Fully Homomorphic Encryption Encapsulated Difference Expansion for Reversible Data Hiding in Encrypted Domain

Yan Ke, Min-Qing Zhang, Jia Liu, Ting-Ting Su, and Xiao-Yuan Yang

Abstract—This paper proposes a fully homomorphic encryption encapsulated difference expansion (FHEE-DE) scheme for reversible data hiding in encrypted domain (RDH-ED). The homomorphic circuits and ciphertext operations are elaborated. Key-switching and bootstrapping techniques are introduced to control the ciphertext extension and decryption failure of homomorphic encryption. A key-switching based least-significant-bit (KS-LSB) data hiding has been designed to realize data extraction directly from the encrypted domain without the private key. In application, the user first encrypts the plaintext and uploads ciphertext to the server. The server embeds additional data into the ciphertext by performing FHEE-DE data hiding and KS-LSB data hiding. Additional data can be extracted directly from the marked ciphertext by the server without the private key. The user owns the private key and can decrypt the marked ciphertext to obtain the marked plaintext. Then additional data or plaintext can be obtained from the marked plaintext by using the standard DE extraction or recovery. The server could also implement FHEE-DE recovery or extraction on the marked ciphertext to return the ciphertext of original plaintext or additional data to the user. Experimental results demonstrate that the embedding capacity and reversibility of the proposed scheme are superior to existing RDH-ED methods, and fully separability is achieved without reducing the security of encryption.

Index Terms—Information security, reversible data hiding in encrypted domain, difference expansion, public key cryptography, fully homomorphic encryption.

I. INTRODUCTION

Reversible data hiding in encrypted domain (RDH-ED) is an information hiding technique that aims to not only accurately embed and extract the additional messages in the ciphertext, but also restore the original plaintext losslessly [1], [2]. RDH-ED is useful in some distortion intolerable applications, such as ciphertext management or retrieval in the cloud, ciphertext watermarking for medical or military use [2]. With the increasing demand for information security and the development of the encrypted signal processing techniques, RDH-ED has been an issue of great attention in the field of privacy protection and ciphertext processing.

From the viewpoint of the cryptosystem that RDH-ED methods are based on, existing RDH-ED methods could be classified into two categories: Symmetric encryption based RDH-ED [1], [3]–[14], and public key encryption based RDH-ED. Symmetric cryptography that has been introduced into RDH-ED includes stream encryption [1], [3]–[6], [13], advanced encryption standard (AES) [7], [8], and RC4 encryption [9].

According to the methods of utilizing the redundancy in the cover for data hiding, symmetric encryption based RDH-ED methods were classified into two categories [1], [2]: “vacating room before encryption (VRBE)” [1], [7], [8], [10], [13] and “vacating room after encryption (VRAE)” [3]–[6]. The room, namely the redundancy in the cover, is vacated for reversible data hiding. The first RDH-ED method was proposed by Zhang for encrypted images [3], and then [4], [5] enhanced its capacity. Qian et al. proposed a similar method to embed data in an encrypted JPEG bit stream [6]. AES was introduced in [7] to encrypt the cover image. Each block containing n pixels could carry one bit data. The embedding rate (ER) is 1/n bits per pixel (bpp). Then difference prediction was introduced before encryption in [8], and AES was used to encrypt pixels except the embedding ones, thus resulting in a better embedding capacity (EC) and reversibility. However, it needed decryption first before data extraction in the above RDH-ED methods, which restricted the practicability in practical applications. The separable RDH-ED was proposed in [11], [12]. Separability has been so far an important attribute of practicability for current RDH-ED.

The redundancy introduced by VRBE or VRAE is independent from the encryption, resulting in the mutual restriction between decryption distortion and the embedding capacity, which is a major obstacle to the realization of separability and a high EC. Two main solutions have been proposed: one is to improve the quality of redundancy introduced before encryption. For example in [13], a separable high embedding algorithm was proposed by making full use of prediction error introduced before encryption. Second, the correlation of the
plaintext is preserved in the ciphertext, so that RDH in spatial
domain, such as difference expansion technique (DE) [15],
histogram shifting technique (HS) [16]–[18], could be imple-
mented in the encrypted domain. For example in [14], a new
framework of RDH-ED was proposed, in which a specific
stream cipher was used to preserve the correlation between the
neighboring pixels. The above mentioned symmetric encryp-
tion based algorithms are fast and efficient in practice, which
has significant research value and technological potential in
the future.

However, there are also technical defects in symmetric
encryption based RDH-ED. The correlation of plaintext would
be destroyed because of the confusion and diffusion principles
of symmetric encryption. To achieve reversible data hiding,
it usually needs to introduce embedding redundancy. Since it is
difficult to vacate room after encryption, the current attention
focuses more on the VRBE methods [13], by which more
computational expense is introduced into the client side. The
preprocessing in the plaintext is similar to data compression.
The compression capability determines the performance of
RDH-ED. As for the methods of preserving plaintext correla-
tion by a specific encryption [14], it currently mainly relies on
reusing the same random sequence to encrypt different pixels
in one block. It could provide certain security guarantees,
but key reusing would weaken the encryption intensity of the
symmetric encryption in theory. The more correlation among
ciphertext is remained, the more the encryption intensity is
reduced. The RC4 encryption was declared breached in 2013
[19], RDH-ED based on early RC4 has certain limitations in
future security applications. In addition, symmetric encryption
requires a geometrically increasing amount of encryption keys
with the increasing number of communication participants.
The local key storage cost is high for each user.

Compared with symmetric encryption, public key encryp-
tion has some advantages for RDH-ED, which is worthy of our
attention: first, public key encryption requires a linear increas-
ing amount of key usage in the communication network. The
local key storage cost is only the private key of the user’s own,
while all the public keys are publicly released. It has been
widely used in electronic finance and network communication
protocols, which provides application prospects for RDH-ED.
Second, public key encryption introduces ciphertext extension,
namely, the redundancy from the ciphertext itself. Through
a certain embedding strategy [28], we could select embed-
ding positions and improve EC effectively. Third, flexible
cryptosystems of the public key encryption, especially the
homomorphic encryption, provide reliable technical supports
for RDH-ED. However, there are still technical limitations and
application dilemmas in public key based RDH-ED. We shall
discuss those in Section II. This paper focuses on the current
state of public key based RDH-ED, aiming at making full
use of learning with Error (LWE)-based fully homomorphic
encryption (FHE) technique to implement DE encapsulation.
A novel RDH-ED method is proposed, which is superior to
the current public key based RDH-ED in practicality, security
and reversibility.

The rest of this paper is organized as follows. The following
section introduces the art of state about public encryption
based RDH-ED. Section III reviews the processes of DE and
introduces the techniques of FHE, key-switching, and
bootstrapping. Section IV describes the detailed processes of
the proposed full homomorphic encryption encapsulated
difference expansion. In Section V, the three judging standards
of RDH-ED, including correctness, security and efficiency, are
discussed theoretically and verified with experimental results.
Finally, Section VI summarizes the paper and discusses future
investigations.

II. RELATED WORK

Currently, researches of public key encryption based RDH-
ED are mainly based on Paillier encryption [20]–[26] and LWE
encryption [27]–[29].

Probabilistic and homomorphic properties of Paillier
encryption allow the third party, i.e., the cloud servers, to con-
duct operations directly on ciphertext without knowing the pri-
ivate key, which shows potential for more flexible realizations
of RDH-ED. The first Paillier encryption based RDH-ED was
proposed by Chen et al. [20]. Shiu et al. [21] and Wu et al. [22]
improved the EC of [20] by solving the pixel overflow prob-
lem. Those algorithms were VRBE methods. Li et al. in [25]
proposed a VRAE method with a considerable EC by utilizing
the homomorphic addition property of Paillier encryption and
HS technique. The above algorithms were all inseparable.

Data extraction was implemented only in the plaintext domain.
It was a crucial bottleneck of public key encryption based
RDH-ED to realize data extraction directly from the encrypted
domain. Wu et al. proposed two RDH-ED algorithms for the
encrypted images in [24]: a high-capacity algorithm based
on Paillier cryptosystem was presented for data extraction
after image decryption. The other one could operate data
extraction in the encryption domain. Zhang et al. [23] pro-
posed a combined scheme consisting of a lossless scheme
and a reversible scheme to realize separability. Data was
extracted from the encrypted domain in the lossless scheme
and from the plaintext domain in the reversible scheme.
In [26], Xiang embedded the ciphertext of additional data into
the least significant bits (LSBs) of the encrypted pixels by
employing homomorphic multiplication. Only the ciphertext
of additional data could be obtained during extraction directly
from ciphertext. To distinguish the corresponding plaintext of
the ciphertext without the private key, a one-to-one mapping
table from ciphertext to plaintext was creatively introduced.
The ciphertext of additional data for embedding was not from
encryption but from the mapping table. However, the exposure
and accumulation of a large number of the mapping tables to
an untrusted third party might increase the risk of cryptanalysis
in theory, while the Paillier algorithms cannot resist adaptive
chosen ciphertext attack [30].

LWE based RDH-ED was first proposed in [27] by quanti-
fying the LWE encrypted domain and recoding the redundancy
from ciphertext. Ke et al. fixed the parameters for LWE
encryption and proposed a multilevel RDH-ED with a flexible
applicability and high EC in [28]. However, the data-hiding
key used for extraction overlapped partly with the private key
for decryption, thus resulting in limitation for embedding by a
third party. LWE encryption in [27], [28] was used as public key encryption method, which is not FHE method. In [29], separability could be achieved by preserving correlation from the plaintext in the ciphertext through a modified somewhat LWE encryption. However, the correlation among ciphertext was strong, and it was theoretically vulnerable to cryptanalysis attacks.

In summary, the public key based RDH-ED has difficulties in the implementation of separability. This paper proposes a novel scheme of FHE encapsulated DE (FHEE-DE) to realize data hiding in encrypted domain based on LWE. Modified bit-addition and bit-subtraction circuits for FHEE-DE are designed. Key-switching and bootstrapping are introduced to control the ciphertext extension and decryption failure, which provides the feasibility of introducing other existing RDH methods. A fidelity constraint is introduced, which has distinctly enhanced the Peak Signal Noise Ratio (PSNR) of the directly decrypted images. To realize separability, a key-switching based least significant bit data hiding (KS-LSB) method is proposed to support the extraction directly from the encrypted domain without the private key. Finally, to improve the efficiency, we propose an efficient FHEE-DE method, and fully separability is achieved without reducing the security of LWE encryption.

### III. Preliminaries

#### A. Difference Expansion

The difference expansion technique is an important part of the early RDH algorithms, appearing together with the histogram shifting technique. It is characterized by specific numerical modification of specific pixels, making it reversible under the premise of fault tolerance.

In Tian’s algorithm [15], the earliest DE algorithm, two adjacent pixels $X$ and $Y$ from an image $I$ can be used to hide one additional bit $b_s$, where $0 \leq X, Y \leq 255$ and $b_s \in \{0, 1\}$. First, the difference $h$ and average value $l$ (integer) of $X$ and $Y$ are computed as following:

$$h = X - Y$$  \tag{1}
$$l = \left\lfloor \frac{X + Y}{2} \right\rfloor$$  \tag{2}

The plaintext is

$$p \in \{0, 1\}.$$  \tag{3}

The embedded pixels $X'$ and $Y'$ can be obtained by substituting $h'$ into Eqs. (3), (4).

$$X' = l + \left\lfloor \frac{h + 1}{2} \right\rfloor$$  \tag{3}
$$Y' = l - \left\lfloor \frac{h}{2} \right\rfloor$$  \tag{4}

Assuming $X > Y$, and $\lfloor . \rfloor$ is the floor function meaning “the biggest integer less than or equal to” while $\lceil . \rceil$ is the ceiling function.

**Data hiding:** The embedded difference $h'$ is calculated:

$$h' = 2 \times h + b_s$$  \tag{5}

Then, the original pixels $X$ and $Y$ can be recovered by using Eqs. (3), (4).

![Fig. 1. The principle of difference expansion.](image)

#### B. Full Homomorphic Encryption

A cryptosystem that supports both homomorphic addition and homomorphic multiplication on ciphertext is known as fully homomorphic encryption [33]. Paillier encryption supports only homomorphic multiplication, which is semi-homomorphic encryption. Such a scheme enables the construction of programs for any desirable functionality, which can be run on encrypted inputs to produce an encryption of the result.

Craig Gentry, [33] using LWE in lattice based cryptography, described the first construction for a FHE scheme. Gentry’s scheme supports both addition and multiplication operations on ciphertext, from which it is possible to construct circuits for performing arbitrary complex computation. A brief review of FHE [33] is as following:

The private key is denoted as $s$, and the public key $A$ is generated by $s$ and $e$ satisfying Eq. (8), where $e$ is sampled randomly:

$$A \cdot s = 2e$$  \tag{8}

**Encryption:**

The plaintext is $p \in \{0, 1\}$. Set $m = (p, 0, 0, \ldots, 0)$. Generate a 0-1 random sequence $a_r$ uniformly and output the
ciphertext:
\[ c = m + A^T a_r \]  \hspace{1cm} (9)

**Decryption:**
\[
\left[ \left( c, s \right) \right]_q = \left[ \left( m + A^T a_r, s \right) \right]_q = \left[ \left( m^T s + (A^T a_r)^T s \right) \right]_q = \left[ \left( p + a^T 2e \right) \right]_q = p \]  \hspace{1cm} (10)

where \( \left[ . \right]_q \) means to perform modulo \( q \). The correctness lies in that the total introduced noise could be restrained to meet:
\[ a^T e < q/4 \]  \hspace{1cm} (11)

To demonstrate the FHE ability, we assume the LWE ciphertext \( c_1, c_2 \) are:
\[ c_1 = m_1 + 2r_1 + p_1q, \quad c_2 = m_2 + 2r_2 + p_2q. \]

**FHE. Add:**
\[ c_1 + c_2 = (m_1 + m_2) + q (p_1 + p_2) + 2 (r_1 + r_2) \]  \hspace{1cm} (12)

where the correctness lies in that the total introduced noise could be restrained to meet:
\[ (r_1 + r_2) < q/4 \]  \hspace{1cm} (13)

**FHE. Multiply:**
\[ c_1 \otimes c_2 = m_1 m_2 + 2 (m_1 r_2 + m_2 r_1 + 2r_1 r_2) + q (c_2 p_1 + c_1 p_2 - q p_1 p_2) \]  \hspace{1cm} (14)

where the correctness lies in that the total introduced noise could be restrained to meet:
\[ m_1 r_2 + m_2 r_1 + 2r_1 r_2 < q/4 \]  \hspace{1cm} (15)

**C. Key-Switching [34]**

There is data expansion in LWE encrypted ciphertext [28], but in FHE, a secondary expansion would occur when ciphertext got multiplied. The homomorphic multiplication between the ciphertext matrices returns the ciphertext tensor product, and the private key is also subjected to the tensor product operation before being used to decrypt the new ciphertext. Therefore, the amount of data will again expand geometrically.

In our scheme, the ciphertext of the pixel bits will get expanded after each multiplication. It also occurs after addition or subtraction between encrypted pixels due to the cases of bit carry or bit borrow, resulting in a large number of multiplication and exclusive or operations among ciphertext of pixel bits. If the secondary expansion cannot be eliminated or controlled, the amount of ciphertext data can produce an excessively extension that is unacceptable in practice. Key-switching can effectively eliminate the extension by replacing the extended ciphertext with new ciphertext of any shorter length without decrypting it, and ensure the new ciphertext corresponds to the same decryption as the extended ciphertext.

We use the key-switching technique to eliminate the ciphertext secondary expansion in FHEE-DE, that is, key-switching is implemented following each homomorphic operation. What is more, a key-switching based LSB data hiding method is proposed in this paper.

**D. Bootstrapping Encryption [35]**

The introduced noise, on the one hand, provides the security guarantee of the LWE algorithm. On the other hand, noise superposition will also affect the correctness of decryption. Usually, the one-way fluctuation interval of required noise cannot exceed a quarter of the encrypted domain. Homomorphic operations result in the superposition of noise as shown in Eqs. (11)(13)(15), which makes the correctness of decryption unstable. Key-switching cannot eliminate noise superposition. Therefore, the decryption overflow problem is an important issue to be considered in FHEE-DE. Usually, the simplest method is to limit the standard deviation of the sampling noise distribution, so that the noise fluctuation range is very small and the overflow will not occur after several times of superposition. Although efficient, it is not enough to support many or even theoretically infinite holomorphic operations.

In this paper, we use bootstrapping to reduce the superposed noise. The sketch of bootstrapping is as shown in Fig. 2: ciphertext1 carries superposed noise. A public key is used to encrypt ciphertext1 and the private key simultaneously. The encrypted private key is then used to decrypt the ciphertext2 (the encrypted ciphertext1) to obtain ciphertext3. The decryption process eliminates the noise in ciphertext1, but retains the noise from the bootstrapping encryption in ciphertext3. Therefore, bootstrapping does not change the data length of ciphertext or the key, but it can restore the total noise amount after several times of superposition into the noise amount introduced by only one bootstrapping encryption (for more details in [35]).

Key-switching and bootstrapping are used to ensure FHEE-DE based RDH-ED has good practicability and scalability. The usage and parameter requirements will be introduced in detail in the following sections, and some adaptive modifications will be made to meet the specific requirements of reversibility. The main drawback of the two techniques is the large number of public keys required. Fortunately, users only need to store the private key for decryption. The public keys are all released publicly on the internet or the cloud instead of stored locally after their generation. The storage problem caused by the large amount of public keys can be ignored in practice, which is also the advantage of public key cryptosystem in application.
Fig. 3. The two schematic flows of the applications of FHEE-DE: (a) Data hiding in the third server side; (b) Data hiding in both the client and server side.

Besides, the operation of generating public keys is not strictly in series with other processes, so parallel optimization techniques can be used to improve the efficiency. The parallel optimization is not the consideration of this paper.

IV. FULL HOMOMORPHIC ENCRYPTION ENCAPSULATED DIFFERENCE EXPANSION

A. Framework of FHEE-DE

As shown in Fig. 3, the application framework of FHEE-DE consists of two types: (a) Data hiding in the third server side. (b) Data hiding in both the client and server side. We take type a as an example to explain the application: the user encrypts the plaintext and uploads its ciphertext to the server. The server performs FHEE-DE data hiding and KS-LSB data hiding to obtain the marked ciphertext. FHEE-DE data hiding ensures that the decryption would contain additional data. KS-LSB data hiding enables the server to extract additional data directly from the marked ciphertext. For the marked ciphertext, there are four cases: a) the client obtains and decrypts it directly by using private key to obtain the marked plaintext. DE extraction or recovery can be implemented to obtain additional data or the plaintext losslessly. b) Additional data can be extracted directly by the server without using the private key, which enables the (trusted or untrusted) third party to manage ciphertext flexibly under the premise of keeping plaintext secret. c) The third server returns new ciphertext through FHEE-DE recovery. The client user can obtain the plaintext by decrypting the new ciphertext. d) FHEE-DE extraction returns the encrypted additional data. The client user can obtain the additional data by decrypting the encrypted additional data.

The notation of the main variables in the proposed scheme is shown in Table I:

B. Universal FHEE-DE

1) Preprocessing of DE:

   a) Overflow/underflow and fidelity constraints: The plaintext is a $512 \times 512$ image $I$. $I$ is divided into non-overlapping pixel pairs. Each pair consists of two adjacent pixels. We take one pair of pixels, denoted as $(X, Y)$, as an example to introduce our scheme, where $0 \leq X, Y \leq 255$. The additional bit is $b_s \in \{0, 1\}$. As grayscale values are bounded in $[0, 255]$, we have $h$ and $l$ according to Eqs. (1), (2):

$$0 \leq l + \left\lfloor \frac{h + 1}{2} \right\rfloor \leq 255\tag{16}$$
$$0 \leq l - \left\lfloor \frac{h}{2} \right\rfloor \leq 255\tag{17}$$
TABLE I  
NOTATIONS OF THE MAIN VARIABLES

| Notation        | Representation                                      |
|-----------------|-----------------------------------------------------|
| $X, Y \in \mathbb{Z}_{256}$ | A pair of two adjacent pixels.                      |
| $b \in \mathbb{Z}_{256}$ | The difference of $X$ and $Y$.                      |
| $l \in \mathbb{Z}_{256}$ | The average value (integer) of $X$ and $Y$.         |
| $b'_e, b'_f, b'_l, b'_t \in \{0, 1\}$ | The $t$-LSB of $X, Y, h$ or $l$ ($t = 1, 2, \ldots, 8$). |
| $c'_e, c'_f, c'_l, c'_t \in \mathbb{Z}_q$ | Ciphertext encrypted from $b'_e, b'_l, b'_t$ ($t = 1, 2, \ldots, 8$). |
| $b \in \{0, 1\}$ | The additional bit for embedding.                    |
| $c_m$ | Ciphertext encrypted from $b_c$.                   |
| $h \in \{0, 1\}$ | To-be-embedded bit by KS-LSB.                       |

To avoid overflow or underflow problems after data hiding, we use a map matrix, denoted as $M_{\text{ava}} \in \{0, 1\}^{512 \times 512}$, to indicate available pixel pairs. Value “1” indicates the bigger pixel within an available adjacent pixel pair for DE data hiding. The difference $h$ of an available pair should satisfy the following constraints [15]:

$$|h| \leq \min(2(255 - l), 2l + 1) \tag{18}$$

$$|2 \cdot h + b_s| \leq \min(2(255 - l), 2l + 1) \tag{19}$$

for $b = 0$ or 1.

We add an extra fidelity constraint to limit the distortion in the marked plaintext. The fidelity parameter $h_{\text{fid}}$ is introduced here: the available pixel pairs are preferentially selected with a smaller pixel difference:

$$h \leq h_{\text{fid}} \tag{20}$$

$M_{\text{ava}}$ would be lossless compressed as side information of the ciphertext to superimpose on the host signal.

b) Parameters setting and function definition: The cryptosystem is parameterized by the integers: $n$ (the length of the private key), $q \in (n^2, 2n^2)$ (the modulus), $d \geq (1+\varepsilon)(1+n)\log_2 q$ (the dimension of the public key space), $1 > \varepsilon > 0$. $q$ is a prime, and all the operations in the cryptosystem are performed modulo $q$ in $\mathbb{Z}_q$, $\beta = \left[\log_2 q\right]$. We denote the noise probability distribution on $\mathbb{Z}_q$ as $\chi$, $\chi = \overline{\mathcal{N}}_{aq}$, where the discrete Gaussian distribution $\overline{\mathcal{N}}_{aq} = \{\lfloor qx \mod q \rfloor x \sim N(0, a^2)\}$, and $N(0, a^2)$ is standard normal distribution, $\lfloor qx \rfloor$ denotes rounding $qx$ to the nearest integer [28].

Definition 1: The private key generating function:

$$s = SKGen_{n, \chi}(.) \tag{21}$$

which returns the private key $s \in \mathbb{Z}_q^n$. $s = (1, t)$, where $t \in \mathbb{Z}_{q^{-1}}^{n-1}$ is sampled from the distribution $\chi$.

Definition 2: The public key generating function:

$$A = PKGen_{d, \sigma, \chi}(s) \tag{22}$$

in which a matrix $W \in \mathbb{Z}_{q^d}^{d \times (n-1)}$ is first generated uniformly and a vector $e \in \mathbb{Z}_{q^d}^d$ is sampled from the distribution $\chi$, then the vector $b \in \mathbb{Z}_{q^d}^d$ is obtained:

$$b = Wt + 2e \tag{23}$$

the $n$-column matrix $A \in \mathbb{Z}^{d \times n}$ is consisting of $b$ followed by $-W$, $A = (b, -W)$. $A$ is returned as the public key.

Remark: Observe that $A \cdot s = 2e$ for Eq. (8).

Definition 3: The encrypting function:

$$c = Enc_A(p) \tag{24}$$

which returns a vector $c$ as the ciphertext of one bit plaintext $p \in \{0, 1\}$ with the private key $A$: Set $m = (p, 0, 0, \ldots, 0) \in \mathbb{Z}_q^n$. Generate a random vector $a_r \in \mathbb{Z}_q^d$ uniformly and output $c$:

$$c = m + A^T a_r \tag{25}$$

Definition 4 [34]: The function $BitDe(x)$, $x \in \mathbb{Z}_q^n$, decomposes $x$ into its bit representation. Namely, it outputs $(u_1, u_2, u_3, \ldots, u_\beta) \in \mathbb{Z}_q^\beta$, $x = \sum_{j=0}^{\beta-1} 2^j \cdot u_j, u_j \in \mathbb{Z}_q^n$.

Definition 5: The decrypting function:

$$p = Dec_d(c) = \left[\left[\langle c, s \rangle\right]_d\right]_2 \tag{26}$$

which returns the plaintext bit $p \in \{0, 1\}$ with the private key $s$. If the inputs of the decryption function are in binary form, we could regard such a function as a decryption circuit, denoted as $Dec_s(C), C = BitDe(c), S = BitDe(s)$.

Definition 6 [34]: The function $Powersof(x)$, $x \in \mathbb{Z}_q^n$, outputs the vector $(x, 2x, 2^2x, \ldots, 2^{\beta-1}x, e) \in \mathbb{Z}_q^{\beta+1}$.

Next, we will give the procedure of key-switching, which takes ciphertext $c_1$ under private key $s_1$ and outputs new ciphertext $c_2$ from the same plaintext under the private key $s_2$.

Definition 7 [34]: The switching key generating function:

$$B = SwitchKGen(s, s_2) \tag{27}$$

where $s_1 \in \mathbb{Z}_q^{n_1}$, $s_2 \in \mathbb{Z}_q^{n_2}$ are two private keys, $A_{\text{temp}} = PKGen_{n_1 \times n_2, \chi}(s_2)$. The matrix $B \in \mathbb{Z}_q^{n_1 \times n_2}$ can be obtained by adding $Powersof(s_1)$ to $A_{\text{temp}}$‘s first column.

Ciphertext $c_2$ can be obtained by using the switching key:

$$c_2 = BitDe(c_1)^T \cdot B \tag{28}$$

The secondary data expansion of the ciphertext is resulted from the homomorphic multiplication. Namely, we need to operate key-switching after each tensor product of ciphertext. Specifically, we would transform $e \otimes e$ under $s \otimes s$ into $c$ under $s$. Therefore, the switching keys for eliminating the secondary data expansion should be:

$$B = SwitchKGen(s \otimes s, s) \tag{29}$$

where $s \otimes s \in \mathbb{Z}_q^{n^2}, s \in \mathbb{Z}_q^n$.

c) Key distribution: In our scheme, there is a key-switching based LSB data hiding method proposed to ensure that the servers could directly extract additional data from ciphertext without using the private key. We generate a pseudo-random binary sequence $k$ for the servers to randomly scramble the additional data before KS-LSB data hiding. The switching key for KS-LSB data hiding is:

$$B_{\text{LSB}} = SwitchKGen(s, s) \tag{30}$$

where $s \in \mathbb{Z}_q^n$. The distributions of keys are in Table II:
2) Encryption: For the pixel pair (X, Y), whose i LSBs are denoted by \( b_i^X, b_i^Y \) \( (i = 1, 2, \ldots, 8) \), each bit is encrypted by LWE encryption with a new public key. We omit the symbol “\( \Lambda \)” in Eq. (24) for short in this paper: \( c_i^X = \text{Enc}(b_i^X), c_i^Y = \text{Enc}(b_i^Y) \), \( i = 1, 2, \ldots, 8 \).

3) FHE Encapsulated DE Data Hiding:

a) Calculation circuits design: To realize FHE encapsulated DE, we designed the calculation circuits as Fig. 4 shows. Compared with traditional circuits, we made some simplification on the carry or borrow cases of the highest/lowest bit and the positive or negative sign judgment, which was mainly based on the bit length of pixels and the overflow/underflow constraints in preprocessing. It should be noted that homomorphic circuits share the same internal relationships and operation types as the above calculation circuits, except that it is arithmetic operations which are performed modulo 2 in calculation circuits (Fig. 4(a)) while those would be matrix operations and performed modulo \( q \) in homomorphic circuits (Fig. 4(b)).

Pixel adding circuit \( \text{Add}\^a \) of \( (X + Y) \) in binary form is as following:

\[
(b_{\text{sum}}^8, b_{\text{sum}}^7, \ldots, b_{\text{sum}}^1) = \text{Add}\^a(b_X^8, b_X^7, \ldots, b_X^1; b_Y^8, b_Y^7, \ldots, b_Y^1) \quad (31)
\]

In \( \text{Add}\^a \), there are eight refreshing in order from 1 to 8 due to the bit carry case. After each refreshing, one more bit of the sum would be outputted and the inputs would be refreshed. In refreshing \( i (i =1 \text{ to } 8) \):

\[
b_{\text{sum}}^i = b_{X}^i + b_{Y}^i, \quad b_{\text{diff}}^i = b_{X}^i - b_{Y}^i, \quad \left( b_{X}^{i-1} \ldots b_{X}^j \ldots b_{X}^1 \right) \left( b_{Y}^{i-1} \ldots b_{Y}^j \ldots b_{Y}^1 \right), \quad j = 1, 2, \ldots, 8 - i.
\]

In this section, assuming \( X > Y \), the subtracting circuit \( \text{Sub}^a \) of \( (X-Y) \) is designed (we could confirm the bigger one between a pair \( (X, Y) \) according to the map \( M_{\text{ava}} \)):

\[
(b_{\text{diff}}^8, b_{\text{diff}}^7, \ldots, b_{\text{diff}}^1) = \text{Sub}^a(b_X^8, b_X^7, \ldots, b_X^1; b_Y^8, b_Y^7, \ldots, b_Y^1) \quad (32)
\]

In \( \text{Sub}^a \), there are eight refreshing in order from 1 to 8 due to the bit borrow case. The internal relationship between inputs and outputs is exhibited in Fig. 4(c). After each refreshing, one more bit of the difference would be outputted and the minuend would be refreshed. In refreshing \( i (i =1 \text{ to } 8) \):

\[
b_{\text{diff}}^i = b_{X}^i + b_{Y}^i, \quad b_{\text{diff}}^i = b_{Y}^i + b_{X}^i, \quad b_{\text{diff}}^i = b_{Y}^i - b_{X}^i, \quad \left( b_{X}^{i-1} \ldots b_{X}^j \ldots b_{X}^1 \right) \left( b_{Y}^{i-1} \ldots b_{Y}^j \ldots b_{Y}^1 \right), \quad j = 1, 2, \ldots, 8 - i.
\]

b) FHE encapsulated DE data hiding: Step 1: Calculate \( c_i^l(i = 1, 2, \ldots, 8) \). According to Eq. (1), \( (c_8^X, c_8^Y, \ldots, c_1^Y) = \text{Sub}^a(c_8^X, \ldots, c_1^X; c_8^Y, \ldots, c_1^Y) \).

Step 2: Calculate the encrypted \( (X + Y),(c_{X+Y}^X, c_{X+Y}^Y) = \text{Add}^a(c_X^8, \ldots, c_X^1; c_Y^8, \ldots, c_Y^1) \).

Step 3: Calculate \( c_{\text{temp0}} = \text{Enc}(c_0) \), and the encrypted \( l \), namely \( (c_8^l, c_7^l, \ldots, c_1^l) \), can be obtained according to Eq. (2): \( (c_8^l, c_7^l, \ldots, c_1^l) = (c_{\text{temp0}}, c_X^8, \ldots, c_X^1, c_Y^8, \ldots, c_Y^1) \).

Step 4: Calculate \( c_{\text{temp}} = \text{Enc}(c_l) \) and refresh \( c_{\text{temp}} = \text{Enc}(0) \).

The encrypted \( h^l, (c_h^8, c_h^7, \ldots, c_h^1) \), can be obtained according to Eq. (5): \( (c_h^8, c_h^7, \ldots, c_h^1) = \text{Add}^a(c_h^8, \ldots, c_h^1, c_{\text{temp0}}; c_{\text{temp0}}, c_{\text{temp0}}, \ldots, c_{\text{temp0}}, \text{ncl}) \).
Step 5: Calculate $c_{\text{temp}_1} = \text{Enc}(1)$, and then calculate the encrypted $(h',+1, (e_{h'+1}^8, e_{h'+1}^7, \ldots, e_{h'+1}^1)) = \text{Add}^a (e_{h'}^8, e_{h'}^7, \ldots, e_{h'}^1)$, $c_{\text{temp}_0}, c_{\text{temp}_0'}, \ldots, c_{\text{temp}_0'}, c_{\text{temp}_1}$.

Step 6: Refresh $c_{\text{temp}_0} = \text{Enc}(0)$. The encrypted $X'$ and $Y'$ after DE data hiding are restored according to Eqs. (3)-(4): 

\[
\left( e_{X'}^8, e_{X'}^7, \ldots, e_{X'}^1 \right) = \text{Add}^a (e_{X'}^8, e_{X'}^7, \ldots, e_{X'}^1); \\
\left( e_{Y'}^8, e_{Y'}^7, \ldots, e_{Y'}^1 \right) = \text{Sub}^b (e_{Y'}^8, e_{Y'}^7, \ldots, e_{Y'}^1);
\]

Following each homomorphic multiplication, key-switching is implemented to eliminate the secondary data expansion of ciphertext. The bootstrapping is implemented for every 10 homomorphic multiplication or 100 homomorphic addition to control noise excessive stacking.

4) Key-Switching Based LSB Data Hiding: Step 1: Randomly scramble the additional data sequence $b_i$ by using data hiding key $k$ to obtain the to-be-embedded data $b_i$:

\[
b_i = k \oplus b_s
\]

where $b_t \in b_i$. Denote the last element of $e_{c_{LX}}$ as $e_{c_{LX}}$, whose LSB would be replaced by $b_t$. $X$ is the “1” signed pixel by $M_{av_a})$.

Step 2: If $b_t = \text{LSB}(c_{LX}), c_{c_{LX}}^{1}$ maintains the same, or if $b_t \neq \text{LSB}(c_{LX}), c_{c_{LX}}^{1}$ is refreshed by: $c_{c_{LX}}^{1} = \text{BitDe}(e_{c_{LX}}^{1})^T$.

$B_{\text{LSB}}$.

Step 3: Repeat Step 2 until $\text{LSB}(c_{LX}) = b_t$.

The marked ciphertext is obtained: $e_{c_{LX}}^{1}$ and $e_{c_{LX}}^{1}(i = 1, 2, \ldots, 8)$.

According to the framework in Fig. 3(a), after receiving the marked ciphertext, the client user could implement the decryption on the marked ciphertext to obtain $X'$ and $Y'$ by using $s$: $b_{y'} = \text{Dec}_s(e_{y'}^{1}), b_{y'} = \text{Dec}_s(e_{y'}^{1}), (i = 1, 2, \ldots, 8)$. The additional data could be extracted according to DE extraction (Eq. (6)) and the original pixels could be recovered according to DE recovery (Eqs. (7), (3), and (4)).

5) LSB Extraction From the Marked Ciphertext: Additional data could be directly extracted from ciphertext without the private key $s(X$ is the “1” signed pixel by $M_{av_a})$:

\[
b_t = \text{LSB}(c_{LX})
\]

6) FHE Encapsulated DE Recovery: FHE encapsulated DE recovery is implemented by the servers to return a new ciphertext of the plaintext without any added data.

Step 1: Calculate $c_{c_{LX}}^{i}$, $e_{c_{LX}}^{i}$, $e_{c_{LX}}^{i}$ ($i = 1, 2, \ldots, 8$) by using $(e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i})$ according to Eq. (1): $(e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, \ldots, e_{c_{LX}}^{i}) = \text{Sub}^b (e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, \ldots, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i}, e_{c_{LX}}^{i})$.

Step 2: Refresh $c_{\text{temp}_0}, c_{\text{temp}_0'} = \text{Enc}(0)$. The encrypted $h$ can be obtained according to Eq. (7): $(e_{h'}^{8}, e_{h'}^{7}, \ldots, e_{h'}^{1}) = (c_{\text{temp}_0}, c_{\text{temp}_0'}, c_{\text{temp}_1})$.

Step 3: Calculate the encrypted $(X' + Y')$: $(e_{X' + Y'}^{8}, e_{X' + Y'}^{7}, \ldots, e_{X' + Y'}^{1}) = \text{Add}^a (e_{X'}^{8}, e_{X'}^{7}, \ldots, e_{X'}^{1}, e_{Y'}^{8}, e_{Y'}^{7}, \ldots, e_{Y'}^{1})$.

Step 4: Refresh $c_{\text{temp}_0} = \text{Enc}(0)$, and the encrypted $l$, $(e_{l}^{8}, e_{l}^{7}, \ldots, e_{l}^{1})$, can be obtained according to Eq. (2): $(e_{l}^{8}, e_{l}^{7}, \ldots, e_{l}^{1}) = (c_{\text{temp}_0}, c_{\text{temp}_0'}, c_{\text{temp}_1})$.

Step 5: Refresh $c_{\text{temp}_0} = \text{Enc}(1)$. And calculate the encrypted $(h' + 1, (e_{h' + 1}^{8}, e_{h' + 1}^{7}, \ldots, e_{h' + 1}^{1}) = \text{Add}^a (e_{h'}^{8}, e_{h'}^{7}, \ldots, e_{h'}^{1}, c_{\text{temp}_0}, c_{\text{temp}_0'}, c_{\text{temp}_0'})$.

Step 6: Refresh $c_{\text{temp}_0} = \text{Enc}(0)$. The encrypted $X$ and $Y$ are restored according to Eqs. (3)-(4):

\[
\left( c_{\text{X}'}^{8}, c_{\text{X}}^{7}, \ldots, c_{\text{X}}^{1} \right) = \text{Add}^a (e_{\text{X}}^{8}, e_{\text{X}}^{7}, \ldots, e_{\text{X}}^{1}) ;
\left( c_{\text{Y}}^{8}, c_{\text{Y}}^{7}, \ldots, c_{\text{Y}}^{1} \right) = \text{Add}^b (e_{\text{Y}}^{8}, e_{\text{Y}}^{7}, \ldots, e_{\text{Y}}^{1}) ; c_{\text{temp}_0} = c_{\text{temp}_0'}, c_{\text{temp}_0'}.
\]

According to the Fig. 3(a), after receiving the restored ciphertext, the client user could implement the decryption to obtain the original pixels $X$ and $Y$ by using $s$:

\[
b_{X} = \text{Dec}_s(e_{X'}^{1}), b_{Y} = \text{Dec}_s(e_{Y'}^{1}), (i = 1, 2, \ldots, 8).
\]

7) FHE Encapsulated DE Extraction: It shares the same step as Step 1 in FHE encapsulated DE recovery to obtain the encrypted $h'$, $(e_{h'}^{8}, e_{h'}^{7}, \ldots, e_{h'}^{1})$. The encrypted $b_{L}$ is $e_{b_{L}}^{1}$.

After receiving the encrypted additional data, the client user could implement the decryption to obtain the embedded data: $b_{L} = \text{Dec}_s(e_{b_{L}}^{1})$.

C. Efficient FHEE-DE

1) Preprocessing of DE: Efficient FHEE-DE shares the same preprocessing as universal FHEE-DE.

2) Encryption: The client user calculates the $(h, l)$ of $(X, Y)$ first. The $(h, l)$ would be encrypted as ciphertext which would be uploaded to the server: $e_{h}^{1} = \text{Enc}(b_{h})$ and $e_{l}^{1} = \text{Enc}(b_{l})$ ($i = 1, 2, \ldots, 8$).

3) FHE Encapsulated DE Data Hiding: Calculate $c_{bs} = \text{Enc}(b_{s})$ and refresh $c_{\text{temp}_0} = \text{Enc}(0)$. The encrypted $h'$, $(e_{h'}^{8}, e_{h'}^{7}, \ldots, e_{h'}^{1})$ can be obtained according to Eq. (5): $(e_{h'}^{8}, e_{h'}^{7}, \ldots, e_{h'}^{1}) = \text{Add}^a (e_{h}^{8}, e_{h}^{7}, \ldots, e_{h}^{1}, e_{b_{L}}^{1}, c_{\text{temp}_0}, c_{\text{temp}_0'}, c_{\text{temp}_0'}$. $c_{bs}$).

Return $e_{h'}^{1}$ and $e_{l}^{1}$ $(i = 1, 2, \ldots, 8)$ as the DE embedded ciphertext.

4) Key-Switching Based LSB Data Hiding: Step 1: The same as Step 1 of Key-switching based LSB data hiding in universal FHEE-DE. Denote the last element of $e_{h'}^{1}$ as $e_{c_{LH}}$, whose LSB would be replaced by $b_t$. 

Step 2: If $b_t = \text{LSB}(c_{LH}), c_{LH}^{1}$ maintains the same, or if $b_t \neq \text{LSB}(c_{LH}), c_{LH}^{1}$ is refreshed by: $c_{LH}^{1} = \text{BitDe}(e_{LH}^{1})^T$. $B_{\text{LSB}}$.

Step 3: Repeat Step 2 until $\text{LSB}(c_{LH}) = b_t$.

The marked ciphertext is obtained: $e_{c_{LH}}^{1}$ and $e_{c_{LH}}^{1}(i = 1, 2, \ldots, 8)$.

After receiving the marked ciphertext, the client user could decrypt it and obtain the marked $(h', l')$ by using $s$: $b_{L}^{s} = \text{Dec}_{s}(e_{LH}^{1}), b_{L}^{s} = \text{Dec}_{s}(e_{LH}^{1}), (i = 1, 2, \ldots, 8)$. Then DE extraction and recovery could be implemented.

Efficient FHEE-DE shares the same $\text{LSB}$ extraction from the marked ciphertext as universal FHEE-DE.
5) FHE Encapsulated DE Recovery: Refresh $c_{\text{temp0}} = Enc(0)$, $(c^8_1, c^8_2, ..., c^8_8)$ can be obtained according to Eq. (7): $(c^8_1, c^8_2, ..., c^8_8) = (c_{\text{temp0}}, c^8_1, c^8_2, ..., c^8_8)$. The unmarked $c^i_j$ and $c^i_j$ ($i = 1, 2, ..., 8$) are obtained. The client user could decrypt them to obtain $h$ and $l$.

6) FHE Encapsulated DE Extraction: The encrypted $b_h$ is obtained by $c_{bs} = c^1_h$. The client user could decrypt it to obtain the embedded data.

V. THEORETICAL ANALYSIS AND EXPERIMENTAL RESULTS

A. Correctness

The correctness of the proposed scheme includes the lossless restoration of plaintext and the accurate extraction of the embedded data. To test the performances of the proposed scheme, we have tested on 1000 8-bit grayscale images of size 512 $\times$ 512 from image libraries USC-SIPI (http://sipi.usc.edu/database/database.php?volume=misc) and Kodak (http://r0k.us/graphics/kodak/index.html). The experimental results of eight test images (as shown in Fig. 5) were demonstrated in this paper. The preprocessing of DE, LWE encryption & decryption, key switching, and KS-LSB were all implemented on MATLAB2010b with a 64-bit single core (i7-6800K) @ 3.40GHz. We referred to the method in [36] to realize on NVIDIA Titan-XP GPU card. We performed bootstrapping operations after each homomorphic circuit calculation.

Parameters setting: Solving the LWE problem with given parameters is equivalent to solving Shortest Vector Problem (SVP) in a lattice with a dimension $\sqrt{n \log_2(q) / \log_2(\delta)}$. Considering the efficiencies of the best known lattice reduction algorithms, the secure dimension of the lattice must reach 500 ($\delta = 1.01$) [37], [38]. An increase in $n$ will result in a stronger security and a high ciphertext extension [28]. To balance security and the practical efficiency, we set $n = 240$, $q = 57601$, $d = 4573$ ($\sqrt{240 \log_2(57601) / \log_2(1.01)} \approx 514.18 > 500$). To limit the distortion of the marked plaintext, we set $h_{\text{fid}} = 10$. These settings were applied to all the following experiments.

TABLE III

| Image      | Max EC | PSNR1 | PSNR2 | PSNR3 |
|------------|--------|-------|-------|-------|
| Lena       | 110195 | 42.1171 | ∞     | ∞     |
| Baboon     | 69286  | 41.3894 | ∞     | ∞     |
| Crowd      | 104882 | 42.4764 | ∞     | ∞     |
| Tank       | 108953 | 40.4472 | ∞     | ∞     |
| Peppers    | 110558 | 40.5025 | ∞     | ∞     |
| Plane      | 114834 | 42.8519 | ∞     | ∞     |
| Boat       | 97619  | 40.5030 | ∞     | ∞     |
| Truck      | 114437 | 41.0175 | ∞     | ∞     |
| Average    | 103847 | 41.3525 | ∞     | ∞     |

1) Reversibility of Plaintext Recovery: In the proposed scheme, there are two cases of plaintext recovery: a) the user directly decrypts the marked ciphertext to get the marked plaintext. We calculated the PSNR of the marked plaintext, named by PSNR1. And then the plaintext can be obtained after DE recovery. We then calculated the PSNR of the recovered plaintext as PSNR2. b) The third server implements FHEE-DE recovery on the marked ciphertext to obtain new ciphertext. The user receives the new ciphertext and decrypts it to obtain the plaintext. The PSNR of the plaintext is named by PSNR3.

In the experiment, we obtained the available pixel pairs for difference extension according to the constraints in Eqs. (18)-(20). For the ciphertext of the available pixel pairs, two encrypted pixels would carry one bit of additional data by FHEE-DE, and an extra bit by KS-LSB. However, the two bits data embedded are the same. We counted the two bits as one bit of embedding capacity. Therefore, the maximum EC is only related to the number of the available pixel pairs.

The values of PSNR1-3 with the maximum EC are listed in Table III. From the results of PSNR1, it could be seen that there is distortion in the marked plaintext of FHEE-DE. Both PSNR2 and PSNR3 are $\approx \infty$, indicating that the recovered plaintext has achieved no distortion.

We continue to analyze the PSNR1 of the test images at different EC. The marked images of Lena with different $h_{\text{fid}}$ are as shown in Fig. 6. According to the principle of DE in Fig. 1, the smaller the difference is, the smaller modification the embedded pixels have. When performing a non-full-embedded experiment, the available pixel pairs were preferentially selected with a smaller $h_{\text{fid}}$. In Table IV, we list the EC at different $h_{\text{fid}}$ and the corresponding PSNR1 values of the eight test images. Fig. 7 shows the relationship between the EC and PSNR1 of test images.

A comparison of the PSNR1 was made on 1000 test images among four most representative methods [23], [25], [26], [29] and the proposed one. The results have shown that the PSNR1 of the proposed scheme is higher than most existing schemes. As shown in Fig. 8 for example, we demonstrated the PSNR1 in the methods of [15], [23], [25], [26], [29] and the proposed scheme on images Lena and Plane. In [15], available pixel pairs were selected without the fidelity constraint in Eq. (20). In [27], [28], plaintext can be losslessly obtained by directly decrypting the marked ciphertext, thus there exists no marked plaintext.

2) Accuracy of Data Extraction: There are three cases of data extraction. We have made the comparison bit by bit.

Fig. 5. The test images. (a) Lena; (b) Baboon; (c) Crowd; (d) Tank; (e) Peppers; (f) Plane; (g) Boat; (h) Truck.
### TABLE IV

PSNR1 (dB) versus EC (bits) at different $h_{fid}$

| Image  | $h_{fid}=5$ | $h_{fid}=3$ | $h_{fid}=2$ | $h_{fid}=1$ | $h_{fid}=0$ |
|--------|-------------|-------------|-------------|-------------|-------------|
| Lena   | 86605       | 50232       | 50232       | 52104       | 11434       |
|        | 45.6073     | 52.2932     | 52.2932     | 56.6637     | 64.7369     |
| Baboon | 42522       | 20702       | 20702       | 24646       | 4210        |
|        | 47.9412     | 55.9499     | 55.9499     | 60.7332     | 69.0304     |
| Crowd  | 86962       | 64164       | 64164       | 46810       | 26504       |
|        | 47.0505     | 52.1256     | 52.1256     | 55.9208     | 61.0445     |
| Plane  | 100746      | 71114       | 71114       | 50764       | 19966       |
|        | 46.1582     | 51.2680     | 51.2680     | 54.8316     | 62.5390     |
| Peppers| 88017       | 42991       | 42991       | 26044       | 8791        |
|        | 45.4989     | 52.9404     | 52.9404     | 57.5117     | 65.9745     |
| Tank   | 77787       | 43832       | 43832       | 20988       | 16843       |
|        | 45.6201     | 52.5652     | 52.5652     | 60.6655     | 63.1064     |
| Boat   | 65848       | 34351       | 34351       | 21595       | 7414        |
|        | 46.3052     | 53.8847     | 53.8847     | 58.3573     | 66.6448     |
| Truck  | 88688       | 53971       | 53971       | 23973       | 18198       |
|        | 45.5865     | 51.8615     | 51.8615     | 58.0245     | 62.6675     |

![Fig. 6](image_url) Fig. 6. The test images. (a) Original Lena; (b) Marked Lena with $h_{fid}=10$, PSNR1 = 42.1171 dB; (c) Marked Lena with $h_{fid}=2$, PSNR1 = 52.2932 dB; (d) Marked Lena with $h_{fid}=0$, PSNR1 = 64.7369 dB.

![Fig. 7](image_url) Fig. 7. PSNR1 (dB) of FHEE-DE with different EC (bit) on the test images.

between the extracted data and the additional data with an EC of 100,000 bits on the test images Lena, Crowd, Tank, Peppers, Plane, and Truck. The realization of the three cases is the embodiment of the separability of the proposed scheme:

a) The third-party server directly extracts the embedded data from the marked ciphertext by using KS-LSB extraction.

b) The user decrypts the marked ciphertext to obtain the marked plaintext, and then uses DE extraction to extract data. As shown in Fig. 9(b), there is no error.

c) The third-party server performs FHEE-DE extraction on the marked ciphertext to obtain the encrypted embedded data. The user decrypts it to obtain the embedded data. As shown in Fig. 9(c), there is no error.

### B. Security

Security of RDH-ED mainly includes two aspects:

a) Data hiding should not weaken the security of the original encryption or leave any hidden danger of security cracking.

b) The embedded information cannot be obtained by an attacker without the extraction key or the private key.
In this paper, all the homomorphic operations are equivalent to the operations of re-encryption [33]. The encryption security can be theoretically guaranteed by the principles of FHE. The ciphertext maintains a random state of meaningless noise before and after data hiding. The processes of implementing FHEE-DE and KS-LSB on the ciphertext are equivalent to the processes of re-encrypting the ciphertext, which would not reveal anything about the private key or reduce the encryption security. The additional data is encrypted by LWE encryption before FHEE-DE data hiding, or scrambled using sequence encryption by the third party before KS-LSB data hiding, which ensure the confidentiality of the additional data. During the transmission or processing by third-party servers, the third party does not obtain any information related to the client user’s private key, nor did it expose any relationship between plaintext and its corresponding ciphertext. Even if public keys used in the re-encryption are all generated by the same private key, there is a random variable participating in the generation process, i.e., $e$ in Eq. (23), thus ensuring the independence among public keys. Due to the random variable $a_i$ in Eq. (25), different ciphertext encrypted by the same public key would be also independent from each other, even if the different ciphertext were corresponding to the same plaintext. That is the advantage of the public key cryptosystem in application. In summary, the security of the proposed scheme can realize the security that LWE encryption has achieved. What is more, the security of LWE encryption reaches anti-quantum algorithm analysis, while Paillier algorithms cannot resist quantum algorithm analysis [38], [39].

C. Efficiency

1) Public Key Consumption: The key consumption of Paillier encryption based RDH-ED [20]–[26] is $O(1)$ to embed one bit additional data. As for the proposed scheme, there are three types of public keys: $A$, $B$, and $B_{LSB}$ as shown in Table II. Although they are not stored locally after generation, their consumption determines the number of the operations of key switching and bootstrapping during data hiding, which is directly related to the efficiency of FHEE-DE. Therefore, it is necessary to analyze the public key consumption in different operations in FHEE-DE. The analysis of addition circuit (Add") is as follows:

In refreshing $i$ ($i = 1, 2, \ldots, 8$): There are 1 bit-addition and $\sum_{\mu=1}^{i-1}$ bit-multiplication. Therefore, the total amount of homomorphic addition is 8, and the total amount of homomorphic multiplication is 84. Following each homomorphic multiplication, there would be once more key-switching. The bootstrapping was implemented for every 10 homomorphic multiplication or 100 homomorphic addition, thus resulting in 93 public keys consumed in Add".

In the same way, we got Table V.

To embed one bit of additional data, there are $7$ add" and $3$ sub" in universal FHEE-DE and 1 add" and 0 sub" in efficient FHEE-DE. Obviously, compared with the universal FHEE-DE, the key consumption is reduced from $O(100)$ to $O(10)$ by efficient FHEE-DE. Therefore, the number of key switching and bootstrapping can be also reduced. Since the computational complexity and the elapsed time of bootstrapping are much higher than other processes, the efficient FHEE-DE has higher operational efficiency than universal FHEE-DE.

The universal FHEE-DE starts from the encrypted image itself, which requires no preprocessing on the image before encryption. However, the efficient FHEE-DE requires obtaining the means and differences of the pixel pairs as plaintext rather than the original pixels. So in terms of technical applicability, the technical scalability of universal FHEE-DE is better than the efficient one.

The secondary expansion of ciphertext needs to be eliminated by using key-switching technique. A new ciphertext can be obtained by performing only once matrix multiplication between a switching key and the old ciphertext.

KS-LSB data hiding is to randomly change the LSB of specific ciphertext by key switching until the LSB is the same as the to-be-embedded bit. Let the number of times of key switching performed for one bit embedding be $\lambda$, that is, the public key consumption of KS-LSB for one bit embedding is $\lambda$. Since the LSB of the ciphertext is 0 or 1 randomly appeared with a probability of 0.5, $\lambda+1$ obeys the geometric distribution as shown in Table VI. It demonstrates that it would be a small probability event with a probability less than 3% to operate more than 4 times key switching to realize one bit embedding. The theoretical value of $\lambda$ is 0.8906. We performed 1000 KS-LSB data hiding tests to obtain that the actual $\lambda$ was 0.995 on average.

2) Cost of Storage and Elapsed Time: The public key encryption algorithms, such as the Paillier algorithm and
TABLE VI

| \( A \) | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| \( \beta \) | 0.5 | 0.25 | 0.125 | 0.0625 | 0.0313 | 0.0156 |

TABLE VII

| Operation | Encryption | Decryption | Key-switching | Bootstrapping |
|---|---|---|---|---|
| Elapsed time (ms) of Once Operation in FHEE-DE | 2.0597 | 0.0067 | 0.1054 | 0.4922 |

the LWE algorithm, have ciphertext extension, which was discussed in detail in [28]; under the condition of non-quantum computing, to reach the same security level of LWE encryption using \( n = 240 \), Paillier encryption based RDH-ED [20]–[26] have an extension of 128–256. LWE encryption based RDH-ED [27]–[29] have an extension of 256. The proposed FHEE-DE scheme has an extension of 256.

Due to the application of the separability of RDH-ED, ciphertext is usually stored in the server or the cloud, the local storage cost of users is not too much. However, the elapsed time of encryption, decryption, data hiding, and extraction is related to the efficiency of users in practice. \( n \) is the length of the private key. The calculation complexity of Paillier encryption is \( O(n^3) \) while that of LWE encryption is \( O(n^2) \) [39]. In FHEE-DE, the calculation complexities of encryption, key switching and bootstrapping are \( O(n^2), O(n) \) and \( O(n^3) \) [33], [34], [36]. The brief structure and linear operations of LWE provide LWE based algorithms with a lower time consumption than Paillier encryption [39], which might be significant in practice.

We demonstrate the average elapsed time of 1000 tests of the main operations in Table VII. The elapsed time is specifically the time (milliseconds) when one bit plaintext gets decrypted, or one public key is generated and consumed for key-switching or bootstrapping. The elapsed time of bootstrapping was obtained on the GPU card, and the others were obtained on the CPU card. The settings are as mentioned in Section V.A.

VI. CONCLUSION

The main contributions of this paper are as following:

1. We propose a FHEE-DE scheme to realize data hiding in encrypted domain based on LWE. The homomorphic circuits and ciphertext operations are elaborated. A fidelity constraint is introduced to enhance the PSNR1 of the marked plaintext. Techniques of key-switching and bootstrapping are utilized to control the ciphertext extension and decryption failure.

2. KS-LSB data hiding in the encrypted domain has been proposed, which supports the data extraction directly from the encrypted domain without the private key.

3. We propose an efficient version of FHEE-DE, which could improve the efficiency compared with the universal FHEE-DE by simplifying FHE operations. The technical scalability is not as good as the universal FHEE-DE.

Experimental results demonstrate that the performances of the proposed scheme in EC and reversibility are superior to most existing RDH-ED methods, and fully separability was achieved without reducing the security of LWE encryption. Future investigation will focus on introducing more RDH methods in image spatial domain into the encrypted domain and optimizing the technique of FHEE-DE to further improve the efficiency.

REFERENCES

[1] 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, no. 3, pp. 553–562, Mar. 2013.
[2] 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.
[3] X. Zhang, “Reversible data hiding in encrypted image,” IEEE Signal Process. Lett., vol. 18, no. 4, pp. 255–258, Apr. 2011.
[4] 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.
[5] 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.
[6] Z. Qian, X. Zhang, and S. Wang, “Reversible data hiding in encrypted JPEG bitstream,” IEEE Trans. Multimedia, vol. 16, no. 5, pp. 1486–1491, Aug. 2014.
[7] W. Puech, M. Chaumont, and O. Strauss, “A reversible data hiding method for encrypted images,” Proc. SPIE, vol. 6819, Mar. 2008, Art. no. 68191E.
[8] W. Zhang, K. Ma, and N. Yu, “Reversibility improved data hiding in encrypted images,” Signal Process., vol. 94, pp. 118–127, Jan. 2014.
[9] M. Lü, 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.
[10] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, “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.
[11] X. Zhang, “Separable reversible data hiding in encrypted image,” IEEE Trans. Inf. Forensics Security, vol. 7, no. 2, pp. 826–832, Apr. 2012.
[12] H.-Z. Wu, Y.-Q. Shi, H.-X. Wang, and L.-N. Zhou, “Separable reversible data hiding for encrypted palette images with color partitioning and flipping verification,” IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 8, pp. 1620–1631, Aug. 2017.
[13] 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, no. 7, pp. 1670–1681, Jul. 2018.
[14] F. Huang, J. Huang, and Y.-Q. Shi, “New framework for reversible data hiding in encrypted domain,” IEEE Trans. Inf. Forensics Security, vol. 11, no. 12, pp. 2777–2789, Dec. 2016.
[15] J. Tian, “Reversible data embedding using a difference expansion,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 8, pp. 890–896, Aug. 2003.
[16] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, “Reversible data hiding,” IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 3, pp. 354–362, Mar. 2006.
[17] B. Ou, X. Li, Y. Zhao, R. Ni, and Y.-Q. Shi, “Pairwise prediction-error expansion for efficient reversible data hiding,” IEEE Trans. Image Process., vol. 22, no. 12, pp. 5010–5021, Dec. 2013.
[18] X. Li, B. Yang, and T. Zeng, “Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection,” IEEE Trans. Image Process., vol. 20, no. 12, pp. 3524–3533, Dec. 2011.
[19] N. J. AlFardan et al. (2013), On the Security of RC4 in TLS and WPA. [Online]. Available: http://cr.yp.to/streamchiper/rc4biases-20130708.pdf
[20] Y.-C. Chen, C.-W. Shia, and G. Hong, “Encrypted signal-based reversible data hiding with public key cryptosystem,” J. Vis. Commun. Image Represent., vol. 25, no. 5, pp. 1164–1170, Jul. 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 Process., Image Commun., vol. 39, pp. 226–233, Nov. 2015.

[22] X. Wu, B. Chen, and J. Weng, “Reversible data hiding for encrypted signals by homomorphic encryption and signal energy transfer,” J. Vis. Commun. Image Represent., vol. 41, pp. 58–64, Nov. 2016.

[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] H.-T. Wu, Y.-M. Cheung, and J. Huang, “Reversible data hiding in Paillier cryptosystem,” J. Vis. Commun. Image Represent., vol. 40, no. 10, pp. 765–771, Oct. 2016.

[25] M. Li and Y. Li, “Histogram shifting in encrypted images with public key cryptosystem for reversible data hiding,” Signal Process., vol. 130, no. 1, pp. 190–196, Jan. 2017.

[26] 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. 2099–2110, Nov. 2018.

[27] Y. Ke, M. Zhang, and J. Liu, “Separable multiple bits reversible data hiding in encrypted domain,” in Proc. Digit. Forensics Watermarking-15th Int. Workshop (IWWD) LNCS, vol. 10082, Beijing, China, 2016, pp. 470–484.

[28] Y. Ke, M.-Q. Zhang, J. Liu, T.-T. Su, and X.-Y. 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.

[29] L. Xiong, D. Dong, Z. Xia, and X. Chen, “High-capacity reversible data hiding for encrypted multimedia data with somewhat homomorphic encryption,” IEEE Access, vol. 6, pp. 60635–60644, 2018.

[30] P. Paillier and D. Pointcheval, “Efficient public-key cryptosystems provably secure against active adversaries,” in Advances in Cryptology—ASIACRYPT’99 (Lecture Notes in Computer Science), vol. 1716. Berlin, Germany: Springer, 1999, pp. 165–179.

[31] E. Chrysochos, V. Fotopoulos, and A. N. Skodras, “A new difference expansion transform in triplets for reversible data hiding,” Int. J. Comput. Math., vol. 88, no. 10, pp. 2016–2025, Jul. 2011.

[32] T.-C. Lu and C.-C. Chang, “Lossless nibbled data embedding scheme based on difference expansion,” Image Vis. Comput., vol. 26, no. 5, pp. 632–638, May 2008.

[33] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in Proc. 41st ACM Symp. Theory Comput. (STOC), vol. 9, 2009, pp. 169–178.

[34] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(Leveled) fully homomorphic encryption without bootstrapping,” ACM Trans. Comput. Theory, vol. 6, no. 3, pp. 1–36, Jul. 2014.

[35] C. Gentry, S. Halevi, and N. P. Smart, “Better bootstrapping in fully homomorphic encryption,” in Proc. Int. Workshop on Public Key Cryptogr., LNCS, vol. 7293, 2012, pp. 1–16.

[36] I. G. N. Chilotti et al., “Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds,” in Advances in Cryptology—ASIACRYPT 2016 (Lecture Notes in Computer Science), vol. 10031. Berlin, Germany: Springer, 2016, pp. 3–33.

[37] N. Gama and P. Q. Nguyen, “Predicting lattice reduction,” in Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn., Istanbul, Turkey, 2008, pp. 31–51.

[38] M. Ruckert and M. Schneider. (2010). Estimating the Security of Latticed-Based Cryptosystems. [Online]. Available: http://eprint.iacr.org/2010/137.pdf

[39] J. Hoffstein, J. Pipher, and J. H. Silverman, “NTRU: A ring-based public key cryptosystem,” in Algorithmic Number Theory (Lecture Notes in Computer Science), vol. 1423. Berlin, Germany: Springer, 1998, pp. 267–288.