Lossless Data Hiding in Encrypted Images Compatible With Homomorphic Processing

Hao-Tian Wu, *Senior Member, IEEE*, Yiu-Ming Cheung, *Fellow, IEEE*, Zhenwei Zhuang, Lingling Xu, and Jiankun Hu, *Senior Member, IEEE*

***Abstract*—Reversible data hiding in ciphertext has potential applications for privacy protection and transmitting extra data in a cloud environment. For instance, an original plain-text image can be recovered from the encrypted image generated after data embedding, while the embedded data can be extracted before or after decryption. However, homomorphic processing can hardly be applied to an encrypted image with hidden data to generate the desired image. This is partly due to that the image content may be changed by preprocessing or/and data embedding. Even if the corresponding plain-text pixel values are kept unchanged by lossless data hiding, the hidden data will be destroyed by outer processing. To address this issue, a lossless data hiding method called random element substitution (RES) is proposed for the Paillier cryptosystem by substituting the to-be-hidden bits for the random element of a cipher value. Moreover, the RES method is combined with another preprocessing-free algorithm to generate two schemes for lossless data hiding in encrypted images. With either scheme, a processed image will be obtained after the encrypted image undergoes processing in the homomorphic encrypted domain. Besides retrieving a part of the hidden data without image decryption, the data hidden with the RES method can be extracted after decryption, even after some processing has been conducted on encrypted images. The experimental results show the efficacy and superior performance of the proposed schemes.**

***Index Terms*—Homomorphic processing, image encryption, lossless data hiding, Paillier cryptosystem, randomness.**

Manuscript received 23 May 2021; revised 29 August 2021 and 10 January 2022; accepted 25 March 2022. Date of publication 15 April 2022; date of current version 17 May 2023. This work was supported in part by the Natural Science Foundation of Guangdong Province of China under Grant 2021A1515011798; in part by the National Natural Science Foundation of China under Grant 61772208 and Grant 61672444; in part by the NSFC/RGC Joint Research Scheme under Grant N_HKBU214/21; in part by the RGC General Research Fund under Grant 12201321; in part by the Hong Kong Baptist University under Grant RC-FNRA-IG/18-19/SCI/03 and Grant RC-IRCMs/18-19/SCI/01; in part by the Innovation and Technology Fund of Innovation and Technology Commission of the Hong Kong Government under Grant ITS/339/18; and in part by Shenzhen Science and Technology Innovation Commission (SZSTC) under Grant SGDX20190816230207535. This article was recommended by Associate Editor S. Ozawa. *(Corresponding author: Yiu-Ming Cheung.)*

Hao-Tian Wu, Zhenwei Zhuang, and Lingling Xu are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: wuht@scut.edu.cn; cszhenwei0512@mail.scut.edu.cn; csllxu@scut.edu.cn).

Yiu-Ming Cheung is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: ymc@comp.hkbu.edu.hk).

Jiankun Hu is with the School of Engineering and Information Technology, The University of New South Wales, Australian Defence Force Academy, Canberra, ACT 2610, Australia (e-mail: j.hu@adfa.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3163245.

Digital Object Identifier 10.1109/TCYB.2022.3163245

I. Introduction

REVERSIBLE data hiding (RDH) in cipher media for privacy protection has recently drawn attentions in the community (e.g., [1] and [2]). As extra data have been hidden in a cipher image, an original plain-text image may be obtained after decrypting the cipher image. Meanwhile, the data hidden in the cipher image can be extracted before or after decryption. In the literature, RDH in encrypted images (RDH-EI) has been proposed to send useful data to the receiver, such as the schemes in [2]–[29]. For instance, RDH-EI can be performed by exploiting the redundancy in the encrypted domain for content annotation and authentication (e.g., [2]–[6]).

The RDH-EI methods can be classified regarding whether data extraction is separable from image decryption. In [3], a separable scheme is designed to extract the data hidden in an encrypted image without decrypting it. Moreover, the original plain-text image can be obtained when image decryption is allowed. To increase the embedding capacity of the separable methods, advances have been made by adopting prediction [4], distributed source coding [10], most significant bits (MSBs) prediction [11], and so on.

As it is inconvenient to leverage the redundancy in the plain-text images for RDH in the encrypted domain, a preprocessing is performed in [7] to vacate space before encryption (VSBE). Accordingly, the existing methods may be divided into two classes. The first class of methods spares space in the plain-text images before encrypting them (e.g., [7]–[9]). The vacated values have been reversibly hidden in the preprocessed image (e.g., by using the RDH methods in [30]–[36]), while the spared space is used to accommodate extra data after image encryption. The preprocessed image should be kept unchanged so that the values hidden in it can be extracted to recover the original image. If the encrypted image has been altered, the original image may not be correctly recovered. The other class of methods spare space in the encrypted domain for data embedding (e.g., [3]). Since no preprocessing is required, it is feasible to process an encrypted image to generate the desired one before embedding data into it. As data hiding is directly performed in an encrypted domain (e.g., by using a stream cipher), the embedding capacity may be relatively low.

There are mainly two types of cryptosystems adopted in the existing RDH-EI schemes. The first type is to perform encryption with a stream cipher, which is implemented with low complexity. However, processing in the encrypted domain is hardly allowed with a cipher stream, such as the

schemes proposed in [2]–[19]. The second kind of encryption achieves "privacy homomorphism" [37] to enable processing in the encrypted domain by using the cryptosystems proposed in [38]–[42]. So a desired plain-text image may be obtained after decrypting the processed cipher image. The drawbacks of homomorphic cryptosystems such as Paillier's [38] include high complexity and increased data size. Nevertheless, the RDH-EI schemes proposed in [20]–[29] and the ones proposed in this article do not increase the file size of an encrypted image by exploiting redundancy in encrypted images.

As processing in a homomorphic encrypted domain (hereinafter denoted by homomorphic processing for short) is useful in some privacy-preserving applications (e.g., [43]–[45]), the data hidden in encrypted images may be easily altered by outer processing. In the schemes proposed in [20], [23], [24], and [27], a preprocessing is conducted so that the original image is modified before being encrypted. After the encrypted image undergoes processing in the encrypted domain, a processed plain-text image is generated after decryption, which is not corresponding to the original image but the preprocessed image. So neither the original nor the exact desired image can be obtained after some processing has been conducted on the encrypted image. When no preprocessing is required (e.g., [26] and [28]), the original plain-text image is altered by performing data embedding in the encrypted domain. Even with the lossless data hiding schemes relying on homomorphic and probabilistic properties (e.g., the lossless scheme in [23] and the algorithms proposed in [21], [25], and [29]), the original image is not changed by data embedding, but processing on encrypted images will change the hidden data.

As compatibility with processing in the encrypted domain has not been fully considered in designing the RDH-EI schemes in [20]–[29], a new preprocessing-free and lossless data hiding method called random element substitution (RES) is proposed for the Paillier cryptosystem. In particular, data embedding is conducted by substituting the to-be-hidden bits for the random element in a cipher value so that the plain-text value is preserved. The RES method can be applied in or after the process of encryption, while the decrypted plain-text value is needed to extract the hidden data from the cipher value. To achieve data extraction before decryption as well, it is combined with the self-blinding (SB) method proposed in [25] so that two lossless data hiding schemes are generated for encrypted images. With either scheme, a processed image can be directly obtained after the encrypted image undergoes the desired homomorphic processing. In addition, the data hidden in the encrypted image can be correctly extracted before processing in the encrypted domain, before image decryption, or after decryption. Even after some processing has been applied in the encrypted domain, the data hidden with the RES method may still be extracted. The experimental results on the USC image set [46] have demonstrated efficacy and superior performance of the proposed schemes. Compared with the schemes in [23]–[29], the proposed ones are more suitable for the scenarios where homomorphic processing is required.

The remainder of this article is organized as follows. Section II introduces the Paillier cryptosystem, the related work, and our contributions. The RES method is presented in Section III. Then, two new schemes are generated in Section IV. The experimental results obtained with the two schemes are given in Section V and the performances are compared with the schemes in [23]–[29]. Finally, a conclusion is drawn in Section VI, while the proof of data extraction with the RES method is given in the Appendix.

TABLE I
SOME NOTATIONS

Notation	Description
p, q	Two large prime numbers used to set up a Paillier cryptosystem
N	The product of p and q
$\lambda(N)$	The least common multiple of $p-1$ and $q-1$
n	The bit length of N
$\mathbb{Z}_N^*$	The set of integers relatively prime to N
$\mathbb{Z}_{N^2}^*$	The set of integers relatively prime to N^2
g	A randomly generated number in $\mathbb{Z}_{N^2}^*$ that N divides the order of g
$\mathbf{K}$	A public key of a Paillier cryptosystem, consisting of N and g
$L(u)$	The function gives the quotient by dividing $u-1$ by N, i.e., $\frac{u-1}{N}$
mod	The modulo operation
μ	The modular multiplicative inverse of $L(g^{\lambda(N)} \bmod N^2)$
$\mathbf{k}$	The private key corresponding to $\mathbf{K}$, including $\lambda(N)$ and μ
r, r_a, r_b	Integers randomly chosen from $\mathbb{Z}_N^*$
m, i	Two plain-text values
c_r	A cipher value generated by encrypting m with r, g and N
$\mathbf{e_K}[\cdot]$	The encryption operation with $\mathbf{K}$
$\mathbf{d_k}[\cdot]$	The decryption operation with $\mathbf{k}$
b	A bit value to be embedded
b'	An extracted bit value
$\mathbf{I}$	A plain-text image, consisting of a set of plain-text pixel values
$\mathbf{e_K}(\mathbf{I})$	An encrypted image, consisting of a set of cipher values
d_x	The decimal value of x bit values $\{a_1 a_2 \ldots a_x\}$
m_x	The decimal value of $x+1$ bit values $\{a_1 a_2 \ldots a_x 1\}$
c_m	A cipher value generated by encrypting m with m_x, g and N
D_A	A piece of message to be extracted after image decryption
D_{B1}, D_{B2}	Two pieces of message to be extracted before image decryption

II. Background and Related Work

In this section, the Paillier cryptosystem is first introduced, followed by the additive homomorphism, the SB method proposed in [25] and the other RDH schemes for homomorphic encrypted images. For the convenience of reference, some notations used in this article are listed in Table I.

A. Paillier Cryptosystem

The Paillier cryptosystem proposed in [38] is a probabilistic asymmetric algorithm for public-key cryptography with the decisional composite residuosity assumption. To set up such a cryptosystem, two large prime numbers p and q are found so that their product is relatively prime to $(p-1)\cdot(q-1)$. The product of p and q is denoted by N, which is included in the public key denoted by $\mathbf{K}$. A randomly generated number $g \in \mathbb{Z}_{N^2}^*$ is also included in $\mathbf{K}$ under the condition that N divides the order of g. A big integer c_r is generated after encrypting a plain-text value $m \in [0, N-1]$ with $\mathbf{K}$ by

$$c_r = \mathbf{e_K}[m] = g^m \times r^N \bmod N^2 \tag{1}$$

where an integer r is randomly chosen from $\mathbb{Z}_N^*$ but it does not belong to $\mathbf{K}$. To encrypt a digital image, a string of big integers are generated after encrypting all pixel values in it in sequence. To decrypt m from the cipher value c_r, the private key $\mathbf{k}$ is required, which consists of $\lambda(N)$ and μ. $\lambda(N)$ is the least common multiple of $p-1$ and $q-1$, while μ is the modular multiplicative inverse of $L(g^{\lambda(N)} \bmod N^2)$, where $L(u) = (u-1/N)$. The plain-text m can be decrypted from c_r by

$$m = \mathbf{d_k}[c_r] = L\left[{c_r}^{\lambda(N)} \bmod N^2\right] \cdot \mu \bmod N. \tag{2}$$

Hereinafter, the encryption with $\mathbf{K}$ and decryption with $\mathbf{k}$ are denoted by $\mathbf{e}_{\mathbf{K}}[\cdot]$ and $\mathbf{d}_{\mathbf{k}}[\cdot]$, respectively. A plain-text image is obtained after decrypting every cipher value contained in an encrypted image and no overflows will be caused if all the decrypted pixel values are in a predefined range (e.g., $[0, 255]$ for a 8-bit grayscale value).

B. Additive Homomorphism

Given two cipher values $\mathbf{e}_{\mathbf{K}}(m_1)$ and $\mathbf{e}_{\mathbf{K}}(m_2)$, which are obtained in the same Paillier cryptosystem by encrypting two plain-text values m_1 and m_2, another cipher value $\mathbf{e}_{\mathbf{K}}(m')$ can be generated by

$$\begin{aligned}\mathbf{e}_{\mathbf{K}}(m') &= [\mathbf{e}_{\mathbf{K}}(m_1) \cdot \mathbf{e}_{\mathbf{K}}(m_2)] \bmod N^2 \\ &= \left(g^{m_1} \times r_1^N \bmod N^2\right)\left(g^{m_2} \times r_2^N \bmod N^2\right) \bmod N^2 \\ &= g^{m_1+m_2}(r_1 r_2)^N \bmod N^2. \end{aligned} \tag{3}$$

The obtained cipher value can be decrypted with the private decryption key $\mathbf{k}$ to generate a new plain text by

$$\begin{aligned}\mathbf{d}_{\mathbf{k}}\left[\mathbf{e}_{\mathbf{K}}(m')\right] &= \mathbf{d}_{\mathbf{k}}\left[g^{m_1+m_2} \times (r_1 r_2)^N \bmod N^2\right] \\ &= (m_1 + m_2) \bmod N. \end{aligned} \tag{4}$$

Hence, the addition (modulo N) in the plain-text domain is conducted in the Paillier cryptosystem by applying (3).

The additive homomorphism has been used to embed extra data in the encrypted domain, such as in [25] and [26]. As shown in (1), an integer within $[0, N-1]$ is encrypted to produce a cipher value with bit length $2n$, where n denotes the bit length of N. As shown in (3), a pixel value within $[0, 255]$ can be doubled in the encrypted domain by multiplying the cipher value with itself. In addition, an extra bit value can be easily embedded by multiplying the cipher value of 1 to embed 1 or doing nothing to embed 0. After decrypting the cipher value with $\mathbf{k}$, the hidden bit can be extracted from the decrypted value by modulo 2, while the original pixel values are recovered by dividing the decrypted value by 2.

C. Data Embedding With Self-Blinding Property

In some cases, it is desirable to extract the hidden data in the encrypted domain. For example, an image sender hides extra information in an encrypted image, which is sent to the cloud server. Since the private key $\mathbf{k}$ is not known by the cloud server, the hidden data should be extracted without decrypting the encrypted image. To extract the hidden data in the encrypted domain, the SB property has been exploited in [23], [25], and [29] by modifying a cipher value without changing its plain-text value.

1) Property of Self-Blinding: As shown in (1), a random element is used in the encryption of a Paillier cryptosystem so that different cipher values may be generated from the same plain-text value. That means multiple cipher values may be decrypted to the same plain-text value. More precisely, the SB property indicates that a cipher value can be changed without affecting its original plain text, that is

$$\begin{aligned}\mathbf{d}_{\mathbf{k}}\left[\mathbf{e}_{\mathbf{K}}(m) r_a^N \bmod N^2\right] &= \mathbf{d}_{\mathbf{k}}\left[\left(g^m\ r^N \bmod N^2\right) r_a^N \bmod N^2\right] \\ &= \mathbf{d}_{\mathbf{k}}\left[g^m\ (r r_a)^N \bmod N^2\right] \\ &= m \bmod N \end{aligned} \tag{5}$$

where r_a and r are both random elements in $\mathbb{Z}_N^*$.

2) Data Hiding Based on the Self-Blinding Property: The SB property can be exploited to hide extra data in a cipher value, such as in the SB method proposed in [25]. For example, a bit value denoted by b can be obtained from a cipher value $\mathbf{e}_{\mathbf{K}}(i)$ if

$$\mathbf{e}_{\mathbf{K}}(i) \bmod 2 = b. \tag{6}$$

If $\mathbf{e}_{\mathbf{K}}(i) \bmod 2 \neq b$, $\mathbf{e}_{\mathbf{K}}(i)$ should be modified to make the condition in (6) hold. Based on the SB property, $\mathbf{e}_{\mathbf{K}}(i)$ is multiplied by r_a^N, where $r_a \in \mathbb{Z}_N^*$ generates a new cipher value by

$$c_{r_a}(i) = \left[\mathbf{e}_{\mathbf{K}}(i) \times r_a^N\right] \bmod N^2. \tag{7}$$

From (5), it can be known that $\mathbf{d}_{\mathbf{k}}[c_{r_a}(i)] = i \bmod N$. Hence, (7) can be iteratively performed until $c_{r_a}(i) \bmod 2 = b$. By finding out an appropriate cipher value that corresponds to the same plain text, one bit value is embedded in $c_{r_a}(i)$, which can be extracted by

$$b' = c_{r_a}(i) \bmod 2. \tag{8}$$

Moreover, (7) can be applied to embed more bits into one cipher value. For instance, a string of s bit values, which is denoted by $b_1 b_2 \ldots b_s$, can be extracted from a cipher value $\mathbf{e}_{\mathbf{K}}(i)$ if

$$\mathbf{e}_{\mathbf{K}}(i) \bmod 2^s = b_1 b_2 \ldots b_s. \tag{9}$$

If the condition in (9) does not hold, another cipher value $c_r^s(i)$ can be generated from $\mathbf{e}_{\mathbf{K}}(i)$ by applying (7) so that $c_r^s(i) \bmod 2^s = b_1 b_2 \ldots b_s$. By iteratively applying (7) to make (9) hold, s bits are simultaneously embedded. The complexity of searching for the appropriate cipher value is exponentially increased with the bit number to be hidden. In a Paillier cryptosystem with 1024-bit N, up to 14 bits can be hidden into one cipher value in our experiments. Given the hiding rate (i.e., s bits per cipher value) is known, the embedded bits can be retrieved from $c_r^s(i)$ by

$$b'_1 b'_2 \ldots b'_s = c_r^s(i) \bmod 2^s \tag{10}$$

where $b'_1 b'_2 \ldots b'_s$ are the extracted string of bit values.

D. RDH Schemes in Homomorphic Encrypted Domain

Besides the SB method, there are several schemes designed for RDH in the homomorphic encrypted domain. Except the schemes based on integer modulo [20], the learning with errors (LWE)-based public-key cryptography [21] and the fully homomorphic encryption [22], most of the schemes are developed for Paillier cryptosystem [23]–[29].

Based on the encryption proposed in [47], an RDH-EI scheme is proposed in [20] by adding a randomly generated number to a pixel value and processing the sum with modulo 256 operation. By encrypting the pixels in the same cross with the same number, the differences between those pixels are preserved and used for data embedding. By exploiting the additive homomorphism, data extraction before and after

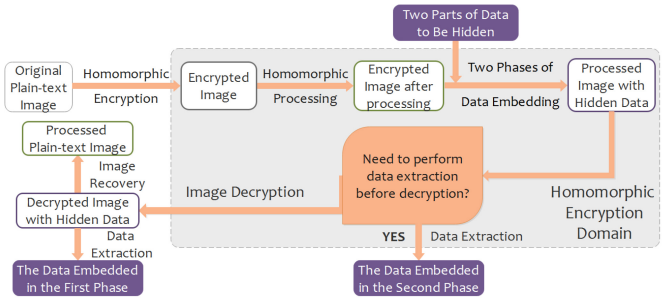


Fig. 1. Flowchart of RDH in a homomorphic encrypted domain after processing.

decryption are both achieved. However, the encrypted image can hardly be further processed because of the cross division adopted in image encryption.

In [23], a combined data hiding scheme is presented for encrypted images in a Paillier cryptosystem. In particular, multilayer wet paper coding [48] is adopted to replace the LSB planes of cipher-text pixels with new values for data embedding. The hidden data can be extracted in the encrypted domain without affecting image decryption. In addition, preprocessing is conducted before image encryption by histogram shrinkage. With the combined scheme, data extractions in both the encrypted and plain-text domains are enabled. Moreover, the original plain-text image can be recovered after decryption and reversing the histogram shrinkage.

In [24], another scheme based on the Paillier cryptosystem is proposed, which is called mirroring cipher-text group (MCG). A reference pixel is chosen to perform data embedding in an MCG unit, while the same bits can be extracted in the encrypted domain or after image decryption. A preprocessing is performed by using the RDH method in [30] to keep the vacated bits in the preprocessed image. Similar to the schemes in [23] and [24], an improved scheme is proposed in [27] by adopting the RDH method in [31] for preprocessing and using the SB method to embed the data to be extracted in the encrypted domain. The RDH-EI schemes in [23], [24], and [27] are not compatible with homomorphic processing due to the preprocessing before image encryption.

In [25], two preprocessing-free methods are developed for the Paillier cryptosystem. One is the SB method, which is similar to the lossless scheme in [23]. Similarly, another RDH-EI method based on probabilistic and homomorphic properties is proposed in [29] by setting up a mapping between the cipher values and secret bits. These schemes do not modify the plain-text values to embed the data to be extracted without image decryption. The other method proposed in [25] is called value expansion (VE), which embeds bits to be extracted after decryption by changing plain-text values in the encrypted domain. In [26], another method is proposed by histogram shifting in the encrypted domain to embed bits to be extracted after decryption. In other words, the VE method and the method in [26] modify the corresponding plain-text values in the encrypted domain for data embedding.

As shown in Fig. 1, the two methods proposed in [25] are further combined in [28] to embed two parts of data after homomorphic processing. The VE method is first applied to embed the data to be extracted after image decryption, while the SB method is applied to hide the data to be extracted without decryption. If homomorphic processing has been applied before data embedding, the processed image will be generated after decryption. In [21], a multilevel RDH-EI scheme is proposed by exploiting the controllable redundancy of LWE-based public-key cryptography. Moreover, a difference expansion-based method is proposed in [22] for fully homomorphic encryption [41]. Nevertheless, the data hidden with the aforementioned schemes (e.g., [20]–[29]) will be easily altered by the processing conducted in the encrypted domain.

E. Our Contribution

As compatibility with processing in an encrypted domain has not been fully considered in designing the RDH-EI schemes in [20]–[29], a preprocessing-free and lossless data hiding method called RES is proposed. Different from the existing schemes, the to-be-hidden bits are used to form random element of a cipher value without changing its plain-text value. Besides, homomorphic processing can be applied to an encrypted image with hidden data to generate the desired image. As the data embedded with the RES method can be extracted after image decryption, the SB method is also adopted to embed the data to be extracted without decryption so that two lossless data hiding schemes are generated for encrypted images, respectively.

Our contributions in this article are summarized as follows. First, data extraction after image decryption is achieved by applying the proposed RES method without changing the plain-text image. As the RES method can be applied in or after the process of encryption without any preprocessing, a mathematical proof of data extraction has been given in the Appendix. Second, two lossless data hiding schemes compatible with processing in the encrypted domain are developed, which do not interfere with the usage of an encrypted image. In other words, a desired image can always be generated after homomorphic processing is conducted on an encrypted image with hidden data. Lastly, the hidden data can be correctly extracted from an encrypted image in multiple scenarios, even after some processing has been conducted on an encrypted image with hidden data. Overall, these properties make our proposed schemes more suitable for RDH in the encrypted images requiring further processing.

III. Random Element Substitution Method

To embed the data to be extracted after decryption, both the VE method in [25] and the histogram shifting method in [26] modify a cipher value to change its plain-text value accordingly. To keep the plain-text value unchanged by data embedding, a preprocessing-free RDH-EI method called RES is proposed by using a string of bit values to form the random element of a cipher value.

A. Random Element Substitution

In the encryption of a plain-text value m with (1), randomness is introduced by employing a random element r in $\mathbb{Z}_N^*$. That means various cipher values may be generated by encrypting m with different random elements from $\mathbb{Z}_N^*$. For embedding a string of x bit values $\{a_1 a_2 \ldots a_x\}$ into a cipher

value of m, the decimal value of $a_1a_2 \dots a_x$, which is denoted by d_x, is used to replace r in (1) to generate a cipher value c_d by $c_d = g^m {d_x}^N \bmod N^2$. In this way, d_x is used to introduce randomness in encrypting the plain text m. To correctly decrypt m from c_d, d_x must be relatively prime to N. To include the case that $\{a_1a_2 \dots a_x\}$ are all zeros, an extra bit 1 is appended so that the decimal of the $x+1$ bit values $\{a_1a_2 \dots a_x\ 1\}$ is nonzero. By denoting the decimal value as m_x, we have

$$c_m = g^m {m_x}^N \bmod N^2. \tag{11}$$

When m_x is relatively prime to N, multiplying g^m with ${m_x}^N$ does not affect the decryption of m.

To ensure the correct decryption, the bit length of m_x (i.e., $x+1$) needs to be appropriately chosen. Since N is the product of two prime numbers p and q, m_x must be relatively prime to N if it is smaller than either of p and q. The values of p and q are not included in $\mathbf{K}$ because they are used to generate the private decryption key $\mathbf{k}$. Nevertheless, $m_x < p$ and $m_x < q$ can be easily satisfied by reducing the value of x. For instance, x can be chosen below $(n/4)$, where n is the bit length of N to test if m_x is relatively prime to N.

B. Data Extraction After Random Element Substitution

As shown in (2), m can be decrypted from the cipher value c_m with $\mathbf{k}$ on the receiver side. With the decrypted m, another integer m_{cx} can be generated by

$$m_{cx} = c_m \cdot \left(g^m\right)^{-1} \bmod N = {m_x}^N \bmod N \tag{12}$$

where $(g^m)^{-1} \bmod N$ is the modular multiplication inverse of g^m, which is an integer so that $g^m \cdot (g^m)^{-1} \equiv 1 \pmod N$. To remove the exponent N in (12), the modular multiplication inverse of N denoted as $N^{-1} \bmod \lambda(N)$ should be found so that $N \cdot N^{-1} \equiv 1 \pmod{\lambda(N)}$, where $\lambda(N)$ is included in $\mathbf{k}$. Then, the decimal value m_x can be obtained by

$$m_x = {m_{cx}}^{N^{-1} \bmod \lambda(N)} \bmod N. \tag{13}$$

Hence, the hidden bits can be extracted with (12) and (13), provided that m has been decrypted from c_m with (2). The detailed proof of (13) will be provided in the Appendix.

Besides m_x, another integer r in $\mathbb{Z}_N^*$ can be employed in encryption. Then, a cipher value c_{rm} is generated by

$$c_{rm} = g^m (rm_x)^N \bmod N^2. \tag{14}$$

Since both m_x and r are relatively prime with N, rm_x is also relatively prime with N. Hence, multiplying g^m with $(rm_x)^N$ does not affect the decryption of m. That means m can be decrypted from c_{rm} with $\mathbf{k}$. Similar to extracting m_x from c_m as shown in (12) and (13), rm_x can be retrieved from c_{rm} if $rm_x \in \mathbb{Z}_N^*$. After that, the embedded value m_x can be calculated from rm_x by knowing the value of r.

C. Data Embedding in Encrypted Domain

Instead of embedding m_x in the encryption of m, m_x can also be embedded in the encrypted domain by multiplying $m_x^N \ mod\ N^2$ with the cipher value $c_r = g^m r^N \bmod N^2$ to generate c_{rm} in (14). As the big integer N is included in the public key $\mathbf{K}$, data embedding can be conducted not only in encryption but also in the encrypted domain.

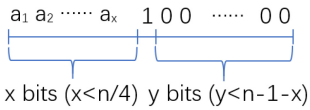


Fig. 2. Hidden bit values $a_1a_2 \dots a_x$ can be obtained by shifting out the rightmost 0s and the adjacent bit value 1, where n is the bit length of the big integer N in a Paillier cryptosystem.

In summary, a string of x bit values and an extra bit value "1," whose decimal value is denoted by m_x, can be used as random element in encryption of m to generate the encrypted value c_m. Suppose m_x is relatively prime with N, it can be correctly extracted by first decrypting c_m. Based on the SB property, m_x can also be embedded in the encrypted domain, while a random element r in $\mathbb{Z}_N^*$ is employed in encryption to introduce randomness. As $r \cdot m_x$ is relatively prime with N, rm_x can be obtained from c_{rm} if the bit length of rm_x is less than n. In that case, m_x can be obtained from rm_x by knowing r.

D. Combining the SB Method With the RES Method

As the plain-text value is unchanged by applying either the RES method or the SB method to embed data in a cipher value, the two methods are combined to achieve data extraction in multiple scenarios. The RES method is first applied because the data embedded with it can be kept unchanged after some processing is performed in the encrypted domain.

For embedding s bit values $\{b_1b_2 \dots b_s\}$ into the cipher value $c_m = g^m {m_x}^N \ mod\ N^2$, c_m is iteratively multiplied by 2^N without changing the plain-text value (i.e., m). Suppose that the following equation holds after c_m is multiplied with 2^N for y times

$$\left[g^m \left(m_x \cdot 2^y\right)^N \bmod N^2\right] \bmod 2^s = m_s \tag{15}$$

where m_s is the decimal value of $b_1b_2 \dots b_s$. The s-bit value embedded with the SB method can be extracted with the modulo operation, that is

$$m_s' = \left[g^m \left(m_x \cdot 2^y\right)^N \bmod N^2\right] \bmod 2^s \tag{16}$$

where m_s' is the decimal value of the string of s bit values and $m_s' = m_s$ if (15) holds.

Note that an extra bit value "1" has been appended at the end of $\{a_1a_2 \dots a_x\}$ when applying the RES method. Given that $x+1+y < n$, $m_x \cdot 2^y$ can be correctly extracted from the cipher value $g^m (m_x \cdot 2^y)^N \bmod N^2$ by first obtaining the plain-text value m. As shown in Fig. 2, the bit values $\{a_1a_2 \dots a_x\}$ can be obtained by right shifting the binary value of $m_x \cdot 2^y$. Specifically, the rightmost y 0s (which were introduced by multiplying c_m with 2^N for y times) are shifted out until the first bit value 1 is encountered. If $m_x \cdot 2^y \geq N$ (i.e., $x+1+y \geq n$), the hidden data m_x may be destroyed. As y is exponentially increased with s, the hiding rate (i.e., s) of the SB method should be controlled to keep m_x intact.

In the case that a random element r in $\mathbb{Z}_N^*$ is employed in encryption, the cipher value $c_{rm} = g^m (rm_x)^N \bmod N^2$ is also

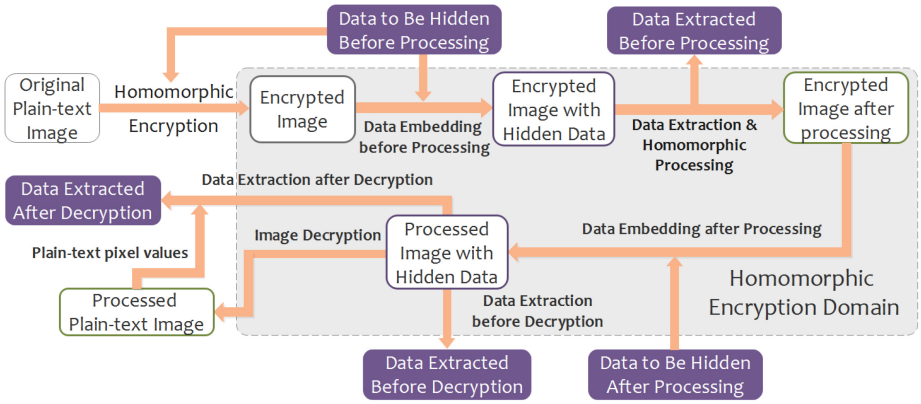


Fig. 3. Framework of RDH in a homomorphic encrypted domain before and after processing, respectively.

iteratively multiplied with 2^N to embed a string of s bit values $\{b_1 b_2 \ldots b_s\}$. Suppose that

$$\left[g^m\left(rm_x \cdot 2^z\right)^N \bmod N^2\right] \bmod 2^s \;=\; m_s \tag{17}$$

after multiplying c_{rm} with 2^N for z times. The decimal value hidden with the SB method can be extracted by

$$m'_s \;=\; \left[g^m\left(rm_x \cdot 2^z\right)^N \bmod N^2\right] \bmod 2^s. \tag{18}$$

Given that $rm_x \cdot 2^z < N$, $rm_x \cdot 2^z$ can be correctly extracted by first decrypting the plain-text value m as discussed in Section III-A. By choosing an odd r, rm_x is also odd so that it can be separated from the z 0s introduced by multiplied c_{rm} with 2^N. As there are z 0s at the end of the extracted value $rm_x \cdot 2^z$, rm_x can be obtained by right shifting $rm_x \cdot 2^z$ to remove the rightmost 0s. With the random element r used in encryption, the decimal value m_x can be obtained from rm_x. If $z \geq n - 1 - x - bl(r)$ where $bl(r)$ represents the bit length of r, m_x cannot be extracted but m can be correctly decrypted. Similarly, the hiding rate s should be controlled to make the data embedded with the RES method extractable.

IV. Two Lossless Data Hiding Schemes

In this section, a new framework for RDH-EI is proposed, which is different from the flowchart illustrated in Fig. 1. In the proposed framework, data embedding can be conducted before or after processing in the homomorphic encrypted domain. Moreover, two schemes are generated by adopting both of the RES and SB methods for data embedding in different stages.

A. New Framework for RDH-EI

Since applying the RES method does not change the plain text of a cipher value, some processing may be conducted in the encrypted domain to generate the desired image. As shown in Fig. 3, data embedding is conducted in image encryption by adopting the RES method. The SB method is adopted in the encrypted domain to embed the data to be extracted without decryption. The data hidden with the SB method can be extracted before any processing is conducted on the encrypted image. After homomorphic processing is performed, both the RES method and the SB method can be applied to generate the processed image with the hidden data. As the data embedded with the RES method may be kept unchanged by some processing in the encrypted domain, two schemes are proposed regarding in which stage it is adopted.

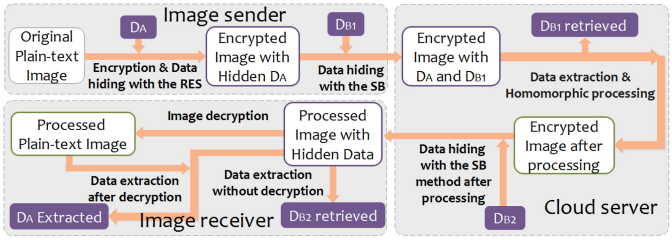


Fig. 4. Schematic of the proposed Scheme I for lossless data hiding in encrypted images.

B. Scheme I: Applying the RES Method in Image Encryption

In this scheme, the receiver needs to set up a Paillier cryptosystem, send the public key **K** (i.e., N and g) to the image sender and cloud server, and keep the private key **k** for image decryption and data extraction. A potential application scenario of Scheme I is as follows. In the medical care sector, medical images in archive must be encrypted by regulation. After storing the encrypted medical images of a patient, the medical doctor would like to add additional information, such as blood test results and diagnosis, which happen at a different time. With our proposed scheme, the medical doctor can embed a piece of new information directly into the encrypted medical images in archive without going through an extra process of decrypting the encrypted medical images in archive, embedding the hidden data, and encrypting the images with hidden data. In some cases, the encrypted images with hidden data need to be sent to the patient via cloud. After processing the encrypted image with hidden data if needed, the cloud server has the capability to add additional information into the encrypted image. After receiving the encrypted image, the patient may extract the message hidden by the cloud server from the ciphertext, obtain the original or processed medical image by decrypting the encrypted image with the private key and extract the message hidden by the hospital by referring to the decrypted image.

As illustrated in Fig. 4, the RES method is applied in the encryption by the image sender to transmit a piece of message

D_A to the receiver. In addition, the image sender may transmit another piece of message D_{B1} to the cloud server, while the cloud server sends a piece of message D_{B2} to the receiver. The procedure of this scheme consists of three stages.

1) Encryption Stage: The messages D_A and D_{B1} are embedded into an encrypted image denoted by $\mathbf{e_K(I)}$, which is obtained by encrypting a plain-text image $\mathbf{I}$ with the public key $\mathbf{K}$ in a Paillier cryptosystem. The encrypted image $\mathbf{e'_k(I)}$ is obtained after embedding D_A and D_{B1} into $\mathbf{e_K(I)}$ with the following details.

1) To encrypt a pixel value m in $\mathbf{I}$ to generate a cipher value c_m with (11), a decimal value denoted by m_x (including x bit values in D_A and an extra bit "1") is embedded. Each of the last four pixel values (denoted by i) is encrypted by employing a random element r in $\mathbb{Z}_N^*$ to generate a cipher value $c = g^i r^N \bmod N^2$.
2) An s-bit value m_s in D_{B1} is embedded by iteratively multiplying c_m by 2^N to make (15) hold, where y is the time of multiplying c_m by 2^N. The last four cipher values are not used to embed D_{B1} into $\mathbf{e_K(I)}$.
3) Four bits are used to represent the value of s and then embedded into the last four cipher values by iteratively multiplying c by 2^N until $c \cdot (2^N)^u \bmod 2 = k$, where k is one of the bit values representing s while u is the time of multiplying c by 2^N.

2) Homomorphic Processing Stage: The cloud server extracts the message D_{B1} from $\mathbf{e'_k(I)}$ before processing it, then embeds another message D_{B2} into the processed image, which is denoted by $\mathbf{e_K(I}_c\mathbf{)}$. The processed image with the hidden D_A and D_{B2} is generated and denoted as $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$.

1) Extract the four bits hidden in the last four cipher values of $\mathbf{e'_k(I)}$ by $c \cdot (2^N)^u \bmod 2$ to obtain the value of s.
2) The message D_{B1} is extracted from the other cipher values by applying (16) and collecting the extracted bits.
3) Apply homomorphic processing on $\mathbf{e'_k(I)}$ to generate a new encrypted image $\mathbf{e_K(I}_c\mathbf{)}$. For instance, $g^m(m_x 2^y)^N \bmod N^2$ is modified to $g^{m'}(m_x 2^y)^N \bmod N^2$, while $g^i(r2^u)^N \bmod N^2$ is modified to $g^{i'}(r2^u)^N \bmod N^2$.
4) Apply the SB method to embed D_{B2} into $\mathbf{e_K(I}_c\mathbf{)}$ without using the last four cipher values. For instance, $g^{m'}(m_x 2^y)^N \bmod N^2$ is modified to $g^{m'}(m_x 2^{y'})^N \bmod N^2$.
5) Apply the SB method to embed the value of v (4 bits) into the last four cipher values in $\mathbf{e_K(I}_c\mathbf{)}$ by iteratively multiplying $g^{i'}(r2^u)^N \bmod N^2$ by 2^N until

$$\left[g^{i'}\left(r2^{u'}\right)^N \bmod N^2\right] \bmod 2 = k \tag{19}$$

where k is a bit value to be embedded and $u' - u \geq 0$ is the time of multiplication with 2^N to make (19) hold.

3) Receiver Stage: The receiver extracts D_{B2} from $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$ without decrypting it, then extracts D_A after $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$ is decrypted with the private key $\mathbf{k}$. The processed image $\mathbf{I}_d$ is obtained after decrypting every cipher value in $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$.

1) Extract the four bits hidden in the last four cipher values of $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$. For a cipher value $c' = g^{i'} \cdot (r2^{u'})^N \ mod\ N^2$, extract one bit value by $c'\ mod\ 2$ to form the 4-bit value v.

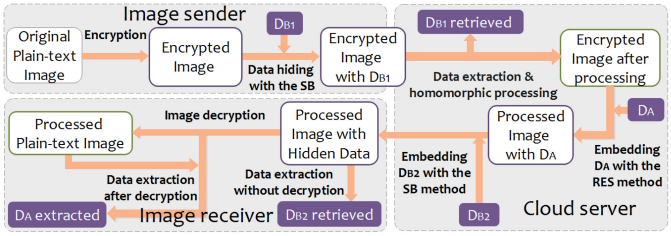


Fig. 5. Schematic of the proposed Scheme II for lossless data hiding in encrypted images.

2) Directly extract D_{B2} from the rest cipher values in $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$. With the obtained value of v, a v-bit binary value m_v can be extracted from a cipher value by modulo 2^v operation.
3) With the private key $\mathbf{k}$, the processed image $\mathbf{I}_d$ can be obtained after decrypting every cipher value in $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$. The original image $\mathbf{I}$ is obtained if the encrypted image has not been processed in the encrypted domain.
4) With a decrypted value m' in $\mathbf{I}_d$ except the last four pixels, obtain the modular multiplication inverse of $g^{m'}$ so that $m_x 2^{y'}$ can be obtained as calculating m_x in (13).
5) Obtain m_x from $m_x 2^{y'}$ in the way as shown in Fig. 2 by locating the bit value 1 adjacent to the rightmost y' 0s.

C. Scheme II: Applying the RES Method After Encryption

In this scheme, the receiver also needs to set up a Paillier cryptosystem, send the public key $\mathbf{K}$ (i.e., N and g) to the image sender and cloud server. In addition, the image sender and receiver share the random elements to be used in encryption. As illustrated in the diagram in Fig. 5, the RES method is applied in the encrypted domain by the cloud server to send the confidential message D_A to the receiver. Meanwhile, the image sender can transmit a piece of message D_{B1} to the cloud server, and the cloud server can send another message D_{B2} to the receiver. The procedure also includes three stages.

1) Encryption Stage: An encrypted image denoted by $\mathbf{e_K(I)}$ is obtained by encrypting a plain-text image $\mathbf{I}$ with the public key $\mathbf{K}$. The encrypted image $\mathbf{e}^s_\mathbf{k}\mathbf{(I)}$ is obtained after embedding D_{B1} into $\mathbf{e_K(I)}$ with the following details.

1) A cipher value c_r is generated by encrypting a pixel value m in $\mathbf{I}$ with (1), where $r \in \mathbb{Z}_N^*$ is a random integer.
2) An s-bit value $b_1 b_2 \ldots b_s$ in D_{B1} is embedded by iteratively multiplying c_r with 2^N to make (9) hold. For example, the multiplication is iteratively conducted for z times. The last four cipher values are not used to embed D_{B1} into $\mathbf{e_K(I)}$.
3) Four bits are used to represent the value of s and then embedded into the last four cipher values by iteratively multiplying c_r with 2^N.

2) Homomorphic Processing Stage: The cloud server extracts the message D_{B1} from $\mathbf{e}^s_\mathbf{k}\mathbf{(I)}$ before processing it, then embeds D_A and D_{B2} into the processed image, which is denoted by $\mathbf{e_K(I}_c\mathbf{)}$. The processed image with the hidden D_A and D_{B2} is generated and denoted as $\mathbf{e}^*_\mathbf{K}\mathbf{(I}_c\mathbf{)}$.

1) Extract the four bits hidden in the last four cipher values of $\mathbf{e}^s_\mathbf{k}\mathbf{(I)}$ with modulo 2 operation to obtain the value of s.

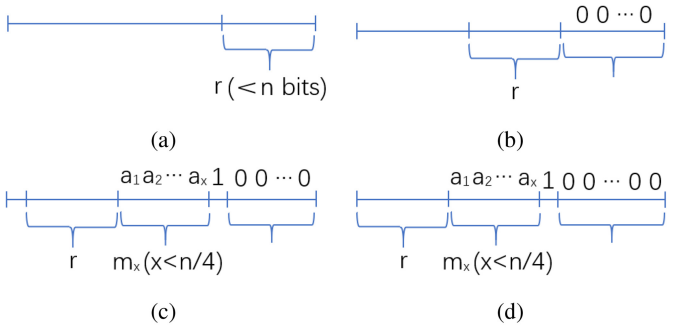


Fig. 6. Random element in a cipher value after: (a) encryption with $r \in \mathbb{Z}_N^*$ by multiplying with r^N, (b) iteratively multiplying with 2^N (i.e., applying the SB method), (c) multiplying with ${m_x}^N$ to apply the RES method, and (d) iteratively multiplying with 2^N.

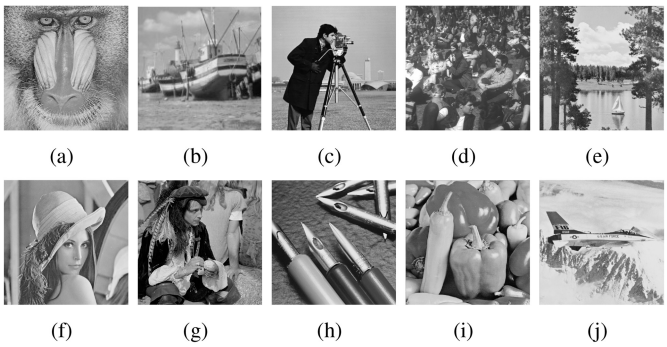


Fig. 7. Gray-level test images with the original size of 512×512. (a) Mandrill. (b) Boat. (c) Cameraman. (d) Crowd. (e) Sailboat. (f) Lena. (g) Male. (h) Pens. (i) Peppers. (j) F-16.

2) The message D_{B1} is extracted from the other cipher values by conducting modulo 2^s operation and collecting the extracted bits.
3) Apply homomorphic processing on $\mathbf{e}_\mathbf{k}^s(\mathbf{I})$ to generate a new encrypted image $\mathbf{e}_\mathbf{K}(\mathbf{I}_c)$.
4) Embed D_A into $\mathbf{e}_\mathbf{K}(\mathbf{I}_c)$ by multiplying a cipher value with ${m_x}^N$ except the last 4 ones. Note that an extra bit 1 is included in m_x following the x bit values.
5) Embed D_{B2} into the cipher image with the hidden D_A by applying the SB method at a rate of v bits per pixel (bpp).
6) Apply the SB method to embed the value of v (4 bits) into the last four cipher values.

3) Receiver Stage: When $\mathbf{e}_\mathbf{K}^*(\mathbf{I}_c)$ is received, the receiver extracts D_{B2} without image decryption, then extracts D_A after the processed image is decrypted with the private key $\mathbf{k}$.

1) Extract the four bits hidden in the last four cipher values of $\mathbf{e}_\mathbf{K}^*(\mathbf{I}_c)$ to obtain the value of v.
2) Extract D_{B2} from the other cipher values in $\mathbf{e}_\mathbf{K}^*(\mathbf{I}_c)$ by performing modulo 2^v operation.
3) Generate a plain-text image $\mathbf{I}_d$ with the private key $\mathbf{k}$ after decrypting every cipher value in $\mathbf{e}_\mathbf{K}^*(\mathbf{I}_c)$. The original image $\mathbf{I}$ is obtained if the encrypted image has not been processed in the encrypted domain.
4) With a decrypted value m' in $\mathbf{I}_d$ except the last four pixels, obtain the modular multiplication inverse of $g^{m'}$ so that the random element in each cipher value can be calculated, similar to calculating m_x with (12) and (13).
5) Within the obtained base, the 0s introduced by applying the SB method can be separated by identifying the bit value 1 appended at the end of m_x, as shown in Fig. 6. So m_x can be extracted by knowing the random element r, which is odd and has been used in the encryption stage.

Compared with Scheme I which is suitable for the image sender to transmit confidential information to the receiver, Scheme II is suitable for the cloud server to send a piece of a confidential message to the receiver. In both schemes, the private decryption key is kept by the receiver while the hidden data can be retrieved in multiple scenarios (i.e., after image decryption, before decryption, and before homomorphic processing).

V. Experiments and Numerical Results

To evaluate the performances of the proposed method and schemes, ten grayscale images were downloaded from the USC website [46] and used in the experiments, as shown in Fig. 7. A LOGO image called "SCUT" was also employed for performance testing. All of the test images were converted to gray-level ones at a resolution of 512 × 512 pixels. The programs of data encryption and decryption, image homomorphic processing, data embedding, and extraction were all developed with Java Eclipse SDK and executed on a 64-bit PC with Intel Core CPU @4.2 GHz and 16-GB RAM. The Paillier cryptosystem was built by setting the bit length of N (i.e., n) to 1024.

In the following, the performance of Schemes I and II is first evaluated and compared, including embedding capacity, applications, and compatibility with homomorphic processing. Then, the security of the RES method and two proposed schemes is analyzed, respectively. After that, the computational complexity of the RES method and two schemes is analyzed and compared with some representative schemes for RDH in the encrypted domain (e.g., [9], [11], [23], [25], and [29]). Finally, the RES method and two proposed schemes are compared with the recent RDH schemes based on Paillier cryptosystem such as [23]–[29].

A. Performance Evaluation of Schemes I and II

The performance of the two proposed schemes is evaluated and compared as follows.

1) Data Extraction and Applications: With Scheme I, the message to be extracted after image decryption is hidden by the image sender, such as a digital signature or other content-associated information. With Scheme II, the message to be extracted after image decryption is embedded by the cloud server, such as the confidential information related to the processing that has been conducted. In both schemes, the private decryption key is required to extract the data hidden by applying the RES method. The cryptosystem should be set up by the receiver, and the public key (i.e., N and g) should be previously shared with the image sender and cloud server, respectively. In applying Scheme II, the random elements to be used in encryption need to be shared between the image sender and receiver.

When no processing was conducted on the encrypted image with hidden data, data extraction was correctly performed with

TABLE II
EMBEDDING CAPACITY OF THE PROPOSED SCHEMES ($n = 1024$)

	Bit length of random element	Applying RES in image encryption	Applying the SB method before processing	Applying the SB method after ∼
Scheme I	$bl(r)$=0	255 bpp	6 bpp	6 bpp
		215 bpp	6 bpp	6 bpp
		145 bpp	6 bpp	6 bpp
		75 bpp	6 bpp	6 bpp
		45 bpp	6 bpp	6 bpp
	$bl(r)=\frac{n}{4}$	255 bpp	5 bpp	5 bpp
		215 bpp	5 bpp	5 bpp
		145 bpp	5 bpp	6 bpp
		75 bpp	5 bpp	6 bpp
		45 bpp	5 bpp	6 bpp
		0 bpp	14 bpp	14 bpp
	$bl(r)=\frac{n}{2}$	255 bpp	4 bpp	4 bpp
		215 bpp	4 bpp	4 bpp
		145 bpp	5 bpp	4 bpp
		75 bpp	5 bpp	5 bpp
		45 bpp	5 bpp	5 bpp
		0 bpp	14 bpp	14 bpp
	$bl(r)=\frac{3n}{4}$	255 bpp	0	0
		215 bpp	2 bpp	1 bpp
		145 bpp	3 bpp	3 bpp
		75 bpp	4 bpp	3 bpp
		45 bpp	4 bpp	3 bpp
		0 bpp	14 bpp	14 bpp
	Bit length of random element	Applying the RES after encryption	Applying the SB method before processing	Applying the SB method after ∼
Scheme II	$bl(r)=\frac{n}{16}$	255 bpp	5 bpp	5 bpp
		215 bpp	5 bpp	5 bpp
		145 bpp	5 bpp	6 bpp
		75 bpp	5 bpp	6 bpp
		45 bpp	6 bpp	6 bpp
		0 bpp	14 bpp	14 bpp
	$bl(r)=\frac{n}{8}$	255 bpp	5 bpp	5 bpp
		215 bpp	5 bpp	5 bpp
		145 bpp	5 bpp	5 bpp
		75 bpp	5 bpp	6 bpp
		45 bpp	6 bpp	5 bpp
		0 bpp	14 bpp	14 bpp
	$bl(r)=\frac{n}{4}$	255 bpp	5 bpp	5 bpp
		215 bpp	5 bpp	5 bpp
		145 bpp	5 bpp	5 bpp
		75 bpp	5 bpp	5 bpp
		45 bpp	5 bpp	5 bpp
		0 bpp	14 bpp	14 bpp
	$bl(r)=\frac{3n}{8}$	255 bpp	5 bpp	4 bpp
		215 bpp	5 bpp	4 bpp
		145 bpp	5 bpp	5 bpp
		75 bpp	5 bpp	5 bpp
		45 bpp	5 bpp	5 bpp
		0 bpp	14 bpp	14 bpp

both schemes. By applying the SB method, the image sender can send a message to the cloud server by hiding it in the encrypted image, while the cloud server can send another message to the receiver by hiding it into the encrypted image. Data extraction in three scenarios (i.e., after image decryption, before image decryption, and before homomorphic processing) was achieved with both schemes, respectively.

2) Embedding Capacity: When Scheme I is applied, a hiding rate around 255 bpp is obtained with the RES method for $n = 1024$. As shown in Table II, the hiding rates in the encrypted domain are affected by the hiding rate of the RES method and the bit length of the random element adopted in encryption. In Table II, the random element is denoted by r and its bit length is denoted as $bl(r)$. In general, the hiding rates with the SB method are reduced with a larger $bl(r)$. This is because the times of multiplying a cipher value with 2^N are limited to maintain the data hidden by the RES method. In the experiments, the highest rate of data extraction before homomorphic processing was 6 bpp when the RES method was applied in image encryption. When the sum of $bl(r)$ and the hiding rate with the RES method was close to n, no more data could be embedded with the SB method to maintain the data hidden with the RES method.

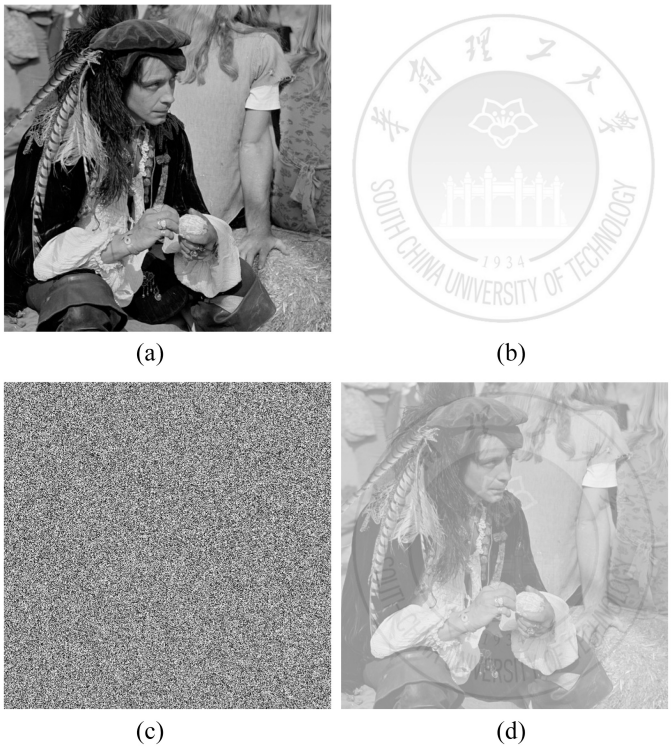


Fig. 8. Average plain-text image can be generated from two input images encrypted with a Paillier cryptosystem. One of the input images has been encrypted with the proposed Scheme I to hide extra data, which can be extracted from the encrypted sum image after decryption. (a) Plain-text input image Man. (b) Plain-text input image SCUT. (c) Encrypted sum image obtained by adding two encrypted images (modulo 256 for illustration). (d) Average image generated by decrypting the sum image and dividing every pixel value by 2.

The hiding rates obtained with Scheme II are also listed in Table II, which are largely affected by $bl(r)$. Different from Scheme I, $bl(r)$ must be larger than zero to introduce randomness in encryption with Scheme II. For each $bl(r)$ value, six data hiding rates were obtained by applying the RES method after homomorphic processing, respectively. For the convenience of comparisons, the same rates were obtained with the RES method as those obtained with Scheme I. It can be seen that the three hiding rates affect each other. As the hiding rates before homomorphic processing can be as high as 14 bpp, the hiding rates with the RES method are reduced to zeros while the hiding rates after homomorphic processing are also 14 bpp. As shown in Fig. 6, the bit length of a random element in a cipher value should not exceed n to ensure that the base is less than N. Otherwise, the data embedded with the RES method cannot be correctly extracted.

3) Compatible With Additive Homomorphic Processing: To test the compatibility with processing in the encrypted domain, two test images "Man" and "SCUT" were used to generate a sum image in the encrypted domain, as shown in Fig. 8.

First, a Paillier cryptosystem was set up by the receiver and the public key was shared with the image sender. In addition, a pixel value m_j in Man was encrypted with (1) by restricting the bit length of random element r less than $(3n/4)$. To embed extra data by applying Scheme I, a pixel value i_j in SCUT was encrypted by using (11), where an $(x+1)$-bit value m_x was used in encryption with $x < (n/4)$. For two pixel values m_j and i_j, their cipher values $g^{m_j} r^N \bmod N^2$ and $g^{i_j} m_x{}^N \bmod N^2$ were multiplied to generate a new cipher value $g^{m_j+i_j}(rm_x)^N \bmod N^2$. The image illustrated in Fig. 8(c) was generated by performing modulo 256 on every cipher value in the sum image. Since both of r and m_x were relatively prime to N, $m_j + i_j$ was correctly decrypted. By dividing $m_j + i_j$ with 2, the average plain-text image was yielded, which is shown in Fig. 8(d). With the values of $m_j + i_j$ and r used in encrypting m_j, the hidden data m_x were correctly extracted from $g^{m_j+i_j}(rm_x)^N \bmod N^2$. Hence, adding two images in the encrypted domain was performed with Scheme I, and the bit values hidden in one encrypted image were correctly retrieved after decrypting the sum image.

With Scheme II, the encrypted image is processed as normal and the desired image is always obtained after decrypting the processed image. In the experiments, two plain-text images "Lena" and "SCUT" were encrypted by adopting random elements r_1 and r_2, respectively. Then, the two encrypted images were added in the encrypted domain so that a sum of encrypted image was obtained. Given that $bl(r_1) + bl(r_2) = (3n/8), 255$ bits were hidden into each cipher value of the sum image by applying the RES method. In addition, extra 4 bits were further hidden into each cipher value by applying the SB method to generate the encrypted image with the hidden data, as illustrated in Fig. 9. The bits hidden by applying the SB method were directly retrieved, while the bits embedded with the RES method were correctly extracted by referring to r_1 and r_2. After dividing every decrypted pixel value by 2, an average image was generated from the encrypted image.

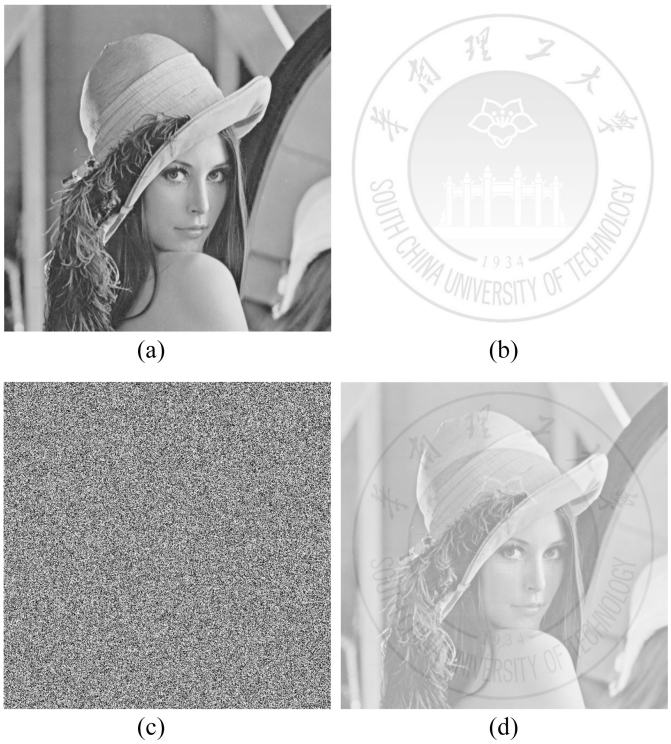


Fig. 9. Average image can be generated after adding two encrypted images in a Paillier cryptosystem, embedding two pieces of messages into the resultant sum image, decrypting the sum image with the hidden data, and dividing every decrypted pixel value by 2. The hidden messages can be extracted before and after image decryption, respectively. (a) Plain-text input image Lena. (b) Plain-text input image SCUT. (c) Encrypted sum image with the hidden data (modulo 256 for illustration). (d) Average image generated from the encrypted sum image by dividing every decrypted pixel value by 2.

B. Security Analysis

Security is an important aspect of data embedding with keys [49]–[51]. Within a Paillier cryptosystem, randomness is introduced by adopting a random element [e.g., r in (1)] in encryption to generate a cipher value. With the RES method, a string of bit values $a_1 a_2 \ldots a_x$ and an extra bit 1, which decimal value is denoted by m_x, are substituted for the random element or appended into the random element. Since the value of N is known in encryption, the bit length (i.e., $x+1$) can be adaptively chosen to ensure that m_x is in $\mathbb{Z}_N^*$. Note that m_x is relatively prime to N when $x+1$ is smaller than the bit lengths of p and q, respectively. Since the modulo N^2 operation is performed in (1) and (15), multiplying $m_x{}^N$ to embed m_x does not undermine the security of encryption. In addition, the bit length of $m_x{}^N$ (i.e., $(x+1)\cdot N$) is longer than that of N^2 (i.e., $2n$) because $N >> n$. Consequently, randomness is always introduced in encryption by applying the RES method for data embedding.

With Scheme I, the data embedded with the RES method can be protected by employing a random element r in encryption, which should be shared by the image sender and receiver. For security enhancement, the data hidden by the SB method can be previously encrypted so that the extracted data need to be decrypted with the correct key. With Scheme II, the encryption is conducted by choosing an odd random element r. Since r^N is multiplied in encryption such as in (1), its bit length (i.e., $bl(r)N$) is much longer than the bit length of N^2 (i.e., $2n$). So the security level of the encryption is not degraded by choosing a relatively smaller r.

C. Computational Complexity Analysis

To embed data into a grayscale image consisting of Y pixels, the computational complexity of applying the RES method is $\mathcal{O}(Y)$ because X bit values can be simultaneously embedded into one cipher pixel value. The computational complexity of applying the VE method in [25] to embed X bit values into one cipher value is $\mathcal{O}(XY)$ because each bit value needs to be embedded at each time. When the SB method in [25] is adopted, the computational complexity is $\mathcal{O}(2^X Y)$ because one out of 2^X possible cipher values needs to be found out to embed X bit values into one cipher pixel value. The computational complexity of data embedding with the scheme in [29] is also $\mathcal{O}(2^X Y)$ because the modified cipher value should fall into one of 2^X subintervals to embed X bit values into one cipher value. In Schemes I and II, the RES method is combined

TABLE III
COMPUTATIONAL COMPLEXITY OF APPLYING THE PROPOSED SCHEMES AND THOSE IN [9], [11], [23], [25], AND [29] TO EMBED X BPP VALUE IN AN IMAGE WITH Y PIXELS

Scheme	Scheme [9]	Scheme [11]	Scheme [23]	VE method [25]	SB method [25]	Scheme [29]	RES method	Scheme I/Scheme II
Encryption scheme	Cipher stream		Paillier cryptosystem (n=1024)					
Embedding capacity (bpp)	≈ 1	≤ 1	< 1	≤ 1015	≤ 14	> 3	≤ 1022	$255+5+5$
Data embedding	$\mathcal{O}(KLY)$+$\mathcal{O}(Y log\sqrt{Y})$	$\mathcal{O}(Y)$	$\mathcal{O}(Y^3)$	$\mathcal{O}(XY)$	$\mathcal{O}(2^X Y)$	$\mathcal{O}(2^X Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(2^{\frac{X}{3}+1}Y)$
Data extraction	$\mathcal{O}(KLY)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(XY)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$

TABLE IV
PERFORMANCE COMPARISONS BETWEEN THE PROPOSED SCHEMES AND THOSE IN [23]–[29]

Reversible data hiding scheme based on Paillier cryptosystem	Preprocessing -free/preserving file size of an encrypted image	Data extraction before or after image decryption	Preserving plain-text values in data embedding	Data extraction after some homomorphic processing	Compatibility with some homomorphic processing	Embedding capacity (for a 1024-bit N)	
						maximum data extraction rate in encrypted domain	maximum data extraction rate after decryption
Combined scheme [23]	×/✓	before or after	×	×	×	close to 1 bpp	about 0.5 bpp
MCG scheme [24]	×/✓	before or after	×	×	×	about $\frac{12}{35}$ of the rate in preprocessing	
Scheme in [27]	×/✓	before or after	×	×	×	12 bpp	≥ 1 bpp
SB method [25]	✓/✓	before decryption	✓	×	partially	14 bpp	0
VE method [25]	✓/✓	after decryption	×	×	partially	0	1015 bpp
Method in [26]	✓/✓	after decryption	×	×	partially	0	1016 bpp
Scheme in [28]	✓/✓	before or after	×	×	partially	14 bpp	1014 bpp
Method in [29]	✓/✓	before decryption	✓	×	partially	> 3 bpp	0
Proposed RES	✓/✓	after decryption	✓	✓	✓	0	1022 bpp
Proposed Scheme I	✓/✓	before or after	✓	✓	✓	6 bpp (before processing) 6 bpp (before decryption)	255 bpp
Proposed Scheme II	✓/✓	before or after	✓	✓	✓	5 bpp (before processing) 5 bpp (before decryption)	255 bpp

with the SB method to embed three parts of data, respectively. Given that the same amount of data is hidden at each stage, the computational complexity for a total hiding rate of X bpp is $\mathcal{O}((2^{(X/3)}+2^{(X/3)}+1)Y)$, which can be simplified to $\mathcal{O}(2^{(X/3)+1}Y)$. In addition, the computational complexity of the schemes in [9], [11], and [23] has been analyzed in the literature (e.g., [9] and [29]), which is summarized in Table III. Note that in the computational complexity of the scheme in [9], K is the number of atoms in the dictionary, and L is the number of nonzero elements in each coefficient vector.

The computational complexity of extracting the hidden data is $\mathcal{O}(Y)$ for the schemes in [23] and [29], the SB method, and the RES method, which is lower than the complexity of data embedding. The reason is that X bit values can be simultaneously and directly extracted from one cipher value with these schemes, respectively. The computational complexity of data extraction with Schemes I and II is also $\mathcal{O}(y)$ because only the RES method and the SB method are adopted. With the scheme in [9], data extraction is also computationally easier because smooth areas have been marked in data embedding and there is no more need to sort them for data extraction. Specifically, the complexity of data extraction is $\mathcal{O}(KLY)$ by removing the second part $\mathcal{O}(Y\log\sqrt{Y})$. As for the scheme in [11] and the VE method, the computational complexity of data extraction is the same as that of data embedding because one-bit value is extracted at each time. From Table III, it can be seen that the proposed RES method has lower computational complexity than the schemes based on the Paillier cryptosystem. Since there are usually thousands of pixels in an image, Schemes I and II have lower computational complexity than the scheme in [23] for $X \leq 3$. Meanwhile, Schemes I and II have lower computational complexity than the scheme in [29] and the SB method for $X > (3/2)$. Despite that the Paillier cryptosystem has much higher computational complexity than encryption with a cipher stream, the RES method has close or even lower computational complexity than the RDH schemes in [9] and [11] for data embedding.

D. Comparisons With Schemes Based on Paillier Cryptosystem

In Table IV, the properties of the proposed schemes are compared with those of the schemes in [23]–[29]. The schemes in [20]–[22] were not included in the comparisons because they are not for the Paillier cryptosystem. In Table IV, the maximum data extraction rate in the encrypted domain represents the amount of data that can be extracted without decryption, while the one after decryption is the amount of bits that can be retrieved after a plain-text image is obtained from the encrypted one.

The combined scheme proposed in [23], the MCG scheme proposed in [24], and the scheme proposed in [27] are similar because a preprocessing is required in applying them. As data extraction can be separated from image

decryption, they are not compatible with homomorphic processing.

No preprocessing is needed with the method in [26], the VE method, the SB method, or the method proposed in [29]. With the SB method and the method in [29], data embedding and extraction are both performed in the encrypted domain, which can be separated from image decryption. With the VE and the histogram shifting method in [26], the embedded data can only be extracted in the plain-text domain so that data extraction cannot be separated from decryption. Moreover, the original image was changed after applying the two methods for data embedding in the encrypted domain. Although homomorphic processing may be performed before applying the methods in [25], [26], and [29] and the scheme proposed in [28], the embedded data will be altered by the subsequent processing made to the encrypted image with the hidden data.

With the RES method, no preprocessing is needed and the plain-text image is preserved after data embedding. The hidden data can only be extracted after image decryption, and the encrypted image may be used as normal after data embedding in encryption or in the encrypted domain. For n=1024, the maximum data extraction rate after decryption was 1022 bpp.

With the proposed Scheme I, data extraction before and after image decryption is enabled in multiple scenarios (i.e., before homomorphic processing, before image decryption, and after decryption). For $n = 1024$, the maximum data extraction rate after decryption may be set to 255 bpp. In that case, the maximum rates for data extraction before being processed and before being decrypted are both 6 bpp.

In Scheme II, an odd random element should be employed in image encryption, which serves as a secret key to extract the data hidden by the RES method. Similar to Scheme I, the embedding capacity of Scheme II consists of three parts, and the maximum hiding rate with the RES method may be set to 255 bpp for $n = 1024$. When the bit length of the random element is no more than $(n/4)$, the maximum rate before being processed and that before being decrypted are both 5 bpp.

The performance comparisons show that the proposed RES method, Schemes I and II are compatible with some processing in the encrypted domain. The desired plain-text image can be obtained by applying homomorphic processing on encrypted images with hidden data. Besides, comparable data hiding rates are obtained with the proposed schemes in different scenarios, respectively.

VI. Conclusion

We have proposed a new preprocessing-free and lossless data hiding method for the Paillier cryptosystem with low computational complexity, namely, RES. A string of bit values can be simultaneously hidden into a cipher value without changing its plain-text value, while data extraction after decryption has been achieved and mathematically proved. It has been analyzed that randomness introduced in encryption can be exploited to carry extra data without degrading the security level. Besides preserving the file size of an encrypted image, the hidden data may be correctly retrieved after some processing has been conducted in the encrypted domain.

To extract the hidden data in multiple scenarios, two schemes have been proposed by adopting the RES and SB method, respectively. Compared with the state-of-the-art schemes, one advantage of the proposed schemes is that data embedding does not interfere with the usage of encrypted images. Subsequently, an encrypted image with hidden data can be processed as normal, and the desired image can be directly obtained after homomorphic processing is applied. Hence, the proposed lossless data hiding schemes are more suitable for the encrypted images to be processed before decryption. One direction of our future work is to study lossless data hiding schemes for other homomorphic cryptosystems such as those in [39]–[42].

Appendix

The following theorems and definitions in number theory (e.g., [52]) are used to prove (13) in Section III-A.

Euler's Theorem: If a is a positive integer and N is an integer relatively prime to a, then

$$a^{\phi(N)} \equiv 1 \pmod{N}$$

where $\phi(N)$ is the number of positive integers between 1 and N that are relatively prime to N.

Corollary 1: If a and N are two relatively prime positive integers (i.e., their greatest common divisor equals to 1, gcd $(a, N) = 1$), there exists a positive integer d satisfying

$$a \times d \equiv 1 \pmod{N}.$$

Then, d is called the modular multiplication inverse of a, denoted by $a^{-1} \bmod N$. For instance, $a^{\phi(N)-1}$ is an inverse of a modulo N because $a \times a^{\phi(N)-1} = a^{\phi(N)} \equiv 1 \pmod{N}$.

Theorem 1: Let $N = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ be the prime-power factorization of the positive integer N. Then

$$\phi(N) = N\left(1 - \frac{1}{p_1}\right) \cdots \left(1 - \frac{1}{p_k}\right).$$

Definition 1: The universal exponent of a positive integer N is a positive integer U so that

$$a^U \equiv 1 \pmod{N}$$

holds for every integer a relatively prime to N. By Euler's theorem, we know that $\phi(N)$ is a universal exponent of N.

Definition 2: The least universal exponent of a positive integer N is denoted as $\lambda(N)$.

Theorem 2: Suppose that N is a positive integer with prime-power factorization, that is, $N = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, where $e_1, e_2, \ldots, e_k > 0$ are positive integers. Then, the minimal universal exponent of N is given by

$$\lambda(N) = \begin{cases} \phi(N), & \text{for } N = p^e, \text{ with } p = 2 \text{ and } e \le 2, \text{ or } p \ge 3 \\ \frac{\phi(N)}{2}, & \text{for } N = 2^e \text{ and } e \ge 3 \\ \operatorname{lcm}\left(\lambda\left(p_1^{e_1}\right), \lambda\left(p_2^{e_2}\right), \ldots, \lambda\left(p_k^{e_k}\right)\right), & \text{otherwise.} \end{cases}$$

Now, we prove that the value m_x embedded into a cipher value c_m in (11) can be extracted with (13), that is

$$m_x = {m_{cx}}^{N^{-1} \bmod \lambda(N)} \bmod N$$

where m_{cx} is calculated by using (12).

Proof: In a Paillier cryptosystem, N is set at the product of two large prime numbers p and q. In *Euler's theorem*, $\phi(N)$ is defined, which can be calculated according to Theorem 1 by $\phi(N) = (p-1)(q-1)$.

When using the RES method, it is required that $m_x \in \mathbb{Z}_N^*$ so that the greatest common divisor $gcd\ (m_x, N) = 1$. By Definitions 1 and 2, we know

$$m_x{}^{\lambda(N)} \bmod N = 1 \tag{20}$$

where $\lambda(N)$ is the least universal exponent of N. Furthermore, $\lambda(N)$ can be calculated according to Theorem 2.

Since $N = pq$ where p and q are two large prime numbers, we know that $\lambda(N)$ is the least common multiple of $\lambda(p)$ and $\lambda(q)$. According to Theorem 1, we know that $\lambda(p) = p-1$ and $\lambda(q) = q-1$ so that $\lambda(N) = lcm\ (p-1, q-1)$, where $\mathrm{lcm}(\cdots)$ represents the least common multiple of a set of positive integers. Hence, $\phi(N)$ is a multiple of $\lambda(N)$. Since $gcd\ (\phi(N), N) = 1$ holds for any Paillier cryptosystem, we know that $gcd\ (\lambda(N), N) = 1$.

According to Corollary 1, there exists a positive integer u satisfying $N \times u \equiv 1 \pmod{\lambda(N)}$, namely, $N \times u = \lambda(N) \times v + 1$, where $v = (N \times u - 1/\lambda(N))$ is a positive integer. So u is an inverse of N modulo $\lambda(N)$, denoted by $u = N^{-1} \bmod \lambda(N)$.

According to (12), i.e., $m_{cx} = m_x^N \bmod N$, we have

$$m_{cx}^u \equiv \left(m_x^N\right)^u \bmod N \equiv m_x^{Nu} \bmod N.$$

$\because Nu = \lambda(N)v + 1$, we have:

$$m_{cx}^u \equiv m_x^{\lambda(N)v+1} \bmod N \equiv \left(m_x^{\lambda(N)}\right)^v \times m_x \bmod N.$$

$\because m_x{}^{\lambda(N)} \bmod N = 1$, i.e., (20), we obtain

$$m_{cx}^u \equiv m_x \bmod N.$$

$\because m_x \in \mathbb{Z}_N^*$, we have

$$m_x = m_{cx}^{N^{-1} \bmod \lambda(N)} \bmod N. \qquad \blacksquare$$

REFERENCES

[1] Y. Q. Shi, X. Li, X. Zhang, H.-T. Wu, and B. Ma, "Reversible data hiding: Advances in the past two decades," *IEEE Access.*, vol. 4, pp. 3210–3237, 2016.

[2] X. Zhang, "Reversible data hiding in encrypted images," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, Apr. 2011.

[3] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 826–832, 2012.

[4] X. Wu and W. Sun, "High-capacity reversible data hiding in encrypted images by prediction error," *Signal Process.*, vol. 104, pp. 387–400, Nov. 2014.

[5] D. Xu, R. Wang, and Y. Q. Shi, "Data hiding in encrypted H.264/AVC video streams by codeword substitution," *IEEE Trans. Inf. Forensics Security*, vol. 9, pp. 596–606, 2014.

[6] F. Huang, J. Huang, and Y.-Q. Shi, "New framework for reversible data hiding in encrypted domain," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 2777–2789, 2016.

[7] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 553–562, 2013.

[8] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, and Y. Y. Tang, "Secure reversible image data hiding over encrypted domain via key modulation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 3, pp. 441–452, Mar. 2016.

[9] X. Cao, L. Du, X. Wei, X. Guoand, and D. Meng, "High capacity reversible data hiding in encrypted images by patch-level sparse representation," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1132–1143, May 2016.

[10] Z. Qian and X. Zhang, "Reversible data hiding in encrypted image with distributed source encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 636–646, Apr. 2016.

[11] P. Puteaux and W. Puech, "An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1670–1681, 2018.

[12] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted JPEG bitstreams," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 6, pp. 1055–1067, Nov./Dec. 2018.

[13] J. He, J. Chen, W. Luo, S. Tang, and J. Huang, "A novel high-capacity reversible data hiding scheme for encrypted JPEG bitstreams," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3501–3515, Dec. 2019.

[14] D. Xiao, F. Li, M. Wang, and H. Zheng, "A novel high-capacity data hiding in encrypted images based on compressive sensing progressive recovery," *IEEE Signal Process. Lett.*, vol. 27, no. 2, pp. 296–300, Feb. 2020.

[15] F. Peng, W. Jiang, Y. Qi, Z.-X. Lin, and M. Long, "Separable robust reversible watermarking in encrypted 2D vector graphics," *IEEE Trans. Circuits Syst. Video Technol*, vol. 30, no. 8, pp. 2391–2405, Aug. 2020.

[16] F. Chen, Y. Yuan, H. He, M. Tian, and H.-M. Tai, "Multi-MSB compression based reversible data hiding scheme in encrypted images," *IEEE Trans. Circuits Syst. Video Technol*, vol. 31, no. 3, pp. 905–916, Mar. 2021.

[17] B. Chen, W. Lu, J. Huang, J. Wen, and Y. Zhou, "Secret sharing based reversible data hiding in encrypted images with multiple data-hiders," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 978–991, Mar./Apr. 2022.

[18] Y. Du, Z. Yin, and X. Zhang, "High capacity lossless data hiding in JPEG bitstream based on general VLC mapping," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1420–1433, Mar./Apr. 2022.

[19] Z. Hua, Y. Wang, S. Yi, Y. Zhou, and X. Jia, "Reversible data hiding scheme in encrypted images using cipher-feedback secret sharing," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jan. 6, 2022, doi: 10.1109/TCSVT.2022.3140974.

[20] M. Li, D. Xiao, Y. Zhang, and H. Nan, "Reversible data hiding in encrypted images using cross division and additive homomorphism," *Signal Process. Image Commun.*, vol. 39, pp. 234–248, Nov. 2015.

[21] Y. Ke, M. Zhang, J. Liu, T. Su, and X. Yang, "A multilevel reversible data hiding scheme in encrypted domain based on LWE," *J. Vis. Commun. Image Represent.*, vol. 54, pp. 133–144, Jul. 2018.

[22] Y. Ke, M.-Q. Zhang, J. Liu, T.-T. Su, and X.-Y. Yang, "Fully homomorphic encryption encapsulated difference expansion for reversible data hiding in encrypted domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2353–2365, Aug. 2020.

[23] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public key cryptography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1622–1631, Sep. 2016.

[24] S. Xiang and X. Luo, "Reversible data hiding in homomorphic encrypted domain by mirroring ciphertext group," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 11, pp. 3099–3110, Nov. 2018.

[25] H.-T. Wu, Y.-M. Cheung, and J. Huang, "Reversible data hiding in paillier cryptosystem," *J. Vis. Commun. Image Represent.*, vol. 40, pp. 765–771, Oct. 2016.

[26] M. Li and Y. Li, "Histogram shifting in encrypted images with public key cryptosystem for reversible data hiding," *Signal Process.*, vol. 130, pp. 190–196, Jan. 2017.

[27] H. T. Wu, Y. M. Cheung, Z. Yang, and S. Tang, "A high-capacity reversible data hiding method for homomorphic encrypted images," *J. Vis. Commun. Image Represent.*, vol. 62, pp. 87–96, Jul. 2019.

[28] H. T. Wu, Y. M. Cheung, Z. Zhuang, and S. Tang, "Reversible data hiding in homomorphic encrypted images without preprocessing," in *Proc. 20th World Conf. Inf. Security Appl.*, vol. 11897, 2019, pp. 141–154.

[29] S. Zheng, Y. Wang, and D. Hu, "Lossless data hiding based on homomorphic cryptosystem," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 692–705, Mar./Apr. 2021.

[30] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, 'Reversible image watermarking using interpolation technique," *IEEE Trans. Inf. Forensics Security*, vol. 5, pp. 187–193, 2010.

[31] H. T. Wu, and J. Huang, "Reversible image watermarking on prediction error by efficient histogram modification," *Signal Process.*, vol. 92, no. 12, pp. 3000–3009, Dec. 2012.

[32] X. Li, W. Zhang, X. Gui, and B. Yang, "Efficient reversible data hiding based on multiple histograms modification," *IEEE Trans. Inf. Forensics Security*, vol. 10, pp. 2016–2027, 2015.

[33] B. Ou and Y. Zhao, "High capacity reversible data hiding based on multiple histogram modification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2329–2342, Aug. 2020.
[34] I. C. Dragoi and D. Coltuc, "Adaptive pairing reversible watermarking," *IEEE Trans. Image Process.*, vol. 25, pp. 2420–2422, 2016.
[35] 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.*, vol. 47, no. 2, pp. 315–326, Feb. 2017.
[36] F. Peng, Z. Lin, X. Zhang, and M. Long, "Reversible data hiding in encrypted 2D vector graphics based on reversible mapping model for real numbers," *IEEE Trans. Inf. Forensics Security*, vol. 14, pp. 2400–2411, 2019.
[37] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*. New York, NY, USA: Academia, 1978, pp. 169–179.
[38] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology (EUROCRYPT)* (Lecture Notes in Computer Science 1592). Heidelberg, Germany: Springer, 1999, pp. 223–238.
[39] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[40] Z. Brakershi and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," *SIAM J. Comput.*, vol. 43, no. 2, pp. 831–871, 2014.
[41] Z. Brakershi, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, 2014.
[42] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based publick key cryptosystem," in *Proc. 3rd Int. Symp. Algorithmic Number Theory*, 1998, pp. 267–288.
[43] M. Barni, T. Kalker, and S. Katzenbeisser, "Inspiring new research in the field of signal processing in the encrypted domain," *IEEE Signal Process. Mag.*, vol. 30, no. 2, p. 16, Mar. 2013.
[44] T. Bianchi, A. Piva, and M. Barni, "Composite signal representation for fast and storage-efficient processing of encrypted signals," *IEEE Trans. Inf. Forensics Security*, vol. 5, pp. 180–187, 2010.
[45] C.-Y. Hsu, C.-S. Lu, and S.-C. Pei, "Image feature extraction in encrypted domain with privacy-preserving SIFT," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4593–4607, Nov. 2012.
[46] "USC-SIPI Image Database." [Online]. Available: http://sipi.usc.edu/database/ (Accessed: Apr. 3, 2022).
[47] A. V. Subramanyam, S. Emmanuel, and M. S. Kankanhalli, "Robust watermarking of compressed and encrypted JPEG2000 images," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 703–716, Jun. 2012.
[48] J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal, "Writing on wet paper," *IEEE Trans. Signal Process.*, vol. 53, no. 10, pp. 3923–3935, Oct. 2005.
[49] Y.-G. Wang, G. Zhu, S. Kwong, and Y.-Q. Shi, "A study on the security levels of spread-spectrum embedding schemes in the WOA framework," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2307–2320, Aug. 2018.
[50] W. Lu, J. Chen, J. Zhang, J. Huang, J. Weng, and Y. Zhou, "Secure halftone image steganography based on feature space and layer embedding," *IEEE Trans. Cybern.*, early access, Oct. 23, 2020, doi: 10.1109/TCYB.2020.3026047.
[51] S. Liu, C. Li, and Q. Hu, "Cryptanalyzing two image encryption algorithms based on a first-order time-delay system," *IEEE Trans. Multimedia*, early access, Sep. 22, 2021, doi: 10.1109/MMUL.2021.3114589.
[52] J. H. Silverman, *A Friendly Introduction to Number Theory*, 4th ed. Upper Saddle River, NJ, USA: Pearson Educ., 2013.

Hao-Tian Wu (Senior Member, IEEE) received the B.E. and M.E. degrees from the Harbin Institute of Technology, Harbin, China, in 2002 and 2004, respectively, and the Ph.D. degree from the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 2007.

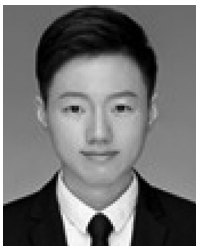
He is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests include reversible data hiding, privacy preservation, homomorphic encryption, and blockchain.

Dr. Wu serves as an Associate Editor for the *EURASIP Journal on Image and Video Processing* and an Invited Reviewer for a number of international journals and conferences, such as IEEE ICME.

Yiu-Ming Cheung (Fellow, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2000.

He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His current research interests include machine learning, pattern recognition, visual computing, and optimization.

Prof. Cheung serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, *Pattern Recognition*, and *Neurocomputing*. He is a Fellow of AAAS, IET, and BCS.

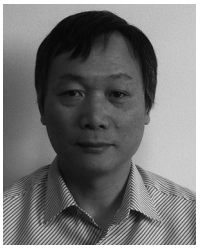


Zhenwei Zhuang received the bachelor's degree from the Guangdong University of Foreign Studies, Guangzhou, China, in 2018, and the Master of Engineering degree from the South China University of Technology, Guangzhou, in 2021.

He is currently with ByteDance Technology Company Ltd., Beijing, China, as an Engineer of Research and Development. His research interests include reversible data hiding in encrypted domain and information hiding in 3-D mesh models.

Lingling Xu received the B.Sc. and M.Sc. degrees in mathematics from Shandong University, Jinan, China, in 2005 and 2008, respectively, and the Ph.D. degree in communication and information system from Sun Yat-sen University, Guangzhou, China, in 2011.

She is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her research interests include cryptography, data security, and privacy preserving in cloud computing and big data.



Jiankun Hu (Senior Member, IEEE) received the B.E. degree from Hunan University, Changsha, China, in 1983, the Ph.D. degree in control engineering from the Harbin Institute of Technology, Harbin, China, in 1993, and the master's by research degree in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2000.

He is a Full Professor with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT, Australia. He has worked with Ruhr University, Bochum, Germany, on the prestigious German Alexander von Humboldt Fellowship from 1995 to 1996, and a Research Fellow with Melbourne University, Melbourne, from 1998 to 1999. His main research interest is in the field of cyber security, including image processing, forensics, and machine learning, where he has published many papers in high-quality conferences and journals, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.

Prof. Hu has served on the editorial board of up to seven international journals, including the top venue IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY as a Senior Area Editor and served as the Security Symposium Chair of IEEE flagship conferences of IEEE ICC and IEEE Globecom. He has obtained ten Australian Research Council (ARC) grants and has served at the prestigious Panel of Mathematics, Information and Computing Sciences, ARC ERA (The Excellence in Research for Australia) Evaluation Committee.