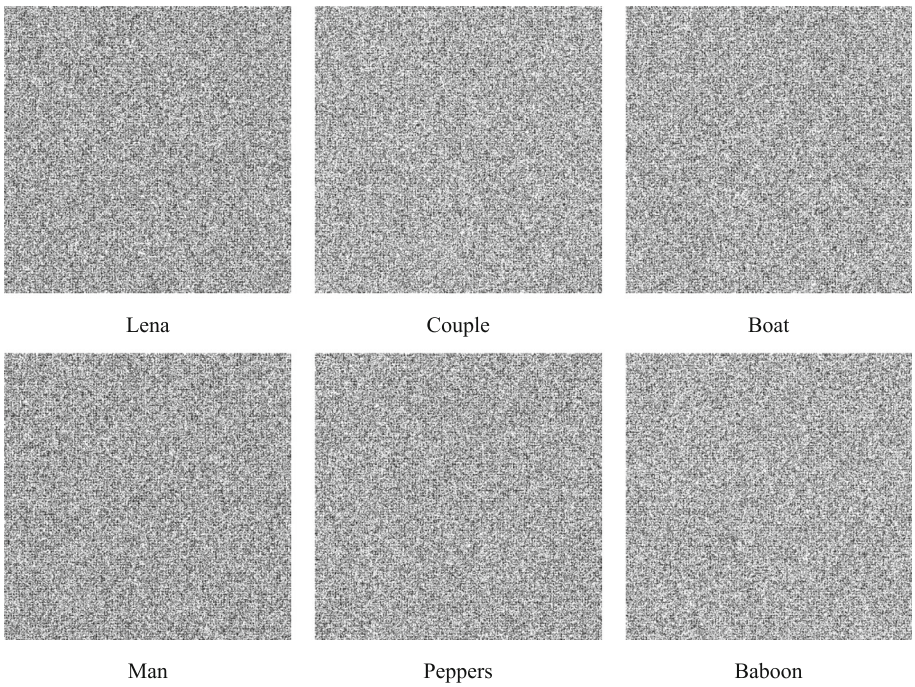


Fig. 6 Encryption results of our work

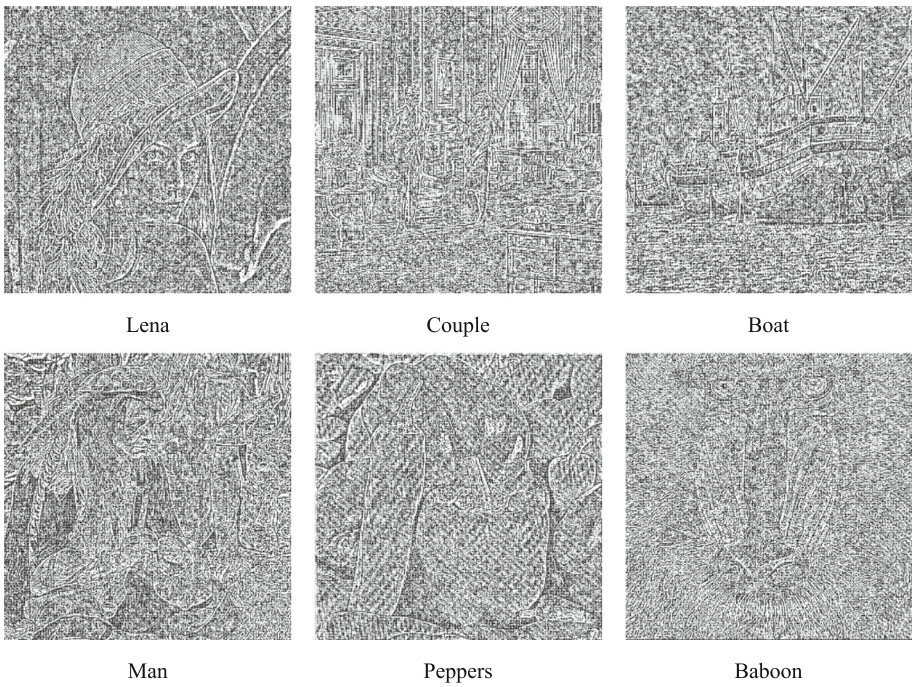


Fig. 7. Encryption results of Xu and Wang's work [18]

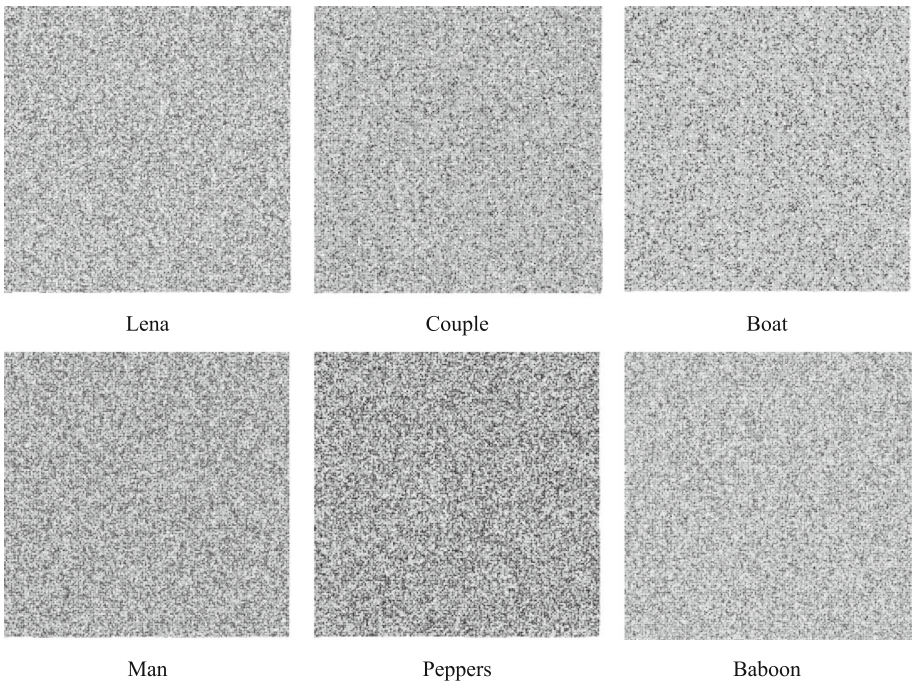


Fig. 8. Encryption results of Li et al.'s work [20]

Table 1 PSNR comparisons among Xu and Wang's work [18], Li et al.'s work [20], and ours

	PSNR (dB)		
	Li et al. (2018)	Xu and Wang (2016)	Proposed
Lena	8.50	8.21	5.92
Couple	8.71	9.15	5.58
Boat	7.19	7.59	5.47
Man	8.25	8.66	6.33
Peppers	7.28	7.48	6.50
Baboon	8.01	9.00	5.13

Note that we reserved $(10000000)_2$, and use $(00000000)_2$ to represent the particular case $E'(i, j) = 0$. We record the coordinates of the prediction errors, which are initially outside the range of [− 127, 127], and save them as a location map, denoted as **O**. Finally, we replace all the non-reference pixels with their corresponding modified prediction errors $E'(i, j)$ to generate a new image, denoted as **I**. It is worth pointing out that the prediction criterion in our work is not limited to Eqs. (1) and (2), and any precise prediction strategy can be used to further improve the efficiency of our entire performance.

2.2 Image encryption

First, we only shuffle $E'(i, j)$ in **I** via an encryption key K_{EY1} to convert $E'(i, j)$ to $E''(i, j)$. For an easier description, we denote the reference pixels as $Y(i, j)$. We use a stream cipher to encrypt $Y(i, j)$ and $E''(i, j)$ separately. Specifically, in our work, we use another encryption key, K_{EY2}, to generate a pseudorandom matrix **R** of size $M \times N$, in which each element is a 7-bit number and denoted as $R(i, j)$. We only encrypt the lowest 7-bit plane of $Y(i, j)$ as

$$Y'(i, j) = Y(i, j) \oplus R(i, j), \tag{5}$$

where $Y'(i, j)$ represents the encrypted value of $Y(i, j)$, and $\oplus$ indicates the exclusive or (XOR) operation.

Table 2 Operation time comparisons among Xu and Wang's work [18], Li et al.'s work [20], and ours. (unit: s)

	Li et al. (2018)	Xu and Wang (2016)	Proposed
Lena	0.6506	0.2766	0.5849
Couple	0.6276	0.2835	0.6749
Boat	0.6588	0.2823	0.6261
Man	0.6542	0.3090	0.5576
Peppers	0.6616	0.2978	0.5820
Baboon	0.6648	0.2912	0.7068
Average	0.6529	0.2901	0.6221

Meanwhile, to encrypt $E''(i, j)$, we need to find two peak points, T_p and T_n, in advance. $E''(i, j)$ is then encrypted as

$$\tilde{E}(i, j) = \begin{cases} E''(i, j), \text{if } E''(i, j) = T_p \text{ or } T_n, \\ E''(i, j) \oplus R(i, j), \quad \text{otherwise}, \end{cases} \tag{6}$$

where $\tilde{E}(i, j)$ represents the encrypted value of $E''(i, j)$. It must be pointed out that if $\tilde{E}(i, j)$ equals T_p or T_n after the XOR operation, we need to add one or subtract one to avoid confusing the original pixels with $E''(i, j) = T_p$ or T_n, and we use a new location map $\mathbf{O_1}$ to record those $\tilde{E}(i, j)$.

Next, we extract the MSB plane of $Y'(i, j)$. Both the MSB plane and the location maps **O** and $\mathbf{O_1}$ are next compressed via LC coding. Here, arithmetic coding is used in our work because all the elements for compression are made up of "0"s and "1"s. We then replace the MSB of $Y'(i, j)$ with the following parts: compression code of the original MSB plane of $Y'(i, j)$ and its length, and compression codes of **O** and $\mathbf{O_1}$ and their lengths. We denote the replacement version of $Y'(i, j)$ as $Y''(i, j)$.

Finally, we composite $\tilde{E}(i, j)$ and $Y''(i, j)$ to generate the final encrypted image, denoted as **I'**, and send it to the cloud server. Fig. 3a, b shows the PEHs before and after the stream cipher, respectively. As can be seen in Fig. 3b, the values approximately obey a uniform distribution after the stream cipher, except for the reserved space.

To explain the security of our encryption methods, we suppose that there is an image of size 512×512 running through the entire encryption process. In the first part of the encryption, there are $512 \times 512 \times \frac{1}{4} = 65,536$ reference pixels that remain unchanged during the shuffling, and $512 \times 512 \times \frac{3}{4} = 196,608$ non-reference pixels that are rearranged. Therefore, there are P_{196608}^{196608} possibilities in total after rearrangement, which is a rather large number for probability analysis. In the second part of the stream cipher, we suppose that 25% of pixels are working as payload, and the remaining 75% of pixels are XOR with a random 7-bit sequence produced by K_{EY2}. As a result, there are $512 \times 512 \times 0.75 \times 2^7 = 25{,}165{,}824$ possibilities after the stream cipher. After running through the entire encryption process, there are $P_{196608}^{196608} \times 25,165,824$ possibilities in total that are different from the original image. It is difficult or even impossible to detect only one original image from such a multitude of probabilities. In addition, Khelifi et al. [28] presented a mathematical security analysis of shuffling followed by a stream cipher.

2.3 Data hiding

After receiving **I'**, due to the lack of K_{EY1} and K_{EY2}, the CP cannot access to the original image. However, he can embed data in $\tilde{E}(i, j)$ without any knowledge of the ori-

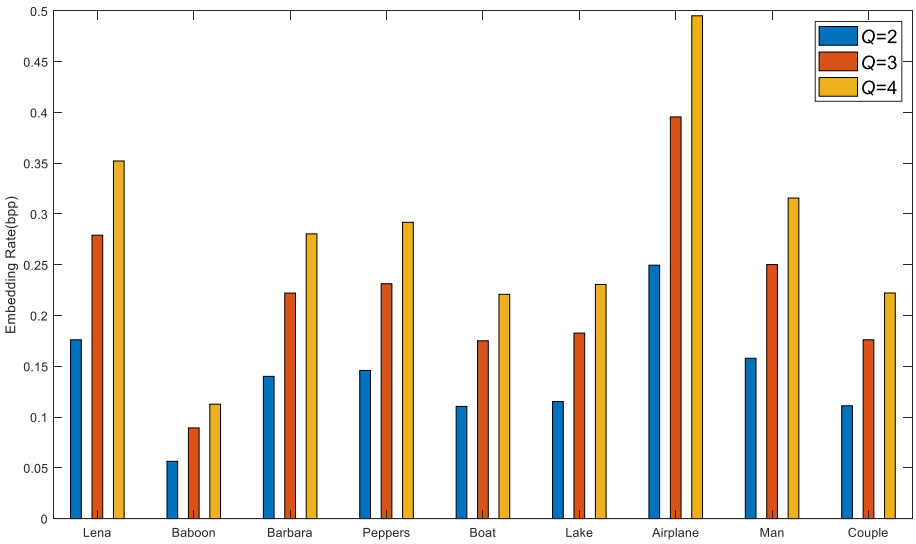


Fig. 9 ERs for nine standard pictures

ginal image. In this paper, owing to the elaborately designed encryption method, applying PEE in the encryption domain becomes feasible. The three steps of our data hiding strategy are listed in detail as follows.

Step 1: The data hider selects the multiary parameter Q, which is suggested to be set to 2, 3, and 4 according to the actual embedding requirement. The reason why Q is recommended to be set to 2, 3, and 4 is that we need to shift histogram during the data embedding process and it will cause inevitably underflow/overflow issues. In our method, we use a location map to deal with these issues. If Q is set to be higher than 4, the size of the location map will increase correspondingly and the embedding capacity might be influenced by the limited space in the bit plane of reference pixels. Before embedding, to increase the security of the proposed method, a new key K_{EY3} is used to shuffle the to-be-embedded message and transfer the message to the corresponding

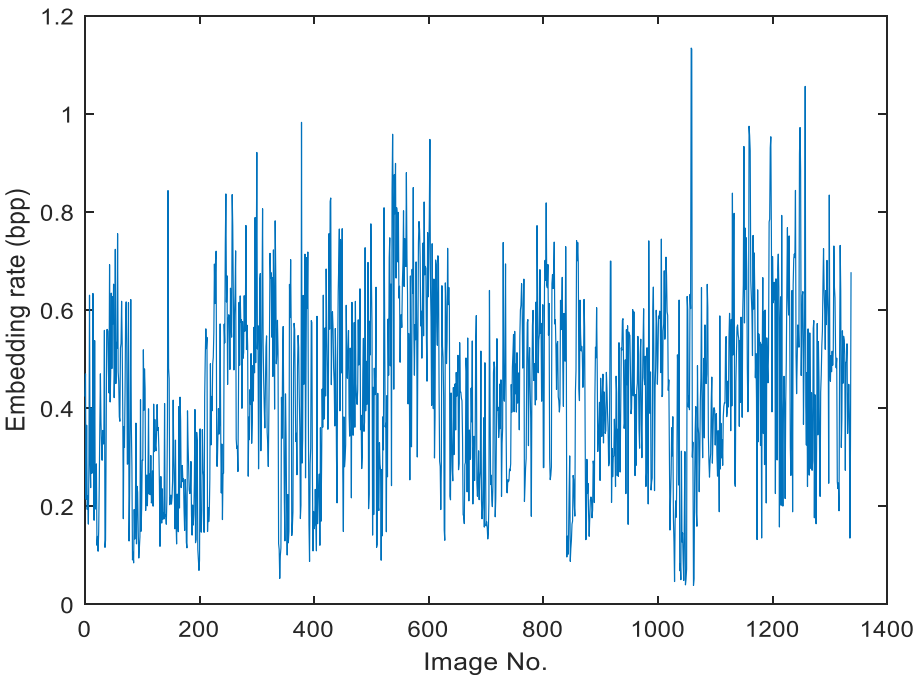


Fig. 10. ERs for UCID images

Table 3 Comparison among four methods in terms of PSNR and maximum ER

Method	Parameter setting	Lena		Couple		Boat		Man		Peppers		Baboon	
		PSNR (dB)	ER (bpp)	PSNR (dB)	ER (bpp)	PSNR (dB)	ER (bpp)	PSNR (dB)	ER (bpp)	PSNR (dB)	ER (bpp)	PSNR (dB)	ER (bpp)
Li et al. (2018)	$T = 1$	51.17	0.338	49.83	0.224	48.36	0.560	50.77	0.586	50.20	0.260	n/a	n/a
Xu and Su (2019)	$\beta = 0$	31.55	0.221	28.06	0.155	29.27	0.154	29.97	0.194	27.36	0.196	24.80	0.078
	$\beta = 1$	30.88	0.377	27.22	0.279	28.72	0.272	29.38	0.340	25.38	0.342	24.47	0.144
Xu and Wang (2016)	$T_n = -1$ $T_p = 0$	49.99	0.176	49.88	0.109	49.86	0.147	49.77	0.124	49.83	0.141	49.58	0.056
	$T_n = -1$ $T_p = 1$	45.98	0.279	46.29	0.164	45.83	0.223	45.88	0.171	45.85	0.200	45.60	0.083
	$T_n = -2$ $T_p = 1$	44.88	0.345	43.98	0.213	45.86	0.266	44.15	0.213	44.23	0.252	43.84	0.103
Proposed	$Q = 2$	58.68	0.176	60.70	0.111	60.73	0.111	59.17	0.158	59.51	0.146	63.65	0.056
	$Q = 3$	53.46	0.279	55.49	0.176	55.52	0.175	53.94	0.250	54.28	0.231	58.43	0.089
	$Q = 4$	50.24	0.352	52.27	0.222	52.30	0.221	50.74	0.316	51.07	0.292	55.19	0.113

n/a not available

Q-ary format. Note that our method belongs to RRBE, and if we encrypt the message before the embedding phase, all we need to do is extract the encrypted message and then decrypt this message in a reverse manner. The peak points T_{p} and T_{n} from the histogram of $\tilde{E}(i,j)$ are then searched for.

Step 2: Shift the histogram of $\tilde{E}(i,j)$ to vacate space for data hiding as

$$\tilde{E}'(i,j) = \begin{cases} \tilde{E}(i,j) + Q{-}1, \text{if } \tilde{E}(i,j) > T_{\mathrm{p}}, \\ \tilde{E}(i,j){-}Q + 1, \quad \text{if } \tilde{E}(i,j) < T_{\mathrm{n}}, \\ \tilde{E}(i,j), \quad \text{otherwise}. \end{cases} \tag{7}$$

where $\tilde{E}'(i,j)$ stands for the shifted value for data hiding. Note that a new location map $\mathbf{O_2}$ is needed to record $\tilde{E}'(i,j)$ with underflow/overflow issues. $\mathbf{O_2}$ can then be compressed and embedded into $Y''(i,j)$ via an MSB replacement, similar to $\mathbf{O}$ and $\mathbf{O_1}$.

Step 3: Embed the message as

$$\tilde{E}''(i,j) = \begin{cases} \tilde{E}'(i,j) + W, \quad \text{if } \tilde{E}'(i,j) = T_{\mathrm{p}}, \\ \tilde{E}'(i,j){-}W, \quad \text{if } \tilde{E}'(i,j) = T_{\mathrm{n}}, \\ \tilde{E}'(i,j), \quad \text{otherwise}, \end{cases} \tag{8}$$

where $\tilde{E}''(i,j)$ denotes the ultimate value after data hiding, and W stands for the Q-ary message. Finally, the marked encrypted image is composited by $Y''(i,j)$ and $\tilde{E}''(i,j)$.

For the ease of understanding, Fig. 4a, b shows two simple examples of the encryption and data embedding processes. In Fig. 4a, we suppose a 3×3 block through the prediction process, in which the pixels valued "1" and "0" are determined to carry additional data. We first apply K_{EY1} to shuffle the prediction errors, and then XOR the block with a random matrix generated by K_{EY2}. Finally, we obtain an encrypted version of this block. Figure 4b illustrates the schematic diagram during the data hiding process when Q is set to 2, 3, or 4. Pixels with values that are not 1 and 0 are shifted by $(Q-1)$ to reserve space for data hiding and are denoted by the dashed lines in Fig. 4(b). Moreover, the pixels valued "1" and "0" are modified to carry additional Q-ary message data, which have been shuffled by K_{EY3} in advance to guarantee the data security, as shown in Fig. 4b by the solid lines.

2.4 Data extraction and image recovery

Because the proposed scheme is separable, the restore process can be divided into two cases, which are described as follows.

Case #1: Data extraction before image recovery

To extract the data correctly, Q, T_{p}, and T_{n} are essential, and all these parameters can be observed from the histogram of $\tilde{E}''(i,j)$. In conjunction with $\mathbf{O_2}$, which can be extracted from $Y''(i,j)$, we can remove the pixels with overflow/underflow issues that may influence the data extraction. Then, W can be extracted as

$$W = \begin{cases} \tilde{E}''(i,j){-}T_{\mathrm{p}}, \text{if } \tilde{E}''(i,j) \in [T_{\mathrm{p}}, T_{\mathrm{p}} + Q{-}1], \\ T_{\mathrm{n}}{-}\tilde{E}''(i,j), \text{if } \tilde{E}''(i,j) \in [T_{\mathrm{n}}{-}Q + 1, T_{\mathrm{n}}]. \end{cases} \tag{9}$$

Eventually, we use K_{EY3} to restore the original message data.

After data extraction, the recovery process can be divided into four steps as

Step 1: Recover the histogram to the condition before embedding, as

Table 4 Comparisons among four methods in terms of SSIM

Method	Parameter setting	Lena	Couple	Boat	Man	Peppers	Baboon
Li et al. (2018)	$T=1$	0.9999	0.9998	0.9998	0.9999	0.9999	n/a
Xu and Su (2019)	$\beta=0$	0.9903	0.9744	0.9827	0.9858	0.9820	0.9387
	$\beta=1$	0.9887	0.9689	0.9803	0.9839	0.9717	0.9340
Xu and Wang (2016)	$T_n=-1$ $T_p=0$	0.9999	0.9999	0.9998	0.9999	0.9999	0.9956
	$T_n=-1$ $T_p=1$	0.9998	0.9998	0.9996	0.9998	0.9998	0.9948
	$T_n=-2$ $T_p=1$	0.9997	0.9995	0.9994	0.9996	0.9997	0.9944
Proposed	$Q=2$	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
	$Q=3$	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
	$Q=4$	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999

n/a not available

$$\tilde{E}(i,j)=\begin{cases}\tilde{E}''(i,j), \text{if } \tilde{E}''(i,j)\in(T_{\mathrm{n}},T_{\mathrm{p}}),\\ T_{\mathrm{p}}, \text{ if } \tilde{E}''(i,j)\in[T_{\mathrm{p}},T_{\mathrm{p}}+Q-1],\\ T_{\mathrm{n}}, \text{ if } \tilde{E}''(i,j)\in[T_{\mathrm{n}}-Q+1,T_{\mathrm{n}}],\\ \tilde{E}''(i,j)+Q-1, \text{if } \tilde{E}''(i,j)<T_{\mathrm{n}}-Q+1,\\ \tilde{E}''(i,j)-Q+1, \text{ otherwise.}\end{cases} \tag{10}$$

Step 2 Extract the compressed code of $\mathbf{O}$, $\mathbf{O_1}$, and original MSB plane of $Y'(i,j)$ from the MSB plane of $Y''(i,j)$. We recover them by using the corresponding decompression method. Next, we replace the MSB plane of $Y''(i,j)$ with the MSB plane of $Y'(i,j)$ to recover $Y'(i,j)$. The original reference pixels can then be recovered as

$$Y(i,j)=Y'(i,j)\oplus R(i,j). \tag{11}$$

Step 3: Increase or decrease $\tilde{E}(i,j)$ by one if its corresponding element in $\mathbf{O_1}$ is equal to one. We then process $\tilde{E}(i,j)$ as

$$E''(i,j)=\begin{cases}\tilde{E}(i,j), \text{ if } (\tilde{E}(i,j)=T_{\mathrm{n}} \text{ or } T_{\mathrm{p}}) \text{ and } (O_1(i,j)=0),\\ \tilde{E}(i,j)\oplus R(i,j), \text{ if } (\tilde{E}(i,j)=T_{\mathrm{n}} \text{ or } T_{\mathrm{p}}) \text{ and } (O_1(i,j)=1),\\ \tilde{E}(i,j)\oplus R(i,j), \text{ otherwise,}\end{cases} \tag{12}$$

where $O_1(i,j)$ is an element in $\mathbf{O_1}$. After the stream cipher, we use K_{EY1} to restore $E'(i,j)$.

Step 4: Modify $E'(i,j)$according to $\mathbf{O}$ as

$$E(i,j)=\begin{cases}E'(i,j)-127, \text{ if } (E'(i,j)<0) \text{ and } (O(i,j)=1),\\ E'(i,j)+127, \text{ if } (E'(i,j)>0) \text{ and } (O(i,j)=1),\\ E'(i,j), \text{ otherwise,}\end{cases} \tag{13}$$

where $O(i,j)$ is an element in $\mathbf{O}$. At last, according to Section 2.1, we compute the prediction values $P(i,j)$ and add them to the corresponding $E(i,j)$ so that we can recover the original image without any error.

Case #2 Data extraction after image recovery

First, we implement the method mentioned in Case# 1 to create an image that includes the additional message. It is worth noting that we should create a new location map $\mathbf{O_3}$ to mark the pixels of $\tilde{E}''(i,j)$ that includes the message and keep them unchanged during the entire recovery process. Next, referring to Section 2.1, we compute the prediction errors for the non-reference pixels and use K_{EY1} to re-shuffle them. Then, we can use $\mathbf{O_3}$ to identify the prediction errors whether message data is included or not and extract the message W according to Eq. (9). Finally, the correct message data can be restored via K_{EY3}.

3 Results and discussion

In this section, we present a series of experiments implemented to demonstrate the effect of the proposed method. The standard images we used in our experiments were *Lena*, *Couple*, *Boat*, *Man*, *Peppers*, and *Baboon*, which are shown in Fig. 5. The size of all test images was 512×512. All the experiments were implemented on a personal computer with an Intel® Core™ i7-4720HQ, GTX 960M and 12 GB RAM.

3.1 Encryption security

Security is a significant indicator of justifying a method for RDHEI. As the owner wants only the intended recipient to perceive the information in the encryption image, we must guarantee the imperceptibility of the encrypted image. In this section, to demonstrate

Table 5 Comparison among four methods in terms of SSIM, PSNR, and ER for UCID

Method	SSIM	PSNR (dB)	ER (bpp)
Xu and Wang (2016)	0.9988	45.17	0.337
Li et al. (2018)	0.9962	35.73	0.858
Xu and Su (2019)	0.9764	27.41	0.365
Proposed	0.9997	49.59	0.433

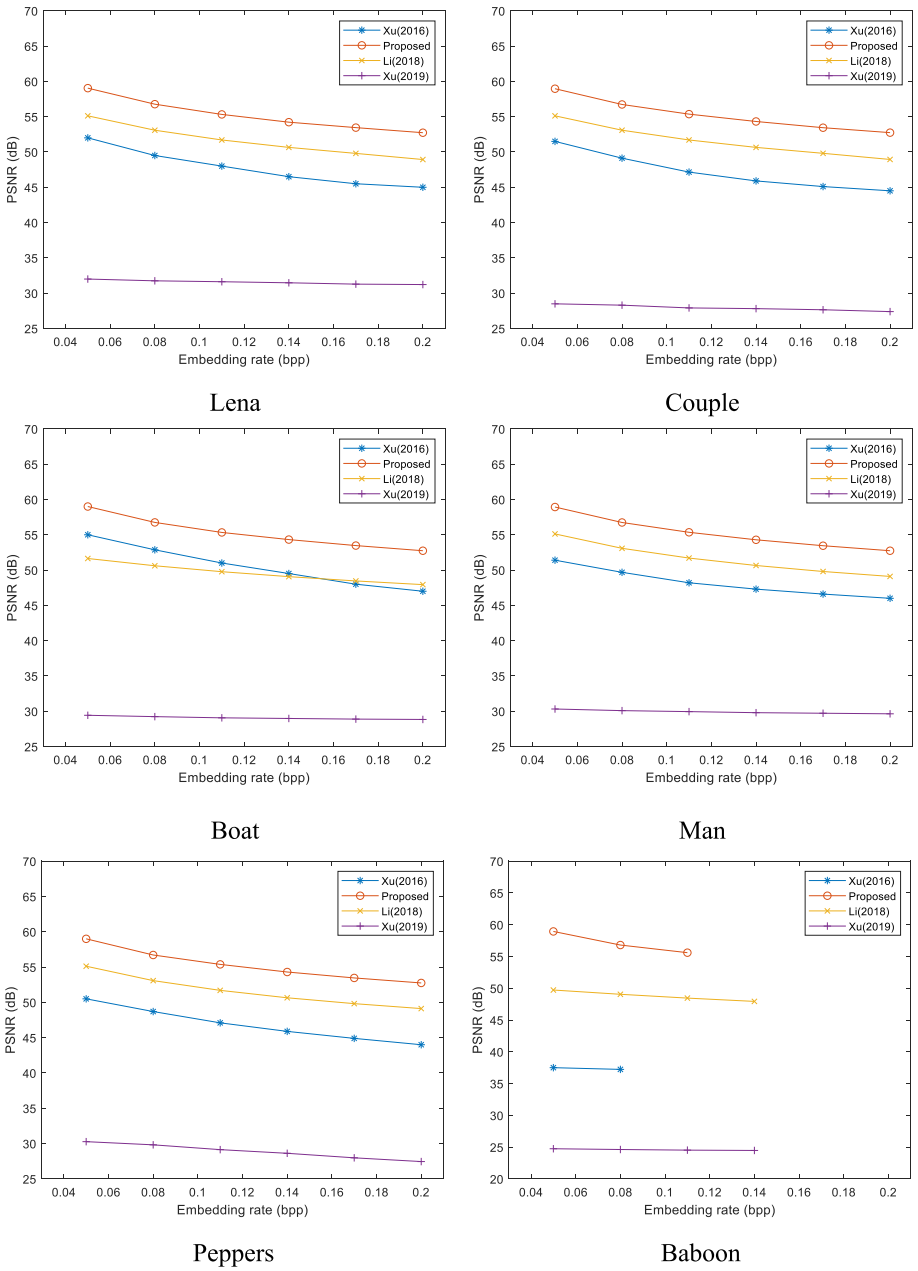


Fig. 11 Performance comparisons between the methods [18, 20, 33], and the proposed method in terms of ER versus PSNR

encryption security, we compare our encryption results with Xu and Wang's work [18] and Li et al.'s work [20] in the two following aspects.

First, a comparison was carried out via subjective visual effects. Figure 6 shows the encryption results of our work on six test images. It can be observed in Fig. 6 that we perceive almost no useful information from the encrypted images. Figures 7 and 8 show the encryption results of [18, 20], respectively. As can be observed in Fig. 7, there are still some contour lines of the original images in the encryption domain, which is a fatal weakness for [18]. It should be pointed out that although the imperceptibility of the encryption images in [20] is similar to ours in Fig. 8, the shuffling strategy applied in [20] was based on block shuffling, and it is more likely that attackers can perceive the contours via a mathematical analysis of an image encrypted with block shuffling than one encrypted with our proposed method, which uses a shuffling strategy based on pixels.

In addition, we used the peak signal-to-noise ratio (PSNR) as an objective evaluation index to demonstrate the imperceptibility of the encryption result. Table 1 lists

Table 6 Operation time comparisons among four methods during the embedding process. (unit:s)

	Xu and Wang (2016)	Li et al. (2018)	Xu and Su (2019)	Proposed
Lena	0.0664	0.4456	0.3164	0.1282
Couple	0.0641	0.4021	0.3199	0.1067
Boat	0.0738	0.3448	0.3435	0.1158
Man	0.0758	0.4156	0.3246	0.1308
Peppers	0.0770	0.4269	0.3060	0.1314
Baboon	0.0918	n/a	0.3055	0.1273

the PSNR results for the three methods. As can be observed from Table 1, the PSNR values from [20] indicate a performance similar to the method from [18]; however, the PSNR results of our work are approximately 75% of the other two methods. From comparisons of the above two aspects, our method performs the best in terms of security.

In addition to a comparison of the security level, we also compared the computing complexity among the three methods, and we used operation time as a metric to evaluate the complexity, as is shown in Table 2. From Table 2, we can find that the method from [18] spent approximately 0.3 s on each test image to complete the encryption process, which is nearly half the time of the method from [20] and the proposed method. However, according to the analysis of the security level, the method in [18] leaves the contour lines in the encryption version of all test images. Moreover, the average operation times for the proposed method and [20] are at the same level. To summarize, our method strikes a satisfactory balance between security and computational complexity.

3.2 Embedding capacity

The embedding capacity is another significant indicator that identifies whether an RDHEI method is efficient. In order to justify the embedding efficiency of the proposed method, we added three more test images, i.e., *Barbara*, *Lake*, and *Airplane*, to our experiments, and we chose three different Q values, i.e., $Q = 2, 3, 4$, to compute the maximum ER for each image with each Q. Figure 9 shows the ER performance of the proposed method on nine test images.

It can be observed from Fig. 9 that the capacity of our method mainly depends on the prediction accuracy and the parameter Q. Specifically, with the increase in Q, the ER also increases. Except for Baboon, the ERs of all other images are higher than 0.2 bits per pixel (bpp), which represents a relatively satisfying result for RDHEI, when Q is set to 4. As our method depends on the prediction to reserve space for data hiding, images with a complex texture, such as Baboon, which has neighboring pixels that vary greatly from one another, cannot be predicted precisely with the existing prediction algorithms so that limited space can be reserved.

For a comprehensive evaluation of the ER of the proposed method, we tested our method on the uncompressed color image database (UCID) [32] with the setting $Q = 4$. Figure 10 shows the ER for each image, and the average ER for the UCID images is 0.433 bpp, which is satisfactory for most embedding applications.

3.3 Comparison and discussion

In this section, we compare the ERs and PSNRs of the recovered images that included additional messages among the four methods. Note that, in this section, we also consider a method from [33], which was further development of [18].

First, we compared the PSNRs and maximum ERs among the four methods with different parameter settings for each method. Their corresponding results are shown in Table 3, where "n/a" strands for "not available, " for the current parameter setting. From Table 3, for the method from [20], the average values of PSNRs and ERs are 50.07 dB and 0.394 bpp, respectively, when T is set to 1. For the method from [33], when β is set to 0, the average PSNR and ER are 29.24 dB and 0.184 bpp, respectively; when β is set to 1, the average PSNR and ER are 28.32 dB and 0.322 bpp, respectively. For the method from [18], when T_n is set to −1 and T_p is set to 0, the average PSNR and ER are 49.87 dB and 0.140 bpp, respectively; when T_n is set to −1 and T_p is set to 1, the average PSNR and ER are 45.97 dB and 0.207 bpp, respectively; when T_n is set to −2 and T_p is set to 1, the average PSNR and ER are 44.62 dB and 0.257 bpp, respectively. For our method, the average values of PSNRs and ERs are 59.76 dB and 0.140 bpp when Q is set to 2, 54.54 dB and 0.222 bpp when Q is set to 3, and 51.32 dB and 0.280 bpp when Q is set to 4. Note that for a fair comparison, five test images, not including Baboon, are used to compute the average PSNRs and ERs with different parameters for all four methods, because the additional message could not be embedded in Baboon in the method from [20]. Comparing the average values between the proposed method and that from [20], the embedding performance of the method from [20] is better than the proposed method when Q is set to 2 or 3, and the embedding performance of the proposed method is close to the method from [20] when Q is set to 4. However, it is worth pointing out that the method proposed in [20] is not entirely separable, and the data can be extracted correctly only from the encryption domain. On the contrary, the proposed method is perfectly separable, and the data can be extracted correctly not only from the encryption domain but also from the plaintext domain. Comparing the average values between the proposed method and the method in [33], we find that the

average ER of the method from [33] is slightly higher than the ER of our method. However, as the cost of higher ER, the average PSNR of [33] is significantly lower than that of the other methods. By comparing the average values between the proposed method and [18], it can be observed that the embedding performances of both methods are similar at a low ER. However, with the increment in the parameters, the increment of the ER of the proposed method is higher than that of the method from [18] at a similar PSNR level.

Because PSNR is not sufficient to show the visual quality of a decrypted image, we also evaluated our method through another metric, SSIM, which stands for the structural similarity between a pair of decrypted and original images. The experimental results are shown in Table 4. From Table 4, for [20], the average value of SSIM is 0.9999 when T is set to 1. For [33], the average values of SSIM are 0.9830 when β is set to 0 and 0.9787 when β is set to 1. For [18], the average values of SSIM are 0.9999 (when T_n and T_p are set to −1 and 0, respectively), 0.9998 (when T_n and T_p are set to−1 and 1, respectively), and 0.9996 (when T_n and T_p are set to −2 and 1). For the proposed method, the average values of SSIM are 0.9999, whether Q is set to 2, 3, or 4. Note that here, for a fair comparison, five test images, not including Baboon, were used to compute the average values. From the results in Table 4, [18, 20], and the proposed method have the same level in terms of SSIM and achieve a better performance than [33].

In addition, we also tested the four methods on UCID, and the average values of SSIM, PSNR, and ER for different methods are shown in Table 5. Note that the parameters T, β, T_n, T_p, and Q were set to 1, 1, − 2, 1, and 4, respectively. As can be observed in Table 5, our method achieves the highest average values in terms of SSIM and PSNR, while the method from [20] achieves the highest average ER.

Next, we compared performance in terms of ER versus PSNR among the four methods. Figure 11 illustrates the performance comparisons among the four methods in terms of ER versus PSNR. From Fig. 11, it is evident that all the PSNR curves of the proposed method are higher than the other three methods under any ER value. It is worth pointing out that the methods in [18, 33], and the proposed method can generate a directly decrypted image while keeping all the embedded data, but [20] cannot keep the embedded data in the directly decrypted image.

In addition to comparisons of the capacity and visual quality of the four methods, we also compared the computing complexity of the four methods during the embedding process. We used operation time as a metric to evaluate the computing complexity, and set T, β, T_n, T_p, and Q to 1, 1, − 2, 1, and 4, respectively. The results are shown in Table 6. The average operation times for [18, 20, 33], and the proposed method are 0.0714 s, 0.4070 s, 0.3220 s, and 0.1226 s, respectively. From Table 6, the method from [20] consumes the most time because the embedding strategy in this method is based on bit operations. Moreover, the operation time of the method from [33] is also higher than the method from [18] and the proposed method, due to the fact that the embedding strategy in the method from [33] is based on module operations. However, both the method from [18] and the proposed method attain efficient results, with both average embedding times lower than 0.2000 s. Note that the reason why our method requires more time than the method from [18] is because of the overflow/underflow processing involved in our embedding process.

4 Conclusions

In this study, a separable multiary RDHEI method that involved prediction and replacement, image encryption, data hiding, data extraction, and recovery of the original image was developed. A stream cipher and shuffling were used to encrypt the original image so that we could obtain a high-security level. The data hider could embed additional data into the image without knowing the original image because we had already vacated space for the data hider. It is worth pointing out that, in our work, the data hider is also capable of adaptively modifying the reserved space according to the actual embedding demand. After the receiver acquires the encrypted image with the additional data, the receiver can extract the entire additional message either from the encryption domain or the plaintext domain because our scheme is entirely separable. It should be noted that there are two limitations in our method: (1) the amount of overflow/underflow data increases dramatically with an increase in Q, and (2) the ER of our method relies heavily on the accuracy of the predictions. Both issues will be addressed in our future research.

Abbreviations

LSB: least significant bit; CP: Cloud provider; RDHEI: Reversible data hiding in encryption domain; RDH: Reversible data hiding; LC: Lossless compression; HS: Histogram shift; DE: Difference expansion; PEE: Prediction error expansion; PEH: Prediction-error histogram; RRBE: Reserving a room before encryption; VRAE: Vacating space after encryption; ER: Embedding rate; CRT: Chinese remainder theorem; RST: Redundant space transfer; RLC: Run-length coding; MSB: Most significant bit; XOR: Exclusive or; PSNR: Peak signal-to-noise ratio; bpp: Bits per pixel

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments.

Authors' contributions

MY designed the scheme and wrote the manuscript. YL and HS did the experiments. HY supervised the algorithm design and experiments, and modified the manuscript. TQ offered the suggestion. All author(s) read and approved the final manuscript.

Authors' information

Mingji Yu is currently working toward a B.S. degree at University of Shanghai for Science and Technology. His research interests include reversible data hiding.

Yuchen Liu is currently working toward a B.S. degree at University of Shanghai for Science and Technology. His research interests include reversible data hiding.

Hu Sun is currently working toward a B.S. degree at University of Shanghai for Science and Technology. His research interests include reversible data hiding.

Heng Yao received the B.S. degree from Hefei University of Technology, China, in 2004, the M.S. degree from Shanghai Normal University, China, in 2008, and the Ph.D. degree from Shanghai University, China, in 2012. Currently, he is with School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, China. His research interests include digital forensics, data hiding, image processing, and pattern recognition. received the B.S. degree from Hefei University of Technology, China, in 2004, the M.S. degree from Shanghai Normal University, China, in 2008, and the Ph.D. degree from Shanghai University, China, in 2012. Since 2012, he has been with School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, China, where he is currently an Associate Professor. His research interests include digital forensics, data hiding, image processing, and pattern recognition. He has contributed more than 30 international peer-reviewed journal papers.

Tong Qiao received the B.S. degree in Electronic and Information Engineering in 2009 from Information Engineering University, Zhengzhou, China, and the M.S. degree in Communication and Information System in 2012 from Shanghai University, Shanghai, China, and the Ph.D. degree in System Optimization and Dependability in 2016 from University of Technology of Troyes, Laboratory of System Modeling and Dependability, Troyes, France. The Ph.D. degree is funded by China Scholarship Council with UT-INSA project. He is currently an assistant professor at Hangzhou Dianzi University, School of Cyberspace. His current research interests focus on steganalysis and digital image forensics.

Funding

This work was supported in part by the National Natural Science Foundation of China (61702332, 61702150), and the Innovation and entrepreneurship training program for college students of USST (XJ2019063).

Availability of data and materials

Not applicable.

Ethics approval and consent to participate

Not applicable.

Consent for publication

All authors agree to publish this paper in this journal.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China. [2]School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China.

Received: 15 October 2019 Accepted: 11 March 2020
Published online: 19 April 2020

References

1. S. Li, X. Zhang, Toward construction based data hiding: from secrets to fingerprint images. IEEE Trans on Image Process **28**(3), 1482–1497 (2019)
2. J. Tao, S. Li, X. Zhang, Z. Wang, Robust image steganography. IEEE Trans Circuits Syst Video Technol **29**(2), 594–600 (2019)
3. C. Qin, Q. Zhou, F. Cao, J. Dong, X. Zhang, Flexible lossy compression for selective encrypted image with image inpainting. IEEE Trans Circuits Syst Video Technol **29**(11), 3341–3355 (2019)
4. C. Qin, P. Ji, C.C. Chang, J. Dong, X. Sun, Non-uniform watermark sharing based on optimal iterative BTC for image tampering recovery. IEEE Multimedia **25**(3), 36–48 (2018)
5. M.U. Celik, G. Sharma, A.M. Tekalp, E. Saber, Lossless generalized-LSB data embedding. IEEE Trans Image Process **14**(2), 253–266 (2005)
6. Z.C. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding. IEEE Trans Circuits Syst Video Technol **16**(3), 354–362 (2006)
7. C. Qin, C.C. Chang, Y.H. Huang, L.-T. Liao, An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism. IEEE Trans Circuits Syst Video Technol **23**(7), 1109–1118 (2012)
8. J. Tian, Reversible data embedding using a difference expansion. IEEE Trans Circuits Syst Video Technol **13**(8), 890–896 (2003)
9. A.M. Alattar, Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans Image Process **13**(8), 1147–1156 (2004)
10. V. Sachnev, H.J. Kim, J. Nam, S. Suresh, Y.Q. Shi, Reversible watermarking algorithm using sorting and prediction. IEEE Trans Circuits Syst Video Technol **19**(7), 989–999 (2009)
11. Y. Hu, H.K. Lee, J. Li, DE-based reversible data hiding with improved overflow location map. IEEE Trans Circuits Syst Video Technol **19**(2), 250–260 (2008)
12. D.M. Thodi, J.J. Rodríguez, Expansion embedding techniques for reversible watermarking. IEEE Trans. Image Process. **16**(3), 721–730 (2007)
13. H. Yao, C. Qin, Z. Tang, Y. Tian, Guided filtering based color image reversible data hiding. J Vis Commun Image R **43**, 152–163 (2017)
14. H. Yao, X. Liu, Z. Tang, Y.C. Hu, C. Qin, An improved image camouflage technique using color difference channel transformation and optimal prediction-error expansion. IEEE Access **6**, 40569–40584 (2018)
15. K. Ma, W. Zhang, X. Zhao, N. Yu, F. Li, Reversible data hiding in encrypted images by reserving room before encryption. IEEE Trans Inf Forensics Security **8**(3), 553–562 (2013)
16. W.M. Zhang, K. Ma, N.H. Yu, Reversibility improved data hiding in encrypted images. Signal Process. **94**, 118–127 (2014)
17. X.C. Cao, L. Du, X.X. Wei, D. Meng, X.J. Guo, High capacity reversible data hiding in encrypted images by patch-level sparse representation. IEEE Trans Cybern **46**(5), 1132–1143 (2016)
18. D.W. Xu, R.D. Wang, Separable and error-free reversible data hiding in encrypted images. Signal Process. **123**, 9–21 (2016)
19. S. Yi, Y.C. Zhou, Binary-block embedding for reversible data hiding in encrypted images. Signal Process. **133**, 40–51 (2017)
20. Q. Li, B. Yan, H. Li, N. Chen, Separable reversible data hiding in encrypted images with improved security and capacity. Multimed. Tools Appl. **77**(23), 30749–30768 (2018)
21. Z.X. Qian, X.P. Zhang, G.R. Feng, Reversible data hiding in encrypted images based on progressive recovery. IEEE Signal Process Lett **23**(11), 1672–1676 (2016)
22. M. Li, Y. Li, Histogram shifting in encrypted images with public key cryptosystem for reversible data hiding. Signal Process. **130**, 190–196 (2017)
23. S. Agrawal, M. Kumar, Mean value based reversible data hiding in encrypted images. Optik **130**, 922–934 (2017)
24. P. Singh, B. Raman, Reversible data hiding for rightful ownership assertion of images in encrypted domain over cloud. AEU-Int J Electron C **76**, 18–35 (2017)
25. D. Xiao, X.P. Xiang, H.Y. Zheng, Y. Wang, Separable reversible data hiding in encrypted image based on pixel value ordering and additive homomorphism. J Vis Commun Image R **45**, 1–10 (2017)
26. Z.L. Liu, C.M. Pun, Reversible data-hiding in encrypted images by redundant space transfer. Inf. Sci. **433**, 188–203 (2018)
27. C. Qin, W. Zhang, F. Cao, X.P. Zhang, C.C. Chang, Separable reversible data hiding in encrypted images via adaptive embedding strategy with block selection. Signal Process. **153**, 109–122 (2018)
28. F. Khelifi, T. Brahimi, J.G. Han, X.L. Li, Secure and privacy-preserving data sharing in the cloud based on lossless image coding. Signal Process. **148**, 91–101 (2018)
29. H.T. Wu, Y.M. Cheung, Z.Y. Yang, S.H. Tang, A high-capacity reversible data hiding method for homomorphic encrypted images. J Vis Commun Image R **62**, 87–96 (2019)
30. Y. Fu, P. Kong, H. Yao, Z. Tang, C. Qin, Effective reversible data hiding in encrypted image with adaptive encoding strategy. Inf. Sci. **494**, 21–36 (2019)
31. C. Qin, X. Qian, W. Hong, X. Zhang, An efficient coding scheme for reversible data hiding in encrypted image with redundancy transfer. Inf. Sci. **487**, 176–192 (2019)
32. G. Schaefer, M. Stich, UCID—an uncompressed colour image database. IS&T/SPIE Electronic Imaging **5307**, 472–480 (2004)

33. D.W. Xu, S.B. Su, Separable reversible data hiding in encrypted images based on difference histogram modification. Secur Commun Netw **2019**, 7480147 (2019)

Publisher's Note