Signal Processing: Image Communication 74 (2019) 78–88

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

A novel secret image sharing with steganography scheme utilizing Optimal Asymmetric Encryption Padding and Information Dispersal Algorithms

Amir M. Ahmadian *, Maryam Amirmazlaghani

Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Keywords:
Secret sharing
Steganography
Image sharing

ABSTRACT

Secret image sharing schemes are employed to safeguard the confidentiality and availability of critical commercial or military images. However, many of the existing image sharing schemes have some security flaws that can reveal key elements of the shared secret image. In this paper, to overcome these weaknesses, we propose a new secret image sharing scheme which utilizes Optimal Asymmetric Encryption Padding (OAEP) and Information Dispersal Algorithms (IDA). The proposed scheme provides computational security, small shadow size, and better performance in share generation and secret reconstruction. We also analyze the security of the proposed scheme formally and prove that the confidentiality of the secret image is guaranteed in the face of computationally bounded adversaries. Additionally, we introduce an edge-based steganography method to conceal the participants' shares in cover images. The proposed technique achieves higher visual quality and better resistance against steganalysis methods. Finally, experimental results confirm the efficiency and effectiveness of the proposed scheme.

1. Introduction

Nowadays with the rapid expansion of computer networks, transmitting data over a network is such a common and popular task that without it our daily routines would encounter numerous difficulties. However, there are critical and confidential intelligence, diplomatic, or commercial data that needs to be protected against unauthorized access or manipulation during transmission and storage.

The most common approaches for preserving the confidentiality of a sensitive data are *cryptography* and *steganography*. Cryptography is the process of converting data from a readable state to apparent nonsense, in such a way that only entities with access to the encryption key can reverse the process and return the data to its initial readable state. On the other hand, steganography does not change the state of data and only conceals it in a carrier medium so that an adversary cannot detect the presence of data and therefore access it.

However, both of these methods have a common pitfall. In cryptography, if the encrypted data or the encryption key becomes lost or corrupted during storage or transmission, the secret data is lost. In steganography, if the carrier media gets damaged or destroyed, in most cases, the embedded secret will become unrecoverable. This problem is commonly known as *the single point of failure*.

To solve this problem, the concept of (t, n)-threshold secret sharing was suggested by Blakley [1] and Shamir [2]. A (t, n)-threshold secret sharing scheme divides the secret data among n participants in such a way that t or more participants can reconstruct the original secret by pooling their shares together, and any less than t shares reveals no *useful* information about the shared secret.

The first secret sharing methods were only suitable for a few types of data, such as passwords and encryption keys. However, with the expansion of network bandwidths and the widespread use of different kinds of digital data such as image, audio, and video, it became necessary to design secret sharing methods for them as well. Particularly, how to share a secret image has attracted the attention of researchers because of the widespread uses of images in real-world applications.

Naor and Shamir [3] proposed a (t, n) secret image sharing scheme called Visual Secret Sharing (VSS) in which n shares are printed on n transparent papers and the secret image can be reconstructed by stacking t shares together. The most important property of VSS is that the secret reconstruction can be done without any computational process, and the secret image can be recovered directly by the human visual system. However, the VSS scheme was only suitable for binary images and it had some major drawbacks such as share size expansion, limited contrast, and reconstructed secret's poor visual quality.

To overcome these problems, with inspirations from Shamir's (t, n)-threshold scheme, Thien and Lin proposed a polynomial based secret image sharing (PBSIS) scheme. The most important characteristics of their method was the reduction of shares size to $\frac{1}{t}$ of the original secret image. Due to the significance of this feature, their method was used as a basis for many succeeding secret image sharing schemes [4–13]. It was believed that their scheme was secure, and any less than t shares

* Corresponding author.
E-mail addresses: amir.ahmadian@aut.ac.ir (A.M. Ahmadian), mazlaghani@aut.ac.ir (M. Amirmazlaghani).

https://doi.org/10.1016/j.image.2019.01.006
Received 15 August 2018; Received in revised form 19 November 2018; Accepted 15 January 2019
Available online 6 February 2019

were not revealing any useful information about the secret image, but as we will show in Section 3, this assumption is not true.

Instead of Shamir's scheme, Blakley's method can also be used as the basis for image sharing [14]. Additionally, the Asmuth–Bloom [15] secret sharing scheme which is based on the Chinese Remainder Theorem (CRT) may also be used for secret image sharing [16]. The major disadvantage of Blakley's and CRT based approaches is their share size which is as large as the secret image. This drawback alongside their significant computational complexity makes them unusable in most practical scenarios.

Some researchers used Linear Memory Cellular Automata (LMCA) as a basis for secret image sharing [17–19]. LMCA's linear computational complexity makes it ideal for sharing large images. However, they have a significant limitation which makes them impractical for real-world applications. In LMCA all of the subsets of participants with t shares cannot recover the secret, and only the subsets with t *consecutive* shares can reverse the LMCA and reconstruct the secret image.

Furthermore, the generated shares of all secret image sharing schemes are noise-like images and they may attract the attention of attackers during transmission or storage. Therefore the use of steganography techniques was proposed to conceal the existence of image shares. Over the years, different steganography methods such as simple LSB substitution [6,9,18,20,21], LSB substitution with Optimal Pixel Adjustment Process (OPAP) [8,12], Exploiting Modification Direction (EMD) [10], Modulus Operation [22], PSNR Estimation [13], and Integer Wavelet Transform [11] were used in secret image sharing schemes.

In our previous paper [23], we used a modified version of Rivest's All-or-Nothing Transform (AONT) [24] in combination with an Information Dispersal Algorithm (IDA) [25,26] to create a computationally secure secret image sharing scheme. The scheme provided small shadow size and had major performance improvements compared to the other methods. However, since the security of Rivest's AONT can only be investigated intuitively, no formal security analysis of the method were given.

As we mentioned before, from all of the introduced secret sharing methods, most of the previous image sharing schemes [4–13] use Shamir's method as their basis. Therefore, in this paper, we first review Shamir's method and the Polynomial-Based Secret Image Sharing (PBSIS) schemes that are based on it. Then we point out some of the security weaknesses of these methods. Next, to overcome these drawbacks, we introduce our computationally secure secret image sharing scheme which is an improved version of our previous work [23]. The proposed scheme uses Optimal Asymmetric Encryption Padding (OAEP) [27] to achieve computational security and an Information Dispersal Algorithm (IDA) [25,26] to generate the shares of participants.

Using OAEP allows us to formally analyze the security of our scheme in the Random Oracle (RO) model. Such analysis is important because in a secret image sharing scheme it is crucial to guarantee the confidentiality of the secret image in the face of adversaries. Therefore, we provide a security analysis to prove that an adversary with limited computational power – while having less than t shares – cannot learn any useful information about the secret image.

As mentioned before, the generated shares are noise-like and may attract the attention of attackers. Therefore, in order to complete the work, it is necessary to provide a way to securely send the generated shares. In this regard, this paper proposes a novel steganographic method based on Fully Exploiting Modification Direction (FEMD) [28] and a simple edge detection algorithm to hide the shares in cover images.

The major contributions of this paper can be summarized as follows:

1. Proposing a computationally secure secret image sharing scheme based on OAEP and IDAs. Using OAEP allows us to ensure the confidentiality of the secret image and an IDA can generate the shares and reconstruct the secret with significantly better performance compared to the other methods.
2. Investigating the security of the proposed scheme in the "Random Oracle" model. This analysis proves that a computationally bounded adversary with less that t shares cannot learn any useful information about the shared secret image.
3. Introducing an edge-based steganographic method which uses an edge detection algorithm to identify the edge areas of an image and then embeds the noise-like shares in those regions. The embedding process takes into account the size of the shares, which means that small shares are embedded only in the most distinct edges and therefore less distortion is introduced into the cover image. This approach achieves high stego-image quality and has better resistance against steganalysis attacks.

The rest of this paper is organized as follows. Preliminaries and related works are described in Section 2. The visual security of polynomial-based secret image sharing schemes is investigated in Section 3. The proposed "secret image sharing with steganography" scheme is introduced in Section 4. Section 5 analyzes the security of the proposed scheme, the experimental results are presented in Section 6, and finally Section 7 states the conclusions of the paper.

2. Preliminaries

In this section, the preliminaries of different parts of the present work have been described. First, Shamir's (t, n) threshold secret sharing scheme is briefly overviewed, and then the secret image sharing schemes that were based on it are introduced. Finally, we will briefly describe notions of the All-or-Nothing Transform (AONT) and Information Dispersal Algorithm (IDA), the two building blocks of our proposed secret image sharing scheme.

2.1. Shamir's (t, n) threshold secret sharing scheme

In 1979, Shamir [2] proposed a (t, n) threshold secret sharing scheme that was based on polynomial interpolation. In his scheme, the secret data S is divided into n shares $S_1, S_2, \ldots, S_n$ in such a way that t $(t \leq n)$ shares can reconstruct the secret using Lagrange's polynomial interpolation, but less than t shares can learn *nothing* about the secret S.

To share the secret S among n participants a polynomial of degree $t-1$ is generated:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \ mod \ p \tag{1}$$

Eq. (1) uses modular arithmetic, and all of the calculations are performed in mod p, which should be a prime number larger than both S and n. Additionally in Eq. (1), a_0 is the secret S and a_1 to a_{t-1} are random numbers chosen uniformly from $[0, p-1]$.

In order to calculate the participants shares, a unique and positive identifier x_i is assigned to the ith participant, and its share is calculated as follows:

$$S_i = f(x_i) \ mod \ p \quad for \ i = 1, 2, \ldots, n \tag{2}$$

From Eq. (2), it is evident that the size of each share S_i is in the range $[0, p-1]$, hence the share's size does not exceed the original secret's size. To recover the secret, when t or more shares (S_i, x_i) are available, the original $t-1$ degree polynomial can be reconstructed using Lagrange polynomial interpolation [2].

2.2. Polynomial based secret image sharing scheme

Based on Shamir's method, Thien and Lin [4] proposed the first polynomial based secret image sharing scheme (which we call **PBSIS** henceforth). Their scheme first permutes a grayscale image randomly using a secret key, then divides it into several non-overlapping blocks. Each block consists of t pixels and they are used as the t coefficients $a_0, a_1, \ldots, a_{t-1}$ of the polynomial (1).

Similar to Shamir's scheme, each participant will have a unique identifier x_i and the value of polynomial in that point $F(x_i)$ will be a pixel of that participants share. This operation is repeated until all of the secret image blocks are processed. Finally, the share pixels of each participant are merged to form a share image (which we will call **shadow** hereafter).

Since PBSIS does not use any random coefficients and t secret pixels are embedded directly in all of the polynomial (1) coefficients, the size of shadows is reduced to $\frac{1}{t}$ of the secret image's. This is especially important because the size of the secret images are usually quite large, and a way to reduce the size of shadows can be beneficial while transmitting them over a network, storing them, or embedding them in a carrier media. As a result of these benefits, **most** of the succeeding secret image sharing schemes used the PBSIS approach for sharing images [4–13]. However, PBSIS's approach weakens the security of the original Shamir's scheme, and we will discuss some of these security weaknesses in Section 3.

2.3. All-or-Nothing Transform

All-or-Nothing Transform (AONT) was first introduced by Rivest [24] as an encryption mode to increase a cryptographic system's security against exhaustive key search attacks without increasing its key length.

AONT is an unkeyed, invertible and randomized transformation that maps a sequence of input blocks $d_1, d_2, \ldots, d_m$ to a sequence of output blocks $c_1, c_2, \ldots, c_{m'}$ under the following conditions:

- If all output blocks $c_1, c_2, \ldots, c_{m'}$ are available, it is fairly easy to invert the transformation and recover the original input blocks $d_1, d_2, \ldots, d_m$.
- Even if one of the output blocks is missing, it is **computationally infeasible** to invert the transformation or learn *any information* about the input blocks.

2.4. Information dispersal algorithm

In a network, an Information Dispersal Algorithm (IDA) distributes a data among n nodes in such a way that the recovery of data is possible if k or more nodes are active, where n and k are parameters satisfying $1 \le k \le n$.

IDA was first introduced by Rabin [25] to guarantee availability and fault tolerance capability of data in a network. The biggest difference between a secret sharing scheme and an IDA is the secrecy of data and shares. While in a secret sharing scheme confidentiality of the original data is very important and shares should not leak any information about it, in an IDA availability of the data is important and IDAs do not deal with the secrecy of information.

The basic idea of an IDA is to add some redundancy to the data and then partition it among n parties. Therefore an information dispersal algorithm can be implemented using any type of erasure codes [26,29]. For example a Reed–Solomon based IDA can be illustrated as a matrix–vector product [26]:

$$\begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,k} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,k} \\ g_{3,1} & g_{3,2} & \cdots & g_{3,k} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n,1} & g_{n,2} & \cdots & g_{n,k} \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix}$$

The $n \times k$ generator matrix G is multiplied by the data vector D and the outcome is stored in the result vector R. Then each element of R is sent to a different node in the network as their share.

Any combinations of t rows of generator matrix must produce an invertible matrix. This way, for every t elements of R, their corresponding rows in G creates a $t \times t$ invertible matrix. The available t elements of R multiplied by the inverted matrix will result in the initial data vector D. Therefore every t elements of R can reconstruct the data vector D.

Vandermonde Matrix [26,29] and *Cauchy Matrix* [25] are some examples of generator matrices with such property.

(a) Camera Man (b) (c) (d)

Fig. 1. Camera man image and generated shares.

Fig. 2. Recovered image with only one shadow.

3. Visual security of the PBSIS based methods

Since in some regions of a secret image, the neighboring pixels are extremely correlated, embedding pixels of the same region to the coefficients of polynomial (1) can result in a shared pixel that is similar to the original pixels of that region. Therefore, some areas of the shadows will remain similar to the secret image, and consequently, expose key elements of the secret image. Fig. 1 illustrates this security issue.

To overcome this issue, [4] proposed to apply permutation to the secret image before the share generation process. Such a permutation breaks the adjacent pixels correlation and results in shadows that are not similar to the secret image and hence do not leak any information about it.

However, using permutation is not a completely secure approach and it is still possible for an attacker or a malicious participant to learn some information about the secret image [12]. For example Fig. 2 depicts the information an adversary can learn about the secret image (1a) while having only one share [23].

In Addition, permutation is not an entirely secure way for image encryption, and it was proven [30] that *some* permutation-only image encryption schemes are not secure against ciphertext-only attacks and practically *all of them* cannot resist known/chosen-plaintext attacks. In 2016, Jolfaei et al. [30] proposed a *chosen-plaintext* attack that can break any *permutation-only* image encryption scheme. Therefore, a secret image sharing scheme that uses a permutation-only image encryption for confidentiality – at least in terms of indistinguishability (IND-CPA) – is not secure.

4. Proposed scheme

4.1. Secret sharing phase

In this section, we are going to propose our computationally secure secret image sharing scheme which uses a two layered structure. Instead of permutation, our scheme uses OAEP [27] to provide confidentiality for the secret image and a systematic IDA [31] to generate the shares.

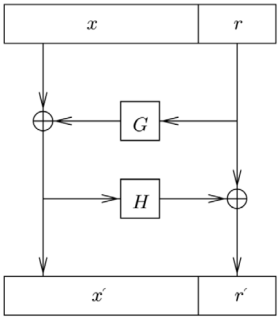


Fig. 3. Diagram of OAEP.

4.1.1. OAEP as an AONT

In this paper we use Optimal Asymmetric Encryption Padding (OAEP) [27] to provide confidentiality. As it was mentioned before, using OAEP allows us to analyze the security of our scheme in the Random Oracle (RO) model.

For a secret data of length n and a security parameter λ, OAEP uses the following functions:

$$G : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{n}$$
$$H : \{0,1\}^{n} \rightarrow \{0,1\}^{\lambda}$$

where G and H are random oracles. Using these functions, $OAEP : \{0,1\}^{n} \times \{0,1\}^{\lambda} \rightarrow \{0,1\}^{n+\lambda}$ is defined as:

$$OAEP^{G,H}(x,y) = x \oplus G(r) \;\|\; r \oplus H(x \oplus G(r))$$

where $\|$ denotes the string concatenation, x is the input message of length n, r is a random string of length λ, and λ is the security parameter of the transform (e.g. $\lambda = 256\ bit$). A diagram of OAEP is depicted in Fig. 3.

In practical applications, H function can be implemented by a hash function such as $SHA-256$ and a mask generation function like $MGF1$ can be used as function G.

4.1.2. Systematic IDA

As it was mentioned before, any erasure code can be used as an IDA. In our scheme, we use a *systematic* Reed–Solomon code as an IDA to generate the shares.

A non-systematic Reed–Solomon code can be easily converted to a systematic one by modifying its generator matrix. However, because of space limitations, we omit the explanation of this procedure and refer the readers to our previous paper [23] or the original articles on the matter [29,31].

Generally, in a non-systematic code, the output does not contain any of the input's symbols. On the other hand, in a systematic code, the input data is directly encoded in the output, therefore, some of the input elements can appear in the output. Thus, a systematic IDA is usually considered *not secure*, however, in our scheme, we provide confidentiality with OAEP, therefore, using a systematic IDA instead of PBSIS can improve the overall performance of our scheme without weakening its security.

4.1.3. Share generation process

Using *OAEP* and *systematic Reed–Solomon based IDA*, the proposed secret image sharing scheme consists of the following steps:

1. Apply OAEP to the secret image S, the result will be the noise-like image C.
2. Partition C into t non-overlapping, equal sized blocks.
3. Take the first unprocessed pixel from each of the t blocks and form the data vector D of the IDA algorithm.
4. Using systematic IDA, map vector D to vector R.
5. The ith element of R is a pixel of the ith participant's shadow image (for $i = 0, 1, \ldots, n$).
6. Repeat steps 3 to 5, until all of the pixels are processed.
7. Finally, concatenate the pixels of each participant together and generate the corresponding shadow image.

Fig. 4 illustrates the share generation procedure.

In most of the practical applications, compared to the large size of secret images, λ is small and negligible. Therefore, in our scheme, the size of shadows is roughly equal to $\frac{1}{t}$ of the secret image size.

As it was mentioned before, the generated shadows are noise-like images, and they can attract the attention of attackers during storage or transmission. To overcome this issue, it is conventional for secret image sharing schemes to hide the shadows in cover images using a steganographic method. Therefore in the next part, we are going to introduce our proposed steganography scheme.

4.2. Steganography phase

In this section, we first introduce the FEMD method [28] which is the basis of our steganography scheme, then we investigate the benefits of embedding in the image's edges. Finally, we combine the FEMD method with a simple edge detection algorithm to create a steganographic scheme that embeds the shadows in the edges of cover images.

4.2.1. FEMD steganography method

In 2011, Kieu and Chang proposed the Fully Exploiting Modification Direction (FEMD) steganography method [28]. Their method had better visual quality compared to LSB substitution and also provided embedding capacities of more than 1 bpp.

FEMD partitions the cover image into non-overlapping blocks of two pixels. In order to embed the secret data into a block, each of its pixels can be increased or decreased by some value, and this modification value determines the embedding capacity of the method. For example, when each pixel can be increased or decreased by *one*, the embedding capacity will be 1.5 bpp.

In the FEMD method, parameter S defines the embedding capacity and the limit of pixels modification. Based on this parameter, the number of bits embedded in each pixel is $k = \frac{\lfloor log_2 S^2 \rfloor}{2}$ and the amount of pixel modification is limited to $r = \lfloor \frac{S}{2} \rfloor$.

Using these parameters, FEMD defines the embedding function F as:

$$F(g_i, g_{i+1}) = [(S-1) \times g_i + S \times g_{i+1}]\ mod\ S^2 \tag{3}$$

In this method, function F is used to generate the 256×256 sized mapping matrix M. To do so, for all of the possible values of pixels g_i and g_{i+1}, the function F is calculated and its result is stored in the element $M[g_i][g_{i+1}]$ of the matrix. In other words:

$$M[g_i][g_{i+1}] = F(g_i, g_{i+1})\ for\ g_i, g_{i+1} = 0, 1, \ldots, 255 \tag{4}$$

As it is evident from Eqs. (3) and (4), for each parameter S, we should generate a different mapping matrix M. Utilizing matrix M, the data embedding procedure of the FEMD method is as follows:

1. Partition the cover image into two-pixel blocks and the secret message to k bit blocks.
2. Convert the first unprocessed block of the secret message to its corresponding decimal value d. We call this value the secret number.
3. Calculate function F for the first unprocessed block of the cover image.
4. If $F(g_i, g_{i+1}) = d$, the embedding process is done and the stego-pixels are g_i, g_{i+1}, otherwise search the matrix M to find an element such as $M[x][y]$ where its value is equal to the secret number d, then the stego-pixels will be equal to x and y respectively. In this search, the element $M[g_i][g_{i+1}]$ is the center and the searching radius is restricted to r.

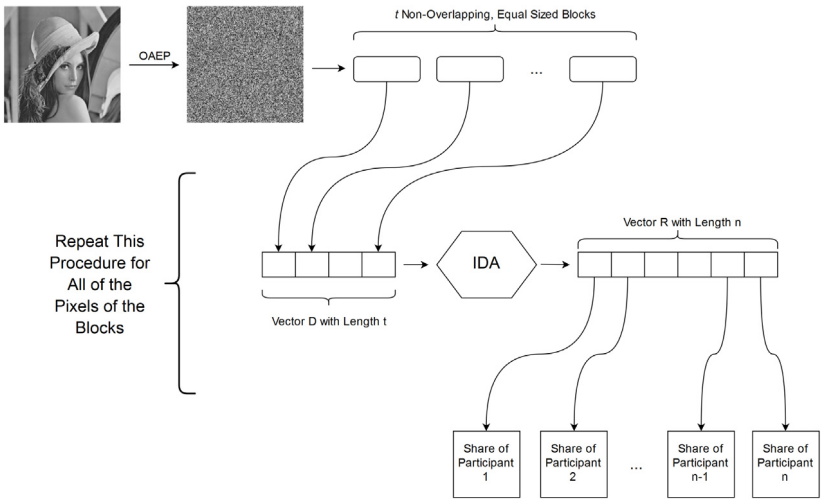


Fig. 4. Share generation process.

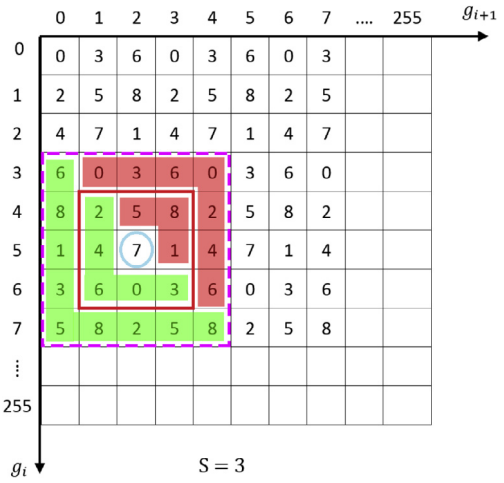


Fig. 5. The mapping matrix M for $S = 3$.

5. Repeat the steps of 2 to 4 until all of the secret message blocks are processed.

For example, Fig. 5 shows the mapping matrix M generated for $S = 3$. Assume that we want to embed the secret number 8 in pixels 5 and 2. Because $F(5,2) \neq 8$, after searching the matrix M in radius 1, we will find the element $M[4][3]$ which has a value of 8. Thus, the stego-pixels are equal to 4 and 3.

The data extraction procedure of the FEMD scheme is fairly straight forward. Knowing the parameter S, calculate function F for each of the stego-image's blocks and then convert its result to binary. One of the advantages of the FEMD method is its ability to embed more than 1 bpp in the cover image, and this capability is controlled by the parameter S. Bigger S increases the search radius and the amount of pixels modification, hence more data can be embedded in the cover image.

4.2.2. Benefits of embedding in the edges

Our steganographic scheme combines the FEMD method with a simple edge detection algorithm because embedding in the edges of an image has some significant advantages:

I. Embedding in the edges reduces the visual distortion of stego-images. Since the pixel values of the edges are more diverse compared to the smooth regions, the human visual system is less sensitive towards changes in them [32]. This decrease in the distortion is not detectable by a general criterion like PSNR, but a criterion such as SSIM which considers the characteristics of the human visual system can detect this distortion reduction very well.

II. Embedding in the edges increases the resistance of stego-images against steganalysis techniques, because:

- In natural images, the LSB of the edge pixels is random and noise-like, but the LSB plane of the smooth regions is not always random, and in some cases, they even contain some textural information about the image. Therefore embedding in these areas will make the LSB planes more and more random, which may lead to visual artifacts and statistical differences between the cover image and its corresponding stego-image [32].
- The edge information of an image is highly dependent on its content, hence the embedding position for each image is different from the other ones, and this makes detecting the stego-images even harder.

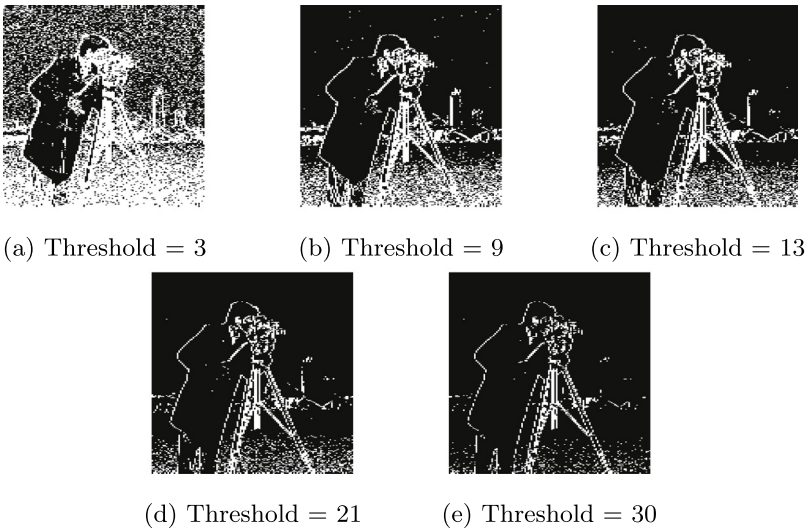
Furthermore, because the size of the generated shadows are relatively large and the cover images usually have a small number of edges, we need an algorithm that can embed more than 1 bit per pixel. Hence we chose the FEMD method because it can embed 1 to 4.5 bpp and achieves better visual quality compared to the LSB substitution method.

4.2.3. The proposed steganography method

In the proposed scheme, similar to the FEMD method, we partition the cover image into non-overlapping blocks of two pixels, and then choose a positive integer as the threshold value. In each block, if the absolute difference of the pixels was greater than or equal to the threshold value, that block is considered an edge block and the secret data will be embedded in it using the FEMD method. Otherwise, the block is ignored, and no embedding will be done in it.

The threshold value determines the number of pixels detected as edges. The higher the threshold value, the more pixels that will be detected as edges. Fig. 6 shows the edges of image (1a) for different threshold values.

In this method, the data is only embedded in the edge blocks, but after the embedding process, the absolute difference of the block's pixels may change and become less than the threshold value. In this case, the search radius of the FEMD method is increased by *one*, and the search process is repeated to find an element equal to the secret data where the difference of its pixels is not less than the threshold.



(a) Threshold = 3 (b) Threshold = 9 (c) Threshold = 13

(d) Threshold = 21 (e) Threshold = 30

Fig. 6. Edges of "Camera Man" image for different thresholds.

In some circumstances, more than one element with such properties can be found, therefore to achieve the best possible quality, the element with the minimum difference from the original pixels is chosen. This way we can ensure that the chosen element produces minimum distortion in the cover image.

For example, see Fig. 5 and assume that the threshold value is equal to 3 and we want to embed data 5 in pixels $(5, 2)$. The resulted stego-pixels will be $(4, 2)$ and their difference is less than 3, so we increase the search radius by one and repeat the search process. The result is pixels $(7, 0)$ and $(7, 3)$. Now in order to find the pixel with minimum distortion, we should calculate their difference from the original pixels:

$$D_1 = |7 - 5| + |0 - 2| = 4$$

$$D_2 = |7 - 5| + |3 - 2| = 3$$

Therefore, the pixels $(7, 3)$ are chosen as the resulted stego-pixels.

In a natural image, the number of pixels that suffer from this problem is very low, so an increase in the search radius does not cause much distortion in the image.

The proposed steganography method consists of the following steps:

1. Take a two-pixel block (p_1, p_2) as input.
2. If the $|p_1 - p_2| >= threshold$, go to step 3, otherwise ignore this block.
3. Embed the secret bits in the block using FEMD method. The result pixels are (p'_1, p'_2).
4. If the result $|p'_1 - p'_2| < threshold$, increase the search radius by 1 and repeat step 3. However, this time ignore all of the previous results.
5. Repeat step 4 until $|p'_1 - p'_2| >= threshold$.
6. Return the pixels (p'_1, p'_2) as the stego-block.

As it was stated before, our scheme can use different thresholds and based on parameter S, various embedding capacities. Hence, before the embedding process, and based on the size of the secret message, the values of threshold and S should be chosen in such a way that the embedding process causes minimum distortion to the stego-image.

The following procedure determines the best values for threshold and S based on the size of the secret message:

1. Initially, an acceptable range for S and the threshold is determined, for example $S = [2, 8]$ and $threshold = [2, 30]$
2. Starting from the smallest S and the biggest threshold, estimate the embedding capacity of the whole image. If this capacity is less than the secret message's size, decrease the threshold by one and repeat the process until reaching the lowest acceptable value for the threshold, then if the capacity was still less than the secret message's size, increase S and set the threshold to its biggest value and start over. Repeat this process until the biggest possible S is reached. Then if the capacity is still less than the secret's size, the message is too large for this image and cannot be embedded in it.

Finally, using the determined values for threshold and S, the data embedding process consists of these steps:

1. Partition the cover image into non-overlapping two-pixel blocks.
2. Convert the shadow image into a binary string (secret message) and then partition it into blocks of length $\lfloor Log_2 S_2 \rfloor$.
3. Ignore the first 5 blocks of the cover image and starting from the 6th block, read the blocks one by one and if the difference of their pixels is more than the threshold, embed the first unprocessed block of the secret message in it using the FEMD method.
4. Repeat step 3 until all of the secret message blocks are embedded in the cover image.
5. In the end, embed the threshold value and S in the first 5 blocks using simple FEMD with $S = 2$ and without edge detection.

Fig. 7 depicts the data embedding process.

In the steganography phase of our scheme, each participant's shadow is converted to a bit string, and then it is embedded in a cover image using the proposed steganography method.

Finally, Fig. 8 illustrates the whole workflow of the proposed method.

5. Security analysis of the proposed scheme

In this section, we use a formal method similar to [33] to investigate the security of our scheme. We prove that the proposed secret image sharing scheme is secure and the confidentiality of the secret image is guaranteed as long as the random oracles of OAEP produce pseudo-random strings in such a way that a computationally bounded adversary cannot distinguish them from entirely random strings.

In this analysis, we suppose that the adversary is in the best possible condition, he has access to $t - 1$ shadows, the generator matrix of the IDA, and the values of n and t are public, and he has access to them. Also, we assume that with access to one shadow, the adversary can learn the size of the secret image. For example, if the size of a shadow is γ bits, the adversary can infer that the secret image's size is $L = \gamma \times t$ bits.

First, we use a Lemma from [34] to show that if an adversary has $t - 1$ shadows of size γ, there are 2^{γ} candidates for the secret. This

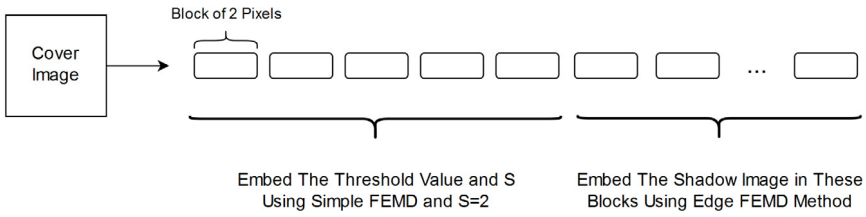


Fig. 7. The data embedding process.

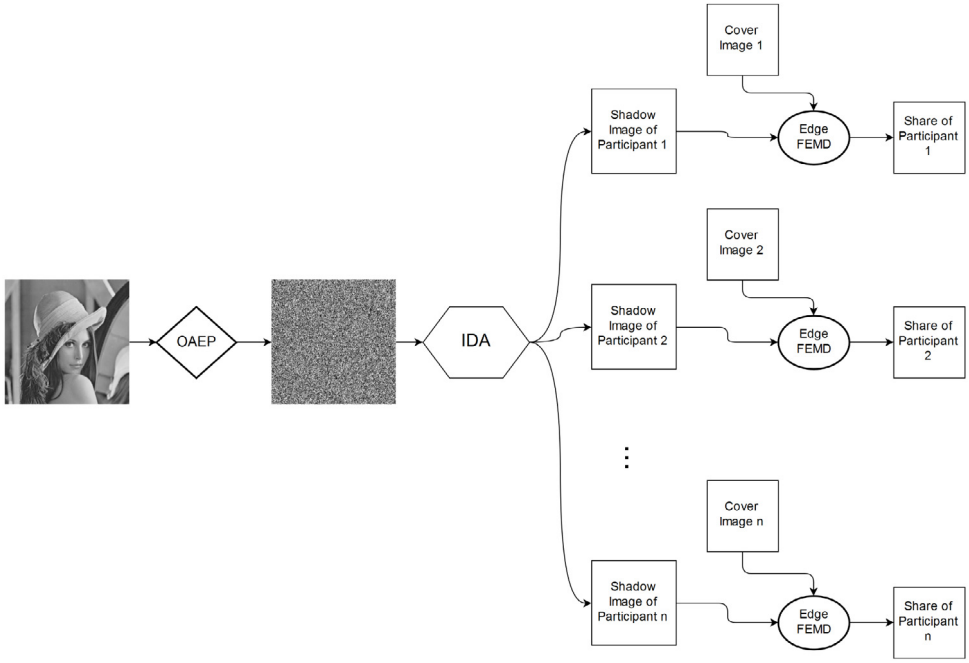


Fig. 8. The workflow of the proposed method.

Lemma does not imply how difficult it would be to find out which of these candidates is the right one. It just shows that there are these many *different* candidates. For the proof of this Lemma please refer to [34].

Lemma 1. *Assume that the secret image S contains L elements, and each element is a member of the finite field $\mathbb{F}$ ($S = s_1, s_2, \dots, s_L$; $s_i \in \mathbb{F}$), and the IDA divides S into n shadows of size γ. Given t-1 shadows, there are 2^{γ} candidates for $s_1, s_2, \dots, s_L$ [34].*

It is important to signify here that the large number of possible candidates cannot guarantee the scheme's security. As described in Section 3, if a secret image is shared using only an IDA, the adversary can learn vast amounts of information about the secret image without having access to t shadows.

However, in our scheme, the secret values $s_1, s_2, \dots, s_L$ are not shared directly and they only act as inputs of the random oracle. Therefore when there are 2^{γ} possible candidates for them – if γ is large enough –, the process of guessing the correct input for the random oracle becomes infeasible.

For example in OAEP, since the output of oracle G is random if an adversary has access to all but γ bits of x' ($\gamma \geq \lambda$), every one of the possible 2^{γ} candidates has an equal probability of being the correct input of oracle H. Now when γ is large enough and the adversary has limited computational power and bounded number of queries to oracle H, his chance to guess the right input for oracle H and thus to find the random value r and the initial secret input is *negligible*.

Now based on these discussions, we analyze the security of the proposed secret image sharing scheme using a modified version of Bellare and Rogaway's method [33]. Therefore, first, we briefly describe their approach in the next part.

PROCEDURE $Share(S)$
$K \xleftarrow{\$} \{0,1\}^{\lambda}$
$C \xleftarrow{\$} Mask_K(S)$
$P \xleftarrow{\$} K \oplus H(C)$
$\mathbf{V} \leftarrow IDA(C||P)$
RETURN $\mathbf{V}$

PROCEDURE $Recover(\mathbf{V}[0], ..., \mathbf{V}[t-1])$
$C||P \leftarrow IDA^{-1}(\mathbf{V}[0], ..., \mathbf{V}[t-1])$
$K = P \oplus H(C)$
$S = Unmask_K(C)$
RETURN S

Fig. 9. Share and recover procedures of a secret sharing scheme using generalized OAEP.

5.1. Bellare and Rogaway's method

Bellare and Rogaway's method uses indistinguishability to investigate the security of computational secret sharing schemes which use encryption to provide confidentiality for the secret data. Their approach uses two games to reduce the security of the secret sharing scheme to the security of its constructor encryption algorithm. These games are played between an *adversary* and a *challenger*.

The **Ind** game analyzes the indistinguishability of the encryption algorithm. This game consists of the following steps:

1. The challenger picks a bit $b \in \{0,1\}$ randomly and generates a random key $K \xleftarrow{\$} \{0,1\}^{\lambda}$
2. The adversary chooses two equal sized secret messages S_0 and S_1 and passes them to the challenger.
3. The challenger picks S_b, encrypts it using K, and passes the cipher-text to the adversary.

Table 1
Notations.

Sign	Description
$\oplus$	Bitwise XOR
H,G	Random oracles
$\|\|$	Concatenating two strings
$\mathbf{V}$	Vector V
$V[i]$	ith element of vector V
$\leftarrow$	Deterministic output
$\xleftarrow{\$}$	Nondeterministic output
$\{0,1\}^{\lambda}$	Binary string of length λ

4. The adversary guesses which of the secret messages were encrypted. The adversary wins if he can guess b correctly.

The second game **Priv**, uses the same approach to analyze the indistinguishability of the secret sharing scheme. In this game, the adversary should distinguish between two distinct secret values while having utmost $t-1$ shares. The steps of this game are as follows:

1. The adversary chooses two equal sized secret messages S_0 and S_1 and passes them to the challenger.
2. The challenger picks a bit $b \in \{0,1\}$ at random, shares S_b using the secret sharing algorithm, and generates share vector $\mathbf{S}$.
3. The adversary can ask the challenger for the ith share and receive $\mathbf{S}[i]$ in return. This process can be repeated $t-1$ times, hence the adversary can get utmost $t-1$ shares.
4. The adversary should guess which of the secret messages were shared, and adversary wins if he can guess b correctly.

Based on these games, the following definitions are made:

- $Adv_{\Pi}^{Priv}(A)$: the advantage of adversary A playing the game $Priv$ against the secret sharing scheme Π.
- $Adv_{\Omega}^{Ind}(B)$: the advantage of adversary B playing the game Ind against the encryption algorithm Ω.

For any computationally bounded adversary, the secret sharing scheme Π and the encryption algorithm Ω are secure in terms of indistinguishability, if $Adv_{\Pi}^{Priv}(A)$ and $Adv_{\Omega}^{Ind}(B)$ are negligible, respectively.

Bellare and Rogaway's method uses these games to prove if an adversary can distinguish between two distinct secret values with only $t-1$ shares, the same adversary can also be used as a *subroutine* to break the indistinguishability of the constructor encryption algorithm. In other words, if the adversary A has a significant advantage against the secret sharing scheme Π, an adversary such as B can use A to break the indistinguishability of the encryption algorithm Ω with the same advantage. For more information about their method, one can refer to [33].

5.2. Analysis of the proposed scheme

As it was mentioned, an encryption algorithm has the property of indistinguishability when an adversary cannot distinguish between the cipher-texts of two distinct inputs. However, in OAEP (and any other AONT) instead of encryption, the secret data is masked with pseudo-random bits which were generated using a random secret key K. Therefore in order to use the method of [33] for analyzing the security of our scheme, we use a generalized version of the OAEP algorithm which consists of these steps:

1. The secret data is masked with the pseudo-random bits. This *masking* procedure can be defined in various ways, for example, OAEP masks the secret data by XORing it with the output of the random oracle G ($C \leftarrow S \oplus G(K)$).
2. The result of the last step is hashed using the random oracle H, and its result is XORed with the random key K, this way K is also masked with a pseudo-random value. In this step, H is a hash function and its output length is equal to the size of random key K.

G_0	G_1
PROCEDURE $Initialize$	PROCEDURE $Initialize$
$K \xleftarrow{\$} \{0,1\}^{\lambda},\ b \xleftarrow{\$} \{0,1\},\ K = K'$	$K, K' \xleftarrow{\$} \{0,1\}^{\lambda},\ b \xleftarrow{\$} \{0,1\}$
PROCEDURE $Share(S_0, S_1)$	PROCEDURE $Share(S_0, S_1)$
$C \xleftarrow{\$} Mask_K(S_b)$	$C \xleftarrow{\$} Mask_K(S_b)$
$P \xleftarrow{\$} K' \oplus H(C)$	$P \xleftarrow{\$} K' \oplus H(C)$
$\mathbf{V} \leftarrow IDA(C\|\|P)$	$\mathbf{V} \leftarrow IDA(C\|\|P)$
PROCEDURE $Corrupt(i)$	PROCEDURE $Corrupt(i)$
RETURN $V[i]$	RETURN $V[i]$
PROCEDURE $Finalize(d)$	PROCEDURE $Finalize(d)$
RETURN $d = b$	RETURN $d = b$

Fig. 10. Games G_0 and G_1.

Now, using the generalized OAEP and the notations of Table 1, the *Share* and *Recover* procedures of a secret sharing scheme are defined in Fig. 9.

In order to analyze these types of secret sharing schemes, we must reduce their security to the indistinguishability of the generalized OAEP, To do so, we present the following theorem and use an approach similar to [33] and [35] to prove it.

Theorem 1. *Assume that A is a privacy adversary against the secret sharing scheme Π and the functions H and $Mask$ are random oracles. Then there exists an adversary B attacking the indistinguishability of the generalized OAEP algorithm Ω such that:*

$$Adv_{\Pi}^{Priv}(A) = Adv_{\Omega}^{Ind}(B)$$

Proof. To prove this theorem, we define the games G_0 and G_1 as depicted in Fig. 10.

The advantage of adversary A against the secret sharing scheme Π is:

$$Adv_{\Pi}^{Priv} = 2.Pr[G_0^A] - 1 \tag{5}$$

Where $Pr[G_0^A]$ is the probability of correctly guessing b in the Finalize procedure of G_0 by A.

The only distinction between the games G_0 and G_1 is that in G_1 the random key K used to mask the secret data in the $Mask$ function (the oracle G in OAEP) is different from the key K' used in $P \xleftarrow{\$} K' \oplus H(C)$.

We claim that:

$$Pr[G_0^A] = Pr[G_1^A] \tag{6}$$

To justify this claim, we assume that the adversary has access to $t-1$ shares and the size of each share is at least λ bits. Additionally, we know that the outputs of $Mask$ and H functions are pseudo-random, and $C \,\|\, P$ is shared using an IDA.

In the worst case scenario (when the size of each share is precisely λ bits) the maximum amount of information that an adversary can learn constitutes of one of the following cases [35]:

1. The adversary learns no information about P but he can learn all of C, hence from the adversary's perspective $H(C) = K \oplus P = K' \oplus P'$ where $P \neq P'$ and $K \neq K'$ are random and unknown strings.
2. The adversary learns all of P but does not know at least λ bits of C, and because he is a computationally bounded adversary, based on Lemma 1 it is computationally infeasible for him to learn

$H(C)$. Therefore from his perspective $P = K \oplus H(C) = K' \oplus H(C)'$ where $H(C) \neq H(C)'$ and $K \neq K'$ are random and unknown strings.

In both cases, the adversary knows one of either C or P and does not know anything about the other, therefore he cannot learn any thing about the random key K. From his perspective, the value that he knows is always equal to the result of XORing two unknown and random strings, thus changing one of these values (in this case K) does not affect his winning chance. This means that a computationally bounded adversary cannot distinguish between K and any other random string (K'), hence (6) holds true [35].

Next, we construct the adversary B attacking the indistinguishability of the generalized OAEP algorithm Ω in such a way that his advantage will be:

$$Adv_{\Omega}^{Ind}(B) = 2.Pr[G_1^A] - 1 \tag{7}$$

Adversary B can achieve this advantage by running A as a subroutine. The steps of such a procedure are as follows:

- Adversary B generates the random key $K' \xleftarrow{\$} \{0,1\}^{\lambda}$.
- As his challenger, adversary B runs A, then A executes the procedure $Share(S_0, S_1)$.
- Adversary B sends S_0 and S_1 to his own challenger and receives $C \xleftarrow{\$} Mask_K(S_b)$ (K is the key generated by the challenger of B).
- Adversary B computes $P = H(C) \oplus K'$ and generates the shares using an IDA algorithm $\mathbf{V} \leftarrow IDA(C \parallel P)$.
- When A executes the $Corrupt(i)$ procedure, B sends $V[i]$ in response.
- Finally, when A outputs d as his guess, B passes it to his challenger as his own guess. Thus, the advantage of B is $2.Pr[d = b] - 1$.

Therefore by combining (5), (6) and (7) we have:

$$Adv_{\Pi}^{Priv}(A) = Adv_{\Omega}^{Ind}(B)$$

Theorem 1 proves that if an adversary such as A can break the indistinguishability of the proposed secret sharing scheme with a significant advantage while having utmost $t-1$ shares, then there is also an adversary such as B that can break the indistinguishability of the generalized OAEP algorithm with the same advantage without having the key K.

However, in the generalized OAEP, the $Mask$ and H functions are random oracles, and one of the most important properties of such oracles is that their output should be *pseudo-random*. Now, if a *computationally bounded* adversary such as B can distinguish between the masked result of two secret values with a significant advantage without having access to the masking key, he practically violated the pseudo-random property of these random oracles.

In conclusion, the proposed secret image sharing scheme is secure in terms of indistinguishability if OAEP uses a random oracle such that an adversary with limited computational power cannot distinguish between *its pseudo-random output* and *an entirely random string*. In other words, as long as the pseudo-random property of the random oracles of OAEP is guaranteed, the security of the proposed scheme can also be ensured.

6. Experimental results

In this section, we will present some experimental results to demonstrate the effectiveness of our "secret image sharing with steganography" method.

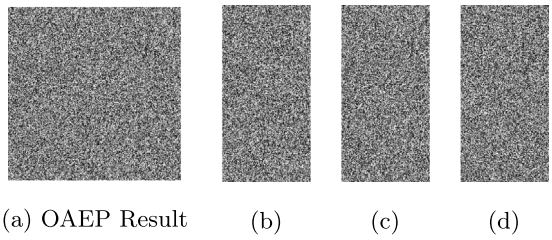


Fig. 11. Experimental results: (a) OAEP transformed image; (c)–(e) generated shadows.

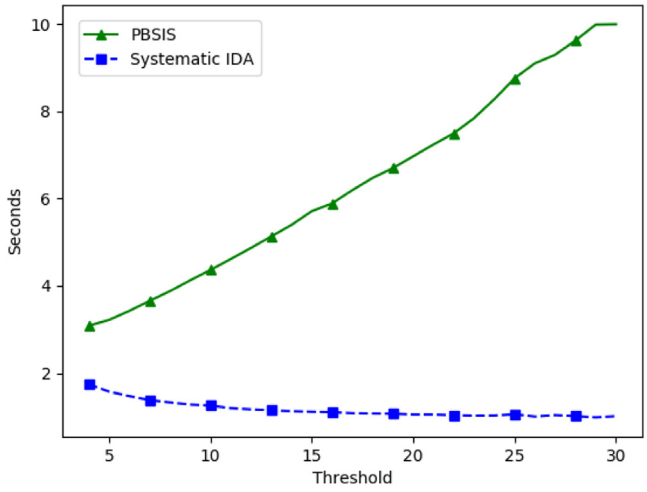


Fig. 12. Execution time for share generation.

6.1. Generated shadows

The results of our secret image sharing scheme are depicted in Fig. 11. In this experiment, a (2, 3)-threshold secret image sharing scheme is applied to a 512 × 512 secret image (1a). For implementing the OAEP transform, we used SHA-256 as the H function and MGF1 as the G function, and λ is 256 bits, The results of this transform are depicted in Fig. 11a, and generated shadows are illustrated in Figs. 11b to 11d.

As it is evident from Fig. 11, the size of each shadow is roughly $\frac{1}{t}$ of the secret image's. Also, shadows are noise-like pictures and they do not have any similarities to the secret image.

6.2. Performance

In this part, we are going to compare the performance of the systematic Reed–Solomon based IDA [26,29] of our scheme with the modified Shamir's scheme of PBSIS methods [4–13].

For this experiment, both the PBSIS scheme and the systematic IDA were implemented in Python 2.7, and a computer running Linux with 2.6 GHz CPU and 8 GB RAM was used to execute the code. Also, in both methods, computations were performed in $GF(2^8)$.

In this experiment, n (the number of participants) is fixed at 30 and the threshold value t is increased from 4 to 30. The average "Share Generation" and "Image Reconstruction" execution time for fifteen 256 × 256 pictures from USC-SIPI Image Database[1] are illustrated in Figs. 12 and 13, respectively.

As it is evident, the systematic IDA is much faster than PBSIS schemes. There are two main reasons for this performance superiority:

[1] http://sipi.usc.edu/database/.

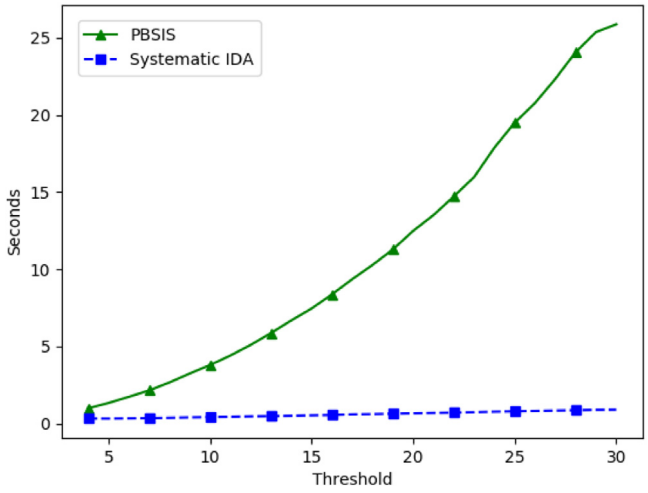


Fig. 13. Execution time for image reconstruction.

1. The computational complexity of the Lagrange interpolation algorithm is $O(t\ log^2 t)$ [19], and Shamir's scheme has to execute this algorithm every time it wants to reconstruct a polynomial. In a secret image sharing scheme, there is a different polynomial for each t pixels, therefore this algorithm has to be executed $\frac{|SecretSize|}{t}$ times during the secret reconstruction process. On the other hand, in a systematic IDA, the generator matrix is constant for all of the data vectors, thus the process of calculating the inverse matrix with $O(t^3)$ complexity is performed only once, and then each of the share vectors are only multiplied by this inverted matrix and the multiplication procedure is a lot faster than the Lagrange interpolation.
2. In a systematic IDA, if any of the first t shares are available during reconstruction, their corresponding block in the initial data is also available without any calculation.

6.3. Steganalysis

We use the SPAM steganalysis method [36] to measure the resistance of our proposed scheme against steganalysis. For this experiment, we used 3000 pictures from the BOSS Image Database.[2]

For LSB substitution, LSBMR [37], FEMD and Edge-FEMD (proposed) steganographic methods, some random messages which created 1.0 bpp capacity were embedded in 1500 randomly selected pictures. Then, half of these 3000 images were used to train the SVM classifier and the other half were used for the test. In this experiment we used a SVM classifier with RBF kernel, so it was necessary to choose the best C and γ values for this kernel before the test, therefore, we used cross-validation to select the best C and γ from the following values:

$$C = 0.001, 0.01, 0.1, \ldots, 10000$$
$$\gamma = 0.0001, 0.001, 0.01, \ldots, 100$$

The ROC curve of this experiment is depicted in Fig. 14.

As it is evident from Fig. 14, our method shows better resistance against steganalysis. Also, it is obvious that the SPAM method can detect LSB substitution with approximately 100 percent accuracy. This is particularly important because most of the previous schemes [6,8,9,12,13,18,20,21] are using LSB substitution for their steganography phase.

[2] http://agents.fel.cvut.cz/boss/.

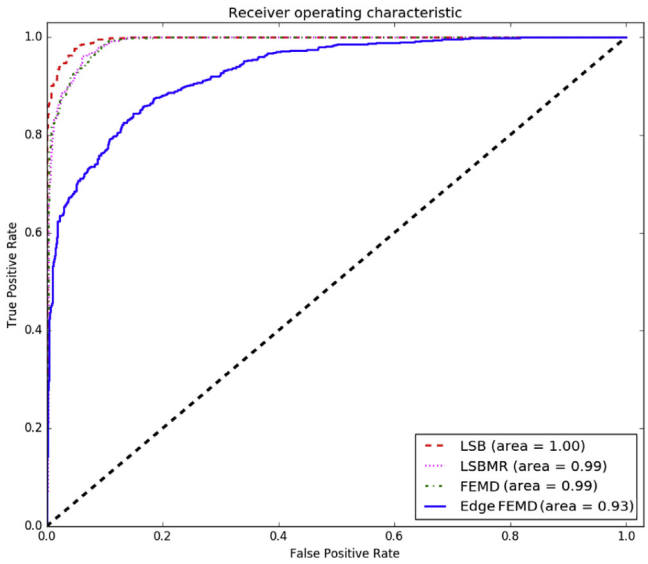


Fig. 14. ROC curve for 1.0 bpp capacity.

Table 2
Visual quality comparison.

Scheme	PSNR (dB)	Scheme	PSNR (dB)
Lin & Tsai [20]	39.21	Yang et al. [21]	40.59
Chang et al. [6]	40.97	Wu et al. [8]	43.54
Ulutas et al. [10]	48.62	Eslami et al. [9]	48.13
Khosravi et al. [11]	43.11	Li et al. [13]	48.14
Proposed [2,30]	**49.74**	**Proposed [0,30]**	**51.42**

6.4. Quality comparison with the previous schemes

In this section, we are going to compare the stego-image quality of some of the previous "secret image sharing with steganography" schemes with the proposed one. In this experiment, a 256 × 256 picture (1a) was used as the secret image of a (2, 4) threshold secret image sharing scheme, and the resulting shadows were embedded in 15 different 512 × 512 grayscale cover images. Table 2 reports the average PSNR values for our proposed method and also previous schemes.

However, in the PBSIS schemes, the size of shadows is $\frac{1}{t}$ of the secret image's, thus increasing t reduces the size of shadows, and smaller shadows result in better visual quality for stego-images. Therefore, in the next experiment, we are going to repeat the last experiment for some of the most important previous schemes [6,8–10,21] using different values for t. In this experiment, the stego-image quality of these schemes for $t = 2, 3, \ldots, 10$ and $n = 10$ have been compared using the PSNR and SSIM criteria, and the results are depicted in Figs. 15 and 16, respectively.

In these experiments, we used the ranges of [0,30] and [2,30] for threshold values. The first range results in the best possible stego-image quality. Although, the visual quality of the second range is less than the first one, it has better resistance against steganalysis techniques. Hence, based on the scheme's requirements, one of these ranges can be adopted.

As it is evident from Table 2 and Figs. 15 and 16, for both threshold ranges, our proposed scheme has better visual quality compared to the previous methods.

7. Conclusion

In this paper, we first demonstrated the security weaknesses of the widely used polynomial-based secret image sharing (PBSIS) scheme. Then we employed optimal asymmetric encryption padding (OAEP) and information dispersal algorithms (IDA) to propose a novel computationally secure secret image sharing scheme. The proposed method

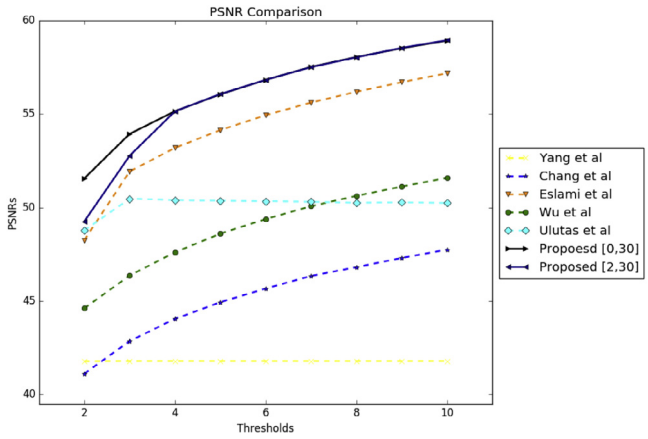


Fig. 15. Comparing the visual quality of stego-images using PSNR criterion.

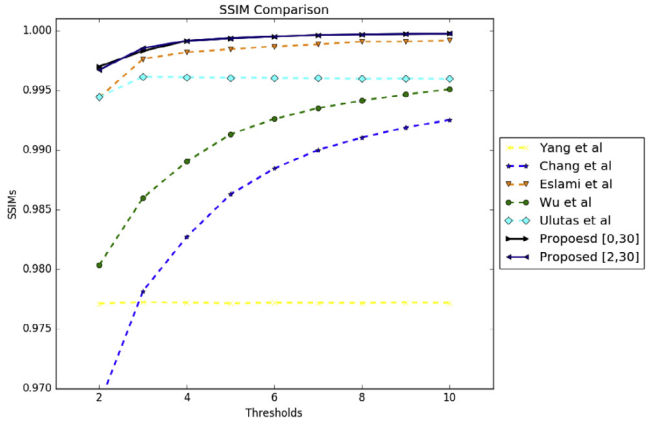


Fig. 16. Comparing the visual quality of stego-images using SSIM criterion.

enjoys the benefits of computational security and small shadow size. Additionally, the experimental results demonstrate its computational efficiency and confirm that the performance of the proposed scheme is higher than PBSIS based methods.

Finally, we proposed a steganography method to embed the shadows in the edges of cover images. Some of the most important features of our proposed steganography scheme are higher stego-image quality and better resistance against steganalysis algorithms.

References

[1] G.R. Blakley, Safeguarding cryptographic keys, in: Proc. of the National Computer Conference1979, Vol. 48, 1979, pp. 313–317.
[2] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[3] M. Naor, A. Shamir, Visual cryptography, in: Workshop on the Theory and Application of of Cryptographic Techniques, Springer, 1994, pp. 1–12.
[4] C.-C. Thien, J.-C. Lin, Secret image sharing, Comput. Graph. 26 (5) (2002) 765–770.
[5] Y.-S. Wu, C.-C. Thien, J.-C. Lin, Sharing and hiding secret images with size constraint, Pattern Recognit. 37 (7) (2004) 1377–1385.
[6] C.-C. Chang, Y.-P. Hsieh, C.-H. Lin, Sharing secrets in stego images with authentication, Pattern Recognit. 41 (10) (2008) 3130–3137.
[7] Y.-Y. Lin, R.-Z. Wang, Scalable secret image sharing with smaller shadow images, IEEE Signal Process. Lett. 17 (3) (2010) 316–319.
[8] C.-C. Wu, S.-J. Kao, M.-S. Hwang, A high quality image sharing with steganography and adaptive authentication scheme, J. Syst. Softw. 84 (12) (2011) 2196–2207.
[9] Z. Eslami, J.Z. Ahmadabadi, Secret image sharing with authentication-chaining and dynamic embedding, J. Syst. Softw. 84 (5) (2011) 803–809.
[10] G. Ulutas, M. Ulutas, V.V. Nabiyev, Secret image sharing scheme with adaptive authentication strength, Pattern Recognit. Lett. 34 (3) (2013) 283–291.
[11] M.J. Khosravi, A.R. Naghsh-Nilchi, A novel joint secret image sharing and robust steganography method using wavelet, Multimedia Syst 20 (2) (2014) 215–226.
[12] J. He, W. Lan, S. Tang, A secure image sharing scheme with high quality stego-images based on steganography, Multimedia Tools Appl. (2016) 1–22.
[13] P. Li, Q. Kong, Y. Ma, Image secret sharing and hiding with authentication based on psnr estimation, J. Inf. Hiding Multimedia Signal Process. 5 (2) (2014) 353–366.
[14] G. Ulutas, M. Ulutas, V. Nabiyev, Distortion free geometry based secret image sharing, Procedia Comput. Sci. 3 (2011) 721–726.
[15] C. Asmuth, J. Bloom, A modular approach to key safeguarding, IEEE Trans. Inf. Theory 29 (2) (1983) 208–210.
[16] M. Ulutas, V.V. Nabiyev, G. Ulutas, A new secret image sharing technique based on asmuth bloom's scheme, in: Application of Information and Communication Technologies, 2009. AICT 2009. International Conference on, IEEE, 2009, pp. 1–5.
[17] G. Alvarez, A.H. Encinas, L.H. Encinas, A.M. del Rey, A secure scheme to share secret color images, Comput. Phys. Commun. 173 (1) (2005) 9–16.
[18] Z. Eslami, S. Razzaghi, J.Z. Ahmadabadi, Secret image sharing based on cellular automata and steganography, Pattern Recognit. 43 (1) (2010) 397–404.
[19] J. Zarepour-Ahmadabadi, M.S. Ahmadabadi, A. Latif, An adaptive secret image sharing with a new bitwise steganographic property, Inform. Sci. 369 (2016) 467–480.
[20] C.-C. Lin, W.-H. Tsai, Secret image sharing with steganography and authentication, J. Syst. Software 73 (3) (2004) 405–414.
[21] C.-N. Yang, T.-S. Chen, K.H. Yu, C.-C. Wang, Improvements of image sharing with steganography and authentication, J. Syst. Software 80 (7) (2007) 1070–1076.
[22] P.-Y. Lin, J.-S. Lee, C.-C. Chang, Distortion-free secret image sharing mechanism using modulus operator, Pattern Recognit. 42 (5) (2009) 886–895.
[23] A.M. Ahmadian, M. Amirmazlaghani, Computationally secure secret image sharing, in: Electrical Engineering (ICEE), 2017 Iranian Conference on, IEEE, 2017, pp. 2217–2222.
[24] R.L. Rivest, All-or-nothing encryption and the package transform, in: International Workshop on Fast Software Encryption, Springer, 1997, pp. 210–218.
[25] M.O. Rabin, Efficient dispersal of information for security, load balancing, and fault tolerance, J. ACM 36 (2) (1989) 335–348.
[26] J.K. Resch, J.S. Plank, Aont-rs: blending security and performance in dispersed storage systems, in: Proceedings of the 9th USENIX Conference on File and Stroage Technologies, in: FAST'11, USENIX Association, 2011, pp. 14–14.
[27] M. Bellare, P. Rogaway, Optimal asymmetric encryption, in: Advances in Cryptology EUROCRYPT'94, Springer, 1995, pp. 92–111.
[28] T.D. Kieu, C.-C. Chang, A steganographic scheme by fully exploiting modification directions, Expert Syst. Appl. 38 (8) (2011) 10648–10657.
[29] J.S. Plank, et al., A tutorial on reed-solomon coding for fault-tolerance in raid-like systems, Softw. Pract. Exp. 27 (9) (1997) 995–1012.
[30] A. Jolfaei, X.-W. Wu, V. Muthukkumarasamy, On the security of permutation-only image encryption schemes, IEEE Trans. Inf. Forensics Secur. 11 (2) (2016) 235–246.
[31] J.S. Plank, Y. Ding, Note: Correction to the 1997 tutorial on reed–solomon coding, Softw. - Pract. Exp. 35 (2) (2005) 189–194.
[32] W. Luo, F. Huang, J. Huang, Edge adaptive image steganography based on lsb matching revisited, IEEE Trans. Inf. Forensics Secur. 5 (2) (2010) 201–214.
[33] P. Rogaway, M. Bellare, Robust computational secret sharing and a unified account of classical secret-sharing goals, in: Proceedings of the 14th ACM conference on Computer and communications security, ACM, 2007, pp. 172–184.
[34] R.W. Lauritsen, Backups with Computational Secret Sharing (Master's thesis), University of Aarhus, 2008.
[35] L. Chen, T.M. Laing, K.M. Martin, Revisiting and extending the AONT-RS scheme: A robust computationally secure secret sharing scheme, in: M. Joye, A. Nitaj (Eds.), Progress in Cryptology - AFRICACRYPT 2017: 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings, Springer International Publishing, 2017, pp. 40–57.
[36] T. Pevny, P. Bas, J. Fridrich, Steganalysis by subtractive pixel adjacency matrix, IEEE Trans. Inf. Forensics Secur. 5 (2) (2010) 215–224.
[37] J. Mielikainen, Lsb matching revisited, IEEE Signal Process. Lett. 13 (5) (2006) 285–287.