Paillier Cryptosystem based Mean Value Computation for Encrypted Domain Image Processing Operations

MOHSIN SHAH, WEIMING ZHANG, HONGGANG HU, and NENGHAI YU, University of Science and Technology of China, China

Due to its large storage facility and high-end computing capability, cloud computing has received great attention as a huge amount of personal multimedia data and computationally expensive tasks can be outsourced to the cloud. However, the cloud being third-party semi-trusted, is prone to information leakage, raising privacy risks. Signal processing in the encrypted domain has emerged as a new research paradigm on privacy-preserving processing over outsourced data by semi-trusted cloud. In this article, we propose a solution for non-integer mean value computation in the homomorphic encrypted domain without any interactive protocol between the client and the service provider. Using the proposed solution, various image processing operations, such as local smoothing filter, un-sharp masking, and histogram equalization, can be performed in the encrypted domain at the cloud server without any privacy concerns. Our experimental results from standard test images reveal that these image processing operations can be performed without pre-processing, without client-server interactive protocol, and without any error between the encrypted domain and the plain domain.

CCS Concepts: • **Security and privacy;** • **Computer systems organization** → *Cloud computing*;

Additional Key Words and Phrases: Encrypted domain, Paillier cryptosystem, cloud computing, secure signal processing (SSP)

ACM Reference format:
Mohsin Shah, Weiming Zhang, Honggang Hu, and Nenghai Yu. 2019. Paillier Cryptosystem based Mean Value Computation for Encrypted Domain Image Processing Operations. *ACM Trans. Multimedia Comput. Commun. Appl.* 15, 3, Article 76 (September 2019), 21 pages.
https://doi.org/10.1145/3325194

1 INTRODUCTION

Cloud computing is an advanced information technology (IT) infrastructure that provides on-demand ubiquitous access to a pool of configurable computing resources—from high-end processing power, storage, networking, and artificial intelligence to natural language processing. In recent years, cloud computing has gained tremendous popularity and growth in business as well as in academia. Reliability, economy, scalability, productivity, speed, and performance are some of

This work was supported in part by the Natural Science Foundation of China under Grants No. U1636201 and No. 61572452, and by Anhui Initiative in Quantum Information Technologies under Grant No. AHY150400. The authors are also grateful to CAS-TWAS President's Fellowship for supporting this work.
Authors' address: M. Shah, W. Zhang, H. Hu, and N. Yu, University of Science and Technology of China, School of Information Science and Technology, Hefei, Anhui, 230026, China; emails: mohsin@mail.ustc.edu.cn, {zhangwm, hghu2005, ynh}@ustc.edu.cn.

1551-6857/2019/09-ART76 $15.00
https://doi.org/10.1145/3325194

the advantages that motivated individuals and organizations to outsource their huge amount of sensitive private data and computationally expensive tasks onto cloud servers. Another benefit of using cloud computing is that organizations can avoid the cost of installation of IT infrastructure, updating operating systems, decommissioning, and disposing of software and hardware when it is out of date (Alhamazani et al. 2015).

Software-as-a-Service (SaaS) is an important service model among the three cloud computing models defined by the National Institute of Standards and Technology (NIST) (Mell et al. 2011). SaaS allows clients to use cloud's applications on demand. Clients can access these applications using thin client interfaces such as web-based email or mobile terminals without knowing or managing the underlying infrastructure such as servers, storage, operating systems or network. In recent years, various photo editing and image enhancement applications, such as Pixlr photo editor (PIXLR n.d.) and Adobe Creative Cloud (AdobeCreativeCloud n.d.), have been using the SaaS services (Ziad et al. 2016). Clients can upload private photographs from their personal devices and can apply different image enhancement tools online. However, while enjoying the online image processing services, outsourcing of private photographs raises serious privacy risks as the cloud server can leak personal information of clients (Lathey and Atrey 2015; Ziad et al. 2016). Most recently, the leakage of private photographs of several Hollywood actresses from the iCloud (Shah et al. 2018) is an example of the privacy breach at the cloud. A straightforward solution to this problem is to encrypt private photographs at the client end with conventional encryption algorithm, such as Advance Encryption Standard (AES) (Pub 2001), before outsourcing it to the cloud. However, the encrypted photographs are first decrypted before any processing and then again encrypted after processing. This solution is applicable when encryption protects private content against attackers while the client and the cloud sever trust each other. Consequently, conventional encryption algorithms are not suited to privacy-preserving cloud-based paradigm as they present challenges for secure signal processing (SSP).

With the increasing demand for protecting outsourced content, SSP has emerged as privacy-preserving signal processing field. SSP allows cloud severs to perform signal processing operations directly on encrypted content without decrypting it (Lagendijk et al. 2013; Troncoso-Pastoriza and Perez-Gonzalez 2013). Recently, various image transforms in the homomorphic encrypted domain have been proposed (Bianchi et al. 2009a, 2009b; Zheng and Huang 2012, 2013). Bianchi et al. proposed the implementation of discrete cosine transform (DCT) in Bianchi et al. (2009a) and the implementation of disctete Fourier transform (DFT) in Bianchi et al. (2009b) in the encrypted domain with Pailler cryptosystem (Paillier 1999). Zheng and Huang implemented discrete wavelet transform (DWT) in Zheng and Huang (2013) and Walsh Hadamard transform (WHT) in Zheng and Huang (2012) with Paillier cryptosystem. Most recently, privacy preserving image processing operations in the encrypted domain with homomorphic public key cryptosystem have emerged (Chen et al. 2018; Mohanty et al. 2016; Rajput and Raman 2018; Ziad et al. 2016). In Mohanty et al. (2016), a method of image scaling and cropping is proposed in the encrypted domain with modified Paillier cryptosystem. This method used bilinear interpolation for scaling. First, bilinear interpolation is computed in the plain domain, then image is divided into tiles and image tiles are encrypted. Encrypted domain image scaling is performed at the cloud server using the multiplication and addition homomorphic properties. Floating point numbers involved in bilinear interpolation are multiplied by a large scaling factor and then rounded to integers to be used with Paillier cryptosystem. This scaling and rounding introduced errors between the plain domain and encrypted domain scaling operations. Later on, various image processing operations such as spatial filtering, edge sharpening and histogram equalization were proposed by Ziad et al. (2016) in the homomorphic encrypted domain with Paillier cryptosystem. Ziad et al. represented floating point numbers in the encrypted domain by encrypted mantissa and un-encrypted exponent. The

division operation in smoothing filter and histogram equalization is realized by a quantization encoding function. Because of the use of quantization encoding function, the method introduced errors between the encrypted domain operations and plain domain operations for local smoothing filtering and edge sharpening. Privacy preserving color transfer in the encrypted domain with Paillier cryptosystem is proposed in Rajput and Raman (2018). The client encrypts the individual channels of the reference image and color transformation values of the target image and send them to cloud server where color transformation is achieved using the homomorphic properties. In Chen et al. (2018), edge detection and image segmentation is proposed in the encrypted domain using Paillier cryptosystem and garbled circuit. The client sends the encrypted image to cloud server where encrypted domain Gaussian filter is applied to the encrypted image to get a smoothed image in the encrypted domain and then encrypted domain Sobel filter is used to detect the edges in the encrypted domain. Gaussian and Sobel filters are approximated to their discrete versions to be implemented with Paillier cryptosystem. Garbled circuit is used for threshold comparison in the encrypted domain for obtaining edges without decrypting the image.

Recently, some privacy preserving image processing methods based on Shamir's Secret Sharing (SSS) scheme have been proposed (Lathey and Atrey 2015; Mohanty et al. 2013; Singh et al. 2018). Mohanty et al. (2013) proposed image scaling and cropping using secret image sharing. Bilinear interpolation is used for scaling images in encrypted domain. The same method as used in Mohanty et al. (2016) for converting floating point numbers to integers is adopted in Mohanty et al. (2013) to implement bilinear interpolation in the encrypted domain. Lathey and Atrey (2015) proposed a method to perform several image processing operations such as spatial filtering, anti-aliasing, edge detection, contrast enhancement, and de-hazing. The original image is encrypted using SSS and the secret shares are distributed among multiple servers in the cloud. Lathey and Atrey's method requires pre-processing of original image at the client side necessary to perform division operation involved in different processing operations at the cloud side. The method is theoretically secure only if the encrypted domain processing operations are performed using n servers with no more than k are colluding. The requirement of non-colluding servers makes this method impractical. Also, the results of the encrypted domain operations performed by this method differs from the results of the plain domain operations. Another privacy preserving method for image processing operations based on SSS and permutation ordered binary (POB) numbers is proposed by Singh et al. (2018). The client creates different shares of the original image using SSS and then further creates two shares for every shares using the POB. The resultant shares are then transformed into frequency domain using the Fourier transform and different low pass and high pass filters are applied in the frequency domain. The integration of POB numbers with SSS enhanced the security of the method and also allowed the method to perform processing directly on the encrypted shares. However, the enhanced security is achieved at the expense of increased complexity and overhead. Furthermore, the method is not error free and introduces errors for some processing operations between the encrypted domain and plain domain.

In this article, a method to perform privacy preserving image processing operations in the homomorphic encrypted domain with Paillier cryptosystem is proposed. All the previously mentioned methods involve pre-processing or encoding function to perform processing on encrypted images. Furthermore, their encrypted domain processing operations differ from the plain domain processing operations with some errors. We propose a method for non-integer mean value computation in the encrypted domain with Paillier cryptosystem without any pre-processing or interactive protocol between the client and the cloud server. Mean value computation involves division operation, which may result in a non-integer value. In this article, a solution to compute the division operation in the homomorphic encrypted domain is proposed. Furthermore, we perform certain image processing operations such as local smoothing filtering, un-sharp masking and histogram

equalization in the encrypted domain, which involve the division operation. Our experimental results from standard test images reveal that these operations can be performed without errors between the encrypted and plain domain, which demonstrates the effectiveness of our method.

The rest of the article is organized as follows. Section 2 briefly describes preliminaries, including homomorphic Paillier cryptosystem, its properties, lattice theory, and basis reduction. Mean value computation in the encrypted domain is elaborated in Section 3. Section 4 is dedicated to the applications of encrypted domain mean in image processing. Experimental results, computational complexity, security analysis, and comparisons are presented in Section 5. Finally, the conclusion is drawn in Section 6.

2 PRELIMINARIES

2.1 Homomorphic Paillier Cryptosystem

The Paillier cryptosystem, which holds both homomorphic and probabilistic properties (Goldwasser and Micali 1984), is based on the computational hardness of Nth residue modulo N^2 of a number. In this article, we adopt Paillier cryptosystem for image encryption and decryption. We direct the readers to Paillier (1999) for a complete mathematical description of the Paillier cryptosystem. Here, we discuss only key setup, encryption and decryption.

Key setup: Select two large prime integers p and q and $N = p \cdot q$, where N is the modulus of the cryptosystem. Let Z_N be the set of integers modulo N, Z_N^* is a subset of Z_N such that the elements of Z_N^* are relatively prime to N, Z_{N^2} be the set of integers modulo N^2 and $Z_{N^2}^*$ is a subset of Z_{N^2} relatively prime to N^2. Randomly select an integer g from $Z_{N^2}^*$. g and N together makes the public key. At the receiver side, given p and q, the receiver first computes $\lambda = lcm(p-1, q-1)$, where lcm stands for least common multiple, and then computes $k = \frac{(g^\lambda \bmod N^2 - 1)}{N}$. The modular multiplicative inverse of k denoted by μ is computed by Equation (1):

$$\mu = k^{-1} \bmod N. \tag{1}$$

λ is the private key.

Encryption: Let $m \in Z_N$ is the plaintext, $c \in Z_{N^2}$ is the corresponding ciphertext. Randomly select r from Z_N^*, then encryption is computed as

$$c = \mathfrak{E}[m, r] = g^m \cdot r^N \bmod N^2. \tag{2}$$

Decryption: Given the private key and ciphertext, decryption is computed as

$$\mathfrak{D}[\mathfrak{E}[m, r]] = m = \frac{(c^\lambda \bmod N^2 - 1)}{N} \cdot \mu \bmod N. \tag{3}$$

2.2 Homomorphic Properties of the Paillier Cryptosystem

The Paillier cryptosystem makes use of Carmichael's Theorem (Carmichael 1913) and has several interesting homomorphic properties, which are given as follows:

$$\mathfrak{D}[\mathfrak{E}[m_1, r_1] \cdot \mathfrak{E}[m_2, r_2]] \bmod N^2] = m_1 + m_2 \bmod N, \tag{4}$$

$$\mathfrak{D}[\mathfrak{E}[m_1, r_1] \cdot \mathfrak{E}[m_2, r_2]^{-1}] \bmod N^2] = m_1 - m_2 \bmod N, \tag{5}$$

$$\mathfrak{D}[\mathfrak{E}[m, r]]^k \bmod N^2] = k \cdot m \bmod N, \tag{6}$$

$$\mathfrak{D}[\mathfrak{E}[m, r]^{k^{-1}} \bmod N^2] = \frac{m}{k} \bmod N, \tag{7}$$

$$\mathfrak{D}[[\mathfrak{E}[m] \cdot P^N] \bmod N^2] = m \bmod N, \tag{8}$$

where k^{-1} is the modular multiplicative inverse of k and P is an integer relatively prime to N. The property given in Equation (8) is termed as self-blinding property. Although the homomorphic Paillier cryptosystem does not allow division operation on encrypted data, because division can lead to float results, we are still able to perform division on encrypted data when the divisor k is a plain value. The property in Equation (7) holds if

(1) The modular multiplicative inverse of k exists, i.e., $\gcd(k, N) = 1$;
(2) m is perfectly divisible by k, i.e., the quotient after division is an integer.

Previous methods cannot perform non-integer mean value computation in encrypted domain because of the limitations of cryptosystems. In this article, we propose a method to perform non-integer mean computation using Equations (4) and (7). We can still obtain the correct result from Equation (7) even if m is not perfectly divisible by k, which will be discussed in Section 3.

2.3 Lattice and Basis Reduction

A lattice is a discrete set of regular arrangement of points in $\mathbb{R}^m$, where $\mathbb{R}$ is the set of real numbers and m represents dimension. A vector space is generated by vectors such that the vectors can be added together and multiplied by real numbers. However, lattice is a discrete subgroup of $\mathbb{R}^m$ generated by a linear combination of its vectors in integer coefficients.

Lattices are used in numerous applications. Due to the hard computational problems associated with lattices, the most prominent application of lattices is in modern cryptosystems. Some of these cryptosystems include the LWE cryptosystems (Regev 2009), Gentry's fully homomorphic cryptosystem (Gentry and Halevi 2011), and the NTRU cryptosystem (Hoffstein et al. 1998). Lattices are used to break public key cryptosystems such as the cryptanalysis of RSA and Knapsack cryptosystem using lattice attacks. The bit security of some cryptosystems can be analyzed using lattices, for example, using the hidden number problem to analyze the bit security of Diffie-Hellman key exchange.

Let $\{b_1, b_2, \ldots, b_n\} \in \mathbb{R}^m$ be a set of n linearly independent vectors, the linear combination of $\{b_1, b_2, \ldots, b_n\}$ generates the lattice $\mathcal{L}$ as

$$\mathcal{L}(b_1, b_2, \ldots, b_n) = \{a_1 b_1 + a_2 b_2 + \cdots + a_n b_b\} = \sum_{i=1}^{n} a_i b_i, \tag{9}$$

where $a_i \in \mathbb{Z}$. The vectors $\{b_1, b_2, \ldots, b_n\}$ that generates $\mathcal{L}$ are called a lattice basis.

Basis reduction is the problem of finding a basis with short and almost orthogonal vectors. Basis reduction is proved invaluable in the fields of cryptology (Nguyen and Stern 2001) and many other fields of mathematics and computer science. Lenstra et al. (1982) used lattice reduction to decompose a polynomial with rational coefficients to irreducible factors. Ritter and Rössner (1997) used lattice reduction for factoring large composite numbers. Poupard and Stern (2000) proposed a recovery system for RSA keys using lattice reduction and rationals. The idea of lattice reduction in $\mathbb{R}^2$ was first presented by Lagrange (1775) and later by Gauss (1966), but combined it is called Lagrange-Gauss basis reduction algorithm. Algorithm 1 illustrates the basis reduction process of Lagrange-Gauss algorithm. In Algorithm 1, $\|\cdot\|$ is the Euclidean norm (ℓ_2 norm), $\langle b, b \rangle$ is the inner product, and $\lfloor \cdot \rceil$ is a function that rounds toward the nearest integer. We propose a more efficient basis reduction algorithm using the extended Euclidean algorithm, which will be described in Section 3.

3 MEAN VALUE COMPUTATION BASED ON PAILLIER CRYPTOSYSTEM

The mean of a discrete set of numbers is the central value. The mean of a set of numbers $v_1, v_2, \ldots, v_k$, denoted by $\bar{m}$, is the sum of the values divided by the total number of elements

ALGORITHM 1: Lagrange-Gauss lattice reduction algorithm

Input: Basis $b_1, b_2 \in Z^2$ of lattice L
Output: Reduced basis $b_1, b_2 \in Z^2$ of lattice L

```
B1 = ||b1||^2
μ = ⌊⟨b1, b2⟩ / B1⌉
b2 = b2 − μ · b1
B2 = ||b2||^2
while B2 < B1 do
    Swap b1 and b2
    B1 = B2
    μ = ⌊⟨b1, b2⟩ / B1⌉
    b2 = b2 − μ · b1
    B2 = ||b2||^2
end while
return (b1, b2)
```

in the set:

$$\bar{m} = \frac{1}{k} \cdot \sum_{i=1}^{k} v_i = \frac{v_1 + v_2 + \cdots + v_k}{k}. \tag{10}$$

The mean $\bar{m}$ may result in integer or float value. Using the homomorphic properties of Paillier cryptosystem given in Equations (4) and (7), the mean can be computed in the encrypted domain as

$$\mathfrak{E}[\bar{m}] = \left(\prod_{i=1}^{k} \mathfrak{E}[v_i] \bmod N^2 \right)^{k^{-1} \bmod N} \bmod N^2. \tag{11}$$

It is worth mentioning to establish the fact that the modular multiplicative inverse of k and the associated multiples always exist given the modulus of the Paillier cryptosystem N. The modular multiplicative inverse of a number x exists if the number is relatively prime to N, i.e., $\gcd(x, N) = 1$. According to the Euler's totient function ϕ, the set of integers that are relatively prime to N is given as

$$\phi(N) = \phi(p) \cdot \phi(q) = S, 1 \leq S \leq N - 1 \tag{12}$$

and

$$S \in Z^+ - \{p, q, pq, ap, aq, apq\},$$

where Z^+ is the set of positive integers and $a \in Z^+$. From Equation (12), it is concluded that if p or q is not equal to k then the relative prime set will contain k and its multiples. Usually, p and q are selected large enough to make N 1,024-bits long, which is a necessary security requirement for the protection of the cryptosystem against attacks. Hence, the modular multiplicative inverse of k and its multiples will always exist.

The encrypted domain mean, computed using Equation (11), when decrypted may result in either the correct mean or large integer depending on whether the plain domain mean is an integer or a float. Let us use Equation (11) to compute the mean of two sets of numbers $s_1 = \{2, 3, 5, 6\}$ and $s_2 = \{2, 4, 5, 6\}$ in the encrypted domain as illustrated in Figures 1(a) and 1(b). For brevity, we use small integers for computing the public key ($N = 117852727$) and fixed random number r for encryption. Figure 1(a) illustrates encrypted domain mean value computation when the corresponding plain domain mean is an integer while Figure 1(b) demonstrates mean value computation

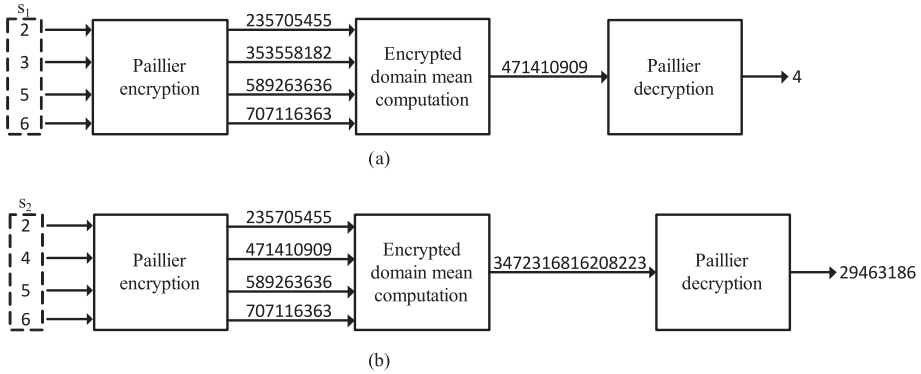


Fig. 1. Encrypted domain mean value calculation. (a) When plain domain mean is an integer. (b) When plain domain mean is a float.

when the corresponding plain domain mean is a float. Note that in Figure 1(b) the decrypted mean value is a large integer, which does not make sense. From mean value computation using Equation (11), it is concluded that non-integer mean value calculation in the encrypted domain will lead to large integer upon decryption. The main contribution of this work is to reduce the large integer value to the correct float mean value.

ALGORITHM 2: Pixel value reduction based on modified extended Euclidean algorithm

INPUT: N, U
OUTPUT: Reduced U

$[u_1, u_2] = [0, N]$
$[v_1, v_2] = [1, U]$
$q = \lfloor u_2/v_2 \rceil$
$[t_1, t_2] = [u_1, u_2] - q \cdot [v_1, v_2]$
$[u_1, u_2] = [v_1, v_2]$
$[v_1, v_2] = [t_1, t_2]$
while $u_2 > \sqrt{N}$ **do**
    $q = \lfloor u_2/v_2 \rceil$
    $[t_1, t_2] = [u_1, u_2] - q \cdot [v_1, v_2]$
    $[u_1, u_2] = [v_1, v_2]$
    $[v_1, v_2] = [t_1, t_2]$
end while
return $U = \lfloor u_2/u_1 \rfloor$

The large integer value in Figure 1(b) does represent the mean value. The large integer value can be reduced to the correct mean value using the two-dimensional lattice theory as mentioned in Section 2.3. Reducing large integer value to the correct mean value in the range $[0, 255]$ (in the case of images) is a two-dimensional lattice reduction problem and Lagrange-Gauss lattice reduction algorithm can be used for reducing large pixel values to optimal values. Here, we propose a more efficient algorithm using extended Euclidean algorithm. We modify the extended Euclidean algorithm to take vector inputs and reduce these vectors to short orthogonal vectors. We refer the readers to Galbraith (2012) for a detailed understanding of the extended Euclidean algorithm. We consider the modulus of Paillier cryptosystem N and large integer value U as independent

Table 1. Step-by-step Results of the Modified Extended Euclidean Algorithm

Step	(u_1, u_2)	(v_1, v_2)	q
1	(1, 29,463,186)	(−3, 29,463,169)	3
2	(−3, 29,463,169)	(4, 17)	1
3	(4, 17)	(−6, 932, 511, 10)	1,733,127

points in a two-dimensional lattice space $\mathcal{L}$. Given N and U, it can be deduced that the vectors $(0, N)$ and $(1, U)$ form a basis for the lattice $\mathcal{L}$. These two basis vectors can be reduced for optimal values. Algorithm 2 adapts the extended Euclidean algorithm for computing the reduced value of U. Algorithm 2 starts with first taking the basis vectors $(0, N)$ and $(1, U)$ and computes the integer quotient q. In the first round, the algorithm computes the reduced vector (t_1, t_2), and then updates u_1 by v_1, u_2 by v_2, v_1 by t_1 and v_2 by t_2. The algorithm goes further if $u_2 > \sqrt{N}$ until we get the reduced basis vectors (u_1, u_2) and (v_1, v_2) in which case $u_2 < \sqrt{N}$. Finally the correct integer mean value is computed using step 13 of the algorithm.

Example: To facilitate understanding of the reduction process, we elaborate it with an example and consider reducing the large integer value illustrated in Figure 1(b). For the brevity of demonstration, we use small prime numbers $p = 10{,}853, q = 10{,}859$ and $N = p \cdot q = 117{,}852{,}727$. According to Algorithm 2, we let $u_1 = 0, u_2 = 117{,}852{,}727$ and $v_1 = 1, v_2 = 29{,}463{,}186$ and $\sqrt{N} = \sqrt{117{,}852{,}727} \approx 10{,}856$. Compute $q = \lfloor u_2/v_2 \rceil = 3$ and $t_1 = u_1 - q \cdot v_1 = -3$, $t_2 = u_2 - q \cdot v_2 = 29{,}463{,}169$, then update $u_1 = v_1 = 1$, $u_2 = v_2 = 29{,}463{,}186$ and $v_1 = t_1 = -3$, $v_2 = t_2 = 29{,}463{,}169$. Since $u_2 > \sqrt{N}$, we compute $q = \lfloor u_2/v_2 \rceil = 1$ and $t_1 = u_1 - q \cdot v_1 = 4$, $t_2 = u_2 - q \cdot v_2 = 17$, then update $u_1 = v_1 = -3$, $u_2 = v_2 = 29{,}463{,}169$ and $v_1 = t_1 = 4$, $v_2 = t_2 = 17$. Again, since $u_2 > \sqrt{N}$, we repeat the process and compute $q = \lfloor u_2/v_2 \rceil = 1{,}733{,}127$ and $t_1 = u_1 - q \cdot v_1 = -6{,}932{,}511$, $t_2 = u_2 - q \cdot v_2 = 10$, then update $u_1 = v_1 = 4, u_2 = v_2 = 17$ and $v_1 = t_1 = -6{,}932{,}511, v_2 = t_2 = 10$. Since now $u_2 < \sqrt{N}$, the algorithm halts and returns the reduced value $u_2/u_1 = 17/4 = 4.25$. The integer mean value can be obtained by taking $floor$ of the reduced float value. Table 1 shows the steps of the modified extended Euclidean algorithm and corresponding values of u_1, u_2, v_1, v_2, and q.

The non-integer mean value can also be computed with a straightforward method by performing the addition in the encrypted domain at the cloud side and performing the division after decryption at the client side. This method avoids costly computations in the encrypted domain but this is unrealistic in some applications. First, in this method the client and the cloud server will require multiple rounds of interaction for exchanging the denominator information. Second, in some applications the cloud server may not disclose its processing algorithm to the client. In such applications the client may not know the denominator to perform division after decryption.

4 APPLICATIONS IN IMAGE PROCESSING IN ENCRYPTED DOMAIN

In this section, we present various low-level image processing operations performed in the encrypted domain. These secure image processing operations are described in the following subsections. Figure 2 illustrates a general architecture of the secure image processing operations in the cloud.

4.1 Local Smoothing Filter

Smoothing is essential in many image processing applications. As smoothing minimizes the effect of sharp changes in color levels, it is mainly used for noise reduction, removal of false contour, and

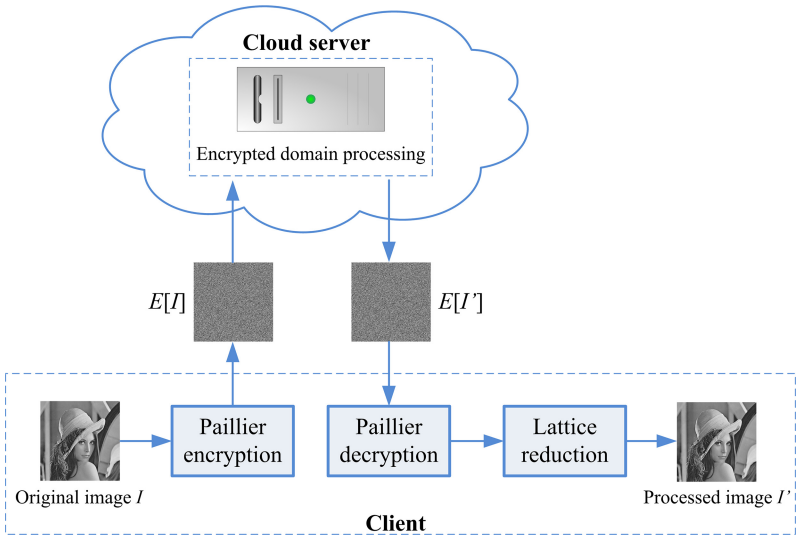


Fig. 2. General architecture of the secure image processing operations in the cloud.

removal of irrelevant details from images, especially when the object to be removed is smaller in size than the filter size.

Local smoothing filter (also called averaging filter) replaces every pixel in an image by the integer average of the pixel values in the neighborhood defined by the filter mask. Let $f(x, y)$ of size $H \times W$ be the input image containing sharp pixel transitions and $w(s, t)$ of size $h \times w$ be the filter mask, where $h = 2a + 1$ and $w = 2b + 1$ and a and b are positive integers. The resultant smoothed image $\bar{f}(x, y)$ is obtained by Equation (13):

$$\bar{f}(x, y) = \frac{1}{h \times w} \cdot \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t), \tag{13}$$

where $a = (h - 1)/$ and $b = (w - 1)/2$.

Smoothing filter can be performed in the encrypted domain using the homomorphic properties of Paillier cryptosystem. Equation (13) can be implemented in the encrypted domain using Equations (4) and (7). For brevity, we express smoothing filter operation for the smallest filter size of 3×3. For a mask of size 3×3, Equation (13) reduces to an average of 9 pixels in the neighborhood that is expressed in the encrypted domain as

$$\mathfrak{E}[\bar{f}(x, y)] = \left[\prod_{x=1}^{3} \prod_{y=1}^{3} \mathfrak{E}[f(x, y)] \bmod N^2 \right]^{9^{-1} \bmod N} \bmod N^2, \tag{14}$$

where $9^{-1} \bmod N$ is the modular multiplicative inverse of 9. The filter mask moves from pixel to pixel and computes the average of the neighborhood and replaces the center pixel value by the average value. When all the pixels in the image are processed, the resultant processed encrypted image is sent back to the client.

After receiving the processed encrypted image, the client decrypts the image with his private key. However, the directly decrypted image contains large pixel values outside the intensity range $[0, 255]$. This is because smoothing filter involve division operation that may result in float value as mentioned in Section 3. Figure 3 illustrates 7×7 pixels of directly decrypted smoothed Lena

72	108	65473845	65473845	39284349	26189602	91663339
108	162	39284404	39284404	160	39284403	78568645
108	162	39284404	39284404	160	39284403	78568645
108	162	39284404	39284404	160	39284403	78568645
104758088	162	65473898	104758140	104758139	78568645	160
108	39284404	78568645	160	104758139	52379150	13094907
78568592	52379150	65473896	78568643	65473896	52379149	13094906

Fig. 3. 7×7 pixels of directly decrypted Lena image. Pixels values in the range $[0, 255]$ are highlighted by white color and large pixels values outside the range $[0, 255]$ are highlighted by gray color.

72	108	107	107	106	107	106
108	162	161	161	160	160	160
108	162	161	161	160	160	160
108	162	161	161	160	160	160
108	162	160	160	159	160	160
108	161	160	160	159	160	159
107	160	158	158	158	159	158

Fig. 4. 7×7 reduced pixels values using modified extended Euclidean algorithm.

image. Algorithm 2 is used to reduce the large integer values of Figure 3 to the correct integer mean values. Figure 4 demonstrates the reduced 7×7 pixels values.

4.2 Sharpening Using Un-sharp Masking

Un-sharp masking is the process of subtracting a smoothed version of an image from the image itself. This method has long been used in the publishing industry for obtaining sharp images. Un-sharp masking in the plain domain can be expressed as

$$f_{us}(x, y) = f(x, y) - \bar{f}(x, y), \tag{15}$$

where $f_{us}(x, y)$ is the unsharp masked image. Using Equation (14) and modular multiplicative inverse property, Equation (15) can be implemented in the encrypted domain as

$$\mathfrak{E}[f_{us}(x, y)] = \mathfrak{E}[f(x, y)] * \mathfrak{E}[\bar{f}(x, y)]^{-1} \bmod N^2. \tag{16}$$

The directly decrypted image obtained after the application of Equation (16) is darker than desired. This is because the areas with slow varying gray levels in the original and smoothed image are identical and the subtraction operation will tend to produce dark areas with low gray levels while edge lines and other discontinuities are significantly sharpened. However, we can obtain a sharpened image with preserved features by adding the original image with the un-sharp masked image as

$$\mathfrak{E}[f_s(x, y)] = \mathfrak{E}[f(x, y)] * \mathfrak{E}[f_{us}(x, y)] \bmod N^2, \tag{17}$$

where $f_s(x, y)$ is the resultant sharp image. The encrypted sharp image $\mathfrak{E}[f_s(x, y)]$ when decrypted by the client may give rise to large pixel values due to the non-integer mean value involved in the smoothing process. The decrypted large pixel values and modulus N can be considered as lattice points and Algorithm 2 can be used to reduce large pixel values to optimal pixel values.

4.3 Histogram Equalization

Histogram of an image is the representation of the number of pixels as a function of their intensity levels. Histogram equalization is a common image processing operation used to adjust the contrast of an image. The aim is to use a transformation function to uniformly distribute the intensity levels over the whole intensity level range. Let $f(x, y)$ be an image of size $H \times W$ and $[0, L-1]$ be the intensity range of the image, then histogram of $f(x, y)$ is defined as

$$h(r_k) = n_k, k = 0, 1, \ldots, L-1, \tag{18}$$

(a) (b) (c) (d)

Fig. 5. Test images. (a) Lena, (b) Airplane, (c) Pirate, (d) Lake.

where r_k is the kth intensity level and n_k is the number of pixels at the kth intensity level. Let r be the intensity level of the image to be transformed and s be the intensity level of the transformed image, then histogram equalization is expressed as a transformation function as given in Equation (19):

$$s_k = T(r_k) = \frac{(L-1)\sum_{i=0}^{k}(n_i)}{HW}. \tag{19}$$

Histogram equalization given in Equation (19) can be implemented in the encrypted domain. The client computes the histogram of the original image using Equation (18), then encrypts the computed histogram and sends it to the cloud server. The cloud server, without knowing anything about the encrypted histogram statistics, implements Equation (19) in the encrypted domain using the properties of the Paillier cryptosystem as

$$\mathfrak{E}[s_k] = \left[\left[\prod_{i=0}^{k} n_i\right]^{(L-1)} \bmod N^2\right]^{(HW)^{-1} \bmod N} \bmod N^2. \tag{20}$$

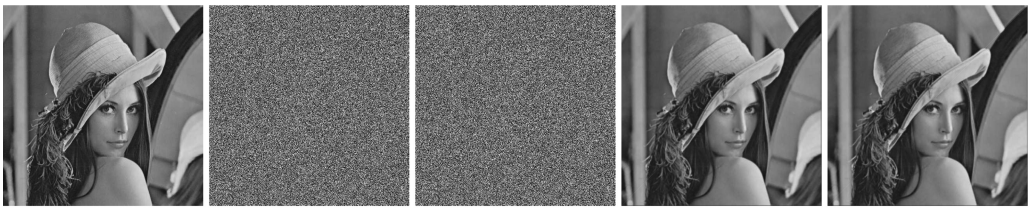
The plain domain transformation function given in Equation (19) involves division by HW. This division operation may result in float values. Because of the division operation, the encrypted domain transformation function may give rise to large values when it is decrypted by the client. Thanks to the lattice theory and basis reduction algorithm, modulus N and decrypted equalized histogram values can be considered as lattice points. Algorithm 2 can be used to reduce decrypted equalized histogram values to optimal values. After obtaining the optimal equalized histogram, the client can apply it to the input image to obtain a contrast adjusted image.

5 EXPERIMENTAL RESULTS AND DISCUSSION

In this section, experiments results are demonstrated to evaluate the performance and effectiveness of the proposed processing operations. Fifty gray-scale images sized 512×512 are selected from the public *CVG-UGR* database[1] for experiments. We show experimental results for only four images Lena, Airplane, Pirate, and Lake. These images are illustrated in Figure 5. All the implementation programs are developed in the experimental environment of MATLAB2016a under 64-bit desktop with Intel(R) Core i7 CPU @3.60 and 8GB of RAM. The modulus of the Paillier cryptosystem is selected 1,024 bits long to ensure minimum security requirement of Paillier cryptosystem.

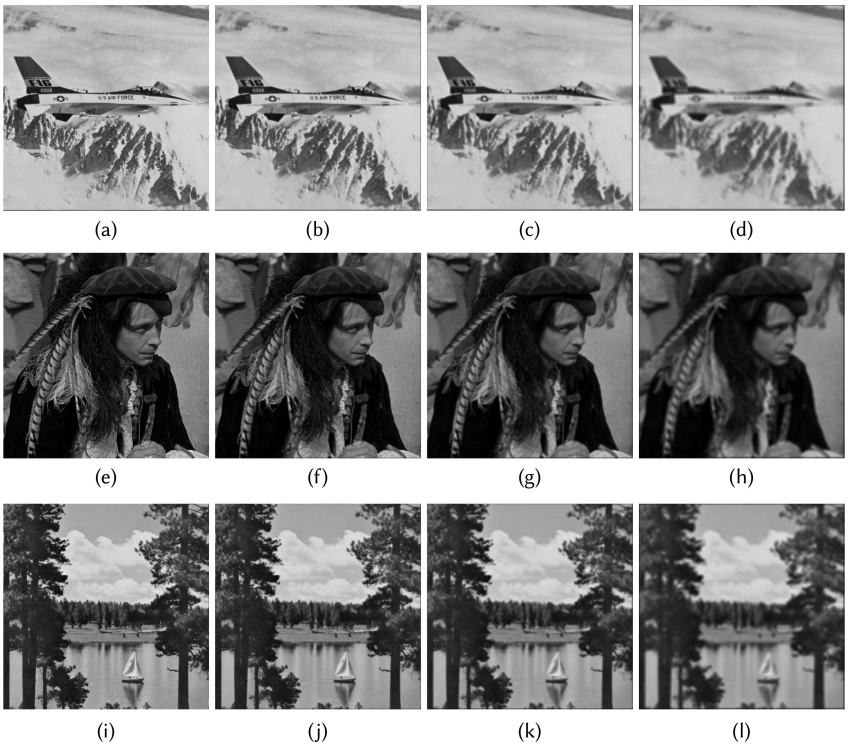
Figure 6 illustrates experimental results for 3×3 smoothing filter in the encrypted domain and plain domain for Lena image. Figure 6(a) is the original Lena image, Figure 6(b) is the encrypted image, Figure 6(c) is the encrypted domain smoothed image, Figure 6(d) is the directly decrypted restored image, and Figure 6(e) is the plain domain smoothed image. It can be observed that Figures 6(d) and 6(e) are visually exactly the same images. Experimental results for various sizes

[1] http://decsai.ugr.es/cvg/dbimagenes/.



(a) (b) (c) (d) (e)

Fig. 6. Experimental results of smoothing filter of size 3×3 for Lena image. (a) Original Lena image, (b) encrypted image, (c) encrypted domain smoothed image, (d) directly decrypted restored image, (e) plain domain smoothed image.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

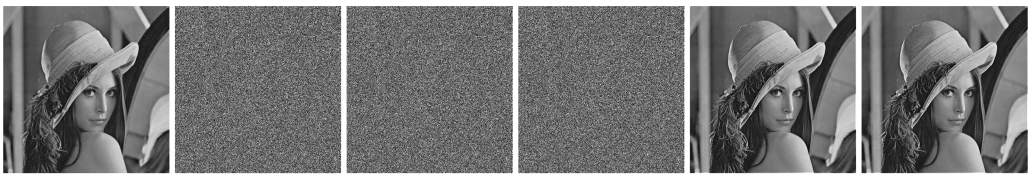
Fig. 7. Results of encrypted domain smoothing filter of various sizes. (a, e, i) original images, (b, f, j) results of 3×3 filter, (c, g, k) results of 5×5 filter, and (d, h, l) results of 9×9 filter.

smoothing filters in the encrypted domain for other test images are demonstrated in Figure 7. A comparison between the PSNR of the images smoothed in the encrypted domain and the plain domain is illustrated in Table 2. Clearly, the PSNR for the processed images in the encrypted and plain domain is the same with no error.

Experimental results for un-sharp masking in the encrypted and plain domain for Lena image are shown in Figure 8. From left to right is the original image, encrypted image, encrypted domain un-sharp maked image (refer to Equation (16)), encrypted domain sharp image (refer to Equation (17)), directly decrypted sharp image and plain domain sharp image. Visual results of encrypted domain

Table 2. Comparison of PSNR of Images Smoothed in the Encrypted Domain and Plain Domain for Various Filter Sizes

Test images	Encrypted domain			Plain domain		
	(3×3)	(5×5)	(9×9)	(3×3)	(5×5)	(9×9)
Lena	31.879	28.253	24.997	31.879	28.253	24.997
Airplane	30.954	26.653	23.002	30.954	26.653	23.002
Pirate	28.020	24.975	22.390	28.020	24.975	22.390
Lake	29.622	25.938	22.622	29.622	25.938	22.622



(a) (b) (c) (d) (e) (f)

Fig. 8. Experimental results of un-sharp masking using 3×3 smoothing filter size in the encrypted domain for Lena image. (a) Original Lena image, (b) encrypted image, (c) encrypted domain un-sharp masked image, (d) encrypted domain sharp image, (e) directly decrypted restored sharped image, (f) plain domain sharp image.

Table 3. Comparison of PSNR for Un-sharp Masking in the Encrypted and Plain Domain

Test images	Encrypted domain	Plain domain
Lena	31.879	31.879
Airplane	30.954	30.954
Pirate	28.020	28.020
Lake	29.622	29.622

un-sharp masking for other test images are illustrated in Figure 9. Table 3 illustrates a comparison between the PSNR of encrypted domain sharpened images and plain domain sharpened images.

Finally, visual results for encrypted domain histogram equalization and the plain domain histogram equalization are shown in Figure 10. Table 4 presents PSNR values for encrypted domain histogram equalized images and plain domain histogram equalized images.

5.1 Complexity Analysis

In this section, we analyze the complexity of the proposed method. In Figure 2, there are two entities: client and cloud server. The client performs Paillier encryption, decryption, and lattice basis reduction, whereas the cloud performs different image processing operations in the encrypted domain. We analyze the complexities at the client side and at the cloud side, respectively.

5.1.1 Client Side Complexity. Paillier encryption as given in Equation (2) involves modular exponentiations. The complexity of $g^m \bmod N^2$ term is $O(log(N)^2 \cdot log(m))$ and the complexity of $r^N \bmod N^2$ term is $O(log(N)^2 \cdot log(N)) = O(log(N)^3)$. The total complexity of encryption is $O(log(N)^2 \cdot log(m)) + O(log(N)^3)$. The complexity of Paillier encryption can be significantly

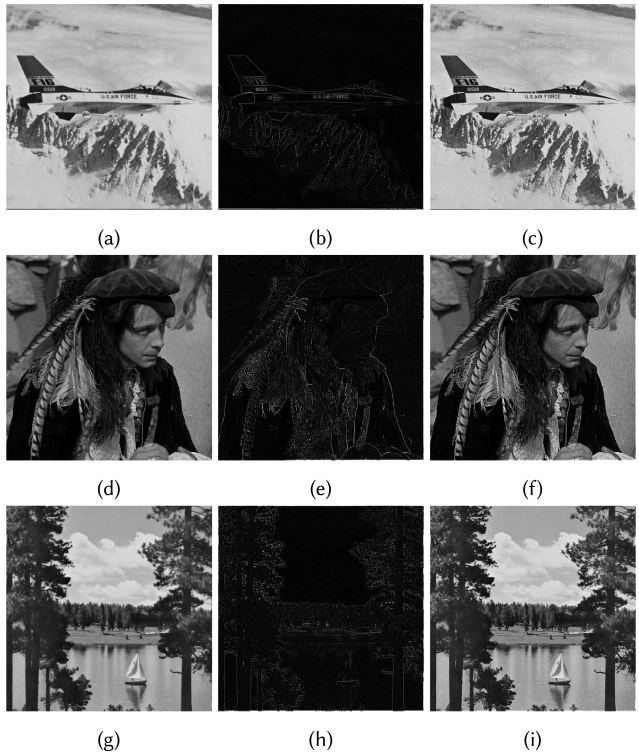


Fig. 9. Image sharpening using un-sharp masking in the encrypted domain. (a, d, g) original images, (b, e, h) un-sharp masked images, (c, f, i) sharp images.

reduced if we take $g = 1 + N$. In this case, the encryption given in Equation (2) modifies to

$$c = \mathfrak{E}[m, r] = (1 + mN) \cdot r^N \bmod N^2. \tag{21}$$

We can pre-compute the $r^N \bmod N^2$ term, since the plaintext m is not required. The real encryption is now computing the $(1 + mN)$ term, since $\bmod N^2$ has no effect if $m < N$. The $(1 + mN)$ term costs $O(log(N) \cdot log(m))$. Hence, the computational complexity of encryption is the complexity of $(1 + mN)$ term multiplied by the complexity of the $r^N \bmod N^2$ term, which ends up to $O(log(N) \cdot log(m))$.

According to Paillier (1999), the most expensive operation in the decryption is c^λ, because μ contains constant terms and can be pre-computed. Therefore, the computational complexity of decryption is $O(log(N)^3)$ (Paillier 1999).

The computational complexity of encryption can be lowered down to five modular multiplications by adopting the fast variant of the Paillier encryption proposed in Jost et al. (2015). The fast variant of Paillier encryption can be used to replace Equation (2) as

$$c = \mathfrak{E}[m, r] = g^m \cdot (g^N)^r \bmod N^2. \tag{22}$$

To efficiently compute the encryption using the above equation, the idea is to pre-compute g^m separately and save it in a table. For an 8-bit image and $N = 1{,}024$-bit security requires $2^8 \times 2 \times 1{,}024 = 2^{19}$ bits that is 64 kilobytes. The noise part $(g^N)^r$ can also be pre-computed. According to the method in Jost et al. (2015), 2^{16} random $(g^N)^r$ are pre-computed and saved in another table, which requires $2^{16} \times 2 \times 1{,}024 = 2^{27}$ that is 16 megabytes. The ciphertext c can be computed by

Fig. 10. Histogram equalization in the encrypted and plain domain. (a–d) original images, (e–h) encrypted domain histogram equalized images, (i–l) plain domain histogram equalized images.

Table 4. Comparison of PSNR for Histogram Equalization in the Encrypted and Plain Domain

Test images	Encrypted domain	Plain domain
Lena	19.469	19.469
Airplane	11.864	11.864
Pirate	13.671	13.671
Lake	24.406	24.406

selecting g^m from the first table and selecting five random values of $(g^N)^r$ from the second table and multiply these values. Hence, encryption can be performed with five modular multiplications.

Now let us compute the complexity of lattice basis reduction of Algorithm 2. The computational complexity of lattice basis reduction is equal to the total number of iterations used to reduce the basis vectors multiplied by the cost of iterations. According to the Lame's theorem (Knuth 1981) and experimental analysis, the number of iterations are $O(log(v))$ bounded by $\sqrt{N}$, where v is the second basis vector. Each iteration of Algorithm 2 computes the quotient q. The computational complexity of dividing a number is $O(log(l+1))$, where l is the bit length of the quotient (Knuth 1981). Hence, the complexity of Algorithm 2 is $O(log(v)) \cdot O(log(l+1))$. The total complexity at the client side including the complexities of encryption, decryption and lattice reduction is the summation of their complexities $O(log(N) \cdot log(m)) + O(log(N)^3) + O(log(v)) \cdot O(log(l+1))$.

Table 5. Computational Complexity at Client Side

Operation	Computational complexity
Paillier encryption	$O(log(N) \cdot log(m))$
Paillier decryption	$O(log(N)^3)$
Lattice reduction	$O(log(v)) \cdot O(log(l+1))$

Data expansion with Paillier cryptosystem is inevitable. Paillier cryptosystem maps the size of plaintext to the range $[0, N]$ where N is the modulus of the cryptosystem. The size of ciphertext space is square of the size of the plaintext space, i.e., the ciphertext is bounded by N^2. The data transmission from the client to the cloud is $2|N|$, where $|N|$ is the bit-length of N. One solution for reducing the expansion factor of $2|N|$ is by adopting the Damgard-Jurik cryptosystem (Damgård et al. 2010), which is a generalization of the Paillier cryptosystem. In the case of an image consisting of n pixels, the data transmission from the client to the cloud is $2|N| \times n$. When 1,024-bit security level is adopted (N = 1,024 bits) then the ciphertext is represented by 2048 bits. In the case of 8-bit image the expansion ratio is $2{,}048/8 = 256$. This expansion ratio can be significantly reduced using composite signal representation proposed by Bianchi et al. (2010). Composite signal representation divides the message sequence into R l-bit messages, packs them together into M messages and encrypt them as unique messages. For mathematical details of composite signal representation refer to Bianchi et al. (2010).

5.1.2 Cloud Side Complexity. The cloud server performs image processing operations in the encrypted domain. We estimate the complexities of encrypted domain smoothing filter, un-sharp masking and histogram equalization.

The smoothing filter is realized in the encrypted domain using Equation (14), which consists of modular multiplications and exponentiation. For an image with n pixels and filter mask of size $s \times s$, the total number of multiplications required to perform smoothing in the encrypted domain are $n \times s^2$, that is complexity of $O(log(n))$. The exponentiation is a fixed value in each computation and hence its complexity is constant. Consequently, the smoothing filter in the encrypted domain requires $O(log(n))$ operations.

Un-sharp masking in the encrypted domain give in Equation (16) involves three operations: encrypted domain smoothing filter, modular multiplicative inverse and multiplication with the original image. As computed earlier, the complexity of encrypted domain smoothing filter is $O(log(n))$. The modular multiplicative inverse is computed using the extended Euclidean algorithm. The complexity of extended Euclidean algorithm is $O(log(N)^2)$, where N is the modulus of the cryptosystem. For an image consisting of n pixels, the complexity of finding the modular multiplicative inverse of all pixels is, therefore, $n \cdot O(log(N)^2)$. Finally, the complexity of multiplying the inverse image and the original image is $O(log(n))$. The total complexity of un-sharp masking in the encrypted domain is $O(log(n)) + n \cdot O(log(N)^2) + O(log(n)) = n \cdot O(log(N)^2) + O(log(n))$.

In the case of histogram equalization in the encrypted domain, for k intensity levels there are k multiplications and two exponentiations as given in Equation (20). The two exponentials are computed once and hence their complexity can be considered constant. The complexity of histogram equalization in the encrypted domain is $O(log(k)^2)$. The computational complexities at the client and cloud are listed in Tables 5 and 6.

5.2 Security Analysis

Concerning the security of the proposed method, as we use semantically secure cryptosystem, the proposed method does not reveal anything about the content of the original image. The security

Table 6. Computational Complexity at Cloud Side

Operation	Computational complexity
Smoothing filter	$O(log(n))$
Un-sharp masking	$n \cdot O(log(N)^2) + O(log(n))$
Histogram equalization	$O(log(k)^2)$

of Paillier cryptosystem is based on the composite residuosity problem (Paillier 1999). A number $z \in Z^*_{N^2}$ is said to be Nth residue modulo N^2 if there exists a number y such that

$$z = y^N \bmod N^2. \tag{23}$$

According to the composite residuosity problem, it is hard for a polynomial time algorithm to decide whether z is an Nth residue or not. We refer the reader to Paillier (1999) for a complete security analysis of the Paillier cryptosystem.

The proposed method provides a solution to the case when condition 2 given in Section 2.2 does not hold. In that specific case, lattice theory and basis reduction is used to obtain the correct results in the plain domain for the computation performed in the encrypted domain. The division property given in Equation (7) is realized using the multiplication property given in Equation (6). The only difference is that the exponentiation in Equation (7) is a modular multiplicative inverse of the divisor k. As a matter of fact, this is similar to the multiplication property. The application of these properties does not leak any information of the original content or statistical profile of the image as the Paillier cryptosystem ensures semantic security. At the client side, the lattice basis reduction is applied after decryption. Only a legitimate client can have the decryption key, so this process is fully secure. In a nutshell, the proposed scheme does not modify any equation in the underlying Paillier cryptosystem so it is not compromising the security of the system.

5.3 Performance Comparison

In this section, we present comparison between the proposed method and four state-of-the-art methods in Ziad et al. (2016), Mohanty et al. (2016), Lathey and Atrey (2015), and Singh et al. (2018). The method in Ziad et al. (2016) used Paillier cryptosystem for encryption and decryption and utilized homomorphic properties of the cryptosystem for processing in the encrypted domain. In Mohanty et al. (2016), modified Paillier cryptosystem and its homomorphic properties are used for encrypted domain processing. The other two methods proposed in Lathey and Atrey (2015) and Singh et al. (2018) utilized Shamir Secret Sharing (SSS) for creating secrete shares and processing on them. A comprehensive comparison between the proposed method and other methods is described as follows.

(1) In Ziad et al. (2016), encrypted domain smoothing filter is performed using quantization encoding function. The division operation involved in the smoothing filter is carried out by multiplying the floating point term by the filter size. The proposed method avoids the use of quantization encoding function and performs the smoothing filter directly in the encrypted domain without using interactive protocol between the client and the cloud server. Furthermore, the method in Ziad et al. (2016) introduced errors between the encrypted domain operations and plain domain operations due to the quantization encoding function. The proposed method, however, does not introduce any errors. This manifests that processing operation performed in the encrypted domain fetch the same operations in the plain domain.

(2) In Mohanty et al. (2016), the floating point numbers involved in bilinear interpolation are mapped to integers using a large scale factor and rounding function. Due to the rounding function, the plain domain result and encrypted domain result is never equal. Furthermore, bilinear interpolation factor, which is computed using the mean value of neighboring pixels, is computed in the plain domain and then used as as exponentiation to encrypted pixels to perform scaling in the encrypted domain at the cloud. In this way, the interpolation factor is exposed to the semi-trusted curious cloud server. The cloud server can use interpolation factors to obtain statistical information about the original encrypted image. Similarly, to achieve reduced size and low complexity, tiles in a super-tile are encrypted with same random number r. With the optimized encryption, images with similar tiles will produce similar ciphertexts. Henceforth, the method in Mohanty et al. (2016) is not semantically secure. In contrast, the proposed method does not leak any information about the original content of the encrypted image as mentioned in Section 5.2. The method in Mohanty et al. (2016) adopts tiles level encryption to recduce the computational complexity and data overhead. However, as described earlier, the benefits of tiles level encryption are achieved at the cost of loss of semantic security. In Mohanty et al. (2016), the authors provided simulation times of encryption, encrypted domain processing operations and decryption, which are code and hardware specific but does not account for theoretical computational complexities. We provide theoretical complexities of Paillier encryption, encrypted domain processing operations and decryption. These complexities are listed in Tables 5 and 6. We also provided solutions to reduce the complexity of Paillier encryption and data expansion as mentioned in Section 5.1.1.

(3) The methods proposed in Lathey and Atrey (2015) pre-processed the original image before making secret shares. Two pre-processing schemes are proposed to modify image pixels to make them completely divisible by the filter size. In this way, the division operation of smoothing filter is performed in the encrypted domain. Moreover, the pre-processing step increases the overhead size of the secret shares. The processing operations performed in the encrypted domain and in the plain domain presented in Lathey and Atrey (2015) are not exactly the same and they differ by some errors. The proposed method performs all the processing operations directly in the encrypted domain without pre-processing the images. Furthermore, the method in Lathey and Atrey (2015) is theoretically secure only if the encrypted domain processing operations are performed using n servers with no more than k are colluding. The requirement of non-colluding servers makes this method impractical. The proposed method works with only one server and uses Paillier cryptosystem with 1,024-bit key length, which is secure against modern factoring attacks.

(4) In Singh et al. (2018), Shamir Secret Sharing (SSS) and POB number system is utilized for creating secret shares and processing in the encrypted domain. The use of POB number system with SSS enhances the security and allows processing operations in the encrypted domain. However, the processing operations performed in the encrypted and plain domain are not exactly the same as the PSNR values are different for results in both domains. Compared to Singh et al. (2018), the PSNR values for plain domain and encrypted domain results are the same for the proposed method.

The method in Singh et al. (2018) has not mentioned the introduced error analytically or quantitatively, we present a comparison of introduced error between the encrypted and plain domain operations for the proposed method against three methods in Ziad et al. (2016), Mohanty et al. (2016), and Lathey and Atrey (2015) in Table 7. The error in Ziad et al. (2016) depends on the precision level of the exponent. A precision level of 10^{-8} introduces error of 0.145 as given in Table 7.

Table 7. Error Comparison between Encrypted and Plain Domain Operations of the Proposed Method Against State-of-the-art Methods

Method	Error between ED and PD
(Ziad et al. 2016)	0.145
(Mohanty et al. 2016)	$-51 \times 10^{1-d} \leq \varepsilon \leq +51 \times 10^{1-d}$
(Lathey and Atrey 2015)	$0 \leq \varepsilon \leq \left\lceil \frac{h \times w}{2} \right\rceil$
Proposed	0

Table 8. Feature Comparison of the Proposed Method Against State-of-the-art Methods

Method	Encryption scheme	Pre-proc.	Errors btw ED and PD	Parties
(Ziad et al. 2016)	Paillier cryptosystem	Yes	Yes	Single-party
(Mohanty et al. 2016)	Modified Paillier cryptosystem	Yes	Yes	Single-party
(Lathey and Atrey 2015)	SSS	Yes	Yes	Multi-party
(Singh et al. 2018)	SSS + POB	No	Yes	Multi-party
Proposed	Paillier cryptosystem	No	No	Single-party

The error in Mohanty et al. (2016) is a function of the scale factor d, whereas, the error in Lathey and Atrey (2015) is bounded by the filter size $h \times w$. Compared to all the methods, the proposed method does not introduce any error, which shows the effectiveness of our method. A comprehensive feature comparison between the proposed method against the four state-of-the-art methods is listed in Table 8.

6 CONCLUSION

In this article, we proposed a method for performing image processing operations involving division in the homomorphic encrypted domain. Because of the limitation of the cryptosystems, previous methods either could not performed such operations in the encrypted domain or performed them with the help of quantization encoding function or pre-processing. We presented a method to perform non-integer mean value computation directly in the homomorphic encrypted domain without any pre-processing or encoding function and without interactive sessions between the client and the cloud sever, based on which, we performed some image processing operations in the encrypted domain, such as local smoothing filtering, un-sharp masking, and histogram equalization. While the previous methods introduced errors between the processing operations performed in the encrypted and plain domain, our experimental results revealed that the processing operations performed in both domains are exactly the same, which shows the feasibility of our method.

Future work is twofold. First, we will extend the proposed work to more complex operations in image, audio, video, and 3D meshes. Second, we will apply the proposed method in the field of reversible data hiding. Many existing plain domain state-of-the-art reversible data hiding methods are not realizable in the homomorphic encrypted domain because of the limitations of the homomorphic cryptosystem to process real numbers, as most of these methods involve division operation.

REFERENCES

Adobe. [n.d.]. Adobe Creative Cloud. Retrieved from https://www.adobe.com/creativecloud.html.

Khalid Alhamazani, Rajiv Ranjan, Karan Mitra, Fethi Rabhi, Prem Prakash Jayaraman, Samee Ullah Khan, Adnene Guabtni, and Vasudha Bhatnagar. 2015. An overview of the commercial cloud monitoring tools: Research dimensions, design issues, and state-of-the-art. *Computing* 97, 4 (2015), 357–377.

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2009a. Encrypted domain DCT based on homomorphic cryptosystems. *EURASIP J. Info. Secur.* 2009, 1 (2009), 716357.

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2009b. On the implementation of the discrete Fourier transform in the encrypted domain. *IEEE Trans. Info. Forensics Secur.* 4, 1 (2009), 86–97.

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2010. Composite signal representation for fast and storage-efficient processing of encrypted signals. *IEEE Trans. Info. Forensics Secur.* 5, 1 (2010), 180–187.

Robert D. Carmichael. 1913. On the numerical factors of the arithmetic forms α n$\pm\beta$ n. *Ann. Math.* 15, 1/4 (1913), 49–70.

Delin Chen, Wenhao Chen, Jian Chen, Peijia Zheng, and Jiwu Huang. 2018. Edge detection and image segmentation on encrypted image with homomorphic encryption and garbled circuit. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'18)*. IEEE, 1–6.

Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. 2010. A generalization of Paillier's public-key system with applications to electronic voting. *Int. J. Info. Secur.* 9, 6 (2010), 371–385.

Steven D. Galbraith. 2012. *Mathematics of Public Key Cryptography*. Cambridge University Press, 24–27.

Carl Friedrich Gauss. 1966. *Disquisitiones Arithmeticae*. Vol. 157. Yale University Press.

Craig Gentry and Shai Halevi. 2011. Implementing gentry's fully-homomorphic encryption scheme. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 129–148.

Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *J. Comput. Syst. Sci.* 28, 2 (1984), 270–299.

Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *Proceedings of the International Algorithmic Number Theory Symposium*. Springer, 267–288.

Christine Jost, Ha Lam, Alexander Maximov, and Ben J. M. Smeets. 2015. Encryption performance improvements of the paillier cryptosystem. *IACR Cryptol.* 2015 (2015), 864. https://www.iacr.org/cryptodb/data/paper.php?pubkey=26497.

Donald E. Knuth. 1981. *The Art of Computer Programming; Volume 2: Seminumeral Algorithms*. Addison-Wesley.

Reginald L. Lagendijk, Zekeriya Erkin, and Mauro Barni. 2013. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Process. Mag.* 30, 1 (2013), 82–105.

Joseph Louis Lagrange. 1775. *Recherches d'arithmetique*. C. F. Voss.

Ankita Lathey and Pradeep K. Atrey. 2015. Image enhancement in encrypted domain over cloud. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 3 (2015), 38.

Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261, 4 (1982), 515–534.

Peter Mell, Tim Grance et al. 2011. The NIST definition of cloud computing. *NIST Spec. Publ.* 800 (2011), 7.

Manoranjan Mohanty, Muhammad Rizwan Asghar, and Giovanni Russello. 2016. 2$DCrypt$: Image scaling and cropping in encrypted domains. *IEEE Trans. Info. Forensics Secur.* 11, 11 (2016), 2542–2555.

Manoranjan Mohanty, Wei Tsang Ooi, and Pradeep K. Atrey. 2013. Scale me, crop me, know me not: Supporting scaling and cropping in secret image sharing. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'13)*. IEEE, 1–6.

Phong Q. Nguyen and Jacques Stern. 2001. The two faces of lattices in cryptology. In *Cryptography and Lattices*. Springer, 146–180.

Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 223–238.

PIXLR. [n.d.]. PIXLR Editor. Retrieved from https://pixlr.com/editor/.

Guillaume Poupard and Jacques Stern. 2000. Fair encryption of RSA keys. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 172–189.

NIST FIPS Pub. 2001. 197: Advanced encryption standard (AES). *Fed. Info. Process. Stand.ards Publ.* 197, 441 (2001), 0311.

Amitesh Singh Rajput and Balasubramanian Raman. 2018. CryptoCT: Towards privacy preserving color transfer and storage over cloud. *Multimedia Tools Appl.* 77, 18 (2018), 24223–24245.

Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. Assoc. Comput. Mach.* 56, 6 (2009), 34.

Harald Ritter and Carsten Rössner. 1997. Factoring via strong lattice reduction algorithms. *IACR Cryptol.* 1997 (1997), 8.

Mohsin Shah, Weiming Zhang, Honggang Hu, Hang Zhou, and Toqeer Mahmood. 2018. Homomorphic encryption-based reversible data hiding for 3D mesh models. *Arabian J. Sci. Eng.* 43, 12 (2018), 8145–8157.

Priyanka Singh, Balasubramanian Raman, and Manoj Misra. 2018. Just process me, without knowing me: A secure encrypted domain processing based on Shamir secret sharing and POB number system. *Multimedia Tools Appl.* 77, 10 (2018), 12581–12605.

Juan Ramón Troncoso-Pastoriza and Fernando Perez-Gonzalez. 2013. Secure signal processing in the cloud: Enabling technologies for privacy-preserving multimedia cloud processing. *IEEE Signal Process. Mag.* 30, 2 (2013), 29–41.

Peijia Zheng and Jiwu Huang. 2012. Walsh-Hadamard transform in the homomorphic encrypted domain and its application in image watermarking. In *Proceedings of the International Workshop on Information Hiding*. Springer, 240–254.

Peijia Zheng and Jiwu Huang. 2013. Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain. *IEEE Trans. Image Process.* 22, 6 (2013), 2455–2468.

M. Tarek Ibn Ziad, Amr Alanwar, Moustafa Alzantot, and Mani Srivastava. 2016. Cryptoimg: Privacy preserving processing over encrypted images. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'16).* IEEE, 570–575.

Received September 2018; revised March 2019; accepted April 2019