Reversible Data Hiding in Paillier Cryptosystem

Hao-Tian Wu, Yiu-ming Cheung, Jiwu Huang

PII: S1047-3203(16)30180-8
DOI: http://dx.doi.org/10.1016/j.jvcir.2016.08.021
Reference: YJVCI 1850

To appear in: *J. Vis. Commun. Image R.*

Please cite this article as: H-T. Wu, Y-m. Cheung, J. Huang, Reversible Data Hiding in Paillier Cryptosystem, *J. Vis. Commun. Image R.* (2016), doi: http://dx.doi.org/10.1016/j.jvcir.2016.08.021

Reversible Data Hiding in Paillier Cryptosystem

Hao-Tian Wu[a], Yiu-ming Cheung[b], Jiwu Huang[c]

[a] *South China University of Technology, School of Computer Science and Technology, Guangzhou, GD 510006, P. R. China.*
E-mail: htwu1981@gmail.com
[b] *Hong Kong Baptist University, Department of Computer Science, Hong Kong.*
E-mail: ymc@comp.hkbu.edu.hk
[c] *Shenzhen University, College of Information Engineering and Shenzhen Key Lab of Media Security, Shenzhen, P. R. China.*
E-mail: jwhuang@szu.edu.cn

Abstract

In this paper, reversible image data hiding in the Paillier cryptosystem is investigated. To transmit additional data in homomorphic encryption domain, two algorithms are proposed for different application scenarios. By exploiting the additive homomorphism, high-capacity data hiding can be accomplished with the first algorithm by conducting value expansion on the encrypted pixel values. But the hidden data can only be extracted after image decryption (i.e., in plain-text domain). With the second algorithm, both of data embedding and extraction can be performed in the encryption domain by exploiting the self-blinding property, while the corresponding plain-text values are unchanged. Compared with the reversible data hiding algorithms designed for encryption with a cipher stream, the proposed ones are more applicable in cloud computing without degrading the security level. Thus the additional data can be losslessly transmitted in the different applications of Paillier cryptosystem.

Keywords: Paillier cryptosystem, value expansion, self-blinding, reversible data hiding, homomorphic encryption.

1. Introduction

In the literature, reversible data hiding (RDH) has been proposed and quite a lot RDH methods have been developed to transmit additional data (e.g. [1]-[7]).

Recently, RDH in the encrypted images has been proposed for authentication and content annotation, such as in [8]-[23]. Although the image entropy is usually increased after encryption, the correlations between the neighboring pixels still exist. By exploiting the correlations, additional data can be hidden into encrypted images and correctly extracted when needed. Meanwhile, the encrypted image before data hiding can be recovered after data extraction, or the original plain-text image can be obtained after decryption and data extraction.

To overcome the drawback that the hidden data can only be extracted after image decryption (e.g., in [10],[11],[12]), the separable RDH in the encrypted images has been achieved (e.g., [8],[9]) by accommodating the additional data in the encryption domain. To reduce the average bit error rate, the complexity of image blocks is evaluated based on the absolute mean difference of multiple neighboring pixels in [13]. In [14], a public key modulation mechanism is adopted so that there is no need to assess to the secret encryption key, while public key cryptography is also used in [15] and [16]. By elaborately selecting the pixels used for data hiding, the visual quality of the decrypted image can be improved by using the method in [17]. Moreover, higher embedding capacity is achieved in [18] by adopting prediction, while the distributed source coding is adopted in [19] to compress the most significant bits in an encrypted image. In addition, some hybrid algorithms (e.g., [20] and [21]) are proposed to enlarge the embedding capacity by reserving room before encryption.

However, the images are all encrypted with a stream cipher in [8]-[21] so that they can hardly be directly processed in the encryption domain. Besides, the security level will be degraded when the image content is revealed in cloud computing. Since the encrypted images cannot be securely processed after decryption, the applications of these RDH algorithms are somehow limited. This drawback can be overcome by conducting RDH in the homomorphic encryption domain (e.g., [22], [23]). To directly process the encrypted data, the special encryption functions called "privacy homomorphism" [24] need to be found. That is, an encrypted result can be generated after processing a cipher-text, which after decryption matches the desired plain-text result.

In the past decades, significant progresses have been made in homomorphic encryption (e.g., [25]-[29]) so as to directly process data in encryption domain. Since there is no need to decrypt the cipher-texts before processing them, the user privacy and data integrity are inherently protected. This property is very useful in modern communication systems, such as in cloud computing (e.g., [30]) and secure voting systems (e.g., [31]). Although it is a challenge to implement complex computations in the fully homomorphic domain (e.g., [26, 27]), quite a few signal processing techniques have been developed (e.g., [32]-[37]). For instance, the discrete fourier transform [33], the discrete cosine transform [34] and the discrete wavelet transform [35] can be applied on the encrypted image, respectively. Reversible data hiding can also be conducted in the homomorphic encryption domain to transmit additional data. Since the encrypted image can be directly processed, the user privacy and confidentiality can be protected. Therefore, conducting RDH in the homomorphic encryption domain can enhance its applicability in cloud computing and other related applications. However, this topic has not received much attention in the community.

In this paper, reversible data hiding in the homomorphic encryption domain is investigated. As shown in Figure 1, two categories of application scenarios are taken into consideration after encrypting a plain-text image and conducting the RDH on encrypted image. Depending on requirements in specific applications, data extraction may be performed before image decryption or after it. Different from the encryption using only simple modular arithmetic as adopted in [22] and [23], two RDH algorithms are proposed based on the additive homomorphic Paillier cryptosystem [25]. Compared with the image RDH algorithms which are designed for encryption with a cipher stream, the proposed ones are more applicable in cloud environment without degrading the security level. Consequently, additional information can be losslessly transmitted in the different applications of the Paillier cryptosystem. In the experiments, the two proposed algorithms are implemented on the USC-SIPI test images in [38].

The rest of this paper is organized as follows. In the following section, the potential applications of RDH in homomorphic encryption domain are discussed.

In Section 3, a high-capacity image RDH algorithm based on Paillier cryptosystem is presented for the scenario of data extraction after image decryption. To perform both of data embedding and extraction in the homomorphic encryption domain, another algorithm is proposed in Section 4. The experimental results are given and analyzed in Section 5 to demonstrate the efficacy of the proposed algorithms. Finally, we draw a conclusion in Section 6.

2. Potential applications of RDH in homomorphic encryption domain

In this section, the potential applications of reversible image data hiding in homomorphic encryption domain are discussed. As shown in Figure 1, after encrypting an original plain-text image, the encrypted image may be directly processed in the homomorphic encryption domain (e.g., [33]-[37]). Besides the aforementioned operations that can be applied, reversible data hiding can also be conducted to generate the encrypted image with hidden data. By using reversible data hiding technique, additional data can be losslessly transmitted between two parties, such as the web user and service provider. Depending on the domain of data extraction, two cases are taken into consideration and the potential applications are discussed as follows.

2.1. Data extraction in encryption domain

If data extraction is required to be performed before image decryption, then the hidden data should be extracted in the homomorphic encryption domain. For instance, the image owner may hide the additional data into the encrypted image to be uploaded. At the web server, the original encrypted image can be blindly recovered after extracting the hidden data. That means both data hiding and extraction are conducted in the homomorphic encryption domain. To keep the hidden data from being changed, the encrypted image with hidden data should not undergo other processing. Another application scenario is for the web administrator or service provider to hide a piece of message into the encrypted image. At the recipient side, the hidden message should be directly

extracted while recovering the encrypted image before data hiding. To keep the hidden data from being illegally extracted, data extraction can be protected with a secret key. Compared with [8], the advantage of RDH in homomorphic encryption domain is that the encrypted image can be further processed after being recovered. Without degrading the security level, quite a few homomorphic operations can be applied to the recovered image. The corresponding plain-text image can be further obtained with the decryption key.

2.2. Data extraction after image decryption

In the second category of applications, the decryption key must be known to perform data extraction. From the decrypted values, the hidden data can be extracted while the processed plain-text image can be obtained. For instance, the service provider may embed some additional data into the encrypted image after processing it. When the encrypted image with hidden data is received, decryption is firstly executed so that both data extraction and image recovery can be carried out in plain-text domain. It can be seen that the user privacy and confidentiality can be protected with the decryption key, which may be the private key of public key cryptosystem. The hidden data cannot be extracted without the decryption key, similar to the cases in [10]-[14].

3. Reversible data hiding by value expansion in Paillier cryptosystem

In this section, a Paillier cryptosystem based RDH algorithm is presented for the scenario where image decryption is required for data extraction.

3.1. Additive homomorphism in Paillier cryptosystem

The Paillier cryptosystem [25] is an additive homomorphic encryption method, which is based on the decisional composite residuosity problem. After encrypting a plain-text value m, a cipher value $\mathbf{e_k}[m]$ can be generated, which is represented by a big integer. Here $\mathbf{e_k}[\cdot]$ represents the encryption operation with a public encryption key $\mathbf{k}$. After encrypting a plain-text digital image, a string of big integers are generated and each of them represents an encrypted pixel value.

As a public-key homomorphic cryptography, the encryption key $\mathbf{k}$ is to be publicly known. That means a plain-text value m_1 can be encrypted by the service provider to generate an encrypted value $\mathbf{e_k}(m_1)$. To add m_1 to another plain-text value m_2 in the encryption domain, the following operation can be performed on their encrypted values $\mathbf{e_k}(m_1)$ and $\mathbf{e_k}(m_2)$ by

$$\mathbf{e_k}(m') = (\mathbf{e_k}(m_1) \cdot \mathbf{e_k}(m_2)) \ mod \ N^2, \tag{1}$$

where N is a big integer generated in the Paillier cryptosystem and included in the public key $\mathbf{k}$. The corresponding plain-text value $(m_1 + m_2) \ mod \ N$ can be obtained by decrypting the cipher value $\mathbf{e_k}(m')$, i.e.

$$(m_1 + m_2) \ mod \ N = \mathbf{d_r}[\mathbf{e_k}(m')] \tag{2}$$

where $\mathbf{d_r}[\cdot]$ represents the decryption operation with the private key $\mathbf{r}$.

3.2. Data embedding by value expansion

Based on the additive homomorphism, a data embedding method named value expansion is developed as follows. To embed a bit value $b \in \{0, 1\}$ into $\mathbf{e_k}(i)$, which is the encrypted value of a pixel value i, another cipher value $\mathbf{e_k}(i')$ is generated by

$$\mathbf{e_k}(i') = \mathbf{e_k}(2i + b). \tag{3}$$

To accomplish this, the following two operations need to be sequentially conducted, i.e.,

$$\mathbf{e_k}(i_d) = (\mathbf{e_k}(i) \cdot \mathbf{e_k}(i)) \ mod \ n^2 \tag{4}$$

and

$$\mathbf{e_k}(i') = \begin{cases} (\mathbf{e_k}(i_d) \cdot \mathbf{e_k}(1)) \ mod \ n^2 & \text{if } b = 1 \\ \mathbf{e_k}(i_d) & \text{if } b = 0 \end{cases}, \tag{5}$$

where $\mathbf{e_k}(1)$ is generated by encrypting the integer 1 with $\mathbf{k}$. For every encrypted pixel value, Eq. (4) and Eq. (5) are sequentially executed so that the same number of bit values as the pixels can be hidden into an encrypted image.

To recover the original pixel value, $\mathbf{e_k}(i')$ needs to be firstly decrypted. From the additive homomorphism, we know that $\mathbf{d_r}[\mathbf{e_k}(i_d)] = \ 2i \ mod \ N$ therefore

$\mathbf{d_r}[\mathbf{e_k}(i')] = (2i + b)\ mod\ N$. As $i \in [0, 255]$ for a grey-level pixel value, we know that $2i + b \in [0, 511]$. When N is a big integer represented with 1024 bits, $(2i + b)\ mod\ N = 2i + b$ so that the original values of i and b can be obtained by

$$i = \lfloor \frac{\mathbf{d_r}[\mathbf{e_k}(i')]}{2} \rfloor, \tag{6}$$

where $\lfloor \cdot \rfloor$ the represents the floor function, and

$$b = \mathbf{d_r}[\mathbf{e_k}(i')] - 2i. \tag{7}$$

The process of value expansion in homomorphic encryption domain can be depicted in Figure 2. For a pixel value i, it will be mapped to $2i + b$ to embed a bit value b. To recover the original value, the encrypted result needs to be decrypted so that the original values of i and b can be obtained by Eq. (6) and Eq. (7), respectively.

3.3. Proposed Algorithm

By exploiting the redundancy in value representation of the Paillier cryptosystem, reversibility of data embedding can be achieved. For an integer within the interval of $[0, 255]$ and another integer in the interval of $[0, 511]$, their encrypted values are represented with the same bits. Therefore, the bit values embedded by applying Eq. (4) and Eq. (5) can be correctly extracted. For the two pixels i and j with the same value, the corresponding cipher values $\mathbf{e_k}(i)$ and $\mathbf{e_k}(j)$ may be different due to the randomness in encryption. So change of a histogram bin i as shown in Figure 2 cannot be correctly depicted in the encryption domain, but in the plain-text domain.

To further exploit redundancy in data representation in the Paillier cryptosystem, value expansion can be iteratively performed on the cipher value. That is, for the resulting $\mathbf{e_k}(i')$ after embedding a bit value b (i.e., $\mathbf{d_r}[\mathbf{e_k}(i')] = 2i + b$), another bit value c can be embedded by applying Eq. (4) and Eq. (5) to generate another cipher value $\mathbf{e_k}(i'')$ so that $\mathbf{d_r}[\mathbf{e_k}(i'')] = 2 \times (2i + b) + c$. Given that

$$[2 \times (2i + b) + c]\ mod\ N = 2 \times (2i + b) + c, \tag{8}$$

both b and c can be correctly extracted from $\mathbf{e_k}(i'')$. After expanding every encrypted pixel value for multiple times, the range of the corresponding plain-text values will be expanded from $[0, 255]$ to $[0, 511]$, then $[0, 1023]$, and so on. The operations of value expansion can be iterated for multiple times until the condition in Eq. (8) is broken. Since the publicly known N is a big integer, multiple bits can be reversibly hidden into an encrypted pixel value.

The procedure of data hiding consists of the following steps.

1) Given an image encrypted in Paillier cryptosystem, a portion of the encrypted pixels are reserved to hide the side information such as the amount of bit values to be hidden. For instance, the first 10 pixels are used to hide the information indicating the number of bits to be hidden in each encrypted pixel value, and the next 16 pixels are used to record the number of encrypted pixel values used for data hiding.

2) Given a string of binary values to be hidden, calculate the number of encrypted pixel values needed with a given data hiding rate. Then hide the side information into the reserved pixels by applying Eq. (4) and Eq. (5) at the rate of one bit per pixel (bpp).

3) Hide the string of binary values into the chosen encrypted pixels by repeating Eq. (4) and Eq. (5) when needed, so that the encrypted image with hidden data is generated.

The procedure of recovering the plain-text image includes the following steps.

1) Firstly, decrypt all big integers in an encrypted image.

2) Extract the side information from the reserved pixels and recover their plain-text values by applying Eq. (6) and Eq. (7).

3) With the extracted side information, extract the bit values from those pixels used for data hiding. Then recover their plain-text values by repeating Eq. (6) and Eq. (7) when needed so that the plain-text image can be obtained.

4. Achieving data extraction in homomorphic encryption domain

Different from the RDH algorithm proposed in Section 3, the algorithm to be presented in this section is applicable to the scenario where data extraction is required before image decryption. By exploiting the self-blinding property of the Paillier cryptosystem, both of data embedding and extraction can be conducted in the homomorphic encryption domain.

4.1. Property of self-blinding in Paillier cryptosystem

Thanks to the randomness in data encryption, a plain-text value may be encrypted into multiple possible cipher-texts in Paillier cryptosystem. That is, the cipher value of a plain-text value m is not unique because several different cipher values may be decrypted to m. Moreover, the self-blinding property in Paillier cryptosystem indicates that

$$\mathbf{d_r}[\mathbf{e_k}(m)p^N] = m \ mod \ N \tag{9}$$

where p is a random element in $\mathbb{Z}_N^*$, which consists of all the integers relatively prime with N. The self-blinding property implies that every cipher value can be changed without affecting the corresponding plain-text value.

4.2. Data embedding with the self-blinding property

By exploiting the randomness in Paillier cryptosystem, both of data embedding and extraction can be conducted in the encryption domain. To embed a bit value b into a cipher value $\mathbf{e_k}(i)$, the following equation should hold:

$$\mathbf{e_k}(i) \ mod \ 2 \ = \ b. \tag{10}$$

If $\mathbf{e_k}(i) \ mod \ 2 \neq b$, $\mathbf{e_k}(i)$ should be changed to another cipher value so as to meet the requirement in Eq. (10). When needed, an integer p relatively prime with N needs to be chosen. Then $\mathbf{e_k}(i)$ is multiplied by p^N to obtain $\mathbf{e_k}(i)p^N$. In the implementation, $\mathbf{e_k}(i)$ is multiplied by $p^N \ mod \ N^2$ instead of p^N because

$$[\mathbf{e_k}(i)(p^N \ mod \ N^2)] \ mod \ N^2 = \mathbf{e_k}(i)p^N \ mod \ N^2. \tag{11}$$

From Eq. (9), we know $\mathbf{d_r}[\mathbf{e_k}(i)p^N] = i \ mod \ N$. It should be noted that the operation in Eq. (11) can be iterated until the condition in Eq. (10) holds. Even when the condition cannot be met with p, another integer q that is relatively prime with N can be used instead of p in Eq. (11) to hide a bit value b. Since the encrypted pixel value may be odd or even due to the randomness in encryption, the modulo 2 operation is adopted in Eq. (10) for data embedding. Thanks to the self-blinding property, the corresponding plain-text value is unchanged by Eq. (11). So there is no need to recover the original cipher value. The only operation to extract the hidden bit value is

$$b' \ = \ \mathbf{e_k}(i) \ mod \ 2, \tag{12}$$

where b' denotes the extracted bit value.

4.3. Proposed Algorithm

For simplicity of illustration, the module 2 operation is adopted in Eq. (10) and Eq. (12) so that one bit can be hidden per encrypted pixel. To increase the hiding rate, module 4 or module 8 may be adopted instead so that multiple bit values can be embedded into one encrypted value. Given an encrypted image and a string of bit values to be hidden, the proposed algorithm can be implemented in the following steps.

1) Scan the encrypted image and check if Eq. (10) holds for an encrypted pixel value and the corresponding bit value.

2) No operation is needed if Eq. (10) holds. Otherwise, the encrypted pixel value needs to be changed by applying Eq. (11) until Eq. (10) holds.

3) The data hiding process is finished after every encrypted pixel value has been checked and changed if needed. Therefore, one bit can be hidden per encrypted pixel by adopting the module 2 operation in Eq. (10).

To extract the hidden data, the encrypted image is scanned in the same order as in data hiding. By applying Eq. (12) on every encrypted pixel value, all of the hidden bit values can be extracted while there is no need to change the encrypted image.

5. Experimental Results

In the experiments, the two algorithms proposed in Section 3 and Section 4 were applied to the USC-SIPI images downloaded from [38]. The test images were in size of 512×512 and converted to the gray-level ones. In implementing the Paillier cryptosystem, the bit length of N was set to 1024. Hereinafter, the algorithm proposed in Section 3 is denoted by the value expansion algorithm, while the algorithm proposed in Section 4 is denoted by the self-blinding algorithm. All of the programs were developed with the Java Eclipse SDK and run on a 64-bit PC with Intel Core CPU @3.2 GHz and 8G RAM. In the following, the performance of the two proposed algorithms will be introduced and then compared with the methods in [8] and [22].

5.1. Performance of the Value Expansion Algorithm

5.1.1. Verifying the reversibility

A string of bit values were randomly generated and hidden into each encrypted image by firstly embedding the side information into a fixed number of the encrypted pixel values as discussed in Section 3.3. To verify reversibility, the encrypted images did not undergo other processing before data hiding. The plain-text images were obtained after image decryption, and directly compared with the original ones. At various hiding rates to be mentioned as follows, all test images were exactly recovered while the bit values hidden in the encrypted images were correctly extracted.

5.1.2. Embedding capacity

To test the embedding capacity, every grey-level value from 0 to 255 was used to hide multiple bit values, respectively. As an original pixel value can be represented with 8 bits, a larger memory space is needed to store a decrypted value as well as the following intermediate values. When 8 bytes were used to store a decrypted value, up to 55 bits were hidden per pixel and correctly extracted. When 4 bytes were allocated to store a decrypted value instead, up to 23 bits per pixel (bpp for short) were hidden and correctly extracted. Even

when 9 bits were allocated for a decrypted value, 1 bpp information could be hidden into an encrypted image and correctly extracted. Note that decryption is performed pixel by pixel and there is no need to keep the intermediate values of the previously decrypted pixels.

In the existing RDH methods for the encrypted images such as [8]-[19], the embedding capacity was far below 1 bpp. Even in hybrid methods such as in [20] and [21], the hiding rate was no more than 1 bpp for most test images. So much higher embedding capacity can be achieved in our proposed algorithm by exploiting the redundancy in data representation of Paillier cryptosystem. When a big integer (with the bit length of 2048) was used to store a decrypted value, up to 1015 bits were hidden into an encrypted pixel and correctly extracted. In that case, one cipher value in the encrypted image can be divided and hidden into no more than three encrypted pixel values to save the bandwidth. At the recipient side, the hidden cipher value was correctly extracted after decrypting the encrypted pixel values used to carry it. Note that the complexity of both data hiding and extraction was increased with the hiding rate.

5.1.3. Security analysis

For security analysis, the original plain-text image of "Lena", the encrypted image with no data hidden, the encrypted images at three different hiding rates (1 bpp, 23 bpp and 55 bpp, respectively), and the recovered one after decryption, are shown in Figure 6. The four images exhibited in Figure 6 (b), (c), (d) and (e) were obtained by performing the arithmetic modulo 256 on the real encrypted images. The encrypted images exhibited in Figure 6 are different, indicating that the encrypted pixel values were changed by data hiding. But all of them look similar because the encrypted pixel values were randomly distributed within the interval of [0, 255] after performing the arithmetic modulo 256.

Since the value expansion operations were all performed in homomorphic encryption domain, the encrypted images with the hidden data were protected by Paillier cryptosystem. Given that the encrypted images with the hidden data were not further processed, the plain-text images recovered from the four

encrypted images were all identical to the original one, as shown in Figure 6 (f).

5.2. Performance of the Self-Blinding Algorithm

When applying the self-blinding algorithm, every encrypted pixel value was processed by using Eq. (11) till the requirement in Eq. (10) was satisfied. The experimental results show that data hiding can be conducted on every encrypted image without changing the corresponding plain-text value. To increase the data hiding rate, multiple bits can be embedded into one cipher value. For instance, to embed a 3-bit binary value $b_1b_2b_3$ into one cipher value $\mathbf{e_k}(i)$, the following condition should be met.

$$\mathbf{e_k}(i) \ mod \ 2^3 \ = \ b_1b_2b_3. \tag{13}$$

The requirement in Eq. (13) could be met by repeatedly applying Eq. (11) if needed so that a data hiding rate of 3 bpp was achieved. In the experiments, the hiding rate was further increased up to 12 bpp by embedding a 12-bit binary value into one cipher value. Meanwhile, the complexity of data embedding will be exponentially increased with the hiding rate.

Since data embedding and extraction were both performed in the encryption domain, an encrypted image was always protected by Paillier cryptosystem before and after data hiding. Therefore, the security level was not degraded by applying the self-blinding algorithm to transmit the additional information in the applications of Paillier cryptosystem.

5.3. Performance comparison

The performance of the proposed two algorithms was compared with that of [8] and [22], as shown in Table 1. It can be seen that the data embedding domain of the proposed two algorithms is different from that of [8] and [22], which are based on encryption with a stream cipher. The data extraction in [8] and [22] is separable from image decryption, indicating that data extraction can be arbitrarily performed before or after image decryption. As for the proposed two algorithms, the data extraction domain is fixed (one is in plain-text domain

and the other is in Paillier encryption domain) so that the suitable one should be chosen according to the specific applications.

As an 8-bit pixel value was encrypted into a 2048-bit big integer for $N = 1024$ in the Paillier cryptosystem, the embedding capacity of the proposed algorithms is much higher than that of [8] and [22]. In that case, the highest embedding rate is 1015 bpp with the value expansion algorithm, while up to 12 bpp can be achieved with the self-blinding algorithm. It should be noted that hiding additional data into the encrypted image does not increase the size of the encrypted image. For instance, the data expansion algorithm can be used in transmitting the processed encrypted image to the web user so that decryption can be firstly performed. By embedding a portion of the encrypted pixels into the others, the total size of the encrypted image can be reduced up to 49.56% to save the bandwidth. With the self-blinding algorithm, additional data can also be transmitted without image decryption so that the security level is not degraded at all. Lastly, additive homomorphism can be provided by the proposed algorithms so that the homomorphic processing can be applied before data hiding, or after extracting the data hidden with the self-blinding algorithm. Compared with the method in [22], more useful processing (e.g., [33]-[37]) can be performed on the images encrypted in the Paillier cryptosystem without infringing the user privacy and confidentiality.

6. Conclusion

In this paper, reversible image data hiding in the homomorphic encryption domain has been investigated. Depending on if image decryption is required for data extraction, two algorithms have been proposed for Paillier cryptosystem, respectively. The first algorithm is suitable for the scenario where image decryption is required for data extraction. In contrast, the second algorithm is suitable for the scenario where both of data embedding and extraction need to be conducted in homomorphic encryption domain.

The experimental results on test images have shown that high data hiding

capacity can be achieved by exploiting the redundancy in Paillier cryptosystem. Since the user privacy and data integrity can be more securely protected by homomorphic encryption, the proposed algorithms are more applicable in the cloud computing, compared with the existing methods designed for encryption with a stream cipher. Consequently, additional information can be losslessly transmitted in different applications of Paillier cryptosystem by using the proposed algorithms.

Acknowledgment

The authors sincerely thank the editors and anonymous reviewers for their valuable comments and insightful suggestions to improve the paper quality.

This work was supported by National Natural Science Foundation of China (No. 61100169, 61272366), Natural Science Foundation of Jiangsu Province of China (BK20151131), and Shenzhen R&D Program (GJHZ20140418191518323).

Reference

[1] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circ. Syst. Video Technol. 13 (8) (2003) 890-896.

[2] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circ. Syst. Video Technol. 16 (3) (2006) 354-362.

[3] V. Sachnev, H.J. Kim, J. Nam, S. Suresh, and Y.Q. Shi, Reversible watermarking algorithm using sorting and prediction, IEEE Trans. Circuits Syst. Video Technol. 19(7) (2009) 989-999.

[4] I.-C. Dragoi, D. Coltuc, On local prediction based reversible watermarking, IEEE Trans. Image Process. 24(4) (2015) 1244-1246.

[5] H.T. Wu, J. Huang, and Y.Q. Shi, A reversible data hiding method with contrast enhancement for medical images, J. Vis. Commun. Image R. 31 (2015) 146-153.

[6] X. Li, W. Zhang, X. Gui, B. Yang, Efficient reversible data hiding based on multiple histograms modification, IEEE Trans. Inf. Foren. Sec. 10(9) (2015) 2016-2027.

[7] J. Wang, J. Ni, X. Zhang, and Y. Q. Shi, Rate and Distortion Optimization for Reversible Data Hiding Using Multiple Histogram Shifting, IEEE Trans. Cybern. in press.

[8] X. Zhang, Separable reversible data hiding in encrypted image, IEEE Trans. Inform. Forensics Secur. 7 (2) (2012) 826-832.

[9] D. Xu, R. Wang, Separable and error-free reversible data hiding in encrypted images, Signal Process. 123 (2016) 9-21.

[10] X. Zhang, Reversible data hiding in encrypted images, IEEE Signal Process. Lett. 18(4) (2011) 255-258.

[11] W. Hong, T. Chen, H. Wu, An improved reversible data hiding in encrypted images using side match, IEEE Signal Process. Lett. 19 (4) (2012) 199-202.

[12] X. Zhang, Z. Qian, G. Feng, and Y. Ren, Efficient reversible data hiding in encrypted images, J. Vis. Commun. Image R. 25(2) (2014) 322-328.

[13] X. Liao, C. Shu, Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels, J. Vis. Commun. Image R. 28 (2015) 21-27.

[14] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, Y. Y. Tang, Secure reversible image data hiding over encrypted domain via key modulation, IEEE Trans. Circ. Syst. Video Technol. 26(3) (2016) 441-452.

[15] 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) 1164C1170.

[16] X. Zhang, J. Wang, Z. Wang, H. Cheng, Lossless and reversible data hiding in encrypted images with public key cryptography, IEEE Trans. Circ. Syst. Video Technol. in press.

[17] C. Qin, X. Zhang, Effective reversible data hiding in encrypted images with privacy protection for image content, J. Vis. Commun. Image R. 31 (2015) 154-164.

[18] X. Wu, W. Sun, High-capacity reversible data hiding in encrypted images by prediction error, Signal Process. 104 (2014) 387-400.

[19] Z. Qian, X. Zhang, Reversible data hiding in encrypted image with distributed source encoding, IEEE Trans. Circ. Syst. Video Technol. 26(4) (2016) 636-646.

[20] K. Ma, W. Zhang, W. Sun, Reversible data hiding in encrypted images by reserving room before encryption, IEEE Trans. Inf. Foren. Sec. 8(3) (2013) 553-562.

[21] X. Cao, L. Du, X. Wei, D. Meng, High capacity reversible data hiding in encrypted images by patch-level sparse representation, IEEE Trans. Cybern. 46(5) (2016) 1132-1143.

[22] M. Li, D. Xiao, Y. Zhang, H. Nan, Reversible data hiding in encrypted images using cross division and additive homomorphism, Signal Process-Image 39(A) (2015) 234-248.

[23] X. Zhang, Commutative reversible data hiding and encryption, Secur. Commun. Netw. 6(11) (2013) 1396-1403.

[24] R. L. Rivest, L. Adleman, M. L. Dertouzos, On data banks and privacy homomorphisms, Foundations of Secure Computation. (1978) 169-179.

[25] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, Advances in Cryptology - EUROCRYPT'99. (1999) 223-238.

[26] C. Gentry, Fully homomorphic encryption using ideal lattices, Proc. the 41st ACM Symposium on Theory of Computing. (2009) 169-178.

[27] C. Gentry, S. Halevi, N. P. Smart, Fully homomorphic encryption with polylog overhead, Proc. Advances in Cryptology - EUROCRYPT'12, LNCS 7237 (2012) 465-482.

[28] Z. Brakershi, C. Gentry, S. Halevi, Packed ciphertexts in LWE-based homomorphic encryption, Proc. Public Key Cryptography - PKC2013, LNCS 7778 (2013) 1-13.

[29] Z. Brakershi, V. Vaikuntanathan, Efficient Fully Homomorphic Encryption from (Standard) LWE, SIAM J. Comput. 43(2) (2014) 831-871.

[30] M. Haghighat, S. Zonouz, M. Abdel-Mottaleb, Security and Privacy in Cloud Computing: Vision, Trends, and Challenges, IEEE Trans. Cloud Comput. 2(2) (2015) 30-38.

[31] R. Cramer, R. Gennaro, B. Schoenmakers, A secure and optimally efficient multi-authority election scheme, European Trans. Telecommun. 8(5) (1997) 481-490.

[32] M. Barni, T. Kalker, S. Katzenbeisser, Inspiring new research in the field of signal processing in the encrypted domain, IEEE Signal Proc. Mag. 30(2) (2013) 16-16.

[33] T. Bianchi, A. Piva, M. Barni, On the implementation of the discrete fourier transform in the encrypted domain, IEEE Trans. Inf. Foren. Sec. 4(1) (2009) 86-97.

[34] T. Bianchi, A. Piva, M. Barni, Encrypted domain dct based on homomorphic cryptosystems, EURASIP Journal on Information Security, (2009) 716357.

[35] P. Zheng, J. Huang, Implementation of the discrete wavelet transform and multi-resolution analysis in the encrypted domain, Proc. 19th ACM International Conference on Multimedia. (2011) 413-422.

[36] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, T. Schneider, Privacy-preserving ecg classification with branching programs and neural networks, IEEE Trans. Inf. Foren. Sec. 6(2) (2011) 452-468.

[37] P. Zheng, J. Huang, Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain, IEEE Trans. Image Process. 22(6) (2013) 2455-2468.

[38] The USC-SIPI Image Database: http://sipi.usc.edu/database/

Table 1: Performance comparison between [8], [22] and the proposed two algorithms

Algorithm	embedding domain	extraction domain	embedding capacity (bpp)	homomorphism
[8]	stream cipher encryption	separable	0.050	no
[22]	stream cipher encryption	separable	0.7702	additive
value expansion (N=1024)	Paillier encryption	plain-text domain	1015	additive
self-blinding (N=1024)	Paillier encryption	Paillier encryption	at least 12	additive

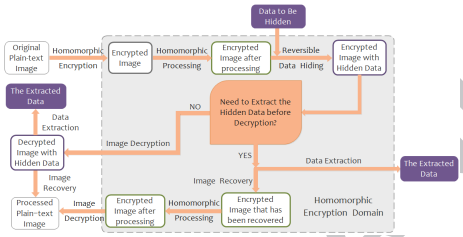

Figure 1: Flowchart of reversible image data hiding in the homomorphic encryption domain.

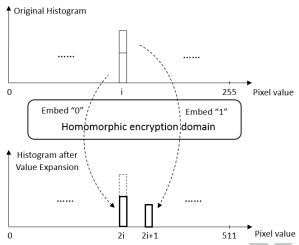

Figure 2: Illustration of reversible data hiding by value expansion.

(a) Original image of "Lena" (b) Encrypted (no data) (c) Encrypted (1 bpp)

(d) Encrypted (23 bpp) (e) Encrypted (55 bpp) (f) Decrypted image

Figure 3: The original test image "Lena", the encrypted images with data hidden by the value expansion algorithm, and the decrypted image.

1) Reversible image data hiding in the Paillier Cryptosystem has been introduced;
2) Two new reversible data hiding algorithms are proposed for the encrypted images;
3) The potential applications of the proposed algorithms have been discussed;
4) High-capacity reversible data hiding can be achieved in the Paillier Cryptosystem.