ISSN (Online): 2455-7838
SJIF Impact Factor (2017): 5.705

EPRA International Journal of Research & Development (IJRD)

Monthly Peer Reviewed & Indexed International Online Journal

Volume: 3, Issue:9, September 2018

Published By :
EPRA Journals

SJIF Impact Factor: 5.705 Volume: 3 | Issue: 9 |September| 2018 ISSN: 2455-7838(Online)

EPRA International Journal of Research and Development (IJRD)

ROBUST REVERSIBLE DATA HIDING BY BINARY-BLOCK EMBEDDING

Pooja Wagh[1]

[1]M.Tech Student, Department Of Computer Science & Engineering, MPCT Gwalior, Madhya Pradesh, India

Dr. Shivnath Ghosh[2]

[2]Associate Professor& H.O.D, Department Of Computer Science & Engineering, MPCT Gwalior, Madhya Pradesh, India

ABSTRACT

The proposed work is based on a binary-block embedding (BBE) framework. This method is used to hide the secret message behind the digital image. The design algorithm is described as RDHEI using BBE. The idea is to hide secret bits in lower bit planes inside the cover image. The lower planes are kept empty for the secret hiding operation in sequential manner. BBE-RDHEI employs a bit-level scrambling process after secret data embedding to spread embedded secret data to the entire marked encrypted image so that it can prevent secret data from loss. A security key design mechanism is proposed to enhance the security level of BBE-RDHEI. The processes of BBE-RDHEI are fully reversible. The secret data and original image can be reconstructed independently and separately. Experiments and comparisons show that BBERDHEI has an embedding rate nearly twice larger than the state-of-the-art algorithms, generates the marked decrypted images with high quality, and is able to withstand the brute-force, differential, noise and data loss attacks.

KEYWORDS: *Binary-Block Embedding (BBE) Reversible data hiding Image encryption Encrypted domain.*

INTRODUCTION

Reversible Data Hiding (RDH) is a technique that slightly alters digital media (e.g. images or videos) to embed secret data while the original digital media can be completely recovered without any error after the hidden messages have been extracted [1]. It is quite useful for various applications in military, medical science or law enforcement, where the original images or videos should not be damaged. A number of RDH methods were proposed in recent years. Histogram shifting (HS)[3] shifts several or the maximum points in histogram bins of the original image to reserve spare space for data embedding [1]. To improve the embedding capacity, prediction-error based HS algorithms were introduced [2–4]. Difference expansion (DE) [5–7] as another type of RDH divides the image into pixel pairs and embeds secret data into the expanded difference values. Integer transforms[9] have been used to modify the values of pixel pairs to embed secret data [8–10]. These RDH methods need the redundancy information of image pixels in original images to embed secret data, such as the statistic or difference information of pixel pairs. They are not suitable for encrypted images that are noise-like and have no redundancy information available. Recently, reversible data hiding in encrypted images (RDHEI) has attracted people's attention. It aims to protect both the original images and secret data simultaneously. For example, the content owner intends to store an original image in the Cloud that is hosted by a third party. To prevent the content of the original image from being exposed to the third party, the content owner encrypts the image before sending it to the Cloud. Meanwhile, the system administrator of the third party is able to add some notations to the encrypted image without knowing its original content. Depending on whether the data extraction and image recovery processes can be performed separately, existing RDHEI methods can be classified into joint and separate methods. For joint methods, Peuch et al. [11] first encrypted each block of the original image by the advanced encryption standard (AES) and then embedded one bit of the secret data into each encrypted block by bit substitution. The encrypted image embedded with secret data is called the marked encrypted image. Secret data extraction is just to obtain the bits in the substituted positions. Original image recovering is accomplished by analyzing the local standard deviation of the marked encrypted image during the decryption procedures. This algorithm has a limited payload and yields the decrypted image with low quality. Another joint RDHEI algorithm proposed by Zhang [12] encrypts the original image using bit-level XOR and then embeds one bit of secret data into each block of the encrypted image by shifting the three least significant bits (LSBs) of half pixels within the block. This algorithm may suffer from incorrect results of data extraction and image recovering[27] in the non-smoothness regions in the image when the block size is relatively small (e.g., 8×8). Hong et al. [13] proposed an improved version of this algorithm by modifying its smoothness measurement function. The error rate of data extraction is reduced for small block sizes. In Wu et al.'s joint method [14], one bit of the secret data is embedded by flipping the ith (i1 ≤ ≤ 6) bit of pixels in a certain group. This method also may suffer from incorrect results of data extraction and image recovery. To allow the receiver with different privileges to obtain different contents (the secret data, the original image or both) from the marked encrypted image, researchers devote themselves to develop separable RDHEI methods. Zhang et al. [15,16] proposed two separable methods that compress the encrypted image to accommodate secret data. In Wu et al.'s separable method [14], one bit of the secret data is embedded by replacing the ith ($i \geq 7$ for the later one) bit of pixels in a certain group. Secret data extraction and image recovering are using the prediction error. Compared with algorithms in [12,13,15], the methods in [14] reduce the number of incorrectly extracted secret data bits and improve the visual quality of the marked decrypted image. Qian [17] proposed a separable RDHEI algorithm using n-nary histogram[3] modification. However, it results in Salt & Pepper noise in the marked encrypted images. Besides, instead of working in the spatial domain, Qian et al. [18] proposed an RDHEI method to embed secret data in the encrypted JPEG bitstream. In [19] and [20], homomorphic encryption is utilized to encrypt the original image. However, image size increases because the used homomorphic encryption algorithm maps the pixel value into a larger data range. In above mentioned RDHEI methods, the content owner does nothing except for image encryption. These methods have a small payload and/or a high error rate in data extraction and image recovering. To overcome these problems, some researchers aim to develop another type of separable RDHEI method by reserving the spare space for secret data embedding before image encryption. In Ma's method [21], it reserves the spare space by embedding some LSBs in a part of the cover image into the rest part of the cover image with using simplified RDH method in [22]. The self-embedding of LSBs ensures the reversibility of image recovering. Zhang et al. [23] selected some pixels and applied a histogram shifting method to their estimation error values for accommodating secret data. Previous RDHEI methods in [12,13,15,14,21] have a limited embedding rate and are under the only situation that the images are for the Cloud storage with no transmission involved and thus no attacks [18]. Considering the scenario that hospitals at different locations build a bridge for co-operation, many medical images embedded with patients' information or treatment history records will be shared among several working teams, thus medical images will be transmitted over public channels that they may inevitably experience some noise and data loss. In this scenario, these RDHEI methods may suffer from secret data loss when the marked encrypted image is partially damaged or lost. For example, the method in [21] uses a part of the LSB plane in the encrypted image to accommodate the secret data when embedding rate is less than 0.2 bpp. If the LSB plane is illegally removed, all secret data will lose. In the separable method in [14], the secret data are embedded in the ith most significant bit (MSB) plane, it will also suffer from complete loss of secret data when this MSB plane is removed or damaged. To improve the embedding rate while enhancing security and robustness[29], this paper introduces the binary-block embedding (BBE) method to embed message bits in binary images. Based on BBE, we further propose a reversible data hiding algorithm in encrypted images (BBE-RDHEI). It first uses BBE to embed binary bits in several LSB planes of the original image into its MSB planes. BBE-RDHEI encrypts the original image and hides the secret data into its LSB planes. A bit-level scrambling process is then employed after secret data embedding to ensure that the proposed BBE-RDHEI can resist the noise and data loss attacks. Using different security[30] keys[28], the receiver is able to obtain the secret data, marked decrypted image, decrypted image, or all of them from the marked encrypted image. Our main contributions in this work are listed as follows:

(1)We propose a new BBE algorithm for reversible data hiding in the encryption domain, which is totally different from traditional RDH methods. BBE can be utilized in different types of images such as binary, gray-scale[24], medical and cartoon images.

(2) Based on BBE, we further propose a method of reversible data hiding in encrypted images, BBE-RDHEI. Compared with existing state-of-the-art methods, it has significantly improved embedding capacity and quality of the marked decrypted image. BBE-RDHEI can also be simplified and utilized for binary images while existing RDHEI methods are designed only for gray-scale images.

(3) To significantly enhance the security level of BBE-RDHEI, we also propose a security key design mechanism such that BBE-RDHEI is able to resist the differential attack while existing RDHEI methods cannot.

(4) To enhance the robustness of RDHEI methods in withstanding noise and data loss attacks, we introduce a bit-level scrambling process to BBE-RDHEI after secret data embedding to spread out embedded secret data over the entire marked encrypted image. As a result, BBE-RDHEI is able to recover most of the secret data even if one bit-plane[6] (e.g., LSB or MSB) of the marked encrypted[25] image is completely removed. Moreover, any bit-level scrambling algorithm can be used in our BBE-RDHEI. This is another security benefit of BBE-RDHEI. The rest of this paper is organized as follows:

(1) Section 2 will introduce the BBE algorithm. Section 3 will propose BBE-RDHEI. Simulation results and comparisons will be provided in Section 5. Section 6 will provide security and robustness analysis of the proposed BBE-RDHEI. Section 7 will draw a conclusion.

(2) 2. Binary-block embedding.

(3) In this section, we propose a binary-block embedding (BBE) algorithm to embed message bits into a binary image.

(4) 2.1. BBE

(5) BBE first divides the binary image[26] into a number of non-overlapping blocks, separates them into two groups named good and bad blocks, respectively. A good block is able to be embedded with messages while a bad one is not. In the message embedding phase, BBE first labels the first 2 or 3 bits of each block with special bits that indicate the block types. Then the rest bits of a good block will be replaced with its structure information and message bits while the rest bits of a bad block will be kept unchanged. Next, we present the BBE algorithm in detail.

(6)2.1.1. Block labeling Assume that a binary image with a size of MN × is able to embed secret data. We first divide the image into a set of non-overlapping blocks with a size of ss ×1 2, where ss,≥ 312. For a certain block, we let n ss =* 1 2 be the total number of pixels within the block, and mn n = min{,} 01be the minimum value of n0 and n1, where n0 and n1 are the numbers of 0 s and 1 s within the block, respectively. According to a threshold na, we then classify these blocks into five categories as shown in Table 1, namely: Good-I/II/III/IV block and Bad block, where a good block is able to embed secret data, while a bad one cannot.

Table 1
Block Types and block-labeling bits

Condition	Blocktype	Blockdescription	Block-labeling bits
$m > n_a$	Bad	Cannot embed data	00
$m = n_0 = 0$	Good-I	all pixels are 1	11
$m = n_1 = 0$	Good-II	all pixels are 0	10
$1 = \leq m \leq n_a$, $n_o < m$	Good-III	most of pixels are 1	011
$1 = \leq m \leq n_a$, $n_1 < n_o$	Good-IV	most of pixels are 0	010

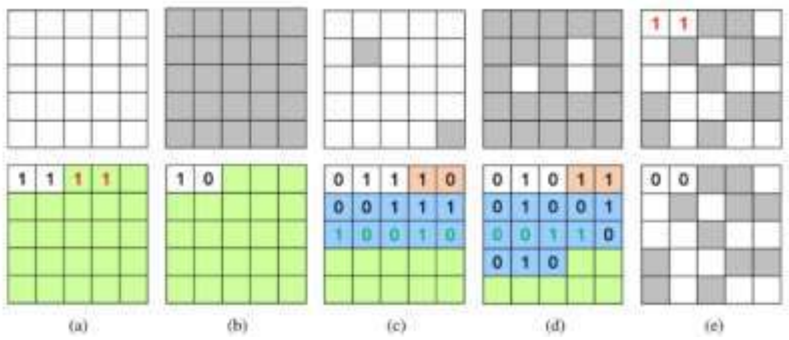


Fig. 1. BBE examples with the block size of 5×5. The first and second rows show the five types of original blocks and their corresponding embedded results. White and gray boxes represent the pixels with values of 1 and 0, respectively. The green, orange and blue areas are utilized to embed payload, parameter m and the positions of m original pixels in the block. (a) and (b) are Good-I and Good-II blocks with c = 23 b and their embedded results; (c) Good-III block with c = 10 b and its embedded result; (d) Good-IV block with c = 7 b and its embedded result; and (e) Bad block withc =− 2b and its result after embedding. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

In a Good-I (Good-II) block, all pixel values are equal to 1 (0), while in a Good-III (Good-IV) block, less than or equal to na pixel values are equal to 1 (0). An illustrative example can also be found in the first line of Fig. 1. To embed the secret data, we first need to determine the block type. As shown in Table 1, Good-I (Good-II) block can be easily distinguished by checking the value of m, where m is obtained once a block is given. In order to distinguish Good-III (Good-IV) blocks from Bad ones, we calculate the threshold na by-

n_a = argmax { n – 3 – max { [$\log_2$ x],1} – x {$\log_2$n] ≥ 0},1 ≤ x ≤ [0.16*n] }

If mn ≥ a, it is a Bad block; otherwise, it is a Good-III (Good-IV) block. In Eq.(1), expressions x max{⌈log ⌉, 1} 2 and x n ⌈log ⌉ 2 represent the bit length to store x and these x pixels' locations, respectively. These 2 parts are utilized to store the structure information of Good-III and Good-IV blocks, and they will be discussed in Section 2.1.2. Thus, after the block is labeled by 3 labeling bits and embedded with the structure information of x pixels, if there is still larger than or equal to 0 bits left, the current block is considered as a good block; otherwise, it is a bad block. Therefore, na is the maximum number of pixels that can be represented by the block itself with a given block size. It is utilized to distinguish bad blocks from good ones. Here, x should be less than n, and the reason why we set x less than or equal to n ⌈0.16* ⌉ will be discussed in Section 2.4.1. After determining the block type, we label each block by replacing its first 2 or 3 pixels with the corresponding block-labeling bits as shown in Table 1. Before labeling blocks, the first 2 or 3 original pixels of each block are picked up and stored with the secret message for the purpose of image recovery at the receiver side.

2.1.2.Structure Information Embedding

In BBE, a good block is self-embedded with its original structure information and may have an additional spare space to accommodate secret message bits. The first 2 bits of each bad block are extracted and embedded into good blocks together with secret messages because they are directly replaced by the block-labeling bits ′00′ after the block labeling procedure. For a Good-I (Good-II) block, no additional structure information needs to be embedded except for two labeling bits. For a Good-III (Good-IV) block, parameter m and the locations of m pixels need to be embedded as the structure information. Here, p bits are utilized to embed parameter m, where p n = max{⌈log ⌉, 1} a2. Then, we use the variable length of bits to store the locations of m pixels. We first scan pixels in a block from top to bottom and left to right to obtain the location index values zzn{}(1 ≤ ≤) ii m i=1 of these m pixels. For the first of m pixels located at z1, without any additional information, z1 could be any integer in the range of n [1,]. Thus, we use n⌈log⌉2 bits to store its location index. For the second pixel located at z2, it could only be in the range of zn[+1,]1. Thus, we can use nz⌈log(−)⌉2 1 instead of n ⌈log ⌉ 2 bits to store its location index. Therefore, the actual location information of m pixels is stored as the distance t { } ii m =1 between the current pixel and its previous pixel, where ti is calculated by Eq. (3). For example, for the second pixel, we store its location information by converting the decimal value t2 into nz ⌈log (−)⌉ 2 1 -bit binary sequence.

In this manner that considers the relative distance between adjacent pixels, we are able to use fewer bits to store the locations of m pixels. We then continue in this process until all m pixels' locations are stored. Thus, qi bits are required to store the ith pixel's location, and totally pq (+∑) im i=1 bits are needed to store the parameter m and m pixels' locations.

$$q_1 = \{ {}^{[log_2 n]}_{\max\{[log_2(n - z_{i-1})],1\}} {}^{for\, i=1}_{for\, 2 \le i \le m}$$

$$t_1 = \{ {}^{z_i}_{z_i - z_{i-1}} {}^{for\, i=1}_{for\, 2 \le i \le m}$$

2.1.3. Message Embedding

The BBE payload consists of two parts: and, where is a bit sequence containing all of the first 2 original bits in each bad block and denotes secret messages. After embedding structure information, we replace the rest bits of a good block with cb bits of payload, where cb is the block capacity and calculated by-

$$c_b = \{n - 3 - p - \sum_{i=1}^{m} q_i for\ 1 \le\ m\ \le\ n_a \quad \begin{matrix} n-2 \\ for\ m=0 \\ otherwise \\ -2 \end{matrix}$$

Algorithm I

```
Input: binary image I, block size $s_1$ x $s_2$, payload P.
1: Divide I into non-overlapping blocks $B_{i,j}$ with a size of $s_1$ x $s_2$ .
Calculate parameters $n_a$, p and $q_i$ using Eqs. (1) and (2).
2: for each block $B_{i,j}$ do
3: Calculate $n_0$, $n_1$, m and $c_b$ according to Eq. (4).
4: if m  > $n_{ac}$
5: set the first two pixels to [0,0].
6: else if m=0 and $n_0$=0
7: set the first two pixels to [1,1], replace other pixels by $c_b$ bits of the payload.
8: else if m=0 and $n_1$=0
9: set the first two pixels to [1,0], replace other pixels by $c_b$ bits of the payload.
10: else if $1 \le m \le n_a$ then
11: if $n_0 < n_1$ then
12 : set the first three pixels to [0,1,1].
13: else
14: set the first three pixels to [0,1,0].
15: endif
16: replace other pixels with the parameter m, locations of m pixels, and $c_b$ bits of the payload.
17: endif
18: end for
Output: Embedded image I.
```

Illustrates an example of BBE. A binary image is divided into 5 blocks with size of 5×5. The payload includes the first two pixels of the bad block in Fig. 1(e) and 61 bits of secret messages. BBE embeds payload into all good blocks one by one. For the block in Fig. 1(d), it is a Good-IV block where most pixels are 0s. Parameters are m=3, p=2, q=5 1 ,qq ==4 23,c=7b. BBE labels the block in Fig. 1(d) by setting its first three pixels to′010′ (the white area), embeds m=3=(11)2 to the subsequent two pixels (the orange area), puts the location of three pixels (white boxes in the original block in Fig. 1(d)) tz== 9=(01001)112,
tzz=−=12−9 = 3= (0011)2212 and
tzz=−=14−12=2=(0010)3322 to the following 13 pixels (the blue region). The remaining 7 pixels (the green area) are utilized to embed payload. For the bad block in Fig. 1(e), its first two original pixels′11′ are embedded at the beginning of the green area in Fig. 1(a). BBE replaces its first two pixels as′00′ to indicate that it is a bad block, and keeps its other pixels unchanged.

2.2. Message extraction and image recovering

The message extraction and image recovering includes two phases: 1) payload extraction and good block recovering; 2) bad block recovering. In Phase 1, the BBE scans the first 3 labelling bits of each block to determine the block type. For a good block, BBE extracts parameter m, the locations of m pixels and payload bits from the block, and then reconstructs the block based on the extracted information. Otherwise, for a bad block, BBE records the block index. In Phase 2, BBE recovers the first 2 pixels of each bad block using the extracted payload and keeps the rest pixels unchanged.

2.2.1. Phase 1

For each image block, we first determine its block type by checking its first 3 pixels. If it is a bad block, we do nothing except for recording its block index. If it is a Good-I (Good-II) block, we obtain the payload bits from the last n(− 2) bits and recover the block by setting all bits to 1 s (0 s for the Good-II block). For a Good-III (Good-IV) block, we first extract the labeling bits and structure information from its first nc (−) b pixels, where cb is the block capacity; and then obtain the payload bits from the rest pixels of the block. To extract secret data from a Good-III (Good-IV) block, we first obtain the raster-scanned bit sequence aa [,…,] n12 from the block, and calculate parameter m from the specific p bits aa [,…,] p 4 3+. Then, we calculate the location index distance t {^} ii m =1 of m pixels by the following bits in an orderly way. Here, t ^ i is the location index distance between the ith and i(− 1) the pixels, and it is sequentially extracted from the subsequent q{ } ii m =1 bits, where qi is calculated by Eqs. (5). After obtaining t ^ i, we then calculate the actual locations of m pixels by Eq. (6). For example, for the first pixel, we obtain its location index zt = ^1 1 according to the following qn= ⌈log ⌉1 2 bits. For the second pixel, because its location can only in the range of tn [^ +1,] 1, the maximum possible bits to store its location index should be q n t = ⌈log (−^)⌉2 21. Thus, we obtain the distance ^ 2 between the first and second pixels from the subsequent q2 bits and calculate the actual location ztt =+ ^21 2 of the second pixel. We continue in this manner until all structure information of m pixels are successfully obtained.

$$\hat{q}_i = \begin{cases} [log_2 n] & for\ i=1 \\ \max\{[log_2(n-z_i)],1\} & for\ 2 \le i \le m \end{cases}$$

$$\hat{z}_i = \begin{cases} \hat{t}_i & for\ i=1 \\ \hat{t}_{i-1+}\hat{t}_i & for\ 2 \le i \le m \end{cases}$$

$$\hat{c}_b = \begin{cases} n-2 & for\ m=0 \\ n-3-p-\sum_{i=1}^{m} \hat{q}_i & for\ 1 \le m \le n_a \\ -2 & otherwise \end{cases}$$

2.2.2. Phase 2 after obtaining the extracted payload and bad block indices

We recover the first two pixels of each bad block using two bits of the payload. Thus, the remaining payload bits are extracted messages. The procedures of message extraction and image recovering of BBE are given in Algorithm.

Algorithm 2. Message extraction and recovering.

Input: Image I with message block size s_1 x s_2,

```
1: Initialization: p= [], bad block index b=[ ].
2: Divide I into non-overlapping blocks $B_{i.j}$ with a size of $s_1$ x $s_2$ .
Calculate parameters n and p.
3: for each block $B_{i.j}$ do
4: Scan $B_{i.j}$ to obtain the pixel sequence [$a_1$,$a_2$,…….,$a_n$].
5: if [$a_1$, $a_2$]=[1,1] then
6: P←[P ,[$a_3$,$a_4$,……..,$a_n$]].$B_{i.j}$←Set all pixels to1.
7: else [$a_1$, $a_2$]=[1,1] then
8: P←[P,[$a_3$,$a_4$,…….,$a_n$]]. $B_{i.j}$←Set all pixels to 0.
9: else [$a_1$, $a_2$]=[1,1] then
10: Calculate m, obtain the m pixels location $z_i$ using eq. (6).
11: P←[P,[$a_{4+p+}\sum_{i=1}^{m} q_1$,……,$a_n$]].
12: if $a_3$=1 then
13: $B_{i.j}$←set all pixels to 1 except for the m pixels with location $z_i$.
14: else
15: $B_{i.j}$←set all pixels to 0 except for the m pixels with location $z_i$.
16: end if
17: else
18: b←[b;(i,j)]. % record the bad block index
19: endif
20: end for
21: Extract B and M from P.
22: for each bad block $B_{i.j}$ by 2 bits of B.
23: end for
```

Output: Recovered image I, message M.

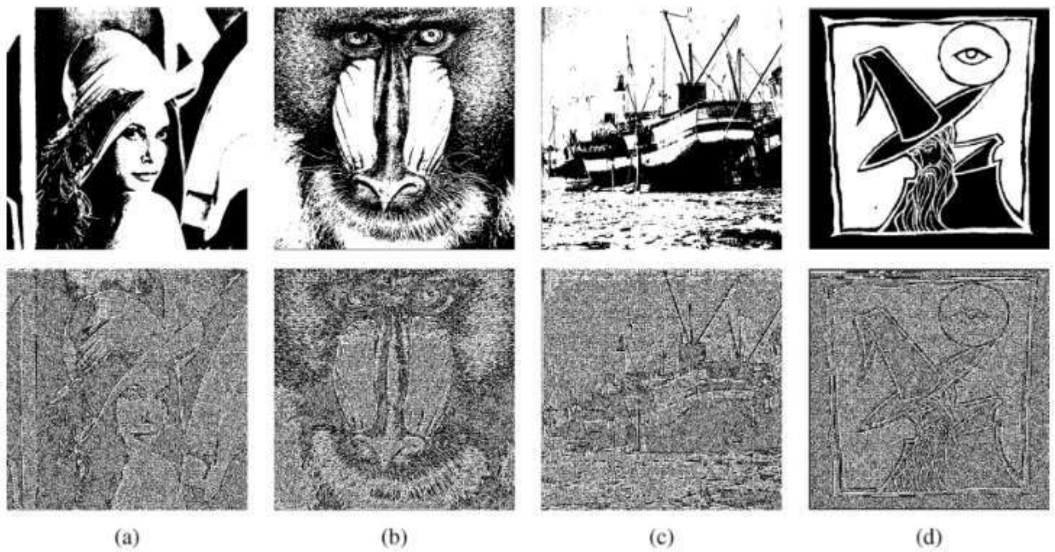


Fig. 2. Embedding results of BBE with a block sizes == 4 12. The first and second rows show the original images and their embedded results with an embedding rate (a) 0.7100 bpp, (b) 0.3536 bpp, (c) 0.6332 bpp and (d) 0.6730 bpp.

2.3. SIMULATION RESULTS

BBE introduces noise to uniform regions, changes the uniform regions, and keeps only the edges. These operations yield an image with more noise. Fig. 2 shows the embedding results of four 512×512 binary images using the BBE algorithm with the block size of 4×4. As we can obverse, the more all-white or all-black blocks the original image contains, the higher embedding rate BBE can achieve. Meanwhile, the embedded images become more noise-like.

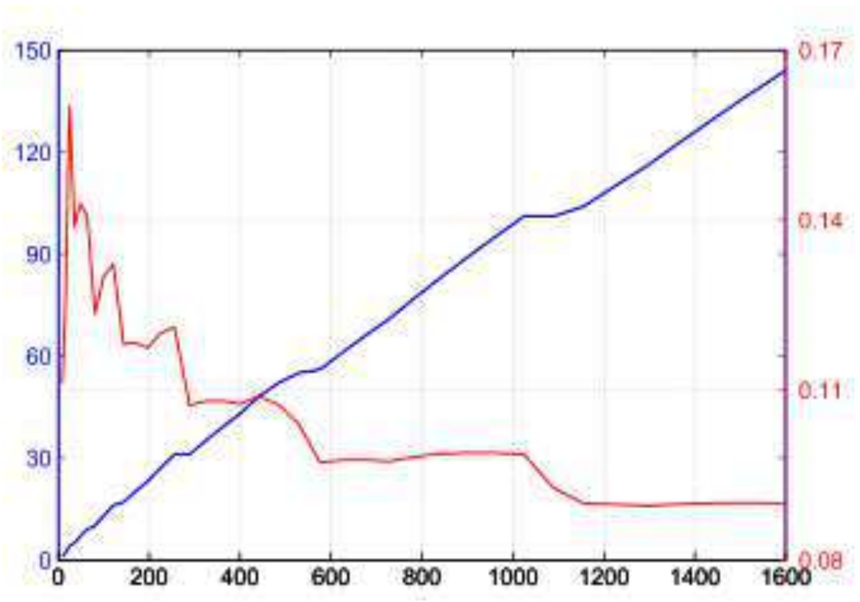
2.4. DISCUSSION

Here, we discuss the threshold na, embedding rate and advantages of BBE.

2.4.1.Threshold na As shown in Eq. (1) and Table 1, the block sizes ×1 2 will influence the value of na, which is a threshold to decide the block types. Fig. 3 visually shows the relationship between the threshold na and block pixel number n. We set ss 3 ≤, ≤ 4012, thus, n ∈ [9, 1600]. As can be seen, na increases with the increase of n. For example, if == 3 12, we can obtain the threshold na=1. This means that a 3×3 image block containing at most one 0 or 1 is considered as a good block, otherwise, it is a bad block. In addition, when the block size is 5×5, n n /a reaches the maximum value of 0.16. Therefore, in Eq. (1), x should be less than or equal to n [0.16*].

2.4.2. Embedding rate To analyze the embedding rate

we apply BBE with different block sizes to 10,000 binary test images that are generated by binarizing the gray-scale images from BOWSBase1 with the threshold calculated by Otsu's method [24]. Table 2 lists the average embedding rates of all test.



n_a/n $n_a n$

Fig.3. Relationship between threshold n_a and block pixel number n.

Table 2

Average embedding rate of 10000 binary images using BBE with different block sizes

Block size S1×S2	Avg. Embedding Rate(.bpp)	Block Size S1×S2	Avg. Embedding Rate(.bpp)
3×3	0.6628	11×11	0.7308
4×4	0.7431	14×14	0.7014
5×5	0.7600	17×17	0.6975
6×6	0.7628	21×21	0.6568
7×7	0.7685	28×28	0.6180
8×8	0.7679	33×33	0.5708
9×9	0.7348	40×40	0.5072

Images with different block sizes. Here we set ss=12. From the result, we can observe that,the avg. embedding rate reaches the maximum value when the block size is 7×7. When block size is less than 7×7, the average embedding rate increases with the block size enlarging, and it decreases when block size is larger than 7×7.

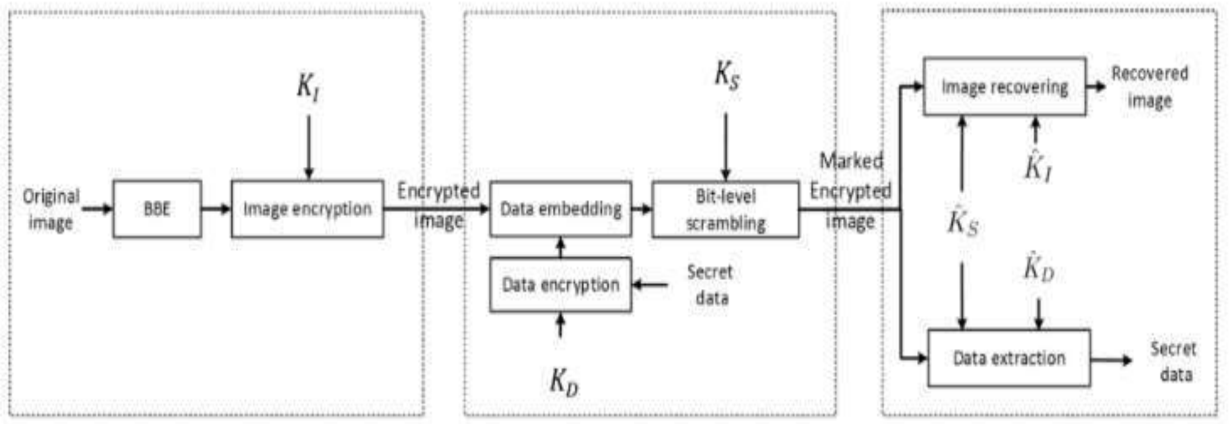
2.4.3. Advantages

The proposed BBE has at least the following advantages. Namely, BBE is able to-(a) achieve a higher embedding rate when the original image contains more all-white or all-black blocks.

(

(b) perform data hiding and image quality degradation within one single step. This is because BBE embeds secret messages by modifying the pixels values in good blocks while keeping the randomness of bad blocks.

(c) Completely recover the secret messages and original image without any error.



Content Owner

Fig.4. The structure of BBE-RDHEI.

3. BBE based reversible data hiding in encrypted images

In this section, we propose a BBE based reversible data hiding algorithm for encrypted images (BBE-RDHEI). The structure of BBERDHEI is shown in Fig. 4. It is composed of three processes: generation of the encrypted image, generation of the marked encrypted image, data extraction/image recovery. These processes are accomplished by the content owner who provides the original image, the data hider who has the secret data to be embedded and the receiver, respectively.

Data Receiver Hider

The content owner uses the BBE algorithm to embed binary bits of lower bit-planes of the original image into its higher bitplanes.such that its lower bit-planes can be reserved for hiding secret data in the subsequent processes. The image is then encrypted using the image encryption key KI. The data hider encrypts the secret data using the data encryption key KD, embeds them into the reserved lower bit-planes in the encrypted image, and scrambles the image using the sharing encryption key KS to generate the marked encrypted image which will be transmitted over public channels. These three encryption keys are randomly generated by users.

3.1. Random sequence generation

Before presenting three processes of BBE-RDHEI in detail, we discuss the security key design and random sequence generation. Their framework is shown in Fig. 5. A secure hash algorithm 2 (SHA) is used to generate two random hash sequences with the inputs of a user-defined security key K and image/secret data, respectively. Then the two hash sequences are XORed to generate the inner random sequence K. K is utilized to initialize a chaotic system to produce the random sequence that will be used for encrypting the original image and secret data. The random sequence K is used for secret data extraction and image recovering. Thus, it is called the decryption key. The length of security key K is user-defined and the decryption key K is with the same length of the output of SHA. Using the framework in Fig. 5, any change in the image/secret data.

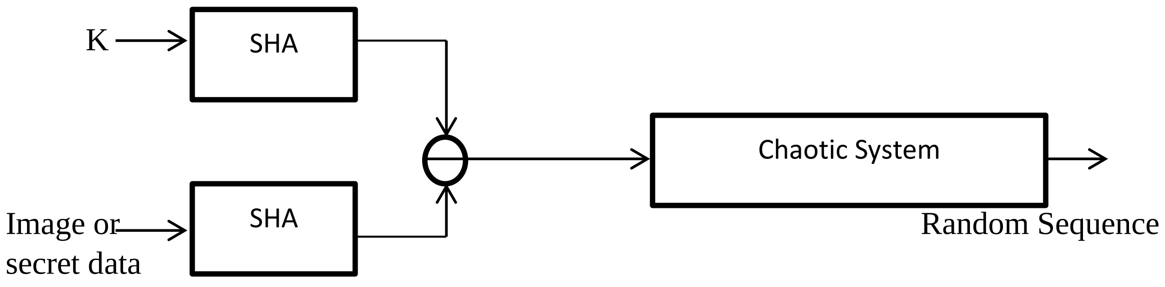


Fig.5. The generation framework of the security key & random sequence.

3.1.1. Security key design BBE-RDHEI has three encryption keys, KI, KD and KS. They all are random bit sequences generated by users. In addition to obtaining the marked encrypted image, BBE-RDHEI also produces three corresponding decryption keys KI, KD and KS for the receiver to extract the secret data, marked decrypted image, decrypted image, or all of them if he holds KS along with KD, KI or both, espectively. These decryption keys are linked with their corresponding encryption keys and contents of images or secret data as shown in Fig. 5. They are defined by

$$\begin{bmatrix} \hat{k}_\iota \\ \hat{k}_D \\ \hat{k}_S \end{bmatrix} = \begin{bmatrix} H(I) \\ H(P) \\ H(E) \end{bmatrix} \oplus \begin{bmatrix} H(k_I) \\ H(k_D) \\ H(k_S) \end{bmatrix}$$

Where K is an SHA; I, P and E represent the original image, secret data, and encrypted image, respectively. Note that users have flexibility to choose any SHA for Eq. (8). In this paper, we select SHA-13 for simulations. Thus, three decryption keys and outputs of having a length of 160 bits. According to Eq. (8)KI is linked to KI and the original image I; KD is linked to KD and the secret data P; KS is linked to KS and the encrypted image E. Thus, any change in the encryption key or the input of will result in a completely different decryption key and thus another chaotic sequence.

3.1.2. Chaotic sequence generation

A chaotic sequence is a random sequence that is sensitive to the parameter and initial value of its chaotic system. Any chaotic system can be used to generate the chaotic sequence, and we choose the Logistic-Sine system (LSS) [25] for demonstrations and it is defined by xyxxy sin πx=((1−))+(4−)()/4)mod1 ii ii+1 (9) where the initial value x0 (x∈[0,1]0) and parameter y(y∈(0,4]) are calculated by Algorithm 3 with a binary hash sequence H. Then, we will use this initial condition (xy,0) in rest of this paper.

Algorithm 3. Generation of initial value and parameter of LSS

```
Input: Binary sequence
H = [ $h_1$,$h^2$,……….$h_{160}$]($h_i \in \{0,1\}$,
1 ≤ i ≤ 160).
1: $u_1 \leftarrow \sum_{i=1}^{40} h_i 2^{40-i}$
2: $u_2 \leftarrow \sum_{i=41}^{80} h_i 2^{80-i}$
3: $v_1 \leftarrow \sum_{i=81}^{120} h_i 2^{120-i}$
4: $v_2 \leftarrow \sum_{i=121}^{160} h_i 2^{160-i}$
5: Initial value X0 $\leftarrow u_1/2^{40}$
6: Parameter $y_0 \leftarrow u_2/2^{40}$
7: for i = 1 to 2 do
8: $x_1 \leftarrow (x_{i-1}u_iv_i/2^{40} + x_{i-1})$ mod1
9: $y_1 \leftarrow (y_{i-1}u_iv_i/2^{40} + y_{i-1})$ mod4
10: end for
11: $x_0 \leftarrow x_2$
12: $y \leftarrow 4 - y_2$
Output Initial conditions ($x_0$,y).
```

SIMULATION RESULTS & COMPARISON

The proposed BBE-RDHEI is implemented in Matlab. All test images in our experiments have a size of 512×512 and the pixel value range of[0, 255]. Fig. 6 shows the simulation results of BBE-RDHEI in the standard gray-scale Lena image with parameter s=8 and the embedding rate r=1.6834 bpp. As we can obverse, the marked encrypted image is a noise-like image. It protects both the original image and secret data. The unauthorized user has extremely difficulty to obtain any useful information from it. Two decrypted images (Figs. 6(c) and (d)) have no visual difference although the LSB lanes of the image in Fig. 6(c) carry secret data. Table 4 lists the average embedding rates of 10,000 images in BOWS Base4 using BBE-RDHEI under different block sizes. As can be seen, when the block size s=5, the images have the maximum embedding rates. When s is larger than 5, the embedding rate decreases while the block size increases. Even s is as large as 20, BBE-RDHEI has an average embedding rate of 1.6874 bpp. To compare the embedding rate, we apply BBE-RDHEI and five existing RDH methods to several selected images as shown in Fig. 7. The results are plotted in Fig. 8. In experiments, we set the block size.

CONCLUSION

In this paper, we have proposed a binary block embedding (BBE) method for embedding messages in binary images. Based on BBE, Graph showing PSNR Vs Embedding Capacity we have proposed a reversible data hiding algorithm in encrypted images (BBE-RDHEI) in which BBE is utilized for reserving the bit space for embedding secret data. BBE-RDHEI employs a bit-level scrambling process after secret data embedding to spread embedded secret data to the entire marked encrypted image. A security key design mechanism is proposed to enhance its security level. Both BBE and BBE-RDHEI have been proved to be reversible. Simulations and comparisons have shown that BBE-RDHEI outperforms other existing methods in terms of the embedding rate and PSNR results of the decrypted images. Security analysis has demonstrated the robustness of BBE-RDHEI in against different attacks.

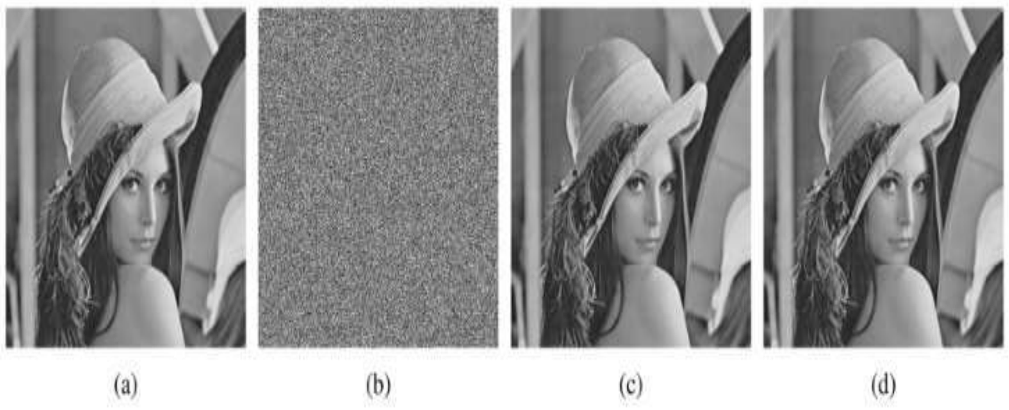


Fig.6 Data hiding & extraction using BBE-RDHEI with the blocl size 8*8 & embedding rate=1.6834bpp.(a)The original image;(b)The marked encrypted image;(c)The marked decrypted image; & (d)The decrypted image.

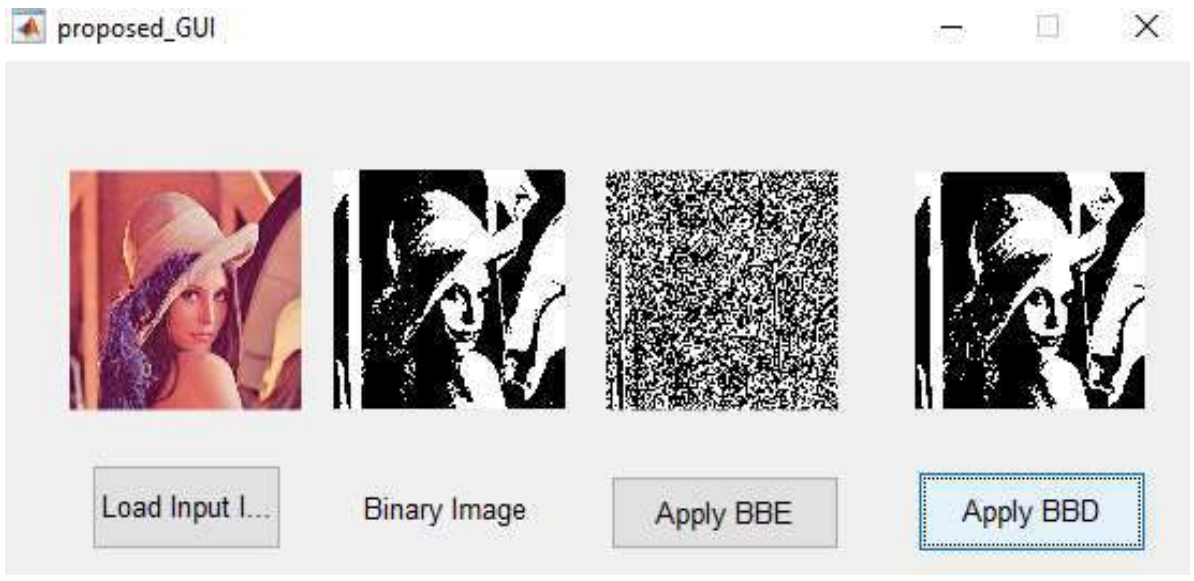


Fig.7 Proposed GUI

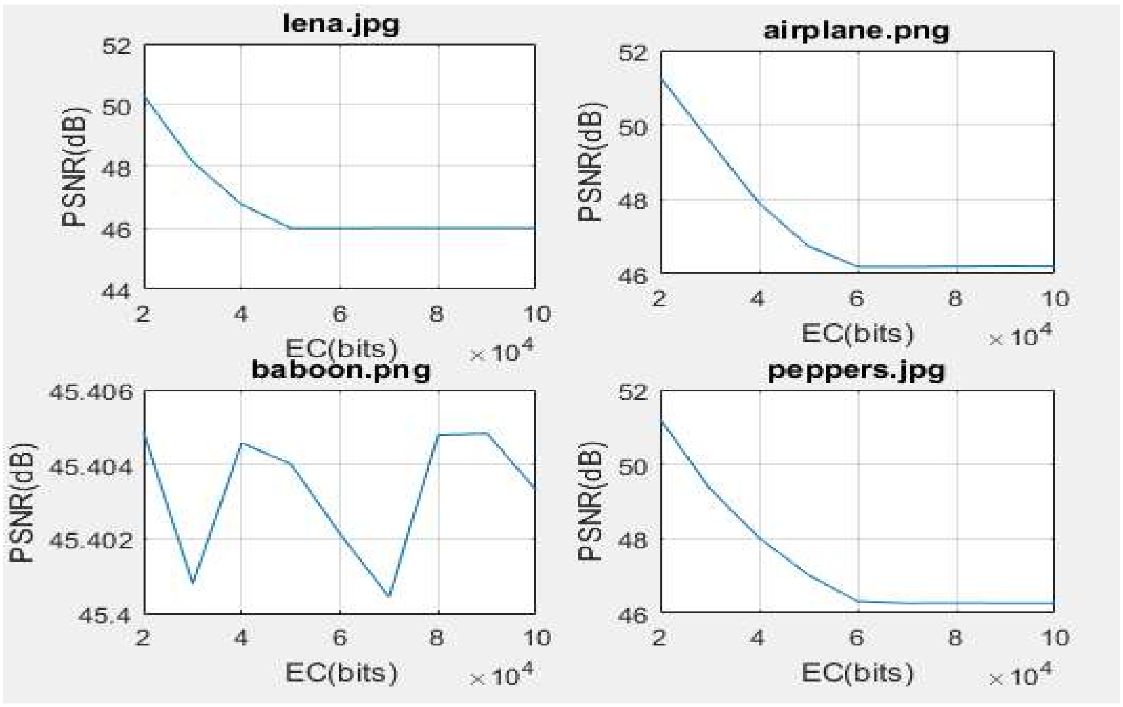


Fig.8 Graph showing PSNR Vs Embedding Capacity

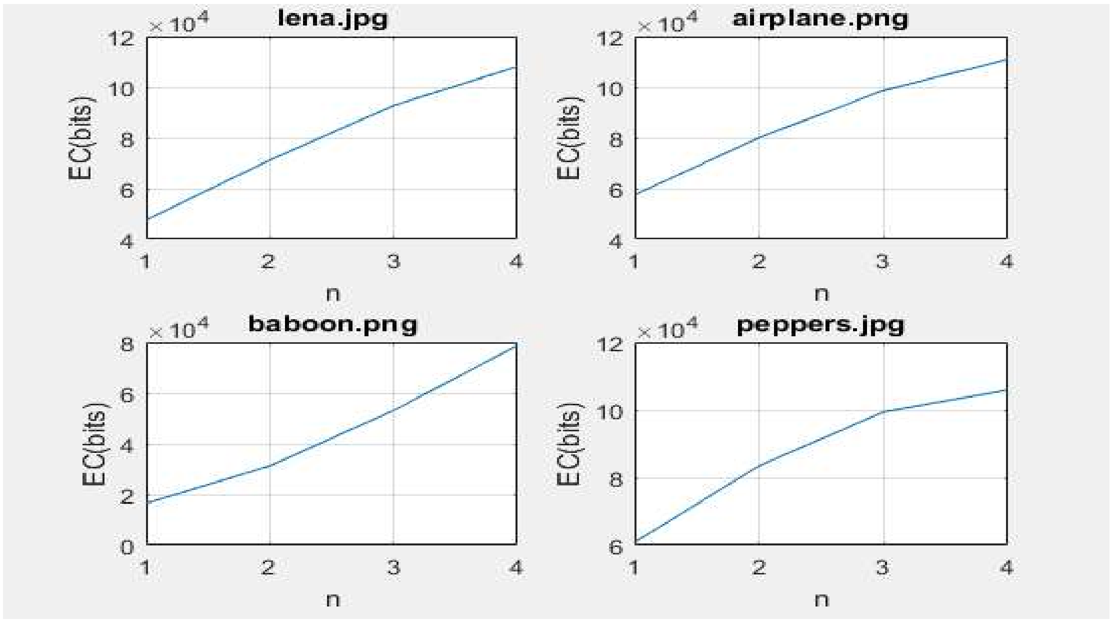


Fig.9 Graph showing embedding capacity vs no bits hidden

REFERENCES

1. *Z. Ni, Y.-Q. Shi, N. Ansari, W. Su, "Reversible data hiding","IEEE Trans. Circuits Syst. Video Technol". 16 (3) (2006) 354–362.*
2. *X. Li, B. Li, B. Yang, T. Zeng, "General framework to histogram-shifting-based reversible data hiding", "IEEE Signal Process. Lett". 22 (6) (2013) 2181–2191.*
3. *P. Tsai, Y.-C. Hu, H.-L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting", Signal Process. 89 (6) (2009) 1129–1143.*
4. *W. Hong, T.-S. Chen, C.-W. Shiu, "Reversible data hiding for high quality images using modification of prediction errors", "J. Syst. Softw." 82 (11) (2009) 1833–1842.*
5. *J. Tian, "Reversible data embedding using a difference expansion", "IEEE Trans. Circuits Syst. Video Technol". 13 (8) (2003) 890–896.*
6. *A. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform", "IEEE Trans. Image Process." 13 (8) (2004) 1147–1156.*
7. *H.-J. Kim, V. Sachdev, Y.Q. Shi, J. Nam, H.-G. Choo, "A novel difference expansion transform for reversible data embedding", "IEEE Trans. Inf. Forensics Secur". 3 (3) (2008) 456–465.*
8. *D. Coltuc, J.-M. Chassery, "Very fast watermarking by reversible contrast mapping", "IEEE Signal Process." Lett. 14 (4) (2007) 255–258.*
9. *X. Wang, X. Li, B. Yang, Z. Guo, "Efficient generalized integer transform for reversible watermarking", "IEEE Signal Process." Lett. 17 (6) (2010) 567–570.*
10. *F. Peng, X. Li, B. Yang, "Adaptive reversible data hiding scheme based on integer"*
11. *M.C.W. Puech, O. Strauss, "A reversible data hiding method for encrypted images, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X," in: Proceedings of SPIE 6819.*
12. *X. Zhang, "Reversible data hiding in encrypted image", "IEEE Signal Process". Lett. 18 (4) (2011) 255–258.*
13. *W. Hong, T.-S. Chen, H.-Y. Wu, "An improved reversible data hiding in encrypted images using side match", "IEEE Signal Process. Lett." 19 (4) (2012) 199–202.*
14. *X. Wu, W. Sun, "High-capacity reversible data hiding in encrypted images by prediction error", "Signal Process." 104 (2014) 387–400.*
15. *X. Zhang, "Separable reversible data hiding in encrypted image", "IEEE Trans. Inf. Forensics Secur." 7 (2) (2012) 826–832.*
16. *X. Zhang, Z. Qian, G. Feng, Y. Ren, "Efficient reversible data hiding in encrypted images", "J. Vis. Commun. Image Represent." 25 (2) (2014) 322–328.*
17. *Z. Qian, X. Han, X. Zhang, "Separable reversible data hiding in encrypted images by n-nary histogram modification", in: Proceedings of the Third International Conference on Multimedia Technology, 2013, pp. 869–876.*
18. *Z. Qian, X. Zhang, W. Shuozhong, "Reversible data hiding in encrypted JPEG bitstream", "IEEE Trans. Multimed". 16 (5) (2014) 1486–1491.*
19. *Y.-C. Chen, C.-W. Shiu, G. Horng, "Encrypted signal-based reversible data hiding with public key cryptosystem", J. Vis. Commun. Image Represent. 25 (5) (2014) 1164–1170.*
20. *X. Zhang, J. Wang, Z. Wang, H. Cheng, "Lossless and reversible data hiding in encrypted images with public key cryptography", "IEEE Transactions on Circuits and Systems for Video Technology" PP (99).*
21. *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 Secur. 8 (3) (2013) 553–562.*
22. *L. Luo, Z. Chen, M. Chen, X. Zeng, Z. Xiong, "Reversible image watermarking using interpolation technique", IEEE Trans. Inf. Forensics Secur. 5 (1) (2010) 187–193.*
23. *W. Zhang, K. Ma, N. Yu, "Reversibility improved data hiding in encrypted images", Signal Process. 94 (2014) 118–127.*
24. *N. Otsu, "A threshold selection method from gray-level histograms"," IEEE Trans. Syst. Man Cybern". 9 (1) (1979) 62–66.*
25. *Y. Zhou, L. Bao, C.P. Chen, "A new 1D chaotic system for image encryption", Signal Process. 97 (2014) 172–182.*
26. *Y. Zhou, W. Cao, C.P. Chen, "Image encryption using binary bitplane", Signal Process. 100 (2014) 197–207.*
27. *H.W. Wang, Shuozhong, "Cyber warfare: Steganography vs. steganalysis", Communications of the ACM 47.*
28. *W. Diffie, M.E. Hellman, "New directions in cryptography", "IEEE Trans. Inf. Theory" 22 (6) (1976) 644–654.*
29. *F.B.M. Barni, T. Furon, "A general framework for robust watermarking security", "Signal Process" 83 (10) (2003) 2069–2084.*
30. *F. Cayre, C. Fontaine, T. Furon, "Watermarking security: theory and practice", "IEEE Trans. Signal Process". 53 (10) (2005) 3976–3987. [31] R. Anderson, Security engineering: A Guide to Building Dependable Distributed Systems, Wiley.*