Multimed Tools Appl (2018) 77:30749–30768
https://doi.org/10.1007/s11042-018-6187-y

Separable reversible data hiding in encrypted images with improved security and capacity

Qi Li[1] · Bin Yan[1] · Hui Li[1] · Na Chen[1]

Received: 8 December 2017 / Revised: 18 May 2018 / Accepted: 22 May 2018 /
Published online: 1 June 2018

Abstract Reversible data hiding (RDH) has to be conducted in the encrypted images when original images are encrypted for privacy protection in some open environments, including the cloud computing. However, the current RDH algorithms in encrypted images with error-free decryption may lead to leakage of image content and low embedding rate. In this paper, a novel RDH algorithm for image in encryption domain is proposed. To improve security, we propose a combined block permutation and a stream cipher into the encryption step, which considers data hiding in later steps. We further increase the embedding rate by proposing bit replacement in prediction error. This scheme has the advantages of built-in embedding flag, error-free decryption and high embedding rate. It can be applied to a wide variety of scenarios: If the recipient has only the data-hiding key, he can extract the hidden data but cannot restore the image; If the recipient has only the image encryption key, he can read the distorted image but cannot extract the hidden data; If the recipient has both keys, he can extract the hidden data and restore the original image completely.

Keywords Personal privacy · Image encryption · Reversible data hiding · Stream encryption · Block permutation

1 Introduction

In recent years, cloud storage and cloud computing are widely used. Since the servers and the users are usually separated from each other, so the data from the users must be uploaded to the cloud server for storage or processing. These data may include sensitive and private information such as personal ID or biometric data. Thus, personal privacy issues have

✉ Bin Yan
yanbinhit@hotmail.com

[1] College of Electronics, Communication and Physics, Shandong University of Science and Technology, Qingdao, 266590, People's Republic of China

become a widespread concern. To protect privacy, the data are usually encrypted before uploading to the cloud server. Without knowing the content of the received data, the server may also need to add additional information for labelling and authentication, *etc*.

A typical application scenario in cloud storage is illustrated in Fig. 1. The user data is an image, which is encrypted in user device 1 (such as a desktop computer), and then is transmitted to cloud server 1. The cloud server 1 embeds a tag into the encrypted image to label the source of this file. The resulting file may need to be transferred to another server, the server 2. In a later time, the user may need to download the file from server 2 to another device, i.e., the device 2, such as his/her cell phone. Then, server 2 needs to extract the hidden data and recover the original encrypted image before sending it to the user. After receiving the recovered image, the user device 2 can decrypt the image. This scenario calls for reversible data hiding (RDH) in encryption domain.

RDH has been proposed in recent years, which can embed the secret data into the meaningful cover media imperceptibly, especially for digital images [14]. It is widely used in many fields such as medical image, copyright certification, activity recognition [5, 6], *etc*. The current algorithms for RDH can be classified into lossless compression [1], difference expansion [16], pixel prediction [9], and histogram shifting [11]. These algorithms are designed for plain image without encryption.

Recently, a lot of works for RDH in encrypted images have been presented [2–4, 10, 13, 15, 17–20]. However, the current algorithms that embed external data into the encrypted images with error-free decryption may lead to leakage of image content and low embedding rate. In this work, we propose a novel RDH scheme in encrypted images, which can provide higher security level and embedding rate.

The main contributions of this work can be summarized as follows:

- A block permutation algorithm adapted to prediction error encryption and RDH.
- A data embedding algorithm by utilizing the extra bits from the reduced dynamic range of the prediction error.

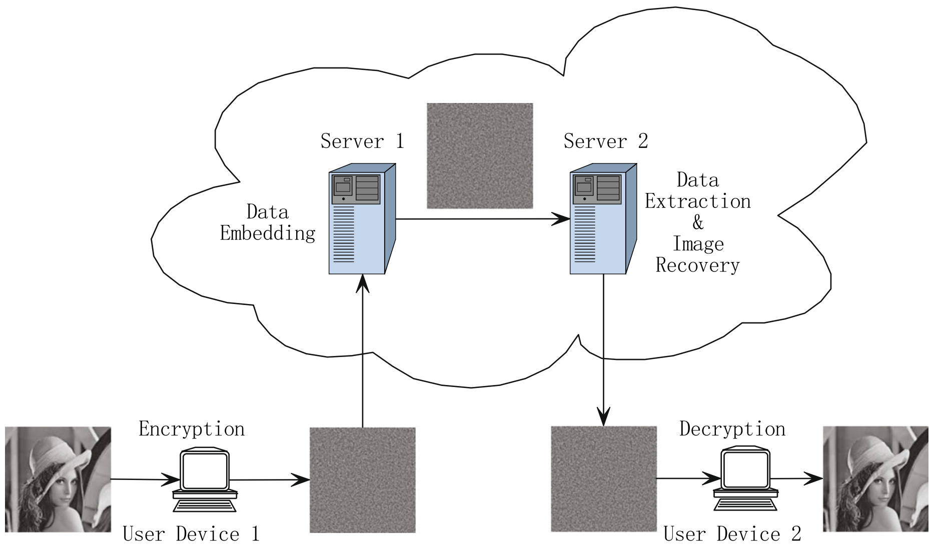


Fig. 1 A typical application scenario of RDH in encryption domain

This paper is organized as follows. After the introduction in Section 1, the related works are reviewed in Section 2. Then, we present our proposed algorithm in Section 3. The experimental results and analysis are discussed in Section 4. Finally, we conclude the paper in Section 5.

2 Related works

Extending RDH for encrypted image is not a trivial task since most RDH relies on local smoothness assumption of the host image. But such an assumption is destroyed in encrypted image since encrypted images are noise-like. In 2011, Zhang first proposed a RDH algorithm in encrypted domains. In Zhang's work [19], the encrypted image was divided into non-overlapping blocks. Then, in each block, external data are embedded by flipping the three least significant bits of half of the pixels. The receiver extracts the data according to the spatial correlation of the natural image. Hong et al. [2] further improved Zhang's [19] algorithm by modifying its smoothness measurement, which reduces the error rate during data extraction. In another work, Qin and Zhang [15] presented an algorithm with capability of image content protection. It only alters the three LSBs of fewer pixels by the elaborate selection. This algorithm has a smaller error rate than the algorithm of [19] and [2], but data extraction and image decryption in these algorithms are carried out at the same time. The receiver must decrypt the image before extracting the hidden data. So, it is not suitable for the application scenario in Fig. 1. The scenario in Fig. 1 requires that the data extraction and decryption of image to be separable.

Zhang [20] proposed a separable embedding method in 2012. The cloud server embeds the external data by compressing the least significant bit in the encrypted image. Qian and Zhang [13] improved the algorithm of [20] by using the distributed source coding. But their algorithm still cannot restore the original image completely.

In [10], Ma et al. proposed a method for embedding external data in encrypted image by vacating room before encryption. To vacate room, this algorithm embeds LSBs of some pixels into other pixels using traditional RDH method. Ma's algorithm is free from error during data extraction and image reconstruction. The embedding capacity in the encrypted image is also improved. However, it requires the sender to implement an additional RDH algorithm. It may not work if the sender is lack of the knowledge of RDH. Homomorphic encryption is also adopted to RDH in the encrypted domain. In [4], the original image is first encrypted by using Paillier encryption. Then, additional data can be embedded easily because of the homomorphic property of the Paillier cipher. Although it achieves good performance in both embedding capacity and quality of the decrypted images, it still suffers from high computational complexity, and the encrypted image is seriously expanded.

Recently, Xu et al. proposed a separable and error-free RDH in encrypted domain [18]. Joint error expansion and histogram shifting are applied to a partially encrypted image. One drawback of this algorithm is the leakage of image content due to partial encryption. In addition, the embedding rate is not high enough. This is the motivation of our work.

This paper aims to improve the security and embedding rate of Xu's algorithm. To improve security, a block permutation algorithm is proposed so that the local dependence in the encrypted image is further mitigated without interfering with the later RDH. To improve the embedding rate, the extra bits from the reduced dynamic range of the prediction error are replaced by hidden data. This algorithm has the advantages of built-in embedding flag, high embedding rate and high security. Furthermore, it can be applied to different application scenarios where the receiver may have either the encryption key or data-hiding key, or both.

There are two main differences between Xu's algorithm and ours. Different from simple partial stream encryption in Xu's algorithm, a block permutation adapted to prediction error encryption and RDH is used to prevent the leakage of image texture. Different from histogram shifting and prediction error expansion in the data embedding procedure of Xu's scheme, a bit replacement in prediction error is proposed to improve the embedding rate.

3 Proposed scheme

In this section, we present our proposed algorithm for RDH in encrypted domain. The overall block diagram of the proposed scheme is illustrated in Fig. 2. As shown in Fig. 2, three parties are involved, the image owner, the cloud server and the receiver.

The image owner encrypts the original plain image to avoid image leakage. This is done by a combination of steam cipher and block scrambling. Then, the encrypted image is sent to the cloud server. The cloud server embeds external data (such as authentication information) into the secret image without knowing the original image content. The secret image containing hidden data is transmitted to the receiving end. The receiver can process it according to the different application requirements, where the receiver may have only the encryption key or the data-hiding key, or both keys. Figure 3a shows a block diagram of image encryption and data hiding. Figure 3b shows the corresponding data extraction and image decryption at the receiving end.

The current algorithm supports three typical situations at the receiving end. If the recipient only wants to authenticate the image or extract the hidden data, he only needs to use the data-hiding key (K_d) to extract the authentication information. If the recipient only wants to understand the contents of the image in general, and does not care about the embedded data, then he can use the image encryption keys (K_p and K_s) to decrypt the image. Then a decrypted image that is close to the original image can be obtained. If the recipient needs to extract the authentication information and the image content, he can first extract the data using the data-hiding key (K_d). Then, he can restore the original image according to the image encryption keys (K_p and K_s). Figure 4 shows the three cases that are described above.

Next, we describe the various steps outlined in Fig. 3.

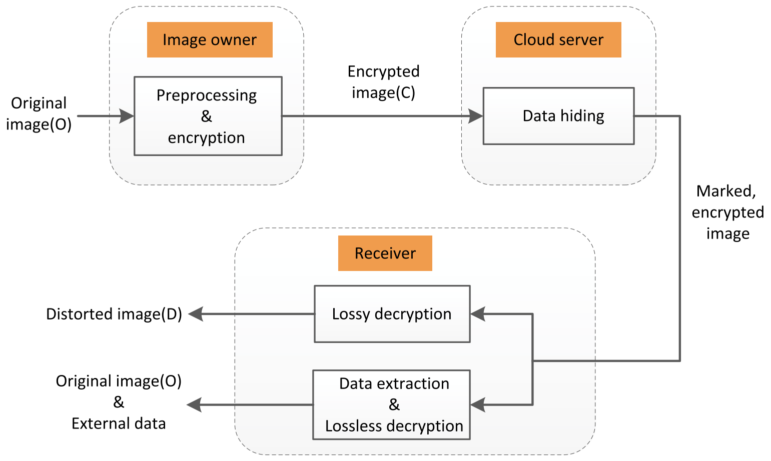


Fig. 2 Block diagram of the proposed scheme

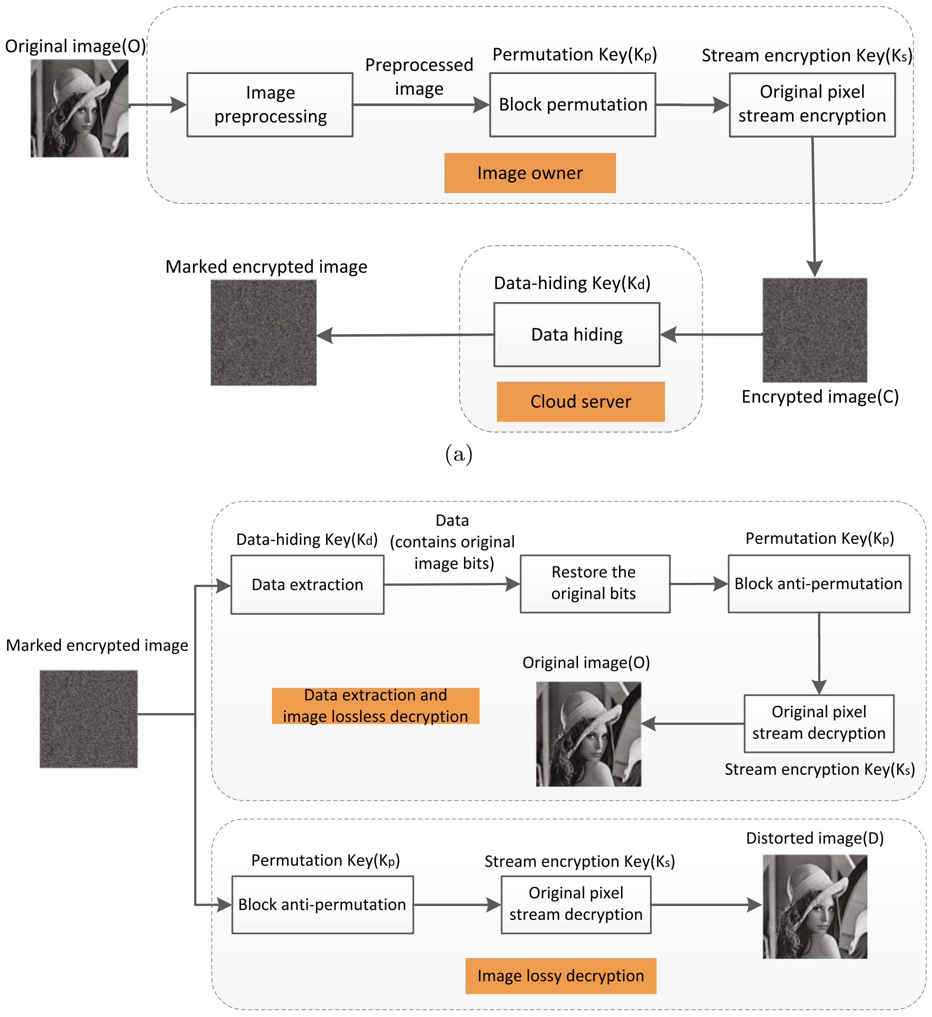


Fig. 3 The block diagrams of the proposed scheme: **a** Image encryption and data hiding, and **b** data extraction and image decryption

3.1 Image preprocessing

This step is done before the image encryption and is implemented by the image owner. Its purpose is reducing the dynamic range of some pixels to allocate space for the later RDH.

Assuming that the original image is an 8-bit quantized gray-scale image with size $M \times N$. The pixel values fall into the range [0,255]. The image is divided into a set of non-overlapping blocks with the size, say 3×3, as shown in Fig. 5. The partition starts from the top left corner and ends at the bottom right corner.

In Fig. 5, each small block corresponds to one pixel. All pixels are divided into two categories, namely, (a) sample pixels (SP) and, (b) non-sample pixels (NSP). The SP is

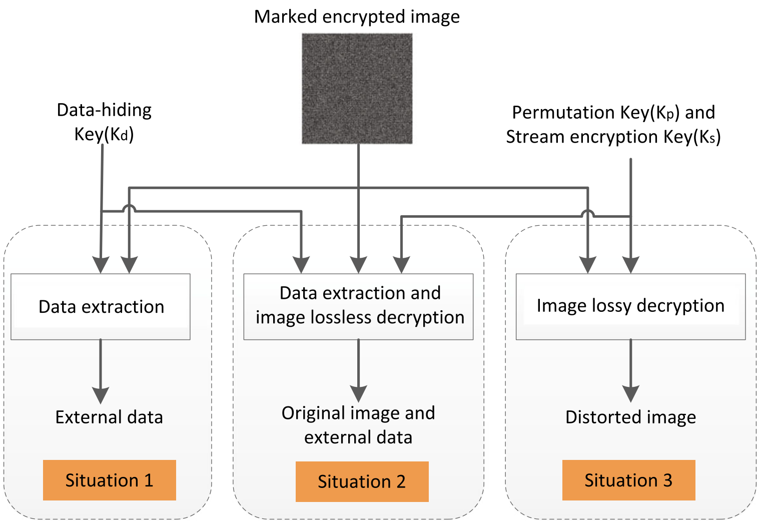
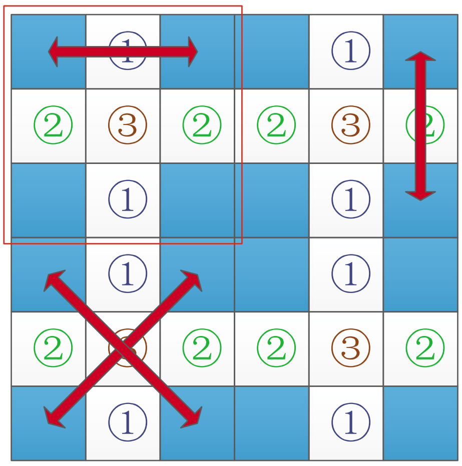


Fig. 4 Three cases at receiving end

made up of all the shaded pixels in the figure, and NSP is made up of all the white pixels in the figure. Prediction is widely used in various applications [7, 8, 12]. Prediction of pixels is also highly important in this work. The pixels belonging to SP are used to predict the NSP. For a 3×3 block, there are four SPs and five NSPs. We record the original pixel in the image as $O(i, j)$, where (i, j) is the pixel position, and $1 \leq i \leq M$, $1 \leq j \leq N$. The pixels marked as '①' in NSP are predicted by two neighborhood pixels in the horizontal direction, and the predicted values are recorded as $O_H(i, j)$. The estimator is:

$$O_H(i, j) = \left\lfloor \frac{O(i, j-1) + O(i, j+1)}{2} \right\rfloor, \tag{1}$$

Fig. 5 Illustration of image preprocessing: block partition and pixel prediction



where $\lfloor x \rfloor$ returns the nearest integer of x towards $-\infty$. The pixels marked as '②' in the NSP are predicted by two neighborhood pixels in the vertical direction, and the predicted values are recorded as $O_V(i, j)$:

$$O_V(i, j) = \left\lfloor \frac{O(i-1, j) + O(i+1, j)}{2} \right\rfloor. \tag{2}$$

The pixels marked as '③' in NSP are predicted by two diagonal neighborhood pixels whose difference are relatively small, and the predicted values are recorded as $O_D(i, j)$. The predictor is:

$$O_D(i, j) = \begin{cases} \left\lfloor \frac{O(i-1,j-1)+O(i+1,j+1)}{2} \right\rfloor, & \text{if } a < b, \\ \left\lfloor \frac{O(i+1,j-1)+O(i-1,j+1)}{2} \right\rfloor, & \text{otherwise,} \end{cases} \tag{3}$$

where

$$\begin{aligned} a &= \left| O(i-1, j-1) - O(i+1, j+1) \right|, \\ b &= \left| O(i+1, j-1) - O(i-1, j+1) \right|. \end{aligned}$$

Here, a smoother pair of diagonal pixels is chosen to improve the accuracy of the predictor. The prediction error $E(i, j)$ is calculated from the predicted value and the NSPs are replaced by the corresponding prediction errors.

As we know, each pixel in the gray-scale image within the range of [0,255] can be represented by an 8-bit binary number. Denote the binary bits of a pixel as $b_0(i, j), b_1(i, j), ..., b_k(i, j)$, where (i, j) is the pixel position and k indicates the k-th bit. Thus, we have:

$$b_k(i, j) = \left\lfloor \frac{O(i, j)}{2^k} \right\rfloor \mod 2, \; k = 0, 1, ..., 7. \tag{4}$$

In order to identify the polarity of the prediction error, we use the method in [18]. The polarity is marked by the most significant bit (MSB). We set the MSB to 1 if the prediction error is negative and set the MSB to 0 if the prediction error is positive. Figure 6 shows an example of the prediction errors 10 and -10. The left most bit, i.e., the MSB is set to 0 for 10 and is set to 1 for -10.

Allocating the MSB for polarity, now only 7 bits can be used to represent the gray value of the prediction error. That is, one can only represent 0(0000000) to 127(1111111). In order to prevent overflow, the values that are less than -127 are mapped back to $[-127, 127]$ by adding 127 to it. The errors that are larger than 127 are subtracted by 127 from it. To help the receiver to recognize these positions, we need a location indicator. The location indicator for the pixel beyond the range of $[-127, 127]$ is marked as 1, otherwise it is marked as 0.

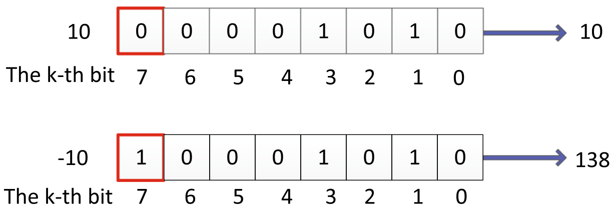


Fig. 6 An example of positive and negative identification of prediction error

Thus, a location map $L(i, j)$ is produced. The mapping and construction of location map is summarized as:

$$\begin{cases} E(i,j) \leftarrow E(i,j) + 127 \text{ and } L(i,j) = 1, & \text{if } E(i,j) < -127, \\ E(i,j) \leftarrow E(i,j) - 127 \text{ and } L(i,j) = 1, & \text{if } E(i,j) > 127, \\ E(i,j) \leftarrow E(i,j) \text{ and } L(i,j) = 0, & \text{if } -127 \leq E(i,j) \leq 127. \end{cases} \tag{5}$$

Most of the prediction error is within the range of [-127,127] because of the local correlation of the image pixels. So the positioning map is mainly composed of ‘0’, which can be compressed to a small number of bits using the lossless compression algorithm, such as run-length code. In order to restore the original image, the lossless compressed location map L will be sent as auxiliary information to the receiver just as done in [18].

It should be noted that the size of block is not limited to 3×3. By dividing the image into 5×5 or 7×7 non-overlapping blocks for processing, the number of pixels belonging to NSP also increases.

3.2 Image encryption

Image encryption is divided into two steps: block permutation and stream encryption. The motivation for using a combined block permutation and a stream cipher is to avoid texture leaks in secret images in Xu’s algorithm. The block permutation disrupts the original position of each block in the image. Stream encryption ensures that the original pixel information in each block is not compromised.

Block permutation: This step permutes all the preprocessed blocks using the permutation key (K_p). We can get a permuted image after this step. It should be noted that this step only permutes the order of the blocks within the image. The pixel position in the block remains unchanged [3]. In the cloud, even if the data hider does not know the permutation key, but he still knows the relative location of the NSP in a block.

Stream encryption: Generate a random number $R(i, j)$ in the range of [0,255] based on the stream encryption key (K_s). The length of $R(i, j)$ is the same as that of the original pixel $O(i, j)$ belonging to SP. And then the bitwise exclusive-or (XOR) operation is performed between each bit plane of $R(i, j)$ and $O(i, j)$. Let $r_k(i, j)$ be the bit plane representation of $R(i, j)$, i.e.,

$$r_k(i,j) = \left\lfloor \frac{R(i,j)}{2^k} \right\rfloor \mod 2, \quad k = 0, 1, ..., 7. \tag{6}$$

Let $b_k(i, j)$ be the bit plane representation of the SP as calculated in (4). Then the encryption for SP can be calculated as:

$$c_k(i,j) = b_k(i,j) \oplus r_k(i,j), \ \ k = 0, 1, 2, \cdots, 7, \forall (i,j) \in \mathcal{S}. \tag{7}$$

where $\mathcal{S}$ is the set of SP in the image. From the encrypted bit planes, we can get the encrypted pixels as:

$$C(i,j) = \sum_{k=0}^{7} c_k(i,j) \cdot 2^k. \tag{8}$$

Eventually, we get the secret image after block permutation and stream encryption. In addition, the stream encryption key (K_s) is also used for encryption if there are edge pixels that cannot be divided into blocks.

3.3 Data embedding

When the cloud server receives the encrypted secret image, external data can be embedded by data hider at the server. First, the data hider encrypts the data according to the data-hiding key (K_d). Then, the prediction error $E(i, j)$ belonging to the NSP in the secret image is sequentially extracted without knowing the original image content. Finally, if the prediction error falls within the range of $[-T, T]$, the additional bits are embedded in a bit-substitution manner. If the prediction error is outside of the range $[-T, T]$, then the pixel is marked. The motivation for using a bit-substitution manner is to increase the embedding capacity. A typical example is given below.

Assuming that $T = 15$, the pixel values within the range of [-15,15] can be represented by the lowest four bits. The MSB is allocated for polarity. In Fig. 7, a binary form of NSP pixels with a prediction error of 10 are presented. The MSB is the polarity mark.

As can be seen from Fig. 7, when the prediction error $E(i, j) \in [-15, 15]$, the pixel value can be expressed by its MSB and the lowest four bits. The 4th, 5th and 6th bits are 0s. We replace the 5th and 6th bits with external data, in order to embed additional information. The 4th bit is used as an embedding flag . When the prediction error falls within the range of $[-15, 15]$, the 4th bit remains unchanged, i.e., 0. But when the prediction error falls outside of the range $[-15, 15]$, its 4th bit is set to 1. Its original bit is saved and embedded into the secret image along with the external data.

The parameter T can be set to 1, 3 or 7. The corresponding range is $[-1, 1]$, $[-3, 3]$, or $[-7, 7]$. The embedding algorithm is the same as the above example. Since the number of prediction error values in these ranges is large, a large amount of external information can be embedded. What's more, the recording step of the positioning map is not necessary since the flag bit is stored inside the pixel value. For example, when $T = 3$, the pixel value can be represented by its 7th bit (positive and negative sign) and the lower two bits (0/1). Its 2nd to 6th bits in its binary form are 0s. We can embed external data by replacing the 3rd to the 6th bits, and the 2nd bit is used as the embedding flag.

3.4 Data extraction and image decryption

Once the receiver obtains the secret image containing the hidden data, he can not only extract the data, but also losslessly decrypt the original image if he has both encryption keys (K_p and K_s) and data-hiding key (K_d). If he has only the data-hiding key (K_d), he can extract the embedded data; If he has only encryption keys (K_p and K_s), he can obtain the

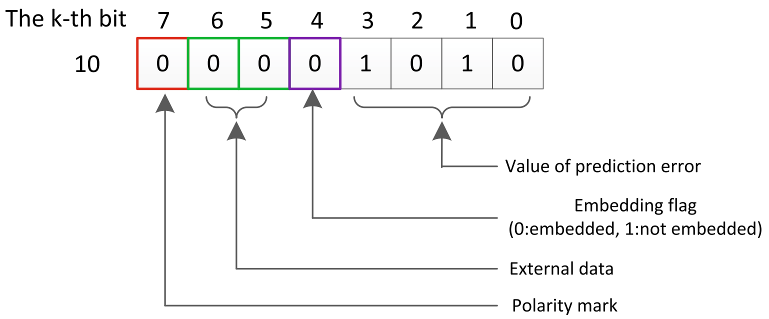


Fig. 7 Binary form of prediction error

distorted image but cannot extract the hidden data. Thus, this algorithm is flexible and can support the above three application requirements.

3.4.1 Data extraction

The recipient only needs to use the data-hiding key (K_d) to extract the authentication information if he only needs to authenticate the image without knowing its content. Take $T = 15$ as an example. The receiver detects the NSP pixels' 4th binary bit sequentially (from top to bottom and from left to right). If it is 0, then extracts the 5th and 6th bits as the data; if it is 1, then skip this bit and continue to detect the next NSP pixels. After all the data are extracted, the data-hiding key (K_d) is used to decrypt the data to obtain the embedded data in plain text form. The original image can be easily restored from the recovered prediction error.

3.4.2 Image lossless decryption

If the receiving end has a higher requirements, he not only needs to extract the embedded information to authenticate the image, but also needs to understand the full content of the image. In this case, the receiver first extracts the data by using the data-hiding key (K_d), as outlined in Section 3.4.1. Then, he restores the original image according to the image encryption keys (K_p and K_s). The detailed implementation steps are as follows:

1. Decrypt the extracted data by using the data-hiding key (K_d) and separate the original bit from the decrypted data.
2. Detect NSP pixels' 4th binary bit in sequence. If it is 0, set the 6th and 7th bit to 0; if it is 1, restore the original 4th bits.
3. Determine whether the error is positive or negative according to the 7th bit (positive and negative sign). Get the predicted error $E(i, j)$.
4. The pixels belonging to SP are decrypted according to the stream encryption key (K_s) to obtain the original SP pixel $O(i, j)$. The prediction pixels $O_P(i, j)$ belonging to NSP which contain $O_H(i, j)$, $O_V(i, j)$, and $O_D(i, j)$ are then calculated according to (1), (2) and (3). The prediction error $E(i, j)$ is calculated as well.
5. The original prediction error that falls outside of $[-15, 15]$ is restored by the following equation according to $E(i, j)$ and the location map L:

$$E(i, j) = \begin{cases} E(i, j) - 127, & \text{if } E(i, j) < 0, L(i, j) = 1, \\ E(i, j) + 127, & \text{if } E(i, j) > 0, L(i, j) = 1. \end{cases} \tag{9}$$

6. The original pixels belonging to NSP are calculated according to the following equation:

$$O(i, j) = O_P(i, j) + E(i, j). \tag{10}$$

7. Inversely permute the blocks to get the original image according to permutation key (K_p).

3.4.3 Image lossy decryption

If the receiver does not want to extract the embedded data, but just wants to understand the contents of the image, he can get an image that is visually close to the original image by using the image encryption keys (K_p and K_s).

1. Detect NSP pixels' 4th binary bit in sequence. If it is 0, set the 6th and 7th bit to 0; if it is 1, then skip it to the next NSP pixel.
2. Determine whether the error is positive or negative according to the 7th bit (positive and negative sign). Get the direct restored predicted error $E'(i, j)$ that is different from the original predicted error.
3. The pixel belonging to SP is decrypted according to the stream encryption key (K_s) to obtain the original SP pixel $O(i, j)$. The prediction pixels $O_P(i, j)$ belonging to NSP which contain $O_H(i, j)$, $O_V(i, j)$, and $O_D(i, j)$ are then calculated according to (1), (2) and (3).
4. The restored pixels belonging to NSP are calculated according to the following equation:

$$O'(i, j) = O_P(i, j) + E'(i, j). \tag{11}$$

   The overflowed pixel can be modified according to the following equation as we just want to know the contents of the image:

$$E'(i, j) = \begin{cases} 0, & \text{if } E'(i, j) < 0, \\ 255, & \text{if } E'(i, j) > 255. \end{cases} \tag{12}$$

5. Inversely permute the blocks to get the distorted image without embedded data according to permutation key (K_p).

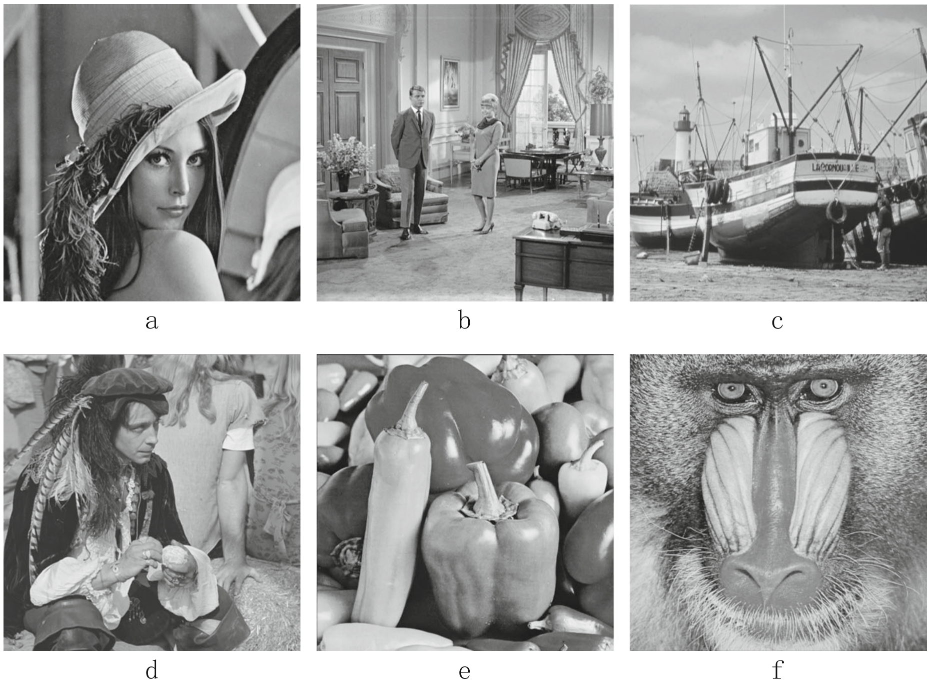


Fig. 8 Original image O: **a**. Lena **b**. Couple **c**. Boat **d**. Man **e**. Peppers **f**. Baboon

4 Experimental results

In this section, we present our experimental result. First, the security of the encrypted image is tested. Then the embedding rate is tested and compared with classical and recent related algorithms. Finally, we present the result of visual quality of the lossy decrypted image.

The experiments are done on six standard testing images, including Lena, Couple, Boat, Man, Peppers, and Baboon, as shown in Fig. 8. All of these images are gray-scale with size 512×512.

4.1 Security of encrypted image

The encrypted image using our algorithm is shown in Fig. 9, where the block size is 3×3. The encrypted image using the method in [18] is shown in Fig. 10. Visual inspection reveals that the the original image contour as shown in Fig. 10 does not appear in the our encrypted image, thanks to the scrambling encryption. This scrambling increases the security performance.

In order to quantitatively characterize the security level, the correlation coefficient ρ_{oc} between the secret image C and the original image O is calculated from the (13) [17].

$$\rho_{oc} = \frac{\mathbb{E}[(O - \mathbb{E}(O))(C - \mathbb{E}(C))]}{\sqrt{\mathbb{D}(O)\mathbb{D}(C)}}, \tag{13}$$

where O and C represent the original image matrix and the secret image matrix, respectively. $\mathbb{E}(O)$ and $\mathbb{D}(O)$ are the expected value and variance of O, respectively; $\mathbb{E}(C)$ and

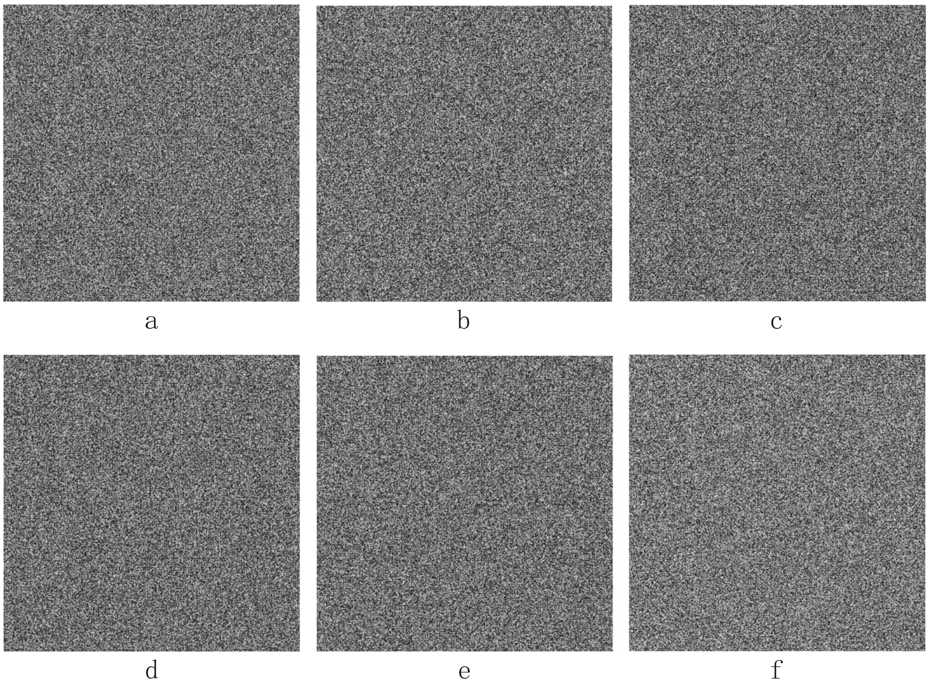


Fig. 9 Encrypted image C: **a**. Lena, PSNR = 8.50dB **b**. Couple, PSNR = 8.71dB **c**. Boat, PSNR = 7.19dB **d**. Man, PSNR = 8.25dB **e**. Peppers, PSNR = 7.28dB **f**. Baboon, PSNR = 8.01dB

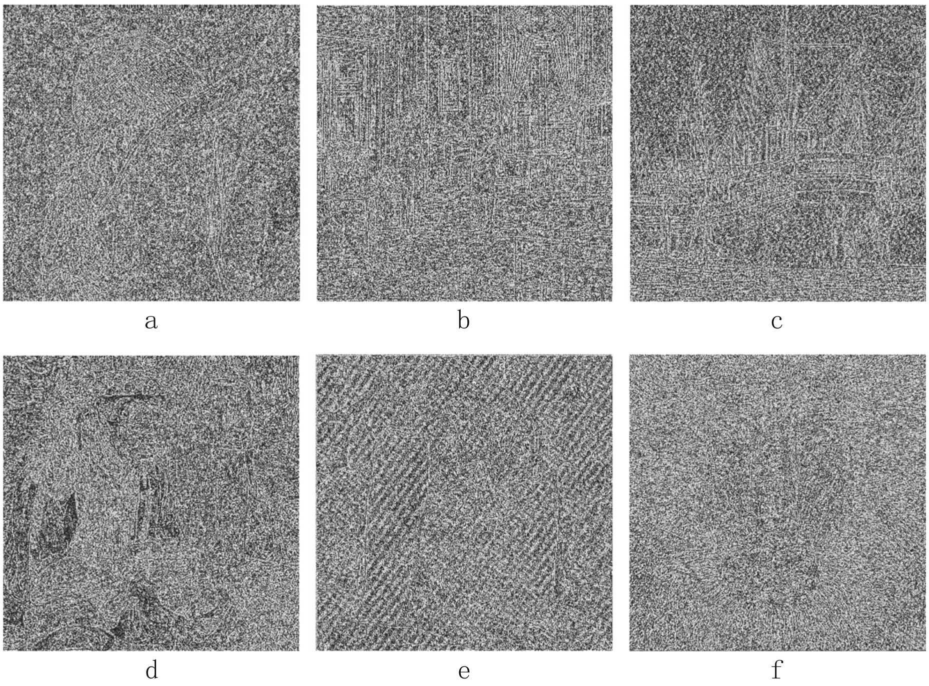


Fig. 10 Encrypted image using Xu's algorithm [18]: **a**. Lena, PSNR = 8.21dB **b**. Couple, PSNR = 9.15dB **c**. Boat, PSNR = 7.59dB **d**. Man, PSNR = 8.66dB **e**. Peppers, PSNR = 7.48dB **f**. Baboon, PSNR = 9.00dB

$\mathbb{D}(C)$ are the expected value and variance of the secret image C, respectively. The expected value and variance are calculated from the sample mean and the sample variance as follows:

$$\mathbb{E}(O) = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} O(i, j), \tag{14}$$

$$\mathbb{D}(O) = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (O(i, j) - \mathbb{E}(O))^2. \tag{15}$$

The correlation coefficients ρ_{oc} for the above images are given in Table 1. It can be seen from Table 1 that the encryption performance is improved. The result from Xu's algorithm [18] is also listed in the second column. We observe that, the correlation coefficient for our algorithm is significantly smaller than those for Xu's algorithm. The correlation coefficient of each image in this encryption scheme tends to be 0.

4.2 Embedding capacity

Tables 2 and 3 show the number of embedded bits (EB) and the embedding rate (ER) for $T = 1$, $T = 3$, $T = 7$, and $T = 15$, where the sizes of the block are 3×3 and 5×5. The number of embedded bits does not include the original bits that need to be embedded together with the external data.

It can be seen from Tables 2 and 3 that, as T goes from one to seven, the embedding rate increases. The embedding rate is the largest when $T = 7$. Take a smaller T, then

Table 1 Correlation coefficient between the secret image and the original image

Correlation coefficient (ρ_{oc})	Proposed	Xu [18]
Lena	−0.0005	0.0381
Couple	−0.0013	0.0885
Boat	−0.0035	0.0509
Man	−0.0006	0.1467
Peppers	−0.0045	0.0509
Baboon	−0.0007	0.1776

most prediction errors are outside of the range of $[-T, T]$ and hence is not suitable for embedding, while a larger T allows more errors to be embeddable. But if we increase T, then less bits are available for embedding. Referring to Fig. 7, when $T = 7$, then three bits are available for embedding external data. But if we choose $T = 15$, only two bits are available for embedding. So there is a tradeoff in choosing T. Our experimental results reveal that using $T = 7$ provides good balance between these two stretching forces.

Comparing Table 2 with Table 3, we notice that, given the same T, the ER increases with the block size. This is because the larger the block size, the more the prediction errors are available, the more bits can be embedded (Fig. 5).

From the embedding capacity table, we also notice that the embedding capacity of Baboon is significantly smaller than other images. When $T = 1$, no external data can be embedded into baboon. The reason is that, baboon has more texture, and there are few prediction errors that can be used to embed data. Obviously, the proposed algorithm achieves high embedding capacity because more predictive errors are used in this algorithm.

4.3 The visual quality of the distorted image

If the receiver has only the encryption key pair (K_p and K_s), then he can decrypt the image and partially recover the modification from RDH, as outlined in Section 3.4.3. So the visual quality of the decrypted image should be high enough so that the distortion cannot be perceived by human visual system. To quantify the visual quality, we use the PSNR measure that is widely used in evaluating the quality of digital images.

Table 2 The embedding capacity when the image is divided into 3 × 3 blocks

	$T = 1$		$T = 3$		$T = 7$		$T = 15$	
	EB	ER	EB	ER	EB	ER	EB	ER
Lena	86032	0.3282	249935	0.9534	336704	1.2844	270532	1.0320
Couple	65662	0.2505	215720	0.8229	295988	1.1291	249628	0.9523
Boat	146854	0.5602	303290	1.1570	328496	1.2531	261388	0.9971
Man	153664	0.5862	294055	1.1217	327408	1.2490	260437	0.9935
Peppers	68146	0.2600	233915	0.8923	335380	1.2794	274033	1.0454
Baboon	—	—	43385	0.1655	133008	0.5074	163531	0.6238

Table 3 The embedding capacity when the image is divided into 5 × 5 blocks

	$T = 1$		$T = 3$		$T = 7$		$T = 15$	
	EB	ER	EB	ER	EB	ER	EB	ER
Lena	98046	0.3740	286396	1.0925	387028	1.4764	311484	1.1882
Couple	70806	0.2701	243436	0.9286	336144	1.2823	285240	1.0881
Boat	165024	0.6295	345791	1.3191	375860	1.4338	299541	1.1427
Man	174030	0.6639	334316	1.2753	374952	1.4303	299247	1.1415
Peppers	82518	0.3148	272601	1.0399	387900	1.4797	315597	1.2039
Baboon	—	—	48766	0.1860	150704	0.5749	186729	0.7123

As Zhang [19] first proposed the application of RDH in encrypted domains and this paper aims to improve the performance of Xu's [18] algorithm, we compared PSNR of the lossy decrypted images in our algorithm with these two algorithms, as shown in Fig. 11.

The experimental results demonstrate that, using block size 3 × 3 and $T = 15$, our algorithm significantly outperforms these reference algorithms. This performance improvement can be explained as follows. For Zhang's algorithm, since 3 LSBs of 50% pixels in each image block are flipped in the data embedding procedure, the visual quality of lossy decrypted images is greatly reduced. In Xu's work, both bidirectional histogram shifting and difference expansion based methods are unified to embed additional data into prediction errors. However, this unified method leads to excessive modification of prediction errors. It can not get high enough visual quality either in the lossy decrypted procedure. In this paper, most prediction errors can be recovered in the lossy decrypted procedure since prediction errors were marked by built-in embedding flags. With the same ER, the PSNR of the lossy decrypted image in our method is significantly higher.

Figure 12 shows the lossy decrypted image for block size 3 × 3 and $T = 15$. In general, for images with simple contents, i.e., less textures, such as Lena and Peppers, the PSNR is above 45dB; For images with more complex textures, such as Baboon, the PSNR is above 35dB. So, the distortion is in general not visible to a human visual system. Due to the setting of the algorithm in this scheme, the distorted image still has a high visual quality.

The source of distortion can be explained as follows. From the algorithm in Section 3.3, when the prediction error $E(i, j) \notin [-T, T]$, the original bits in the position of embedding flag are appended to the end of the data to be embedded. Since all the embedded bits are lost after direct decryption, these bits are also destroyed. This results in distortion to the cover image.

4.4 The execution time of different stages

Computational time is also an important aspect of image processing algorithms. We evaluated the time complexity of the proposed scheme and Xu's scheme with the same embedding rate. These two algorithms are implemented in Matlab 2013b. Table 4 lists the execution time for the different images in the four stages of the proposed scheme and Xu's scheme. It need to be noted that the image encryption time in Table 4 includes the time for image preprocessing.

It can be seen from Table 4 that the time for image encryption and decryption of the proposed scheme is longer than Xu's. The reason is that, the block permutation process

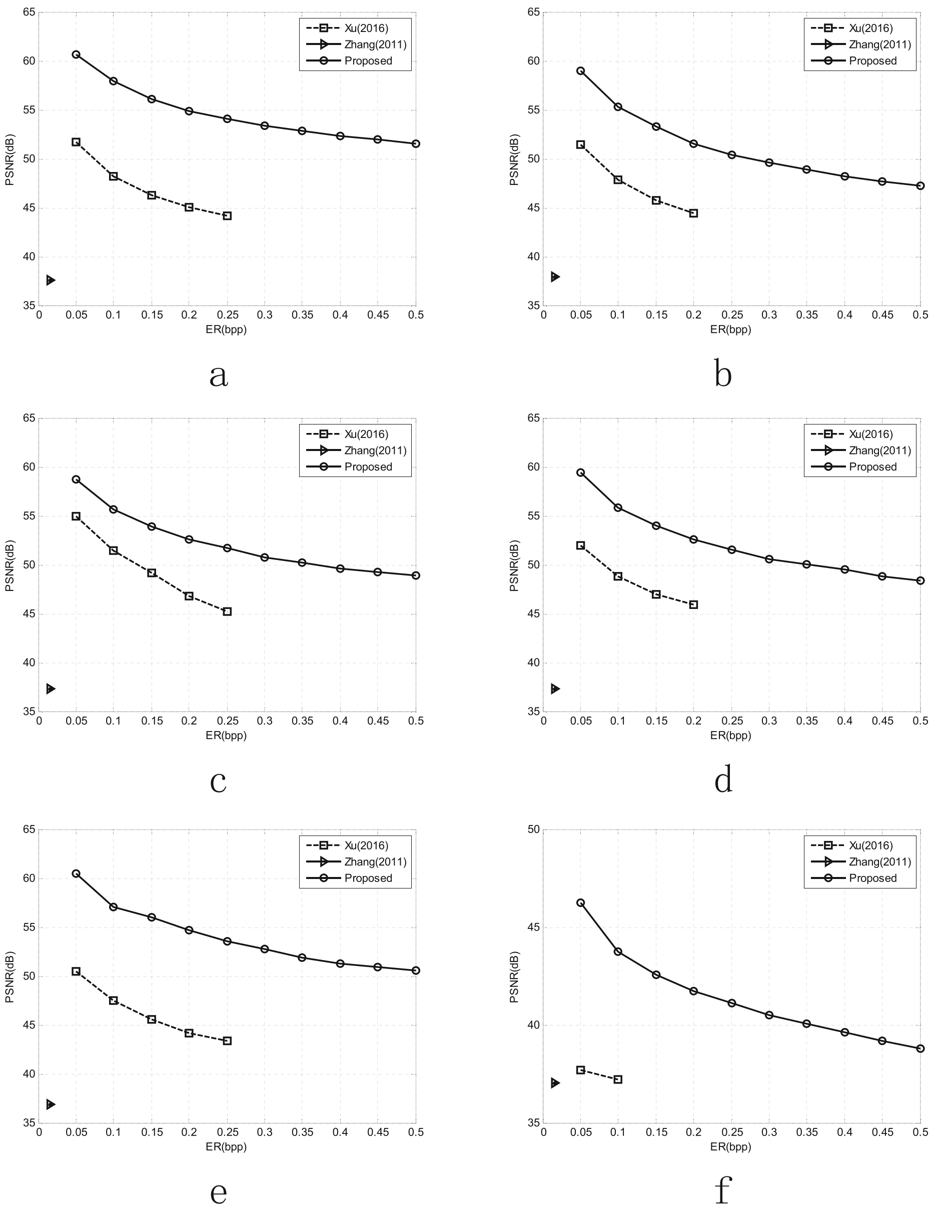


Fig. 11 PSNR comparison for the lossy decrypted images: **a**. Lena, **b**. Couple, **c**. Boat, **d**. Man, **e**. Peppers, **f**. Baboon

is adopted in the proposed scheme to increase the security of encrypted images. This process increases the execution time. We can also see from Table 4 that the time for data embedding of the proposed scheme is less than Xu's. Compared to histogram shifting and prediction error expansion, simple bit replacement according to built-in embedding flag has a lower complexity calculation during the data embedding process. Since the execution time

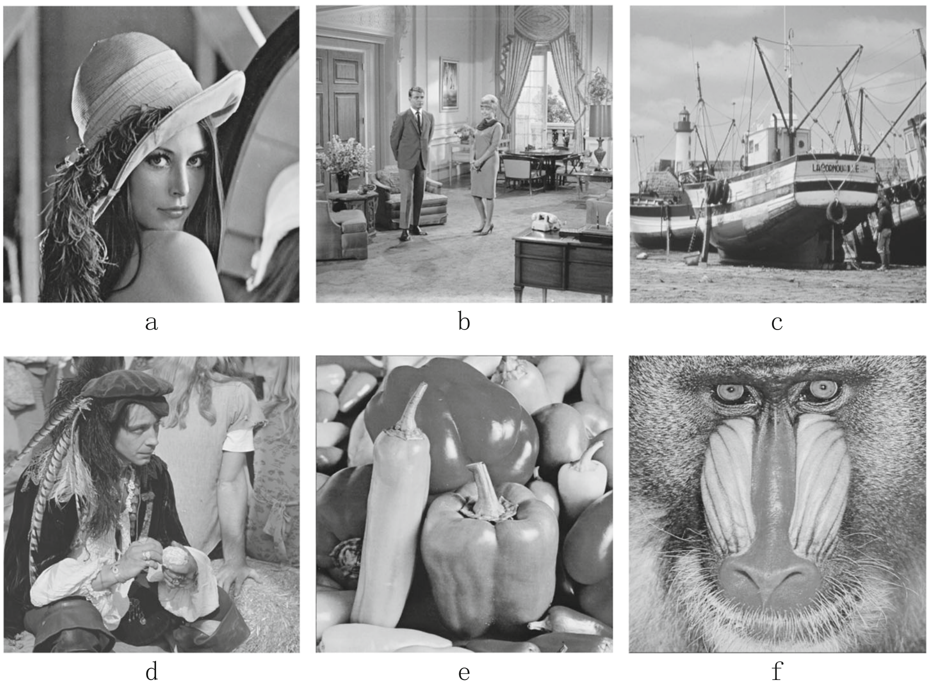


Fig. 12 Distorted images D: **a**. Lena, PSNR = 48.50dB **b**. Couple, PSNR = 44.57dB **c**. Boat, PSNR = 45.94dB **d**. Man, PSNR = 45.64dB **e**. Peppers, PSNR = 47.48dB **f**. Baboon, PSNR = 36.55dB

in different stages does not exceed two seconds, it can satisfy the real-time requirement in applications.

Convergence and stability are important aspects of computer algorithms. Even though the current algorithm is not iterative in nature, but the randomness introduced in encryption stage may arouse stability and convergence problem. Specifically, we should consider the randomness introduced from block permutation and stream encryption. To test the convergence and stability, all the previous experiments are conducted using randomly selected keys. Experimental results demonstrate that, the encoding and decoding can be finished in finite time, as shown in Table 4. Thus, our algorithm is stable under different keys and convergence is validated.

Table 4 The execution time for the different stages (unit)

	Image encryption		Data embedding		Lossy decryption		Data extraction and image decryption	
	Proposed	Xu [18]	Proposed	Xu [18]	Proposed	Xu [18]	Proposed	Xu [18]
Lena	1.5481	0.7887	0.1238	0.4858	1.6069	0.8970	1.7528	1.3859
Couple	1.5316	0.8358	0.1466	0.4869	1.5976	0.9102	1.7399	1.4025
Boat	1.5291	0.8376	0.1271	0.4886	1.5908	0.9326	1.7288	1.3975
Man	1.5271	0.7724	0.1261	0.4879	1.5946	0.9043	1.7515	1.3658
Peppers	1.5373	0.7972	0.1224	0.4778	1.5938	0.9188	1.7432	1.3796
Baboon	1.5497	0.8025	0.1624	0.4843	1.5920	0.9132	1.7837	1.3664

5 Conclusion

This paper designed a separable and error-free RDH in encrypted domain. Using a carefully designed block permutation algorithm adapted to the RDH, the image content leakage problem in Xu's algorithm is further mitigated. To quantitatively characterize the security level, the correlation coefficient between the encrypted image and the original image is calculated. The results show that the proposed algorithm provides correlation that is less than 1/10 of that of the reference algorithm. What's more, a data embedding algorithm utilizing the extra bits from the reduced dynamic range of the prediction error was used to increase the embedding rate. This embedding algorithm has an advantage of built-in embedding flag. The experimental results on test images show that the embedding rate is significantly improved. Furthermore, the PSNR of the lossy decrypted image has better performance. The execution time in different stages is also evaluated. Experiments prove that the execution time consumed by our scheme can satisfy the real-time requirement in applications. The performance analysis implies that the method in this paper can achieve good results for practical applications. It can be used in three cases adapted to different needs.

Since our algorithm is targeted for cloud computing scenarios, so data extraction in decrypted images is not feasible. Our future work is to solve this problem to expand the application in different scenarios.

Acknowledgments This work is supported by the National Natural Science Foundation of China (NSFC) (No. 61272432) and Shandong Provincial Natural Science Foundation (No. ZR2014JL044).

References

1. Celik MU, Sharma G, Tekalp AM, Saber E (2005) Lossless generalized-LSB data embedding. IEEE Trans Image Process 14(2):253–266
2. Hong W, Chen TS, Wu HY (2012) An improved reversible data hiding in encrypted images using side match. IEEE Signal Process 19(4):199–202
3. Huang F, Huang J, Shi YQ (2016) New framework for Reversible data hiding in encrypted domain. IEEE Trans Inf Forensics Secur 11(12):2777–2789
4. Li M, Li Y (2017) Histogram shifting in encrypted images with public key cryptosystem for reversible data hiding. Signal Process 130:190–196
5. Liu Y, Nie LQ, Han L, Zhang LM, Rosenblum DS (2015) Action2activity: recognizing complex activities from sensor data. In: International conference on artificial intelligence, pp 1617–1623
6. Liu Y, Nie LQ, Liu L, Rosenblum DS (2016) From action to activity: sensor-based activity recognition. Neurocomputing 181:108–115
7. Liu Y, Zhang LM, Nie LQ, Yan Y, Rosenblum DS (2016) Fortune teller: predicting your career path. In: Proceedings of the Thirtieth AAAI conference on artificial intelligence, pp 201–207
8. Liu Y, Zheng Y, Liang YX, Liu SM, Rosenblum DS (2016) Urban water quality prediction based on multi-task multi-view learning. In: Proceedings of the Twenty-Fifth international joint conference on artificial intelligence, pp 2576–2582
9. Luo L, Chen Z, Chen M, Zeng X, Xiong Z (2010) Reversible image watermarking using interpolation technique. IEEE Trans Inf Forensics Secur 5(3):187–191
10. Ma K, Zhang W, Zhao X, Yu N, Li F (2013) Reversible data hiding in encrypted images by reserving room before encryption. IEEE Trans Inf Forensics Secur 8(3):553–562
11. Ni Z, Shi YQ, Ansari N, Su W (2006) Reversible data hiding. IEEE Trans Circuits Syst Video Technol 16(3):354–362
12. Preotiuc-Pietro D, Liu Y, Hopkins DJ, Ungar L (2017) Beyond binary labels: political ideology prediction of twitter users. In: Meeting of the association for computational linguistics, pp 729–740
13. Qian Z, Zhang XP (2016) Reversible data hiding in encrypted images with distributed source encoding. IEEE Trans Circuits Syst Video Technol 26(4):636–646

14. Qin C, Chang CC, Hsu TJ (2015) Reversible data hiding scheme based on exploiting modification direction with two steganographic images. Multimedia Tools and Applications 74(15):5861–5872
15. Qin C, Zhang XP (2015) Effective reversible data hiding in encrypted image with privacy protection for image content. J Vis Commun Image Represent 31:154–164
16. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Transactions on Cricuits and Systems for Video Technology 13(8):890–896
17. Xiao D, Xiang YP, Zheng HY, Wang Y (2017) Separable reversible data hiding in encrypted image based on pixel value ordering and additive homomorphism. J Vis Commun Image R 45:1–10
18. Xu DW, Wang RD (2016) Separable and error-free reversible data hiding in encrypted images. Signal Process 123:9–21
19. Zhang XP (2011) Reversible data hiding in encrypted image. IEEE Signal Process 18(4):255–258
20. Zhang XP (2012) Separable reversible data hiding in encrypted image. IEEE Trans Inf Forensics Secur 7(2):826–832

Qi Li received his B.E. degree in communication engineering, from College of Electronics, Communication and Physics, Shandong University of Science and Technology, 2016, P. R. China. He is now pursuing his master degree. His research interests include reversible watermarking and multimedia encryption.

Bin Yan received the B.S degree in applied physics from Qingdao University, P. R. China, in 1996 and the M.S. degree in electrical engineering from Harbin Institute of Technology, P. R. China, in 2002 and Ph. D degree in electrical engineering from Harbin Institute of Technology, P. R, China, in 2007. From 1996-1999, he was an engineer in Goma Company Group. From 2007-2012, he was a lecturer in Shandong University of Science and Technology. From 2015-2016, he was a visiting scholar in Deakin University, Australia. Since 2013, he has been an associate professor with the Communication Engineering Department, Shandong University of Science and Technology. His research interests include statistical signal processing, multimedia signal processing and security.

Hui Li received his B.E. degree in 2015. She is now a postgraduate student in College of Electronics, Communication and Physics, Shandong University of Science and Technology. Her research interest is reversible watermarking.

Na Chen received her Ph.D degree from Xi Dian University, P. R. China, in 2016. She is now with the Department of Communication Engineering, College of Electronics, Communication and Physics, Shandong University of Science and Technology. Her research interests include cryptography and quantum information processing.