Lossless Data Hiding in NTRU Cryptosystem by Polynomial Encoding and Modulation

Hao-Tian Wu, *Senior Member, IEEE*, Yiu-ming Cheung, *Fellow, IEEE*, Zhihong Tian, *Senior Member, IEEE*, Dingcai Liu, Xiangyang Luo, Jiankun Hu, *Senior Member, IEEE*

***Abstract*—Lossless data hiding in ciphertexts (LDH-CT) is to perform data embedding without changing their plaintexts, which can be used to transmit extra data in the applications of homomorphic encryption at little cost. In this paper, two LDH-CT algorithms named Polynomial Encoding (PE) and Polynomial Modulation (PM) are proposed for the "N-th Degree Truncated Polynomial Ring Unit" (NTRU) scheme, respectively. In the PE algorithm, a polynomial is encoded according to a string of bit values and further used to encrypt a plain-text polynomial. After decrypting the ciphertext, the encoded polynomial can be retrieved so that dozens of bit values can be extracted from it. Moreover, the PE algorithm can be combined with a polynomial partitioning strategy to achieve data extraction before decryption as well. In applying the PM algorithm, no parameter setting of an NTRU cryptosystem is changed while a cipher-text polynomial is generated by selectively sampling a polynomial to match the to-be-hidden value. Furthermore, the data hidden with the PM algorithm can be pre-chosen to be extracted without decryption or after decryption, and in each case up to 10 bit values can be hidden into one cipher-text polynomial. The proposed algorithms and schemes are implemented and compared with several schemes developed for NTRU, BGN, LWE and Paillier encryption. Experimental results and performance evaluations demonstrate the efficacy and superiority of the proposed algorithms and schemes.**

***Index Terms*—Homomorphic encryption, lossless data hiding, polynomial, encoding, modulation.**

I. Introduction

With the development of remote services and cloud computing [1], more and more data are uploaded to the server for storage and processing. For privacy preservation and content protection, the data containing sensitive information are often encrypted. A ciphertext may need to be processed but directly decrypting it will cause privacy infringement. To process a ciphertext without disclosing its plaintext, the notion of "privacy homomorphism" was proposed in [2]. With the homomorphic encryption schemes [3]-[9], data processing in encrypted domain has been reported in [10]-[15].

In past decades, reversible data hiding (RDH) in ciphertexts has been intensively studied to embed an amount of data for annotating, tracing and authenticating content (e.g., [16]-[41]). Although extra data are embedded into a ciphertext, the plaintext should not be permanently changed but should be exactly recovered when needed. To achieve this property, redundancy in ciphertexts or their plaintexts needs to be exploited, denoted by vacating room after encryption (VRAE) and vacating room before encryption (VRBE), respectively. For the VRAE-based algorithm, the ciphertext for data hiding is unnecessarily an encrypted image, while the correlations between pixels in a plain-text image are often leveraged in the VRBE-based algorithm. For instance, some sophisticated algorithms have been designed to spare room in the plain-text image before encryption (e.g., [18], [22], [23], [30], [31], [32]). Alternatively, Puteaux and Puech proposed an efficient prediction-based method in [24] so that the most significant bit (MSB) plane can be almost vacated before encryption. Depending on the specific applications, the hidden data need to be retrieved before or after decrypting the ciphertext. For instance, a separable RDH scheme (e.g., [17], [25], [26], [27], [28]) indicates that data extraction can be performed without decryption, while decryption is required for data extraction with the inseparable algorithms (e.g., [22], [23], [39]).

In view of whether a plaintext is changed after hiding data into it, the methods of RDH in ciphertexts can be classified into two categories. In the first category, both of a ciphertext and its plaintext are changed while the plaintext is supposed to be recovered after the hidden data are retrieved. For instance, a cipher value is modified so that its plain-text value is multiplied with two and then added by a bit value (zero or one). After decrypting the modified cipher value and extracting the bit value from the decrypted plaintext by modulo 2, the original plain-text value can be recovered by division with two. In such a case, it is inconvenient to directly process the ciphertext with hidden data. Otherwise, neither the original plaintext can be exactly recovered nor the hidden data can be correctly extracted. Consequently, applying this type of RDH methods likely limits the usages of ciphertexts.

For homomorphic encryption schemes, lossless data hiding in ciphertexts (LDH-CT) has been recently proposed to hide extra data without changing the file size or plaintext of a ciphertext (e.g., [30], [38], [39]). Hereinafter, a RDH method that maintains the plaintext unchanged after hiding data into a ciphertext is denoted as LDH-CT. As a countermeasure to

Hao-Tian Wu and Dingcai Liu are both with School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: wuht@scut.edu.cn, 202021044277@mail.scut.edu.cn.

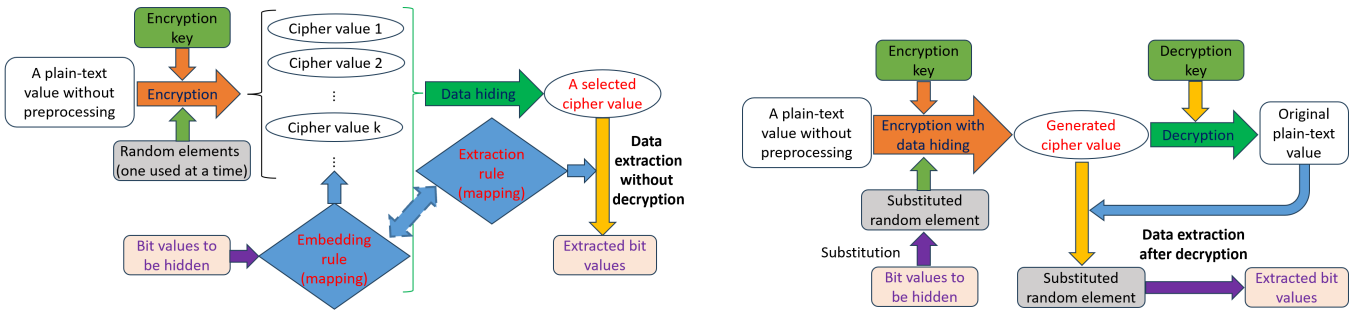
Yiu-ming Cheung is with Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China. E-mail: ymc@comp.hkbu.edu.hk Yiu-ming Cheung is the Corresponding Author.

Zhihong Tian is with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 511370, China. E-mail: tianzhihong@gzhu.edu.cn.

Xiangyang Luo is with Zhengzhou Xinda Institute of Advanced Technology, Zhengzhou 450001, China, and also with Henan Key Laboratory of Cyberspace Situation Awareness, China. E-mail: luoxy_ieu@sina.com.

Jiankun Hu is with School of Engineering and Information Technology, the University of New South Wales, Australian Defence Force Academy, Canberra, ACT 2612, Australia. E-mail: j.hu@adfa.edu.au.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2024.3362592



(a) Lossless data hiding by selecting a proper cipher value

(b) Lossless data hiding by substituting the random element

Fig. 1. Two approaches of lossless data hiding in ciphertexts by exploiting the randomness introduced by homomorphic encryption.

the known plain-text attack, randomness is introduced in the encryption so that one plain-text value corresponds to multiple cipher values. For homomorphic encryption schemes, one or more bit values can be hidden by selecting or generating the cipher values according to designed embedding rules, as shown in Fig. 1. Since the plain-text value is kept unchanged by applying an LDH-CT method, a cipher value containing the hidden data can be used as usual. In addition, an LDH-CT algorithm is called compatible with a type of homomorphic processing if data extraction can still be performed after the ciphertext with the hidden data undergoes the processing.

A. RDH methods for homomorphic encrypted images

For RDH in homomorphic encrypted images, an early method suited for Paillier cryptosystem [4] was proposed by Chen et al. [20]. Specifically, every pixel value is divided into two parts, i.e., the least significant bit (LSB) and the rest bits, which are separately encrypted so that the file size of an encrypted image is doubled. After modifying the LSB part by applying homomorphic multiplication, one bit value is hidden into two adjacent pixels and can be extracted after image decryption. To avoid the increase of file size, the difference expansion technique [42] is adopted in [21] for embedding one bit value into a pair of encrypted pixel values. Since overflows may be caused by applying the difference expansion operation, an encrypted location map needs to be generated and separately sent as side information.

To eliminate the side information, Xiang and Luo proposed another Paillier-based RDH scheme in [31] by vacating selected bit values in a plain-text image. In a group of pixels named mirroring ciphertext group (MCG), a reference pixel is chosen so that data embedding can be carried out by homomorphic multiplication. In particular, the hidden data can be retrieved before image decryption by applying a modular multiplicative inverse operation and referring to a lookup table, while the same data can be extracted after image decryption. In [32], three LSB values of some pixels are reversibly embedded into the rest pixels in a plain-text image for embedding extra bits into the encrypted image. Besides, the self-blinding algorithm proposed in [33] is adopted to embed data that can be extracted without image decryption. Furthermore, several RDH methods have been developed for other homomorphic encryption schemes such as those based on integer modulo (e.g., [28], [29], [36]), the "N-th Degree Truncated Polynomial Ring Unit" (NTRU) (e.g., [40], [41]) and fully homomorphic encryption scheme (e.g., [37]).

The schemes in [20], [21], [28], [29], [31], [32], [36], [37], [40] modify a plain-text image so that they belong to the first category of RDH in ciphertexts. In [30], an early LDH-CT scheme was proposed for Paillier cryptosystem to perform data extraction without decrypting the encrypted image. In particular, several LSB planes of a cipher image are replaced with new values by applying multi-layer wet paper coding technique [43]. Beside the lossless scheme, a reversible scheme was also presented in [30] by conducting histogram shrinkage before image encryption. Similarly, the self-blinding property is utilized in [33] to select a cipher value from those corresponding to the same plain-text value to embed a string of bit values. With the same parameter setting, the hiding rate was increased from less than 1 bpp (bit per pixel) in [30] to about 12 bpps with the method in [33]. In [38], another LDH-CT method is proposed by setting up a mapping between the cipher values and secret bits to be hidden. In addition, an LDH-CT scheme is developed in [34] for the learning with errors (LWE)-based public-key cryptography [9]. In [35], two preprocessing-free schemes using additive and multiplicative homomorphism were proposed for BGN [5] and ELGamal [6] public-key encryptions, respectively.

As shown in Fig.1(a), the data hidden with the schemes in [30], [33], [34], [35], [38] can be directly extracted from the ciphertext given the extraction rule. Thereby, the to-be-hidden data need to be protected by adopting symmetric encryption schemes before being embedded with this kind of LDH-CT methods. In [39], an LDH-CT method named random element substitution (RES) was proposed by using the to-be-hidden bit values to generate a random element in a Paillier cryptosystem. For data extraction, it is critical to retrieve the random element from the cipher value after obtaining the decrypted plaintext. As shown in Fig.1(b), the data hidden with the RES method can only be extracted by the receiver owing the decryption key. Note that the RES method can be combined with the self-blinding algorithm in [33] without affecting each other.

B. RDH schemes for NTRU encryption

For NTRU encryption, two RDH schemes were proposed in [40] and [41], respectively. With the scheme in [40], a plain-text image is partitioned into groups of adjacent pixels. As the

TABLE I
COMPARISON OF HOMOMORPHIC ENCRYPTION SCHEMES

Homomorphic encryption scheme	Parameters		Length of the public key (bit)	Length of the private key (bit)	Homomorphism	Resistant to quantum-computer attacks
RSA [3]	p and q are big primes, $N = p * q$	$\psi = (p-1) * (q-1)$, e: prime to ψ , d: private key	$log_2^N + log_2^e$	log_2^d	Multiplicative	No
Paillier [4]		$g \in \mathbb{Z}^*_{M^2}$: a set of positive integers that are prime to M^2, λ: the least common multiple of p-1 and q-1	$log_2^N + log_2^g$	$log_2^N + log_2^\lambda$	Additive	No
BGN [5]		$g, u \in G$: a cyclic (bilinear) group of order M, $v = u^q$	$log_2^N + log_2^g$ $+log_2^v$	log_2^p	Additive and one-time multiplicative	No
NTRUEncrypt [45]	N: a prime number used to determine the degree of the truncated polynomials in a ring $R = \frac{\mathbb{Z}[x]}{x^N-1}$. p and q: two moduli, $p \ll q$.		$N * log_2^q$	$2N * log_2^p$	Additive and multiplicative	Yes

pixels in one group are encrypted with the same parameters, the differences between a reference cipher value and the other values in the same group are used for histogram shifting. A data hiding rate less than 1 bpp was obtained in [40] and the hidden data can be extracted without image decryption. In [41], the additive homomorphism is utilized by adding the ciphertext of a to-be-hidden bit to that of a pixel value. Similar to the methods in [21], [28], the technique of difference expansion [42] is applied on a plain-text image so that the hidden data can only be extracted after image decryption. To convey useful data in the applications of NTRU encryption, it is desirable to conduct LDH-CT instead of using the RDH schemes in [40] and [41]. However, the LDH-CT schemes developed for other homomorphic cryptosystems cannot be applied in an NTRU cryptosystem where polynomials are used. As the schemes in [40], [41] modify the plain-text image for data embedding, how to hide data into a cipher-text polynomial without changing its plaintext emerges as a challenging issue.

To address the above issues, two LDH-CT algorithms named Polynomial Encoding (PE) and Polynomial Modulation (PM) are proposed for NTRU encryption. In the PE algorithm, a polynomial is encoded according to a string of bit values so that the encoded polynomial complies with the requirements of NTRU encryption. By retrieving the encoded polynomial from the cipher-text polynomial, the bit values hidden in it can be extracted. To achieve data extraction before decryption, the PE algorithm is combined with a polynomial partitioning (PP) strategy to generate a scheme named PE-PP. In applying the PM algorithm, no parameter setting of an NTRU cryptosystem needs to be changed while a cipher-text polynomial is selectively sampled to match the to-be-hidden value. Furthermore, the data hidden with the PM algorithm can be pre-chosen to be extracted without decryption or after decryption. A scheme named PM-hybrid is developed by adopting the PM algorithm to achieve data extraction in different stages.

The proposed algorithms and schemes can be applied in the internet of things systems where NTRU encryption is employed to protect the data being shared. Since the plaintext is unchanged by data hiding, the server can aggregate the shared ciphertexts normally. Extra data can be transmitted to the server or multiple receivers without increasing the file size of a ciphertext. That means the proposed algorithms and schemes can be used to save bandwidth in the case that extra data need to be sent. Besides, a ciphertext can be authenticated after extracting the data newly added in it. Another important application can be privacy-preserving medical data processing. It is well known that medical images, especially 3D medical images, are large in volume, which makes it costly in encryption. With our proposed scheme, medical diagnosis results can be efficiently embedded into the encrypted medial images without increasing their file sizes and then extracted in a confidential way.

C. Our contribution

Our contributions are summarized as follows. Firstly, we propose two NTRU-based LDH-CT algorithms, which can be used to hide data in a cipher-text polynomial without affecting its usages. In particular, the PE algorithm is compatible with additive operations on the cipher-text polynomial. Secondly, much higher embedding capacity can be obtained than the schemes in [40], [41]. Experimental results show that dozens of bit values can be hidden into a cipher-text polynomial by applying the PE algorithm, while up to 10 bit values can be embedded in a cipher-text polynomial by applying the PM algorithm. Thirdly, data extraction in different situations is achieved with the proposed PE-PP and PM-hybrid schemes, respectively. For the first time, data extraction after decryption is enabled by selecting a proper cipher value, which is not included in the two cases illustrated in Fig. 1. Moreover, the security level of an NTRU cryptosystem is not affected by applying the PM algorithm or the PM-hybrid scheme because no parameter setting is changed. The performance evaluation and comparisons with the state-of-the-art schemes developed for Paillier, BGN, LWE, NTRU and homomorphic encryption based on integer modulo in [29], [30], [31], [32], [34], [35], [38], [39], [40] demonstrate the efficacy and superiority of the proposed algorithms and schemes.

The rest part of this paper is organized as follows. The preliminaries of the NTRU encryption is introduced in Section II. In Section III, the PE algorithm is presented with security analysis, where the PP strategy is introduced to generate the PE-PP scheme. In Section IV, the PM algorithm is presented, and then the PM-hybrid scheme is proposed. The experimental results are given in Section V, where the performance of the proposed PE and PM algorithms as well as the PE-PP and PM-hybrid schemes is compared with the state-of-the-art schemes. Finally, we draw a conclusion in Section VI.

II. Preliminaries

In this section, the NTRU encryption scheme is briefly introduced, followed by two properties of homomorphism.

A. NTRU Encryption

The NTRU encryption was first proposed by J. Hoffstein et al. in [7] and regulated in IEEE P1363.1 standard [44] in 2009, which is also called "Number Theory Research Unit". Compared with the partially homomorphic encryption schemes such as [3] and [4], the fully homomorphic encryption schemes support arbitrary computations on ciphertexts. In [8], C. Gentry proposed the first fully homomorphic encryption scheme, which lighted the lattice-based path to homomorphic encryption. In the literature, D. Stehl*é et al.* proposed a revised NTRUEncrypt scheme which can provide higher level of security [45]. In [46], A. L*ó*pez-Alt et al. found the fully homomorphic property in the revised NTRUEncrypt scheme and proposed a multi-key fully homomorphic encryption scheme. The security of NTRU encryption is reduced to the Shortest Vector Problem (SVP) [49] on lattices, which can resist to Shor's algorithm [47] and quantum attacks. As demonstrated in Table I, the efficiency and resistance to quantum-computer attacks [48] makes the NTRU scheme attractive in public-key cryptography, which has been used in applications such as mobile devices and smart cards. Specifically, the NTRU is a ring-based public-key cryptosystem, which is featured with reasonably short and easily created keys, high operation speed and low memory requirements. As an anti-quantum cipher, it consists of two algorithms, NTRUEncrypt [7] for encryption and decryption, and NTRUSign [50] for digital signature.

The NTRUEncrypt system is defined over the truncated polynomial ring, which is denoted by $\mathbf{R} = \frac{\mathbb{Z}[x]}{X^N-1}$, where N is a prime number, $\mathbb{Z}[x]$ denotes the set of polynomials of arbitrary degree with integer coefficients. Since X^N-1 is the modulo polynomial, the set of R contains all the polynomials with degree up to $N-1$. An element $a(x) \in \mathbf{R}$ is a polynomial, which can be represented by $a(x) = a_0 + a_1 \cdot x + \cdots + a_{N-1} \cdot x^{N-1}$ where $a_i \in \mathbb{Z}$ for $i = 0, 1, \cdots, N-1$, so $a(x)$ is usually represented by a vector, i.e.,

$$a(x) = \sum_{i=0}^{N-1} a_i \cdot x^i = [a_0, a_1, ..., a_{N-1}]. \tag{1}$$

The arithmetic operations that can be conducted on R includes addition and multiplication. The addition of two polynomials $[y_0, y_1, ..., y_{N-1}]$ and $[z_0, z_1, ..., z_{N-1}]$ is also a polynomial, i.e., $[y_0 + z_0, y_1 + z_1, ..., y_{N-1} + z_{N-1}]$. By using $\otimes$ to denote the multiplication operation in R, the multiplication of two polynomials $y(x)$ and $z(x)$ denoted by $u(x) = \sum_{k=0}^{N-1} u_k \cdot x^i$ is defined as a cyclic convolution product by

$$u_k = \sum_{i=0}^{k} y_i \cdot z_{k-i} + \sum_{i=k+1}^{N-1} y_i \cdot z_{N+k-i} = \sum_{i+j \equiv k (mod N)} y_i \cdot z_j. \tag{2}$$

TABLE II
Recommended values for six parameters used in NTRU cryptosystem with different security levels

Parameter		N	p	q	d_f	d_g	d_r
Security level	moderate	107	3	64	15	12	5
	high	167	3	128	61	20	18
	highest	503	3	256	216	72	55

Parameter setting: To construct an NTRU cryptosystem [7], six parameters need to be set, which recommended values with respect to security level are given in Table II. Among them, N is a prime number that determines the degree of the ring, while p is a small odd prime used to build the set of plain-text polynomials denoted by $\mathbf{L_m}$ defined by

$$\mathbf{L_m} = \left\{ e(x) \in \mathbf{R} : -\frac{p-1}{2} \leq e_i \leq \frac{p-1}{2} \right\}. \tag{3}$$

The parameter q is a positive integer much greater than p, which is used for modular operations during encryption and decryption. Note p is often set to 3 so that $e_i \in \{-1, 0, 1\}$ for simplicity and $gcd(p, q) = 1$.

Another three parameters (d_f, d_g, d_r) need to be set in constructing an NTRU cryptosystem. Among them, d_f is a positive integer used to build a set of polynomials denoted by $\mathbf{L_f}$. A polynomial $f(x)$ in $\mathbf{L_f}$ has d_f coefficients equal to 1, $d_f - 1$ coefficients equal to -1, and the rest coefficients equal to 0. The d_g is a positive integer used to build a set of polynomials denoted by $\mathbf{L_g}$. A polynomial $g(x)$ in $\mathbf{L_g}$ has d_g coefficients equal to 1, d_g coefficients equal to -1, and the rest coefficients equal to 0. The d_r is a positive integer used to build a set of polynomials denoted by $\mathbf{L_r}$. A polynomial $r(x)$ in $\mathbf{L_r}$ has d_r coefficients equal to 1, d_r coefficients equal to -1, and the rest coefficients equal to 0.

The **Key generation** procedure is as follows.

(1) Randomly sample a polynomial $f(x) \in \mathbf{L_f}$ according to the parameter d_f so that the inverse of $f(x)$ modulo p and the inverse of $f(x)$ modulo q exist. The two inverses of $f(x)$ are respectively denoted by $F_p(x)$ and $F_q(x)$, i.e.,

$$F_p(x) \otimes f(x) \equiv 1 \ (mod\ p), \quad F_q(x) \otimes f(x) \equiv 1 \ (mod\ q). \tag{4}$$

Otherwise, re-sample another polynomial until Eq. (4) holds.

(2) Calculate $F_p(x)$ and $F_q(x)$ according to the obtained $f(x)$, which are also polynomials in $\mathbf{R}$.

(3) Randomly sample a polynomial $g(x) \in \mathbf{L_g}$. Then calculate a polynomial denoted by $h(x)$ by

$$h(x) = F_q(x) \otimes g(x) (mod\ q) \tag{5}$$

After the above steps, the public key $K_{pub} = h(x)$ and the private key $K_{pri} = \{f(x),\ F_p(x)\}$ are generated. Note that

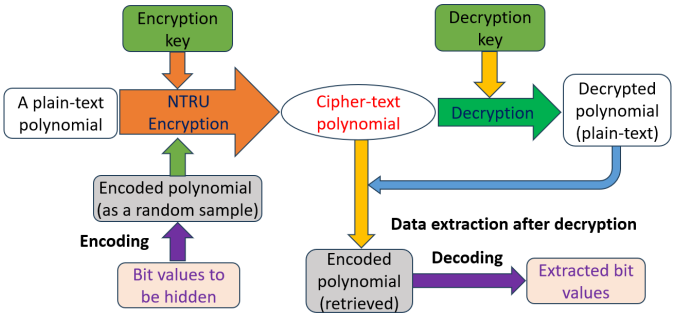


Fig. 2. Procedure of applying the Polynomial Encoding algorithm for lossless data hiding with NTRU encryption.

the operation of a polynomial modulo integer q is to let all coefficients in the polynomial modulo q.

Encryption and Decryption: Given a plain-text polynomial $m(x) \in \mathbf{L_m}$, a cipher-text polynomial $e(x)$ is generated after the encryption consisting of the following two steps.

(1) Randomly sample a polynomial $r(x) \in \mathbf{L_r}$.

(2) Obtain a cipher-text polynomial $e(x)$ by

$$e(x) = Enc(m(x)) = p \cdot r(x) \otimes h(x) + m(x) \ (mod\ q). \quad (6)$$

Given a cipher-text polynomial $e(x)$ and the private key $\{f(x),\ F_p(x)\}$, a plain-text polynomial $m(x)$ is decrypted by first calculating

$$\begin{aligned} a(x) &= f(x) \otimes e(x) \ (mod\ q) \\ &= (p \cdot r(x) \otimes g(x) + f(x) \otimes m(x)) \ (mod\ q), \end{aligned} \quad (7)$$

then shifting $a(x)$ to $[-\frac{q}{2},\ \frac{q}{2}]$, and finally calculating

$$\begin{aligned} b(x) &= F_p(x) \otimes a(x) \ (mod\ p) \\ &= F_p(x) \otimes f(x) \otimes m(x) \ (mod\ p) \\ &= m(x). \end{aligned} \quad (8)$$

B. Homomorphism in an NTRU Cryptosystem

In [45], D. Stehlé *et al.* proposed a revised NTRUEncrypt scheme which has additive and multiplicative homomorphism. Let $e_1(x)$ and $e_2(x)$ respectively be the ciphertexts of the plaintexts $m_1(x)$ and $m_2(x)$ with the same private key $f_d(x)$. The **additive homomorphism** of the revised NTRUEncrypt scheme implies

$$\begin{cases} e_{add}(x) = e_1(x) + e_2(x) \ (mod\ q), \\ Dec(e_{add}(x)) = f_d(x) \otimes (e_1(x) + e_2(x))(mod\ p) \\ \quad = f_d(x) \otimes e_1(x)(mod\ p) + f_d(x) \otimes e_2(x)(mod\ p) \\ \quad = Dec(e_1(x)) + Dec(e_1(x)) = m_1(x) + m_2(x), \end{cases}$$

while the **multiplicative homomorphism** implies

$$\begin{cases} e_{mul}(x) = e_1(x) \otimes e_2(x)(mod\ q), \\ Dec(e_{mul}(x)) = f_d(x) \otimes f_d(x) \otimes (e_1(x) \otimes e_2(x))(mod\ p) \\ \quad = (f_d(x) \otimes e_1(x)(mod\ p)) \otimes (f_d(x) \otimes e_2(x)(mod\ p)) \\ \quad = Dec(e_1(x)) \otimes Dec(e_2(x)) = m_1(x) \otimes m_2(x). \end{cases}$$

Algorithm 1: Encoding of a polynomial

Input: A string of bit values $\mathbf{b} = \{b_1 b_2 \dots b_x\}$, parameter d_r of an NTRU cryptosystem.
Output: $\mathbf{r} \in \mathbf{L_r}$.
function Encoding($\mathbf{b}$)
    $num_{zero} = 0;\ num_{nz} = 0;\ i = 0;$
    while $num_{nz} < 2d_r$ **or** $num_{zero} < N - 2d_r$, **do**
        if $b_i = 1$ **then** $r_i = 1\ or\ -1;\ num_{nz}++;$
        otherwise $r_i = 0;\ num_{zero}++;$
        endif
        $i++;$
        if $num_{nz} = 2d_r$ **then**
            for $i < N$ **then** $r_i = 0;\ i++;$
            end for
        else if $num_{zero} = N - 2d_r$ **then**
            for $i < N$ **then** $r_i = 1\ or\ -1;\ i++;$
            end for
        endif
    return $\mathbf{r} = \{r_0, r_1, r_2, \cdots, r_{N-1}\}$
end function

III. Polynomial Encoding Algorithm

This section presents the details of the PE algorithm, including encoding of a polynomial for data embedding and decoding of the encoded polynomial for data extraction. In addition, the security of applying the PE algorithm is analyzed. The procedure of applying the PE algorithm is illustrated in Fig. 2.

A. Encoding and Decoding of a Polynomial

In the RES method [39], a string of bit values are used to constitute a random element in Paillier cryptosystem. In an NTRU cryptosystem, there is a polynomial randomly sampled from $\mathbf{L_r}$, i.e., $r(x)$ in Eq.(6). Now we discuss how to use a string of bit values $\mathbf{b} = \{b_1 b_2 \dots b_x\}$ to generate a polynomial $r(x)$=$[r_0, r_1, ..., r_{N-1}]$. As stated in Section II-A, there are d_r 1s, d_r $-$1s and $N-2d_r$ 0s in the coefficients of any polynomial in $\mathbf{L_r}$. Consequently, there are $N - 2d_r$ zero coefficients in $[r_0, r_1, ..., r_{N-1}]$ and an encoding procedure is proposed in Algorithm 1 to generate $r(x)$ according to $\mathbf{b}$.

Without loss of generality, a non-zero coefficient in $r(x)$ is used to represent a bit value 1, while a zero coefficient is used to represent a bit value 0. For correct decoding, it is required that the length of bit values to be hidden is up to $N - 1$. The actually hidden bit number is adaptively determined and no less than the minimum of $2d_r$ and $N - 2d_r$. That is, if d_r 1s and d_r $-$1s have been assigned to the coefficients of $r(x)$, then the rest coefficients are assigned with 0s. Vice versa, if $N-2d_r$ 0s have been assigned to the coefficients of $r(x)$, then the rest coefficients are assigned with 1s or $-$1s. In addition, totally d_r 1s and d_r $-$1s are assigned to the coefficients of $r(x)$. After executing Algorithm 1, there are d_r 1s, d_r $-$1s and $N - 2d_r$ 0s in the coefficients of $r(x)$, which is used in encrypting a plain-text value in Eq. (6).

Given an encoded polynomial, the bit values hidden in it are extracted as described in Algorithm 2. According to Algorithm

1, a bit value 1 is extracted from a non-zero coefficient and a bit value 0 is obtained from a zero coefficient in the encoded polynomial. The number of extracted bit values is automatically determined with the proposed encoding algorithm. When either the non-zero or zero coefficients are used up, i.e., a bit value 1 is extracted from the last non-zero coefficient or a bit value 0 is extracted from the last zero coefficient in an encoded polynomial, the decoding process is completed.

For instance, a string of bit values $\{0,1,1,0,1,0,0,1,0,1,1,1,0,1,0,1,0,1,0\}$ are used to encode a random polynomial $r(x)$ in an NTRU cryptosytem with N=107 and d_r=5. As there are 107 coefficients to be assigned in $r(x)$, the first, fourth, sixth, seventh, ninth, thirteenth, fifteenth, seventeenth coefficients are assigned with 0s so that eight 0s are embedded. Meanwhile, the second, third, fifth, eighth, tenth, eleventh, twelfth, fourteenth, sixteenth and eighteenth coefficients are assigned with five 1s and five -1s so that ten 1s are embedded. As d_r=5, no more 1 or -1 can be assigned to $r(x)$ so that the rest coefficients are assigned with 0s. Accordingly, ten 1s can be extracted from the non-zero coefficients in $r(x)$. Meanwhile, eight 0s can be extracted from the first, fourth, sixth, seventh, ninth, thirteenth, fifteenth, seventeenth coefficients in $r(x)$. Then no more bit values can be extracted because there are no more non-zero coefficients left.

B. Encryption and Decryption of A plaintext

Since a polynomial $r(x)$ can be generated according to a string of bit values $\mathbf{b}$ to be hidden, encrypting a plain-text polynomial with $r(x)$ has no difference from that in Eq.(6). As discussed in Section II-A, a plain-text polynomial $m(x)$ is encrypted with a public key $h(x)$, parameters p and q, and an encoded polynomial $r(x)$ to generate a cipher polynomial $e(x)$. With the private key $\{(f(x), F_p(x)\}$, $m(x)$ can be decrypted from $e(x)$ by applying Eq.(7) and Eq.(8). Now we discuss how to retrieve the encoded polynomial $r(x)$ from $e(x)$ to extract the hidden bit values. If a receiver has the private key $f(x)$ to decrypt a plaintext $m(x)$ from the cipher polynomial $e(x)$, a new polynomial can be calculated by

$$\begin{aligned} &p^{-1} \cdot [e(x) - m(x)] \otimes f(x) \ (mod\ q) \\ &= p^{-1} \cdot p \cdot r(x) \otimes h(x) \otimes f(x) \ (mod\ q), \\ &= p^{-1} \cdot p \cdot r(x) \otimes F_q(x) \otimes f(x) \otimes g(x) \ (mod\ q), \\ &= r(x) \otimes g(x) \ (mod\ q). \end{aligned} \tag{9}$$

where $p \cdot p^{-1} \equiv 1 \ (mod\ q)$ and $F_q(x) \otimes f(x) \equiv 1 \ (mod\ q)$.

To obtain $r(x)$ from Eq.(9) by

$$r(x) \otimes g(x) \otimes G_q(x) \equiv r(x) \ (mod\ q),$$

a polynomial $G_q(x)$ needs to be found such that $G_q(x) \otimes g(x) \equiv 1 \ (mod\ q)$ holds. In an NTRU algorithm, a polynomial $g(x)$ has the same number of coefficients valued at 1 and -1, indicating that $g(1) = \sum_{i=0}^{N-1} g_i \cdot 1 = 0$ so that there is no $G_q(x) \otimes g(x) \equiv 1 \ (mod\ q)$. To make it possible to retrieve $r(x)$ from $r(x) \otimes g(x)$, the constraint on $g(x)$ is slightly modified so that a polynomial in $\mathbf{L_g}$ has d_g coefficients equal to 1, $d_g - 1$ coefficients equal to -1, and the rest equal to 0. In this way, $g(1) \neq 0$ so that $G_q(x)$ can be calculated from $g(x)$.

Algorithm 2: Decoding of an encoded polynomial

Input: $\mathbf{r} = \{r_0, r_1, r_2, \cdots, r_{N-1}\} \in \mathbf{L_r}$.
Output: A string of bit values $\mathbf{b} = \{b_1 b_2 \dots b_x\}$ with length automatically determined.
function Decoding(**r**)
    $num_{zero} = 0$; $num_{nz} = 0$; $i = 0$;
    while $num_{nz} < 2d_r$ **or** $num_{zero} < N - 2d_r$, **do**
        if $r_i = 1$ *or* -1 **then** $b_i = 1$; $num_{nz}++$;
        otherwise $b_i = 0$; $num_{zero}++$;
        endif
        $i++$;
    return $\mathbf{b} = \{b_1 b_2 \dots b_{num_{nz}+num_{zero}}\}$
end function

Meanwhile, $g(x)$ can be obtained by multiplying $f(x)$ in the private key with the public key $h(x)$, i.e.,

$$f(x) \otimes h(x) = f(x) \otimes F_q(x) \otimes g(x) \ (mod\ q) = g(x).$$

According to Eq.(6), Eq.(7) and Eq.(8), the existence of the inverse of $g(x)$ modulo q does not affect the encryption and decryption of a plain-text value. After decryption, $r(x)$ can be retrieved from $e(x)$ and further decoded to obtain the bit values hidden in it, so the hidden data can be extracted after decryption given that the encoded polynomial $r(x)$ is intact.

C. Security Analysis

As the polynomial $g(x)$ is made invertible so that its modular multiplicative inverse $G_q(x)$ exists, the public key $h(x) = F_q(x) \otimes g(x) \ (mod\ q)$ is also invertible because the polynomial $F_q(x)$ is invertible. Making the public key invertible does not leak information about the plain-text polynomial $m(x)$. This is due to that the encoded polynomial $r(x)$ is non-invertible so that thee polynomial $r(x) \otimes h(x)$ is non-invertible. Consequently, it is very hard to remove the polynomial $p \cdot r(x) \otimes h(x)$ from the cipher-text polynomial $e(x)$ in Eq. (6) to obtain $m(x)$ without using the private key.

With $G_q(x)$, i.e., the modular multiplicative inverse of $g(x)$, $F_q(x)$ can be obtained from the public key $h(x)$ by

$$G_q(x) \otimes h(x) = G_q(x) \otimes g(x) \otimes F_q(x) \ (mod\ q) = F_q(x).$$

According to Eq.(4), the private key $\{f(x),\ F_p(x)\}$ can be calculated with $F_q(x)$. In fact, the polynomials $g(x)$ and $G_q(x)$ can only be calculated by knowing the private key $\{f(x),\ F_p(x)\}$ because $g(x)$ is randomly chosen from $\mathbf{L_g}$ to generate the public key $h(x)$ in Eq. (5) and will not be used anymore. Thus, obtaining $g(x)$ from $h(x)$ inevitably needs to know $F_q(x)$ and the private key.

D. Polynomial Partitioning Strategy

As the PE algorithm can be applied to hide confidential data to be extracted after decryption, a polynomial partitioning strategy is introduced for data extraction without decryption. In an NTRU cryptosystem, both a plaintext and a ciphertext are represented by a polynomial with N coefficients. With the values suggested in Table I, it can be seen that redundancy

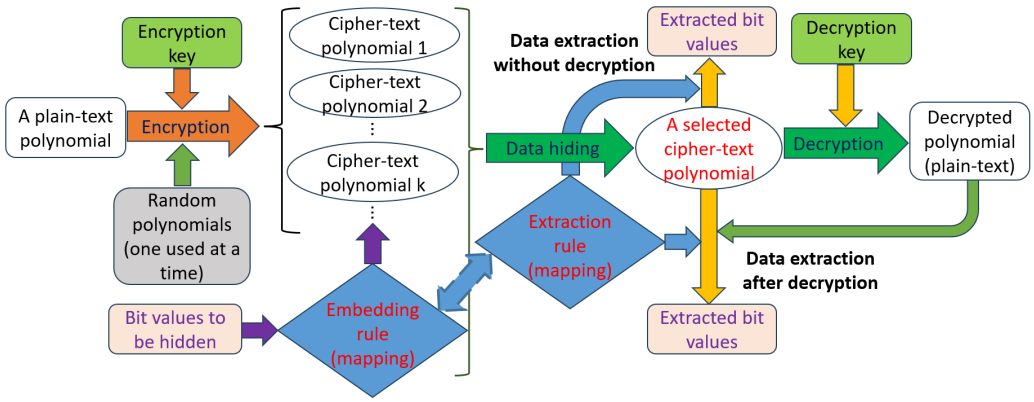


Fig. 3. Procedure of applying the Polynomial Modulation algorithm for lossless data hiding with NTRU encryption.

is introduced in data representation because there are N coefficients needing to be specified to constitute a polynomial. Due to the redundancy, quite a few of coefficients are zeroes so that it is possible to represent a plaintext or a ciphertext with less coefficients, whereby a polynomial partitioning (PP) strategy is proposed to divide the coefficients into two parts. The first part of coefficients are used to represent a plain-text value, while the rest coefficients are vacated in the plain-text domain to carry extra data in the encrypted domain.

Given a plain-text polynomial $m(x) = [m_0, m_1, ..., m_{N-1}]$ where $m_{N-k} = 0$ for $k \in \{1, 2, \cdots, y\}$ and $y < N$, a cipher-text polynomial $e(x)$ can be obtained by sampling a random polynomial $r(x)$ and using the public key $h(x)$ in encryption, as shown in Eq. (6). For data hiding, the last y coefficients in the cipher-text polynomial $e(x)$ is modified according to a string of bit values $\{b_1, b_2, \cdots, b_y\}$ to generate another polynomial $e'(x)$ by

$$e'_k = \begin{cases} e_k, & for \quad 0 \le k < N - y \\ e_k, & if \ N - y \le k < N, \ e_k \ mod \ 2 = b_{N-k} \\ e_k + 1, & if \ N - y \le k < N, \ e_k \ mod \ 2 \ne b_{N-k} \end{cases}$$

In this way, a polynomial $u(x) = [u_0, u_1, ..., u_{N-1}]$ where

$$u_k = \begin{cases} 0, & for \ 0 \le k < N - y \\ e'_k - e_k, & for \ N - y \le k < N \end{cases}$$

is actually added to $e(x)$ to generate $e'(x)$ so that we have

$$e'(x) = p \cdot r(x) \otimes h(x) + m(x) + u(x) \ (mod \ q). \qquad (10)$$

As a result, $m(x)+u(x)$ can be decrypted from $e'(x)$ with the private key $\{(f(x), F_p(x)\}$. Since $m_{N-k} = 0$ for $1 \le k \le y$ and $u_j = 0$ for $0 \le j < N - y$, the plaintext $m(x)$ can be separated from $u(x)$ by knowing the value of y.

From $e'(x)$, a string of y bit values $\{b_1, b_2, \cdots, b_y\}$ can be extracted without decryption by

$$b_k = e'_{N-k} \ mod \ 2, \ \ for \ 1 \le k \le y.$$

Thus, the hidden data can be easily extracted from a ciphertext by knowing the value of y, which is an integer between 1 and N. Besides extracting the bit values hidden with the PP strategy, $r(x) \otimes g(x)$ can be retrieved from $e'(x)$ as discussed in Section III-B to retrieve $r(x)$ given that $g(x)$ is invertible. Therefore, the PP strategy can be combined with the PE algorithm to hide bit values to be extracted before decryption and those to be extracted after decryption. The combination is named **PE-PP scheme**, in which the coefficients in a plain-text polynomial are partitioned for data extraction without decryption and data extraction after decryption. In the next section, another LDH-CT algorithm named polynomial modulation will be further proposed, which can also be used to achieve data extraction in different stages.

IV. Polynomial Modulation Algorithm

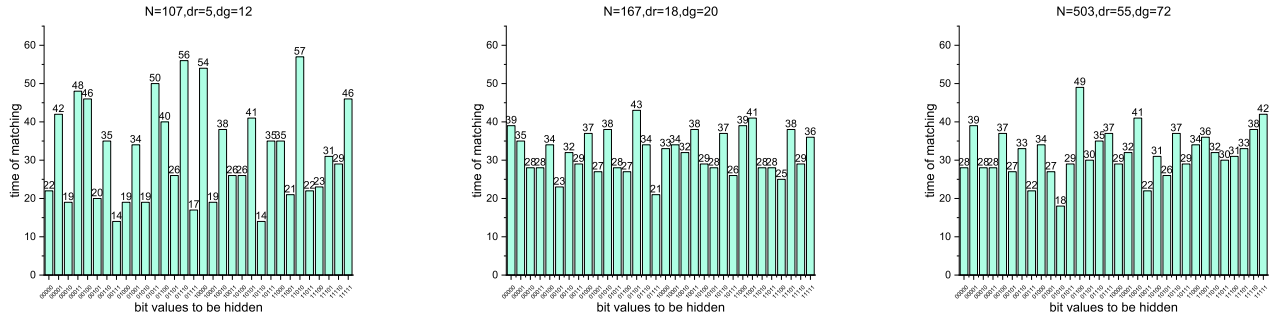
In this section, the polynomial modulation algorithm is presented for NTRU encryption without making the public key invertible such as in the PE algorithm or reserving part of coefficients in the PP strategy. The procedure of applying the PE algorithm is illustrated in Fig. 3. Different from the PE algorithm, the data hidden with the PM algorithm can be pre-chosen to be extracted without decryption or after decryption.

A. Data Hiding by Polynomial Modulation

As shown in Eq. (9), a polynomial $r(x) \otimes g(x)$ can only be obtained after decrypting the plain-text polynomial $m(x)$ with the private key $\{f(x), \ F_p(x)\}$. In the PE algorithm, $g(x)$ is invertible so that $r(x)$ can be retrieved to extract the bit values hidden in it. To perform data hiding without interfering with encryption, all properties and parameter setting of an NTRU are kept unchanged in applying the PM algorithm. That means a polynomial in $\mathbf{L_g}$ still has d_g coefficients equal to 1, d_g coefficients equal to -1 and $N - 2d_g$ coefficients equal to 0, while a polynomial $g(x)$ is randomly sampled from $\mathbf{L_g}$ in key generation. Instead of being generated according to the to-be-hidden values such as in the PE algorithm, $r(x)$ is randomly sampled from $\mathbf{L_r}$ but not arbitrarily. In particular, the sampled $r(x)$ is used to modulate the cipher polynomial $e(x)$ so that the desired information can be extracted from it. That is, $r(x)$ is randomly sampled in trial until the coefficients in $e(x)$ exactly match the bit values to be hidden.

To hide a string of bit values $\mathbf{b} = \{b_1 b_2 \dots b_k\}$, all coefficients in $e(x)$ modulo $k+1$ are calculated and the parity of the frequencies are used for data hiding. For instance, there are six possible values $\{0, 1, 2, 3, 4, 5\}$ after all coefficients in

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2024.3362592



(a) (b) (c)

Fig. 4. The time of hiding every five-bit value after randomly sampling $r(x)$ 1024 times. Note that the hidden bit values are represented by the parity of the frequencies of 1, 2, 3, 4, 5 after all coefficients in a cipher-text polynomial modulo 6 with the parameter settings: (a) $N = 107$, $d_r = 5$, $d_g = 12$; (b) $N = 167$, $d_r = 18$, $d_g = 20$; (c) $N = 503$, $d_r = 55$, $d_g = 72$.

$e(x)$ modulo 6 and their frequencies are counted, which are respectively denoted by $f_0, f_1, f_2, f_3, f_4, f_5$. Without loss of generality, a bit value zero is represented by an even f_i for $i \in \{1, 2, 3, 4, 5\}$ while an odd number f_i stands for a bit value one. The reason of using the parity of $k+1$ frequencies to embed k bits is due to that the total number of coefficients (i.e., N) is odd. As a result, there must be an odd number of odd frequencies so that it is impossible to hide $k+1$ zeros into the parity of $k+1$ frequencies. Nevertheless, it is still possible to embed k bit values into the $k+1$ frequencies by using one frequency to compensate the parity of N. In the example, $r(x)$ is randomly sampled to update $e(x)$ until the parity of f_1, f_2, f_3, f_4, f_5 exactly matches the five-bit value to be hidden. Given a cipher-text polynomial $e(x)$, the modulo operation is conducted to calculate the parity while there is no need to modify $e(x)$ itself.

To show the feasibility of data hiding by using the PM algorithm, the bit values matching the cipher-text polynomials generated with different parameter settings of NTRU encryption were counted. For a cipher-text polynomial $e(x)$, all coefficients modulo 6 were calculated to find the five-bit value that is matched. Then the time of embedding every five-bit value was counted after randomly sampling $r(x)$ 1024 times with the different parameter settings recommended in Table II, respectively. As shown in Fig. 4, every five-bit value occurred dozens of times under each parameter setting of NTRU encryption, which indicates that data hiding with the proposed PM algorithm is feasible in practice. It can be seen that the distributions in Fig. 4(b) and Fig. 4(c) are more uniform than the one shown in Fig. 4(a), indicating that an NTRU cryptosystem with a larger parameter N is more secure.

The procedure of data embedding and that of data extraction are given in Algorithm 3 and Algorithm 4, respectively. In Algorithm 4, the hidden bit values can be obtained after encrypting the to-be-hidden data, which are extracted from the coefficients of a cipher-text polynomial without decryption.

Algorithm 3: Data embedding with PM algorithm

1: **Input**: Plain-text polynomial $m(x)$, k bit values to be hidden, the public key $h(x)$ shared by the receiver.
2: **Output**: Cipher-text polynomial $e(x)$.
3: **Data embedding:**
4: Randomly sample $r(x)$ in $\mathbf{L_r}$ and encrypt $m(x)$ with the public key $h(x)$ to obtain a cipher-text polynomial $e(x)$;
5: Modulo all coefficients in $e(x)$ with $k+1$ and count the frequencies of $\{0, 1, \cdots, k\}$, respectively;
6: **while** the parity of the frequencies of $\{1, \cdots, k\}$ does not match the k bit values to be hidden, **do**
7: Sample another $r(x)$ in $\mathbf{L_r}$ and update $e(x)$;
8: Modulo all coefficients in $e(x)$ with $k+1$ and count the frequencies of $\{0, 1, \cdots, k\}$;
9: **if** there is no match after a certain number of trials,
10: **then** decrease the value of k by one and restart the process of **data embedding**.
11: **endif**
12: Output the cipher-text polynomial $e(x)$ obtained before the modulo operation.

Algorithm 4: Data extraction with the PM algorithm (without decryption)

1: **Input**: Cipher-text polynomial $e(x)$, modulo divisor k+1
2: **Output**: k bit values extracted from $e(x)$.
3: **Data extraction:**
4: Modulo all coefficients in $e(x)$ with $k+1$ and count the frequencies of $\{1, \cdots, k\}$;
5: Obtain k bit values according to the parity of the k frequencies;
6: Output the k bit values extracted from $e(x)$.

B. Data Extraction After Decryption

In some cases, data extraction after decryption is preferred to transmit confidential messages, such as diagnosis made by a doctor on a medical image of a patient. Before being transmitted over Internet, the medical image is encrypted for privacy protection by using a public key provided by the patient while the diagnosis information may be hidden into the encrypted medical image. Instead of being extracted before decryption, the hidden diagnosis information should be

Algorithm 5: Data extraction with PM algorithm (after decryption)

1: **Input**: Cipher-text polynomial $e(x)$, private and public key pair, modulo divisor $k+1$.
2: **Output**: Plain-text polynomial $m(x)$, k bit values extracted from $e(x)-m(x)$.
3: **Data extraction:**
4: Decrypting $m(x)$ from $e(x)$ with the private key;
5: Calculating $r(x)\otimes h(x)$ by $e(x)-m(x)$;
6: Modulo all coefficients in $e(x)-m(x)$ with $k+1$ and count the frequencies of $\{1,\cdots,k\}$;
7: Obtain k bit values according to the parity of the k frequencies;
8: Output the k bit values and the decrypted $m(x)$.

extracted after decryption by the patient owing the private key. In such a case, the PM algorithm can also be applied to hide bit values to be extracted after decryption. Instead of modulating the cipher-text polynomial $e(x)$ as described in Algorithm 3, the polynomial $e(x)-m(x)$, i.e., $r(x)\otimes h(x)$, is modulated by randomly sampling $r(x)$ to match the to-be-hidden bit values. As the polynomial $r(x)$ is randomly sampled, $r(x)\otimes h(x)$ will not be disclosed until $m(x)$ is decrypted from $e(x)$ by using the private key. Consequently, the procedure of data extraction after decryption is given in Algorithm 5, which is quite different from the one in Algorithm 4.

C. PM-hybrid Scheme

To achieve data extraction in different situations, a portion of plain-text polynomials are encrypted by modulating the cipher-text polynomial $e(x)$ to embed the to-be-hidden data while the other polynomials are encrypted by modulating $e(x)-m(x)$ for data embedding. Hereinafter, applying the PM algorithm in the hybrid way is named **PM-hybrid** scheme. In applying the PM-hybrid scheme, the indices of those polynomials encrypted by modulating $e(x)-m(x)$ can be pre-chosen with a secret key. Consequently, only the authorized receiver knowing the secret key can extract the bit values hidden in the coefficients of $e(x)-m(x)$ after decrypting the cipher-text polynomial $e(x)$ to obtain $m(x)$.

V. Experimental Results and Evaluation

To evaluate the performances of the proposed algorithms and schemes, ten test images were used in the experiments for comparison with the schemes in [29]-[40]. All of the test images were converted to grayscale with 512×512 pixels. With the parameters listed in Table I, an NTRU cryptosystem was built by setting p to 3 so that a plain-text coefficient value falls into $\{-1,0,1\}$ according to Eq. (3). Note that an 8-bit pixel value was decomposed into eight bit values so that each bit value was assigned to a coefficient in a plain-text polynomial, while the coefficient value -1 may be used to represent the integer 2, which can be obtained after modulo p. The programs of image encryption/decryption, data embedding and extraction were implemented with Python programming language (3.7 version) and Pycharm compiler on a 64-bit PC with 16G RAM and Windows 10 operating system.

TABLE III
Date hiding rate of proposed PE algorithm and PE-PP scheme

N	d_r	$2d_r$	N-$2d_r$	PE algorithm (bits per polynomial)	PE-PP scheme (reserving N-8 coefficients in a plaintext)
107	5	10	97	20.0	20.1
	8	16	91	32.0	32.0
	10	20	87	40.0	40.0
	30	60	47	93.0	93.8
167	18	36	131	71.9	70.9
	30	60	107	120.0	120.0
	32	64	103	128.0	128.1
	50	100	67	113.9	114.2
503	55	110	393	220.1	220.4
	90	180	323	359.9	360.2
	110	220	383	440.3	440.0
	160	320	183	364.6	365.8

In the following, the PE and PM algorithms are firstly evaluated and compared, including data hiding rate, security and compatibility with homomorphic processing. Then the computational complexity of the two algorithms is analyzed and compared with representative RDH schemes in encrypted domain (e.g., [23], [24], [30], [38], [39], [40]). Finally, the proposed PE and PM algorithms, as well as the PE-PP and PM-hybrid schemes, are compared with various RDH schemes developed for Paillier, BGN, LWE, NTRU and homomorphic encryption based on integer modulo, including [29], [30], [31], [32], [34], [35], [38], [39], [40].

A. Performance of PE and PM Algorithms

1) Data Hiding Rate: With the PE algorithm, the data hiding rate is subject to the bit values to be hidden as well as the values of d_r and N. In the experiments, the bit values to be hidden were randomly generated, which contained the same or close numbers of 0s and 1s. After 2^{10} runs, the average number of bits hidden in a cipher-text polynomial was close to the minimum of $4d_r$ and $2(N-2d_r)$ bits, as shown in Table III with different parameter settings. Averagely, there is a half chance to embed a bit value in a non-zero coefficient or a zero coefficient. As there are $2d_r$ non-zero coefficients and $N-2d_r$ zero coefficients in a random polynomial $r(x)$, the actual hiding rate is about the double of the minimum. The theoretical limit of embedding capacity is to embed N-bit value into a cipher-text polynomial consisting of N coefficients because one coefficient can carry at most one-bit value.

The rate for data extraction after decryption with the PE-PP scheme is also listed, which was very close to that of the PE algorithm. This is because the PP strategy does not interfere with applying the PE algorithm but exploits the redundancy in data representation within an NTRU cryptosystem. Only the first 8 coefficients in a plain-text polynomial were used to represent a 8-bit pixel value while the rest coefficients were vacated for data hiding as described in Section III-D.

The data hiding rate with the PM algorithm is not so high as that of the PE algorithm. As shown in Table IV, at most 10 bit values were hidden in a cipher-text polynomial for N=107, in which case the coefficient values modulo 11 were counted to embed ten bit values at the same time. When N was set to

TABLE IV
DATE HIDING RATE OF PM ALGORITHM AND PM-HYBRID SCHEME

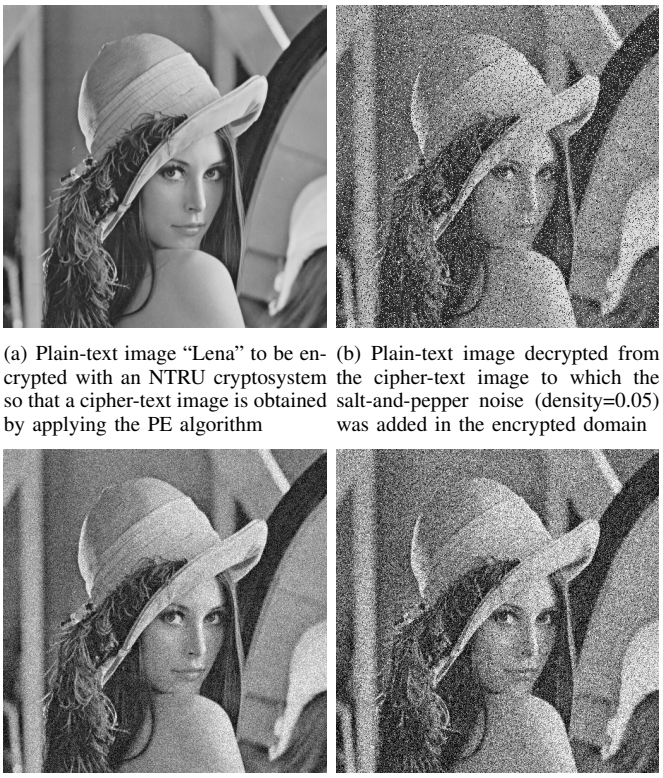
N	d_r	PM algorithm (bits per polynomial)	PM-hybrid scheme (bits per polynomial)
107	5	10	10
	8	10	10
	10	10	10
	30	10	10
167	18	12	12
	30	10	10
	32	10	10
	50	10	10
503	55	10	10
	90	10	10
	110	10	10
	160	10	10

503, at most 10 bit values were also hidden in one cipher-text polynomial, while at most 12 bit values were hidden into one cipher-text polynomial for N=167. The hiding rate slightly changes with the parameter N because the modulo operation is performed on all coefficients. Different from the PE algorithm, increasing the parameter d_r does not increase the hiding rate with the PM algorithm. Although increasing d_r makes the coefficient values in a cipher-text polynomial $e(x)$ more scattered, the maximum data hiding rate with the PM algorithm was stable because all coefficients in a cipher-text polynomial after modulo operation are concentrated. The theoretical limit of embedding capacity with the PM algorithm depends on the space of $p \cdot r(x) \otimes h(x)$ so that a set of cipher-text polynomials can be generated from the same plain-text polynomial. The data hiding rate with the PM-hybrid scheme was the same as that of the PM algorithm.

As less than one bit is hidden into a cipher-text polynomial in [40] and [41], much higher data hiding rates are achieved with the PM algorithm and PM-hybrid scheme, respectively. The improvement is due to that the redundancy introduced by NTRU encryption instead of the redundancy in the plain-text image is exploited by applying the proposed PM algorithm.

2) Security: Security is an important aspect of data hiding and encryption schemes. To apply the PE algorithm, the polynomial $g(x)$ employed in key generation is made invertible. Consequently, the public key $h(x)$ is invertible, which is the multiplication of an invertible polynomial $F_q(x)$ and $g(x)$ as shown in Eq. (5). The impact of making $g(x)$ and $h(x)$ invertible has been analyzed in Section III-C and no obvious vulnerability is found. Furthermore, the security level of an NTRU cryptosystem is intact for applying the PM algorithm because the parameter setting is unchanged by data embedding. In applying the PM-hybrid scheme, the indices of those polynomials from which the hidden data should be extracted after decryption need to be protected by using cryptographic algorithms and a secret key.

3) Compatibility with Homomorphic Processing: With the PE algorithm, the hidden data can be extracted after decryption if the encoded polynomial $r(x)$ is unchanged. Specifically, adding another polynomial to the cipher-text polynomial (such as adding $u(x)$ to $e(x)$ in Eq. (10)) does not change $r(x)$. To verify the compatibility with addition operations on the cipher-text, a cipher-text image with hidden data was processing by adding the salt-and-pepper noise (density=0.05), the Gaussian noise (variance=0.01) and the Gaussian noise (variance=0.05), as shown in Fig. 5. Then the hidden data were correctly extracted after decrypting the cipher-text images processed in the encrypted domain, respectively. It can be seen that the PE algorithm is compatible to the addition operations conducted on ciphertexts in the NTRU-based encrypted domain. With the PM algorithm, the hidden data can hardly be extracted if the coefficients in the cipher-text polynomial $e(x)$ are modified, while data extraction after decryption cannot be correctly performed if the coefficients in the polynomial $e(x) - m(x)$ are changed. Thus, the PE algorithm is suitable when additive operations need to be conducted on the ciphertext. Meanwhile, it is easier to implement the PM algorithm to convey useful data without decrypting the plaintext, such as by a web administrator in the cloud computing environment.



(a) Plain-text image "Lena" to be encrypted with an NTRU cryptosystem so that a cipher-text image is obtained by applying the PE algorithm

(b) Plain-text image decrypted from the cipher-text image to which the salt-and-pepper noise (density=0.05) was added in the encrypted domain

(c) Plain-text image decrypted from the cipher-text image to which the Gaussian noise (variance=0.01) was added in the encrypted domain

(d) Plain-text image decrypted from the cipher-text image to which the Gaussian noise (variance=0.05) was added in the encrypted domain

Fig. 5. A plain-text grayscale image and those decrypted from the cipher-text images to which different noises have been respectively added. The experimental results show that the data hidden in encrypting the plain-text image is unchanged after adding different noises to the cipher-text image, respectively.

B. Computational Complexity

In the following, the computational complexity of applying the PE and PM algorithms is analyzed and compared with the representative schemes for RDH in encrypted domain (e.g., [23], [24], [30], [33], [38]). To embed X bit values by encoding a random polynomial $r(x)$ as shown in Algorithm

TABLE V
COMPUTATIONAL COMPLEXITY OF APPLYING THE PROPOSED SCHEMES AND THOSE IN [23], [24], [30], [38], [39], [40] TO EMBED X BITS PER PIXEL VALUE IN AN IMAGE WITH Y PIXELS

Scheme	Scheme [23]	Scheme [24]	Scheme [30]	Scheme [38]	RES method [39]	Scheme [40]	PE algorithm	PM algorithm
Encryption scheme	Cipher stream		Paillier encryption (a 1024-bit N)			NTRU encryption (N=503)		
Embedding capacity	≈ 1 bpp	≤ 1 bpp	< 1 bpp	> 3 bpp	≤ 1022 bpp	0.8 bpp	220 bpp	10 bpp
Data embedding	$\mathcal{O}(KLY)$+$\mathcal{O}(Ylog\sqrt{Y})$	$\mathcal{O}(Y)$	$\mathcal{O}(Y^3)$	$\mathcal{O}(2^XY)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y^2)$	$\mathcal{O}(NY)$	$\mathcal{O}(2^XY)$
Data extraction	$\mathcal{O}(KLY)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y)$	$\mathcal{O}(Y^2)$	$\mathcal{O}(NY)$	$\mathcal{O}(XY)$

1, every of the N coefficients in $r(x)$ needs to be enumerated and the encoded polynomial is used in encrypting a plain-text polynomial. The computational complexity of applying the PE algorithm on a grayscale image consisting of Y pixels is $\mathcal{O}(NY)$, where X bit values are embedded into one encoded polynomial consisting of N coefficients to generate a cipher-text polynomial. The computational complexity of data embedding with the scheme in [40] is $\mathcal{O}(Y^2)$ because the difference of a reference pixel value and a neighboring pixel value needs to be calculated and the histogram of difference values needs to be calculated for shifting. The computational complexity of applying the PM algorithm is as follows. To embed X bit values by sampling a polynomial $r(x)$ to modulate the cipher-text polynomial as described in Algorithm 3, the computational complexity is $\mathcal{O}(2^XY)$ because at most 2^X permutations of X bit values need to be enumerated for data embedding.

The computational complexity of the scheme in [38] is also $\mathcal{O}(2^XY)$ because one out of 2^X possible cipher values needs to be found to embed X bit values into one cipher value. To embed data into a grayscale image consisting of Y pixels, the computational complexity of applying the RES method [39] is $\mathcal{O}(Y)$ because X bit values simultaneously constitute the random element to generate one cipher pixel value. Note that the parameter N in a Paillier cryptosystem has no relationship with the parameter N in an NTRU cryptosystem. A cipher value in a Paillier cryptosystem has $2N$ bits, while a cipher value in an NTRU has N coefficients. In addition, the computational complexity of the schemes in [23], [24], [30] has been reported in [23], [38]) and summarized in Table V. As for the computational complexity of the scheme [23], K is the number of atoms in the dictionary while L is the number of nonzero elements in each coefficient vector.

The computational complexity of data extraction with the schemes in [30], [38] and the RES method is $\mathcal{O}(Y)$, which is lower than that of data embedding with these schemes. The reason is that X bit values can be simultaneously and directly extracted from one cipher value with these schemes. The complexity of data extraction with the PE algorithm is $\mathcal{O}(NY)$ because every coefficient in the retrieved polynomial needs to be enumerated. The computational complexity of data extraction with the scheme in [40] is $\mathcal{O}(Y^2)$ because the histogram of difference values needs to be calculated. The complexity of data extraction with the PM algorithm is $\mathcal{O}(XY)$ because the frequencies of $1, 2, \cdots, X$ need to be counted. With the scheme in [23], the complexity of data extraction is approximated to $\mathcal{O}(KLY)$ because there is no more need to sort the smooth areas for data extraction. The complexity of data extraction with [24] is the same as data embedding because one bit value is extracted at each time.

C. Comparisons with Schemes for Homomorphic Encryption

In Table VI, the proposed algorithms and schemes are compared with the schemes in [29], [30], [31], [32], [34], [35], [38], [39], [40] for LWE, Paillier, BGN, NTRU and homomorphic encryption based on integer modulo, respectively. The data extraction rate in the encrypted domain represents the amount of bit values that can be extracted without decryption, while the one after decryption is the amount of bit values that can be retrieved after a plain-text image is decrypted.

The scheme proposed in [29] is for RDH in encrypted images by using a rhombus pattern prediction in a plain-text image. A third party may embed data into an encrypted image, which is obtained from a modified plain-text image after additively homomorphic encryption based on integer modulo. The hidden data can only be extracted after direct image decryption and vulnerable to processing made to the encrypted image. The combined scheme proposed in [30], the MCG scheme [31] and the scheme in [32] are similar because a preprocessing is required in applying them, respectively. Although data extraction can be separately performed without image decryption, they are not compatible with homomorphic processing. Besides, the hidden data can hardly be correctly extracted after an encrypted image is processed.

No preprocessing is required to apply the method proposed in [38] for Paillier encryption and the scheme proposed in [35] for BGN encryption, with which both data embedding and extraction are conducted in the encrypted domain. Although resistance to cropping can be achieved by duplicate embedding in the ciphertext, the hidden data are easily altered by other homomorphic processing conducted on the ciphertext. With the RES method in [39], no preprocessing is required and the plain-text value is unchanged after data hiding. The hidden data can only be extracted after decryption, while the encrypted image with hidden data may be used as normal. It has been shown that the data hidden with the RES method can be correctly extracted after applying the self-blinding method in [33] on the ciphertext. For a 1024-bit parameter N, the maximum data extraction rate after decryption is 1022 bpp.

The scheme in [34] was developed for the LWE-based cryptography, with which a multilevel data hiding can be performed by the secret key owner. Similar to the schemes

TABLE VI
PERFORMANCE COMPARISONS BETWEEN THE PROPOSED SCHEMES AND THOSE IN [29], [30], [31], [32], [34], [35], [38], [39], [40]

Reversible/lossless data hiding scheme for various homomorphic encryption	Preprocessing-free/ preserving the file size of an encrypted image	Data extraction without decryption or after decryption	Preserving plain-text content after data hiding	Processing that can be conducted on the ciphertext without changing the hidden data	Embedding capacity (a 1024-bit N for Paillier encryption, N=503 for NTRU)	
					extraction rate in encrypted domain	extraction rate after decryption
Scheme in [29]	×/✓	after decryption	×	×	0	0.498 bpp
Combined scheme in [30]	×/✓	both	×	×	close to 1 bpp	about 0.5 bpp
MCG scheme [31]	×/✓	both	×	×	about $\frac{12}{35}$ of that in preprocessing	
Scheme in [32]	×/✓	both	×	×	12 bpp	$\geq$ 1 bpp
Scheme in [38]	✓/✓	without decryption	✓	cropping	$>$ 3 bpp	0
RES Method [39]	✓/✓	after decryption	✓	self-blinding algorithm [33]	0	1022 bpp
Scheme in [34] for LWE	✓/✓	without decryption	✓	cropping	about 614 bpp	0
Scheme in [35] for BGN	✓/✓	without decryption	✓	cropping	$\geq$ 1 bpp	0
Scheme in [40] for NTRU	×/✓	both	×	×	0.8 bpp	0.8 bpp
Proposed PE Algorithm	✓/✓	after decryption	✓	additive operations	0	220 bpp
Proposed PE-PP Scheme	×/✓	both	✓	cropping	$\leq$495 bpp	220 bpp
Proposed PM Algorithm	✓/✓	without decryption	✓	cropping	10 bpp	0
Proposed PM-hybrid Scheme	✓/✓	both	✓	cropping	10 bpp	10 bpp

in [30] and [38], there is no preprocessing needed and the file size of a cipher value is preserved by applying the multilevel data hiding scheme. The plain-text image is unchanged by data hiding, and a hiding rate about 614 bpp can be obtained. The hidden data can be extracted without decryption, while resistance to cropping can be achieved by duplicate embedding.

The RDH scheme in [40] is for NTRU encryption by modifying the original image before data embedding. The cipher-text image with hidden data can hardly be processed. Otherwise, neither the original nor the desired plain-text image can be obtained after decryption. Similar to the MCG scheme proposed in [31] for Paillier encryption, data extraction is separable from image decryption with the scheme in [40]. No preprocessing is required on the original plain-text polynomial to apply the proposed PE or PM algorithm, nor the file size of a ciphertext is changed. With the PE algorithm, about 220 bits can be hidden into a cipher-text polynomial for an NTRU cryptosystem with N=503, and the hidden data can be correctly extracted after another polynomial is added to the cipher-text polynomial. With the PM algorithm, about 10 bits can be hidden into a cipher polynomial, which may be extracted without decryption or after decryption. The data hidden with the PM algorithm are vulnerable to operations that change the cipher-text polynomial.

By applying the PM-hybrid scheme, data extraction without decryption and after decryption can be both achieved. In applying the PE-PP scheme, the pre-chosen (e.g., eight, sixteen or any number no more than N-8) coefficients in a plain-text polynomial are vacated. As discussed in Section III-D, the plain-text pixel values are not changed by data hiding with the PP strategy. In the experiments, 495 coefficients in a polynomial were used for data hiding so that the same number of bit values were extracted from a cipher-text polynomial before decryption. Similar to the schemes in [34], [35], [38], [39], the data hidden with the proposed algorithms and schemes can be resistant to cropping by duplicate embedding in the ciphertext.

VI. CONCLUSION

In this paper, we have proposed two preprocessing-free algorithms for lossless data hiding with NTRU encryption, namely polynomial encoding (PE) and polynomial modulation (PM). In particular, the plaintext is not changed by data embedding with either algorithm while the usages of the ciphertext are not affected. With the PM algorithm, the security level of a NTRU cryptosystem is intact while the hidden data can be pre-chosen to be extracted without decryption or after decryption. With the PE algorithm, much higher data hiding rate can be achieved, while data extraction can be correctly performed after conducting additive operations on the cipher-text polynomial. The PE and PM algorithms can be applied wherever NTRU encryption is adopted for privacy preservation and security enhancement such as in internet of things systems.

A polynomial partitioning strategy has been proposed and combined with the PE algorithm to generate the PE-PP scheme. Meanwhile, the PM-hybrid scheme has been developed without changing parameter setting of an NTRU cryptosystem. Compared with the state-of-the-art schemes, one advantage of the PE-PP and PM-hybrid schemes is that data hiding does not interfere with the usages of the ciphertext. The performance comparisons demonstrate the advantages and superiority of the proposed schemes, which are suitable for the applications of high-secure NTRU encryption to convey extra data to multiple receivers. Our future work is to study lossless data hiding schemes for other post-quantum cryptography such as ring learning with errors [51].

REFERENCES

[1] M. Haghighat, S. Zonouz, M. Abdel-Mottaleb, "Security and Privacy in Cloud Computing: Vision, Trends, and Challenges," *IEEE Transactions on Cloud Computing*, Vol. 2, No. 2, pp. 30-38, 2015.

[2] R. L. Rivest, L. Adleman, M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, pp. 169-179, 1978.

[3] R. L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, pp. 120-126, 1978.

[4] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Proc. Advances in Cryptology - EUROCRYPT'99, LNCS*, Vol. 1592, pp. 223-238, 1999.

[5] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," *Proc. Theory of cryptography conference*, pp. 325-341, 2005.

[6] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, Vol. 31, No. 4, pp. 469-472, 1985.

[7] J. Hoffstein, J. Pipher, J. H. Silverman, "NTRU: a ring-based public key cryptosystem," *Proc. the 3rd International Symposium on Algorithmic Number Theory*, pp. 267-288, 1998.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Proc. the 41st ACM Symposium on Theory of Computing*, pp. 169-178, 2009.

[9] Z. Brakershi, V. Vaikuntanathan, "Efficient fully homomorphic encryption from (Standard) LWE," *SIAM Journal on Computing*, Vol. 43, No. 2, pp. 831-871, 2014.

[10] M. Barni, T. Kalker, S. Katzenbeisser, "Inspiring new research in the field of signal processing in the encrypted domain," *IEEE Signal Processing Magazine*, Vol. 30, No. 2, pp. 16-16, 2013.

[11] T. Bianchi, A. Piva, M. Barni, "On the implementation of the discrete fourier transform in the encrypted domain," *IEEE Transactions on Information Forensics and Security*, Vol. 4, No. 1, pp. 86-97, 2009.

[12] T. Bianchi, A. Piva, M. Barni, "Composite signal representation for fast and storage-efficient processing of encrypted signals," *IEEE Transactions on Information Forensics and Security*, Vol. 5, No. 1, pp. 180-187, 2010.

[13] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Transactions on Information Forensics and Security*, Vol. 6, No. 2, pp. 452-468, 2011.

[14] C. Y. Hsu, C. S. Lu, S. C. Pei, "Image feature extraction in encrypted domain with privacy-preserving SIFT," *IEEE Transactions on Image Processing*, Vol. 21, No. 11, pp. 4593-4607, 2012.

[15] P. Zheng, J. Huang, "Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain," *IEEE Transactions on Image Processing*, Vol. 22, No. 6, pp. 2455-2468, 2013.

[16] X. Zhang, "Reversible data hiding in encrypted images," *IEEE Signal Processing Letters*, Vol. 18, No. 4, pp. 255-258, 2011.

[17] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, Vol. 16, No. 5, pp. 826-832, 2012.

[18] K. Ma, W. Zhang, W. Sun, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Transactions on Information Forensics and Security*, Vol. 8, No. 3, pp. 553-562, 2013.

[19] X. Wu, W. Sun, "High-capacity reversible data hiding in encrypted images by prediction error," *Signal Processing*, Vol. 104, pp. 387-400, 2014.

[20] Y. C. Chen, C. W. Shiu and G. Horng, "Encrypted signal-based reversible data hiding with public key cryptosystem," *Journal of Visual Communication and Image Representation*, Vol. 25, No.5, pp. 1164-1170, 2014.

[21] C. W. Shiu, Y. C. Chen and W. Hong, "Encrypted image-based reversible data hiding with public key cryptography from difference expansion," *Signal Processing: Image Communication*, Vol. 39, pp. 226-233, 2015.

[22] 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 Transactions on Circuits and Systems Video Technology*, Vol. 26, No. 3, pp. 441-452, Mar. 2016.

[23] X. Cao, L. Du, X. Wei, D. Meng, "High capacity reversible data hiding in encrypted images by patch-level sparse representation," *IEEE Transactions on Cybernetics*, Vol. 46, No. 5, pp. 1132-1143, 2016.

[24] P. Puteaux, W. Puech, "An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images," *IEEE Transactions on Information Forensics and Security*, Vol. 13, No. 7, pp. 1670-1681, 2018.

[25] S. Yi and Y. Zhou, "Separable and Reversible Data Hiding in Encrypted Images Using Parametric Binary Tree Labeling," *IEEE Transactions on Circuits and Systems Video Technology*, Vol. 21, No. 1, pp. 51-64, 2019.

[26] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted JPEG bitstreams," *IEEE Transactions on Dependable and Secure Computing*, Vol. 15, No. 6, pp. 1055-1067, 2018.

[27] 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 Transactions on Information Forensics and Security*, Vol. 14, No. 9, pp. 2400-2411, 2019.

[28] R. Anushiadevi, and R. Amirtharajan, "Separable reversible data hiding in an encrypted image using the adjacency pixel difference histogram," *Journal of Information Security and Applications*, Vol. 71, No. 103407, 2023.

[29] R. Anushiadevi, and R. Amirtharajan, "Design and development of reversible data hiding- homomorphic encryption & rhombus pattern prediction approach," *Multimedia Applications and Tools*, 2023. DOI: 10.1007/s11042-023-15455-1

[30] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public key cryptography," *IEEE Transactions on Circuits and Systems Video Technology*, Vol. 26, No. 9, pp. 1622-1631, 2016.

[31] S. Xiang, X. Luo, "Reversible data hiding in homomorphic encrypted domain by mirroring ciphertext group," *IEEE Transactions on Circuits and Systems Video Technology*, Vol. 28, No. 11, pp. 3099-3110, 2018.

[32] H. T. Wu, Y. M. Cheung, Z. Yang, S. Tang, "A high-capacity reversible data hiding method for homomorphic encrypted images," *Journal of Visual Communication and Image Representation*, Vol. 62, pp. 87-96, 2019.

[33] H. T. Wu, Y. M. Cheung, and J. Huang, "Reversible data hiding in Paillier Cryptosystem," *Journal of Visual Communication and Image Representation*, Vol. 40, pp. 765-771, Oct. 2016.

[34] Y. Ke, M. Zhang, J. Liu, T. Su, X. Yang, "A multilevel reversible data hiding scheme in encrypted domain based on LWE," *Journal of Visual Communication and Image Representation*, Vol. 54, pp. 133-144, 2018.

[35] B. Chen, X. Wu, W. Lu and H. Ren, "Reversible data hiding in encrypted images with additive and multiplicative public-key homomorphism," *Signal Processing*, Vol. 164, pp. 48-57, 2019.

[36] 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, 2015.

[37] 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, 2020.

[38] S. Zheng, Y. Wang, D. Hu, "Lossless Data Hiding Based on Homomorphic Cryptosystem," *IEEE Transactions on Dependable and Secure Computing*, Vol. 18, No. 2, pp. 692-705, 2021.

[39] H.-T. Wu, Y.-M. Cheung, Z. Zhuang, L. Xu, and J. Hu, "Lossless data hiding in encrypted images compatible with homomorphic processing," *IEEE Transactions on Cybernetics*, Vol. 53, No. 6, pp. 3688-3701, 2023.

[40] N. Zhou, M.Q. Zhang, H. Wang, Y. Ke, and and F.Q. Di, 'Separable reversible data hiding scheme in homomorphic encrypted domain based on ntru," *IEEE Access*, Vol. 8, pp. 81412-81424, 2020.

[41] N. Zhou, M.Q. Zhang, H.Q. Tang, H.N. Zhou,Y.ke, and and F.Q. Di, 'Reversible data hiding algorithm in encrypted domain based on NTRU," *Science Technology and Engineering (in Chinese)*, Vol.20, No. 32, pp. 13285-13294, 2020.

[42] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems Video Technology*, Vol. 13, No. 8, pp. 890-896, 2003.

[43] J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal, "Writing on Wet Paper," *IEEE Transactions on Signal Processing*, Vol. 53, No. 10, pp. 3923-3935, 2005.

[44] W. G. of the C/MM Committee, "IEEE P1363.1 standard specification for public key cryptographic techniques based on hard problems over lattices," 2009.

[45] D. Stehé , R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," *Proc. Annual international conference on the theory and applications of cryptographic techniques*, pp.27-47, 2011.

[46] A. López-Alt, E. Tromer, V. Vaikuntanathan, 'On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," *Proc. the forty-fourth annual ACM symposium on Theory of computing*, pp.1219-1234, 2012.

[47] P. W. Shor, 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, Vol.41, No. 2, pp. 303-332, 1999.

[48] D. J. Bernstein , T. Lang, 'Post-quantum cryptography," *Nature*, Vol. 549, No. 7671, pp. 188-194, 2017.

[49] M. Ajtai, 'Generating hard instances of lattice problems," *Proc. the twenty-eighth annual ACM symposium on Theory of computing*, pp. 99-108, 1996.

[50] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSIGN: Digital signatures using the NTRU lattice," *Proc. Cryptographers track at the RSA conference*, pp. 122-140, 2003.

[51] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," *Proc. EUROCRYPT 2010, Lecture Notes in Computer Science*, Vol. 6110, pp. 1-23, 2010.