Secure Nonlocal Denoising in Outsourced Images

XIANJUN HU, WEIMING ZHANG, KE LI, HONGGANG HU, and NENGHAI YU,
University of Science and Technology of China

Signal processing in the encrypted domain becomes a desired technique to protect privacy of outsourced data in cloud. In this article, we propose a double-cipher scheme to implement nonlocal means (NLM) denoising in encrypted images. In this scheme, one ciphertext is generated by the Paillier scheme, which enables the mean filter, and the other is obtained by a privacy-preserving transform, which enables the nonlocal search. By the privacy-preserving transform, the cloud server can search the similar pixel blocks in the ciphertexts with the same speed as in the plaintexts; thus, the proposed method can be executed fast. To enhance the security, we randomly permutate both ciphertexts. To reduce the denoising complexity caused by random permutation, a random NLM method is exploited in the encrypted domain. The experimental results show that the quality of denoised images in the encrypted domain is comparable to that obtained in the plain domain.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Security

Additional Key Words and Phrases: Image denoising, Johnson-Lindenstrauss transform, nonlocal means, Paillier homomorphic encryption

ACM Reference Format:
Xianjun Hu, Weiming Zhang, Ke Li, Honggang Hu, and Nenghai Yu. 2016. Secure nonlocal denoising in outsourced images. ACM Trans. Multimedia Comput. Commun. Appl. 12, 3, Article 40 (March 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2886777

1. INTRODUCTION

The computable cloud is now prevalent in our daily life, in which customers can remotely store their data to enjoy convenient and effective services [Mell and Grance 2009]. Increasingly sensitive information such as e-mails and finance data are professionally maintained in data centers. They face many basic challenges such as security [Ren et al. 2012], though outsourcing data storage and processing are quite promising. In fact, many corporations and companies are still reluctant to outsource their data to a cloud server, concerned that their data may be leaked or a cloud server

This work was supported in part by the National Natural Science Foundation of China under Grant 61572452 and Grant 61170234, in part by the Strategic Priority Research Program through the Chinese Academy of Sciences under Grant XDA06030601, in part by the National Natural Science Foundation of China (61271271, 61522210), 100 Talents Program of Chinese Academy of Sciences, and the Fundamental Research Funds for the Central Universities in China (WK2101020005).
Authors' addresses: X. Hu, W. Zhang, K. Li, H. Hu, and N. Yu, Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences, School of Information Science and Technology, University of Science and Technology of China, Hefei, 230027, China; emails: hxj2012@mail.ustc.edu.cn, zhangwm@ustc.edu.cn, lee0525@mail.ustc.edu.cn, hghu2005@ustc.edu.cn, ynh@ustc.edu.cn.

DOI: http://dx.doi.org/10.1145/2886777

could abuse their data. Thus, it is necessary for sensitive data to be encrypted for data privacy.

This leads to a need for techniques of signal processing on encrypted data, which obviously is a difficult problem, because we must have a secure encryption scheme or a protocol that allows computations in the encrypted domain [Lagendijk et al. 2013]. Rivest et al. [1978] proposed solving this problem by a scheme called homomorphic encryption, which keeps the algebraic relations between plaintexts and ciphertexts. Later, several homomorphic encryption schemes [ElGamal 1985; Paillier 1999; Damgård and Jurik 2001] were presented, which process encrypted data with only one homomorphic property, such as addition or multiplication. For instance, the Paillier scheme [1999] has additive homomorphism, which means that one can perform the addition of two plaintext signals by multiplying two corresponding encrypted signals. A scheme is called full homomorphic encryption (FHE) if it enables additive and multiplicative homomorphisms at the same time. The first secure FHE scheme was proposed in Gentry [2009], which, from a theoretical perspective, can solve any privacy-preserving computational problem. However, due to the huge computational complexity and ciphertext expansion, the FHE scheme is too inefficient to be applied in practice, even though great improvements have been made [Brakerski et al. 2012; Aguilar-Melchor et al. 2013; Zhou and Wornell 2014]. So far, additive homomorphic encryption is still the most popular scheme used by the privacy-protection community. Based on additive homomorphic encryption, some linear computations have been realized in the encrypted domain, such as discrete Fourier transform [Bianchi et al. 2009b], discrete cosine transform [Bianchi et al. 2009a], discrete wavelet transform [Zheng and Huang 2011, 2013a], and Walsh-Hadamard transform [Zheng and Huang 2013b].

An interesting and challenging problem is how to do nonlinear computations in the encrypted domain with low complexity. In this article, we present a framework to solve the problem of encrypted image denoising that involves some nonlinear operations, such as exponent arithmetic and Euclidean distance. Image denoising is one of the most popular tasks in image processing; there are many classical image denoising algorithms, such as Gaussian filter, neighborhood filter, and nonlocal means (NLM) [Buades et al. 2005]. Among them, NLM and its extensions can reach better performance by exploiting the similarity between the nonlocal pixel blocks with the current block. However, the computational complexity of NLM algorithm is very high because it needs to search for the similar pixel blocks. Such hardness of computation is suitable for being outsourced to the cloud server, but the user may hope to prevent the cloud server from getting the content of the images. Therefore, the cloud server should implement denoising in encrypted images.

Secure multiparty computation (MPC) proposed in Yao [1982] is an important approach to compute arbitrary function in cryptography. In recent years, great improvements have made MPC more practical [Cramer et al. 2001; Orlandi 2011], even though the MPC protocols are still computationally costly.

In particular, the secret sharing scheme has been an efficient technique in encrypted multimedia processing in recent years. This was proposed in Shamir [1979] and Blakley et al. [1979], which supports additive and multiplicative homomorphic properties. A drawback of Shamir's secret sharing is that it requires multiple cloud servers to resist collusion attack [Benaloh 1987]. In Saghaian et al. [2012], a privacy-protected wavelet images denoising using secret sharing was realized. Lathey et al. [2013] and Lathey and Atrey [2015] proposed a novel and efficient scheme to implement image enhancement using secret sharing in the encrypted domain, when it came with arithmetic division operations for nonterminating quotients. In their articles, some of the low-level image processing tasks (e.g., spatial filtering, unsharp masking) were performed. Using secret sharing to deal with complex arithmetic is still very difficult (e.g., using secret sharing

to compute Equation (3) in Section 2). In our article, the cloud server can compute Equation (3) in the ciphertexts with the same speed as in the plaintexts; also, our scheme needs only one cloud server.

In this article, we try to implement the NLM in the encrypted domain, which consists of two operations, that is, nonlocal search and mean filter. Mean filter in the encrypted domain can be realized based on an additive homomorphic cryptosystem such as the Paillier scheme [1999], while nonlocal search is a nonlinear operation. To avoid using a complex cipher algorithm for the nonlinear part, we propose a double-cipher denoising scheme, in which we encrypt the image with two cryptosystems and thus outsource two ciphertexts to the cloud. One ciphertext is generated by the Paillier scheme, which enables the mean filter, and the other is obtained by a distance-preserving transform, Johnson-Lindenstrauss Transform (JL Transform), which enables the nonlocal search. According to this idea, we proposed a preliminary scheme for privacy-preserving NLM in Hu et al. [2014]. However, we found that this scheme may be attacked because JL Transform keeps the position relationship and relative amplitudes of pixels.

In this article, we present a binarization attack on the scheme in Hu et al. [2014] by exploiting the information leaked by JL Transform, by which we can infer a profile of the image from the ciphertext. To resist the binarization attack, we propose the use of random permutation before JL Transform and Paillier encryption. However, random permutation will increase the computational complexity of NLM in a ciphertext. To reduce the complexity, a random NLM method is used in encrypted images. The analysis shows that the novel scheme can resist binarization attack, and the experimental results show that the quality of denoised images in the encrypted domain is comparable to that obtained in the plain domain.

The remainder of the article is organized as follows. In Section 2, we give some preliminaries about the NLM denoising algorithm, Paillier cryptosystem, and JL Transform. A framework of our scheme is presented in Section 3. In Section 4, we describe how to perform image denoising in the encrypted domain in detail. We implement our scheme and give the experimental results in Section 5. In Section 6, we present our conclusions.

2. PRELIMINARY

2.1. Nonlocal Means

The NLM proposed in Buades et al. [2004] was widely used in image denoising. Unlike local smoothing methods, which operate only within a local area, NLM tries to exploit the relativity between the pixels that are not close to each other. We briefly introduce NLM below.

We can describe a discrete noisy image as follow:

$$v(i) = u(i) + n(i), \tag{1}$$

where i is the pixel index in the set I, $v(i)$ is the observed value, and $u(i)$ is the original value. The most simple way to model the effect of noise on a digital image is to add Gaussian noise. Therefore, $n(i)$ is an i.i.d. Gaussian variable with zero-mean and variance σ^2.

The denoised pixel value at position i is obtained by

$$NL(i) = \sum_{j \in \Omega} w(i, j)v(j), \tag{2}$$

where Ω is the search window for the similar pixel. The weights $\{w(i, j)\}$ are determined by the similarity between the i-th and j-th pixels, satisfying $0 \leq w(i, j) \leq 1$ and $\sum_j w(i, j) = 1$. Usually, the similarity can be calculated by the Euclidean distance between the two blocks centered at the i-th and the j-th pixels. Therefore, the weights

are computed as follows:

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{\|v(N_i)-v(N_j)\|_2^2}{h^2}}. \tag{3}$$

Herein, h is used to control the decay of the weights, N_i denotes patches centered in the i-th pixel, $v(N_i)$ denotes the intensity gray level vectors of N_i, $\|\cdot\|_2$ is the Euclidean norm, and $Z(i)$ is the normalizing constant defined as:

$$Z(i) = \sum_{j\in\Omega} e^{-\frac{\|v(N_i)-v(N_j)\|_2^2}{h^2}}. \tag{4}$$

2.2. Paillier Cryptosystem

The Paillier cryptosystem [1999] is one of the well-known probabilistic and homomorphic schemes with an additive homomorphic property, which is realized as follows:

Initialization. Compute $N = pq$, where p, q are selected as two large random prime numbers. Let $\lambda = lcm(p-1, q-1)$ and $g \in Z^*_{N^2}$, where the order of g is a multiple of N. The public key pk is (N, g) and the secret key sk is λ.

Encryption. Take a message $m \in Z_N$, and a random number (blinding factor) $r \in Z^*_N$. The corresponding ciphertext is

$$c = E_{pk}(m, r) = g^m r^N \ mod\ N^2. \tag{5}$$

Decryption. Let the ciphertext $c \in Z^*_{N^2}$, so that the message m can be recovered by

$$m = D_{sk}(c) = \frac{L(c^\lambda \ mod\ N^2)}{L(g^\lambda \ mod\ N^2)} \ mod\ N, \tag{6}$$

where $L(\phi) = (\phi - 1)/N$.

Homomorphism. The additive homomorphism means that the sum of two plaintexts m_1 and m_2 can be obtained by decrypting the product of corresponding ciphertexts.

$$\begin{aligned} D(E(m_1, r_1) \cdot E(m_2, r_2)) &= D(g^{m_1+m_2}(r_1 r_2)^N) \\ &= D(E(m_1 + m_2, r_1 r_2)) = m_1 + m_2. \end{aligned} \tag{7}$$

Moreover, let γ be a constant and m be a plaintext, then γm can be calculated by decrypting the power of the ciphertext.

$$D(E(m, r)^\gamma) = D((g^m r^N)^\gamma) = D(E(\gamma m, r^\gamma)) = \gamma m. \tag{8}$$

2.3. Johnson-Lindenstrauss Transform

JL Transform is a dimension-reduction method preserving Euclidean distance. The Johnson-Lindenstrauss Theorem [Johnson and Lindenstrauss 1984] states that it is possible to project Q points in a space of arbitrarily high dimension onto an $O(\log Q)$-dimensional space, such that the pairwise distances between the points are approximately preserved.

THEOREM 2.1 (JOHNSON-LINDENSTRAUSS THEOREM). *For any $0 < \varepsilon < 1$ and any positive integer Q, there is a positive integer k such that $k \geq 4\ln Q/(\epsilon^2/2 - \epsilon^3/3)$. Then, for any set U of Q points in R^d, there exists a map $f : R^d \to R^k$ such that for any two vectors $\alpha, \beta \in U$, the following inequality holds:*

$$(1-\varepsilon)\|\alpha - \beta\|^2 \leq \|f(\alpha) - f(\beta)\|^2 \leq (1+\varepsilon)\|\alpha - \beta\|^2.$$

Theorem 2.1 means that some d-dimensional vectors can be mapped into k-dimensional vectors, and the Euclidean distance between these d-dimensional vectors can be estimated by corresponding k-dimensional vectors. Usually, f is defined as $f(\alpha) = \alpha P$, where each entry of $P \in R^{d \times k}$ is drawn independently from a Normal distribution with zero-mean and variance $1/k$ [Indyk and Motwani 1998]. Kenthapadi et al. [2012] proposed the following private projection algorithms: Algorithm 1 and Algorithm 2, and also proved that an attacker who knows all except one value of the secret vector cannot recover this value from the JL Transform vector. Algorithm 1 uses a $d \times k$ random matrix P to map a d-dimensional vector α to a k-dimensional vector, and adds Gaussian noise to get α'. Algorithm 2 is about how to use transformed vectors to estimate squared distance in the original space.

ALGORITHM 1: JL Transform-Based Private Projection

Input: d-dimensional vector α; $d \times k$ random matrix P; Noise parameter ζ.
Output: The projected k-dimensional vector α'.

1. $Y = \alpha P$;
2. Generate a k-dimensional $N(0, \zeta^2)$ Gaussian noise vector Δ;
3. $\alpha' = Y + \Delta$.

ALGORITHM 2: JL Transform-Based Distance Recover

Input: Two k-dimensional transformed vectors α' and β'; Noise parameter ζ.
Output: Estimated squared distance between α and β in the original space.

Output $dist^2_{\alpha,\beta} = \|\alpha' - \beta'\|_2^2 - 2k\zeta^2$.

3. PROBLEM STATEMENT AND FRAMEWORK

We consider the cloud server as the "honest-but-curious" model (originally called the semi-honest model, firstly introduced in Goldreich et al. [1987]), which means that the cloud server will honestly perform the designated algorithm, but it is curious to obtain the user's private information from its storage and computation. A resources-constrained client owns one image I, and wants to denoise I with the NLM method. The owner hopes to outsource this work to a cloud server without leaking the content of I.

As mentioned earlier, Paillier homomorphic cryptosystem is an additive homomorphic cryptosystem, which can perform any linear operations in the encrypted domain. However, as shown in Equation (3) and Equation (4), NLM block similarity computation includes nonlinear operations. It is hard to implement the whole NLM algorithm based only on the additive homomorphic property. But if we can compute the values of $\{w(i, j)\}$ with some other methods, this problem will be easily solved. The weights $\{w(i, j)\}$ are about the similarity or distance between two image blocks, thus we can use the JL Transform-based Private Project (Algorithm 1) to compute the weights. Therefore, we propose the following framework for nonlocal denoising in outsourced images.

As illustrated in Figure 1, the owner encrypts I by Privacy Preserving Transform (PPT) and Partially homomorphic cryptosystem (PHE), and gets two encrypted images denoted by $E^{PPT}(I)$ and $E^{PHE}(I)$, respectively. Then, the owner sends $E^{PHE}(I)$ and $E^{PPT}(I)$ to the cloud. With the help of $E^{PPT}(I)$, the cloud server executes nonlocal filter on $E^{PHE}(I)$ and yields a denoised cipher-image $E^{PHE}(I')$ that is sent back to the owner. The owner decrypts $E^{PHE}(I')$ and gets a plaintext denoised image I'. In Section 4, we will elaborate the details of each step.

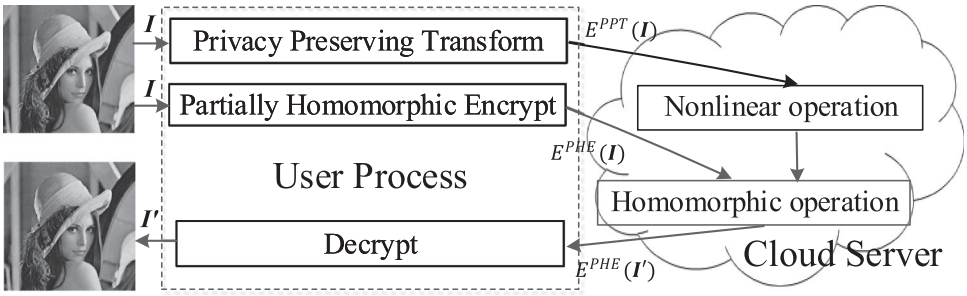


Fig. 1. Framework of double-cipher scheme.

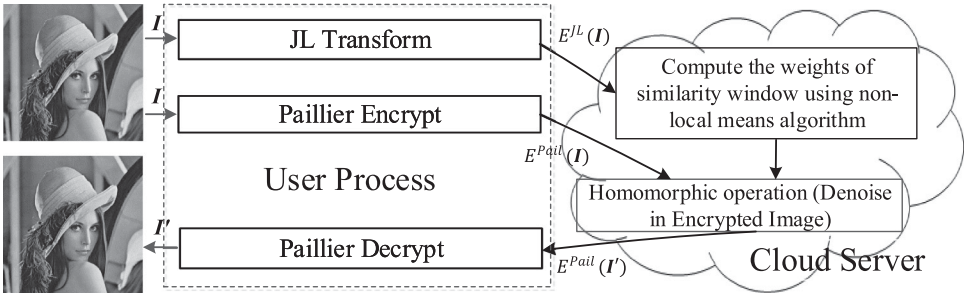


Fig. 2. Scheme I: Preliminary scheme.

4. DOUBLE-CIPHER SCHEME

4.1. Scheme I: Preliminary Scheme

In this section, we will describe our first scheme. The complete Scheme I has 4 algorithms, as illustrated in Figure 2: *Encryption with JL Transform*, *Paillier Encryption*, *Denoising in Encrypted Images*, and *Paillier Decryption*.

Encryption with JL Transform. When encrypting I with Algorithm 1, the owner takes the random matrix P and noise parameter ζ as the key. For each pixel $v(i)$ of I, we take an $s \times s$ block centered at i and stack the block rows as an s^2-dimensional vector, denoted by N_i. With Algorithm 1, the owner projects N_i into a k-dimensional vector $E^{JL}(N_i)$, which is just the ciphertext of $v(i)$. In other words, by JL Transform, each pixel is encrypted into a k-dimensional vector. For marginal pixels, some elements of the block matrix are blank; thus, we fill them with the surrounding pixels.

As we know, a $1 \times s^2$ vector data vector right-multiplied by an $s^2 \times k$ Gaussian projection matrix equals a $1 \times k$ data vector. For $k < s^2$, it is hard to inverse this process to get the original data vector. Thus, we use this procedure to do the JL Transform, and we give a detailed account of how to apply it to our scheme.

Taking the pixels at positions 6, 7, 10, and 11 as an example in Figure 3, we set $s = 3$ and $k = 4$. The 3×3 block centered at each pixel is transformed to a 1×9 vector. The four vectors corresponding to the four pixels are arrayed as a 4×9 matrix α, which is multiplied by a 9×4 projection matrix P to get a 4×4 matrix Y. Then, we add a $N(0, \zeta^2)$ Gaussian noise matrix Δ, and generate the ciphertext α'. Next, we can upload the transformed data α' to the cloud, with which the cloud server can estimate the Euclidean distance of these image blocks and evaluate values of $\{w(i, j)\}$ in Equation (3).

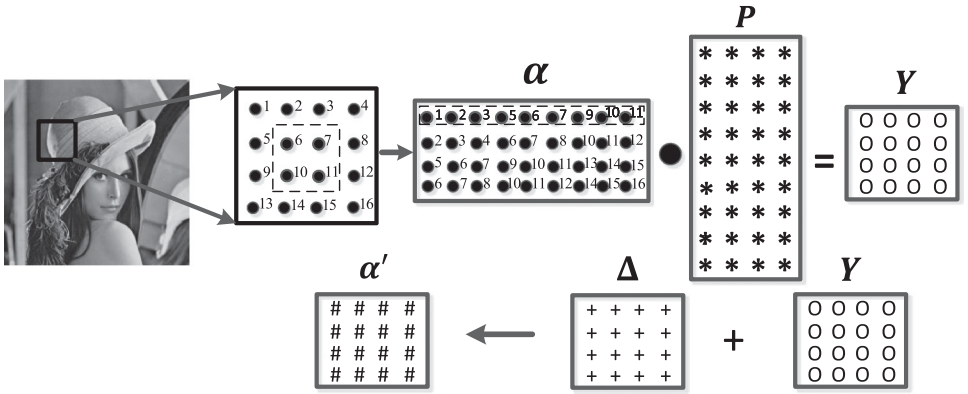


Fig. 3. A toy example of JL Transform. Each row of α' is the ciphertext of image block centered at 6, 7, 10, and 11, respectively.

From this description, for an image consisting of n pixels, the size of transformed data will be $n \times k$, where k is the size of a single transformed block. For instance, when taking $k = 4$ for a 512×512 image, the size of transformed data is $512^2 \times 4$, that is, $k = 4$ is the expansion factor.

Paillier Encryption. For each pixel $v(i)$, take a random number $r_i \in Z_N^*$ and encrypt $v(i)$ by Equation (5) as

$$E^{Pail}(v(i)) = E^{Pail}(v(i), r_i) = g^{v(i)} r_i^N \ mod\ N^2. \tag{9}$$

Denoising in Encrypted Images. As shown in Equation (2) and Equation (3), to perform nonlocal filter, the cloud server should first calculate the weights $\{w(i, j)\}$ that are determined by the Euclidean distance between N_i and N_j. Note that JL Transform preserves Euclidean distance; thus, the cloud server can estimate $\|v(N_i) - v(N_j)\|_2^2$ with the ciphertexts of $v(i)$ and $v(j)$, that is, $E^{JL}(v(N_i))$ and $E^{JL}(v(N_j))$. Therefore, the weights are estimated by

$$w'(i, j) = \frac{1}{Z'(i)} e^{-\frac{\|E^{JL}(v(N_i)) - E^{JL}(v(N_j))\|_2^2 - 2k\zeta^2}{h^2}}. \tag{10}$$

The normalizing constant $Z'(i)$ is obtained by

$$Z'(i) = \sum_{j \in \Omega} e^{-\frac{\|E^{JL}(v(N_i)) - E^{JL}(v(N_j))\|_2^2 - 2k\zeta^2}{h^2}}. \tag{11}$$

With weights $\{w'(i, j)\}$, the cloud server filters the encrypted image $E^{Pail}(I)$ as follows. For each ciphertext $E^{Pail}(v(i))$, the filtered value $E^{Pail}(NL'(i))$ is

$$E^{Pail}[NL'(i)] = \prod_{j \in \Omega} (E^{Pail}[v(j)])^{w'(i,j)} \ mod\ N^2. \tag{12}$$

Having collected all $E^{Pail}(NL'(i))$, the cloud server yields a denoised encrypted image $E^{Pail}(I')$ that will be sent back to the owner.

Note that the weights $\{w'(i, j)\}$ are real numbers, but to calculate Equation (12) according to the Paillier cryptosystem, the values $\{w'(i, j)\}$ have to be quantified as integer numbers. The quantization process is computed as follows:

$$W(i, j) = \lfloor Aw'(i, j) \rceil, \tag{13}$$

where $\lfloor \cdot \rceil$ is the rounding function and A is the scaling factor. Thus, $W(i, j)$ based on Equation (2), Equation (10), Equation (11), and Equation (13) can be computed as follows:

$$W(i, j) = \left\lfloor \frac{A}{Z'(i)} e^{-\frac{\|E^{JL}(v(N_i)) - E^{JL}(v(N_j))\|_2^2 - 2k\zeta^2}{\hbar^2}} \right\rceil, \tag{14}$$

$$NL''(i) = \sum_{j \in \Omega} W(i, j) v(j) = \sum_{j \in \Omega} \lfloor Aw'(i, j) \rceil v(j). \tag{15}$$

Replacing $w'(i, j)$ by $W(i, j)$, Equation (12) will be changed to

$$E^{Pail}[NL''(i)] = \prod_{j \in \Omega} (E^{Pail}[v(j)])^{\lfloor Aw'(i,j) \rceil} \ mod\ N^2. \tag{16}$$

Comparing with denoising in plaintext images, there are two kinds of errors in this procedure. The first is caused by JL Transform, which becomes larger when decreasing the expansion factor k. This error is analyzed in Kenthapadi et al. [2012], who prove that distance recovery in Algorithm 2 is an unbiased estimator of the original Euclidean distance.

The second kind of error is caused by quantization. To analyze the quantizing error, we rewrite Equation (13) as follows:

$$W(i, j) = Aw'(i, j) + \varepsilon_{w_j}, \tag{17}$$

where $|w'(i, j)| \leq 1$ and ε_{w_j} is the error caused by quantization with $|\varepsilon_{w_j}| \leq 1/2$.

Equation (15) will be changed to

$$NL''(i) = \sum_{j \in \Omega} W(i, j) v(j) = \sum_{j \in \Omega} (Aw'(i, j) + \varepsilon_{w_j}) v(j). \tag{18}$$

From Equation (18), the owner of I can estimate $NL'(i)$ by

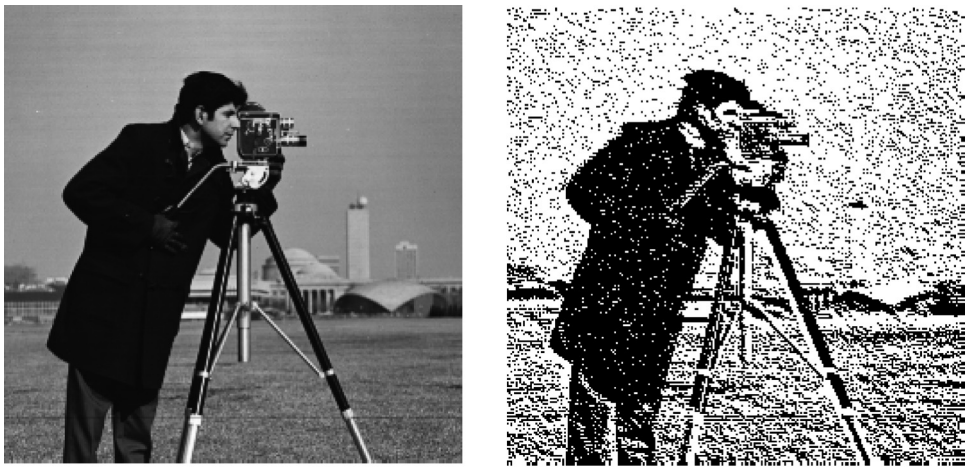
$$\overline{NL'(i)} = \frac{NL''(i)}{A} = NL'(i) + \frac{\sum_{j \in \Omega} \varepsilon_{w_j} v(j)}{A}. \tag{19}$$

The last item $\frac{\sum_{j \in \Omega} \varepsilon_{w_j} v(j)}{A}$ in Equation (19) is the error caused by quantization, which can be controlled by selecting a large enough parameter A. Note that, to get the accurate value of $NL''(i)$ by Paillier decryption, we have to limit $NL''(i) < N$. In other words, the value of A cannot be as large as possible, and must be chosen properly according to the settings of the Paillier cryptosystem. In our scheme, for security reasons, we recommend choosing the product of two primes ($N = p \times q$) for the Paillier cryptosystem, which is longer than 1024 bits. For $NL''(i) \approx A \times NL'(i) < N$, we can get $A < 2^{1024}/2^8 = 2^{1016}$. If we choose $A = 2^7 L^2$, the search window Ω is L^2, and the boundary of the error is

$$\frac{\sum_{j \in \Omega} \varepsilon_{w_j} v(j)}{A} \leq \frac{1/2 \times 255 \times L^2}{2^7 \times L^2} < 1.$$

Paillier Decryption. After receiving $E^{Pail}(I')$, the image owner decrypts it pixel by pixel. According to the homomorphism Equation (7) and Equation (8), the pixel $E^{Pail}(NL''(i))$ is decrypted as

$$NL''(i) = D^{Pail}\left[\prod_{j \in \Omega} (E^{Pail}[v(j)])^{\lfloor Aw'(i,j) \rceil}\right] = \sum_{j \in \Omega} \lfloor Aw'(i, j) \rceil v(j)\ mod\ N. \tag{20}$$



(a) (b)

Fig. 4. Attack on Scheme I: (a) is the original image and (b) is the recovered image by binarization attack.

Then, we use $NL''(i)/A$ to estimate $NL'(i)$ according to Equation (19). Comparing Equation (2) with Equation (20), we conclude that the denoised result obtained in encrypted image is similar to what is obtained in the plaintext image because the weights $\{w'(i, j)\}$ are a good estimation of $\{w(i, j)\}$ thanks to the property of JL Transform.

4.2. Attack on Scheme I

Using Scheme I, the cloud server can get the ciphertext $E^{JL}(I)$ encrypted by JL Transform, which preserves Euclidean distance. But careful readers may notice that the ciphertext $E^{JL}(I)$ keeps the position of the adjacent pixels, as in the plaintext image. Even though attackers may not infer the accurate pixel values, they may get the profile of the image.

Here, we present a simple attack method, shown in Algorithm 3. As the JL Transform described in Section 4.1, for an image of n pixels with the projection dimension k, the size of transformed data will be $n \times k$. First, we calculate the mean of each row in the $n \times k$ transformed data matrix, and get an $n \times 1$ mean vector. From this analysis, the n mean values preserve the relative positions and amplitudes of pixels in the plaintext image. Then, we use image binarization to reset the value of the mean, and reshape this $n \times 1$ binary vector to an image consisting of n pixels. At last, we can get an approximate image of the plaintext image. Figure 4 shows that this attack can successfully recover the profile of the plaintext image.

ALGORITHM 3: Attack on Scheme I: Binarization Attack

Input: $n \times k$ ciphertexts $E^{JL}(I)$ encrypted by JL Transform.
Output: The approximate image I_{app}.

1. Calculate the mean of each row in $n \times k$ matrix $E^{JL}(I)$, get an $n \times 1$ data matrix E_{app};
2. Set the threshold and perform image binarization to E_{app}, then get a binary image I_{app};
3. Output this binary approximate image I_{app}.

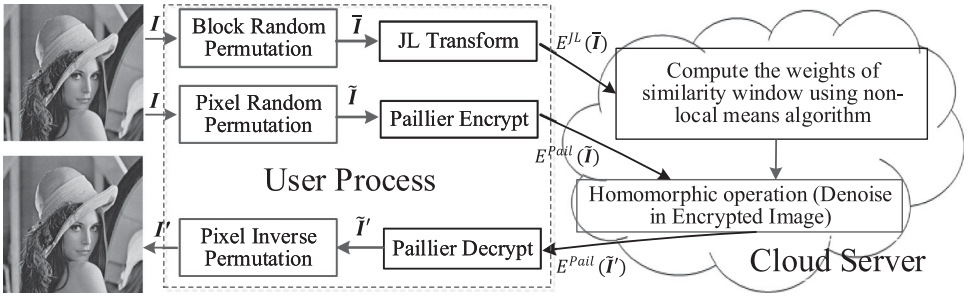


Fig. 5. Scheme II: Security-improved scheme.

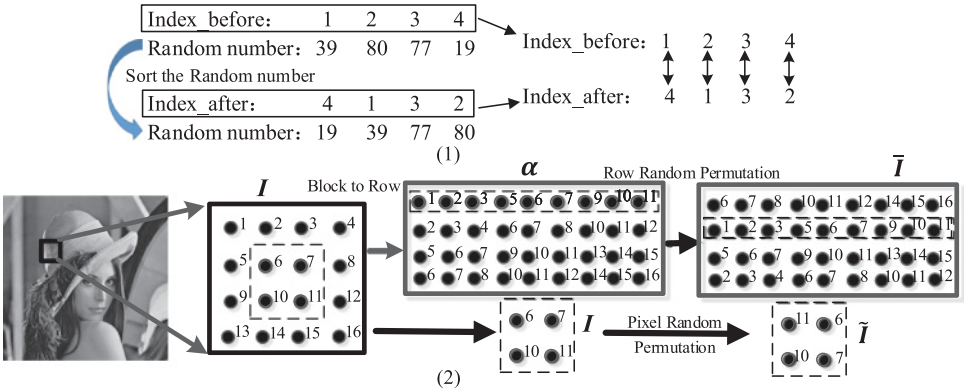


Fig. 6. Random permutation: (1) Generate a pseudorandom permutation sequence; (2) scramble the images.

4.3. Scheme II: Security-Improved Scheme

The privacy leakage shown earlier is caused by the strong correlation of the adjacent pixels. To eliminate this correlation, we propose randomly permutating the images before encryption, then uploading it to the cloud server, as illustrated in Figure 5. We use the method introduced in Deng et al. [2014] to generate a secure pseudorandom permutation sequence. Thus, the complete Scheme II has 5 algorithms: *Random Permutation*, *Encryption with JL Transform*, *Paillier Encryption*, *Denoising in Encrypted Images*, and *Paillier Decryption*. Here, we describe only the random permutation algorithm and its correspondingly modified denoising algorithm in encrypted images.

Random Permutation. To keep the relationship between two ciphertexts encrypted by JL Transform and Paillier encryption, we use two random permutation methods, *Block Random Permutation* and *Pixel Random Permutation*, to permute the two images with the same pseudorandom permutation sequence, respectively. Image I after block and pixel random permutation will be used to perform JL Transform and Paillier encryption, respectively. The owner will perform inverse permutation after getting the denoised encrypted image $\tilde{I}'$. The algorithm of random permutation is described in Algorithm 4.

Figure 6 is an example of **Random Permutation**, which has two parts: (1) generate a pseudorandom permutation sequence and (2) scramble the images.

4.3.1. Generate a Pseudorandom Permutation Sequence. Figure 6 (1) is an example of generating a $4 \rightarrow 4$ pseudorandom permutation sequence. In the first step of Algorithm 4, we use AES to generate four pseudorandom numbers, such as 39, 80, 77, and 14, and

their indices are 1, 2, 3, and 4. Then, we sort the random numbers in an ascending order: 19, 39, 77, and 80, and get a new index: 4, 1, 3, and 2. Thus, $\{1 \leftrightarrow 4, 2 \leftrightarrow 1, 3 \leftrightarrow 3, 4 \leftrightarrow 2\}$ is a pseudorandom permutation sequence.

4.3.2. Scramble the Images. Figure 6 (2) is an example of scrambling the images with block and pixel random permutation, respectively. First, we perform block random permutation, using an example in Figure 3 with $s = 3$. The 3×3 blocks centered at pixel $6, 7, 10, 11$ are transformed to a 4×9 matrix α. Then, the image I will be performed with pixel random permutation, while the matrix α will be performed with row random permutation. After permutation, the indices of pixels in $\tilde{I}$ keep consistent with the indices of rows in $\overline{I}$, because the two permutations use the same random permutation sequence.

ALGORITHM 4: Random Permutation

Input: Image I

Output: Scrambled images $\overline{I}$ and $\tilde{I}$.

1. Compute ciphertexts $C = \{c_1, c_2, \ldots, c_n\}$, where $c_i = E(i, v(i))$ for $i \in \{1, .., n\}$, $v(i)$ is the i-th pixel value, and E is an encryption algorithm (e.g., AES, 3DES);
2. Sort these n ciphertexts in the ascending/descending order and get a new index;
3. Use the new index and the original index to constitute a permutation sequence;
4. Use this permutation sequence to perform block and pixel random permutation;
5. Output the scrambled images $\overline{I}$ and $\tilde{I}$.

Denoising in Encrypted Images. This random permutation may cause some problems for NLM. In the NLM algorithm, the simplest approach to reduce computational complexity is to restrict the search window Ω, as shown in Equation (2). Usually, pixels in such a window are correlative, thus NLM can achieve good performance. However, after random permutation, the local correlation is completely destroyed and the search window will no longer work. The simplest method to solve this problem is directly using full NLM to search the whole image. However, from the analysis in Section 4.5, the computational complexity of full NLM is too large. To accelerate NLM on randomly permutated images, we use the method called Monte Carlo Non-Local Means (MCNLM) [Chan et al. 2014], which randomly computes a small number of image patch distances, according to a designed sampling pattern.

MCNLM is a randomized algorithm to accelerate the full NLM; its basic idea can be described as follows. For the image pixel $v(i)$, we randomly select a number of image patches and compute the weights $\{w_{i,j}\}_{j=1}^{n}$ to estimate the full NLM.

For the image pixel $v(i)$, the sampling process of MCNLM depends on a sequence of Bernoulli random variables $\{J_{i,j}\}_{j=1}^{n}$, $J_{i,j} \in \{0, 1\}$ with the following probabilities:

$$Pr[J_{i,j} = 1] = p_{i,j} \text{ and } Pr[J_{i,j} = 0] = 1 - p_{i,j}.$$

The weights $\{w_{i,j}\}$ are sampled if and only if $J_{i,j} = 1$. For all these probabilities, the sampling pattern is defined as: $\mathbf{p} = [p_{i,j}]_{n \times n}$, $0 < p_{i,j} \leq 1$, and $p_{i,j}$ denotes the probability of the j-th pixel being sampled when we denoise the i-th pixel. Chan et al. [2014] introduced two sampling patterns: uniform sampling and optimal sampling. In the uniform sampling pattern, all $p_{i,j}$ have the same value and are independent of the image itself. In the optimal sampling pattern, calculating $p_{i,j}$ is to solve an optimization problem for estimating the upper bounds of $w(i, j)$, which has two choices to estimate

the upper bounds. In the first choice, the upper bound is estimated by the spatial distance of the i-th and j-th pixels. In the second choice, the upper bound is estimated by the intensity information of the i-th and j-th pixel values.

For the image pixel $v(i)$, the empirical sampling ratio is a random variable, which can be defined as:

$$S_i = \frac{1}{n}\sum_{j=1}^{n} J_{i,j}. \tag{21}$$

Thus, the average sampling ratio is

$$\mathbb{E}[S_i] = \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}[J_{i,j}] = \frac{1}{n}\sum_{j=1}^{n} p_{i,j} \stackrel{def}{=} \xi. \tag{22}$$

The parameter ξ is an important indicator in the MCNLM algorithm to describe the sampling ratio. For a special case $\xi = 1$, the MCNLM becomes the full NLM, and all the image patches are selected with probability one, that is $p_{i,j} = 1$ for all $i, j \in \{1, \ldots, n\}$.

Thus, we can estimate Equation (2), Equation (3), and Equation (4) as follows:

$$w_{i,j} = \frac{\frac{1}{n}\frac{\tilde{w}_{i,j}}{p_{i,j}} J_{i,j}}{\frac{1}{n}\sum_{j=1}^{n}\frac{\tilde{w}_{i,j}}{p_{i,j}} J_{i,j}}, \qquad \tilde{w}_{i,j} = e^{-\frac{\|v(N_i)-v(N_j)\|_2^2}{h^2}}, \tag{23}$$

$$NL(i) = \sum_{j\in\{1,..n\}} w_{i,j}v(j) = \frac{\frac{1}{n}\sum_{j=1}^{n} v(j)\frac{\tilde{w}_{i,j}}{p_{i,j}} J_{i,j}}{\frac{1}{n}\sum_{j=1}^{n}\frac{\tilde{w}_{i,j}}{p_{i,j}} J_{i,j}} = \frac{\sum_{j=1}^{n} v(j)\frac{\tilde{w}_{i,j}}{p_{i,j}} J_{i,j}}{\sum_{j=1}^{n}\frac{\tilde{w}_{i,j}}{p_{i,j}} J_{i,j}}. \tag{24}$$

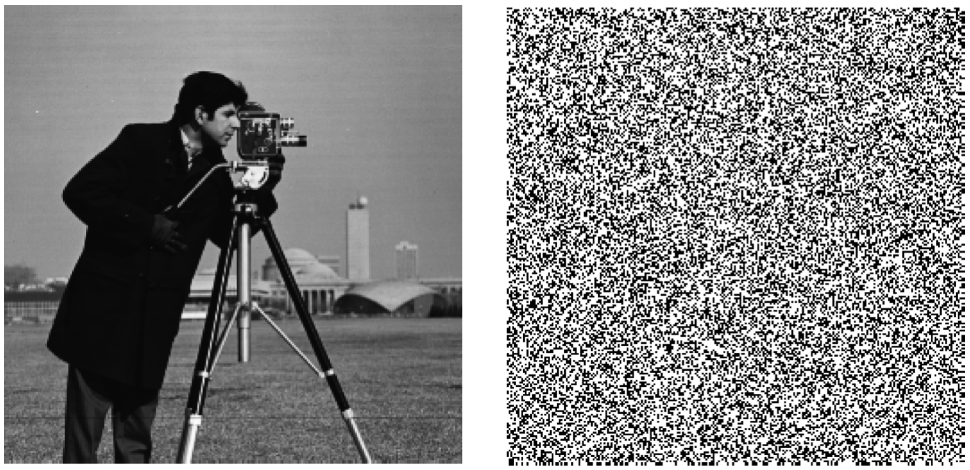
The denoised image using MCNLM can be viewed as an estimation of the denoised image obtained by the full NLM. Chan et al. proved that the probability of the deviation will drop exponentially as the number of pixels n tends to infinity. Therefore, for a large n, the MCNLM algorithm will approach the performance of the full NLM.

We give a complete description of image denoising in the encrypted domain with MCNLM in Algorithm 5.

ALGORITHM 5: Denoising in Encrypted Images with MCNLM

Input: Ciphertexts $E^{JL}(\bar{I})$ and $E^{Pail}(\tilde{I})$. Scaling factor A. Sampling rate ξ.
Output: Denoised ciphertext image $E^{Pail}(\tilde{I}')$.

1. For the noisy pixels $v(i)$ $(i \in \{1, \ldots, n\})$, do
   1. According to the sampling rate ξ, solve the optimization problem, and get $p_{i,j}$;
   2. Generate random variables $J_{i,j} \sim$ Bernoulli$(p_{i,j})$;
   3. If $J_{i,j} = 1$, compute the weight $\tilde{w}_{i,j}$;
   4. Normalize: $w_{i,j} = \tilde{w}_{i,j} / \sum_j \tilde{w}_{i,j}$.
2. end.
3. Compute: $E^{Pail}(\tilde{I}') = \prod_{j\in\Omega} (E^{Pail}[v(j)])^{\lfloor A w_{i,j} \rceil} \ mod\ N^2$;
4. Output the denoised ciphertext image $E^{Pail}(\tilde{I}')$.



(a) (b)

Fig. 7. Scheme II: (a) is the original image; (b) is the recovered image by binarization attack.

4.4. Security Analysis of Scheme II

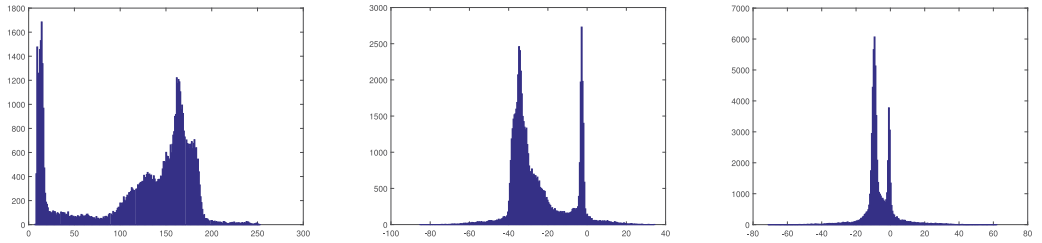
In our Scheme II, we analyze two ciphertexts: ciphertext encrypted by the Paillier scheme and ciphertext encrypted by JL Transform.

Paillier [1999] proved that the Paillier scheme provides semantic security [Goldwasser and Micali 1984], which means that an attacker could not learn any information about the plaintext, except for the length of the plaintext.

Next, we analyze the security of the scrambled image $E^{JL}(\overline{I})$. The pseudorandom permutation sequence in Algorithm 4 is generated by a secure block cipher such as 3DES or AES. As we know, a secure block cipher can be thought of as a secure pseudorandom permutation and its output is computationally indistinguishable from the output of a true random permutation [Katz and Lindell 2007]. Without the private key, the attacker may want to try all possible permutations; however, the number of all possible permutations is $n!$ for an image of n pixels. Thus, for a 512×512 image, the possibility for the attacker to guess the correct permutation is $1/262144!$. Therefore, it is computationally infeasible for any attacker to recover the original image this way.

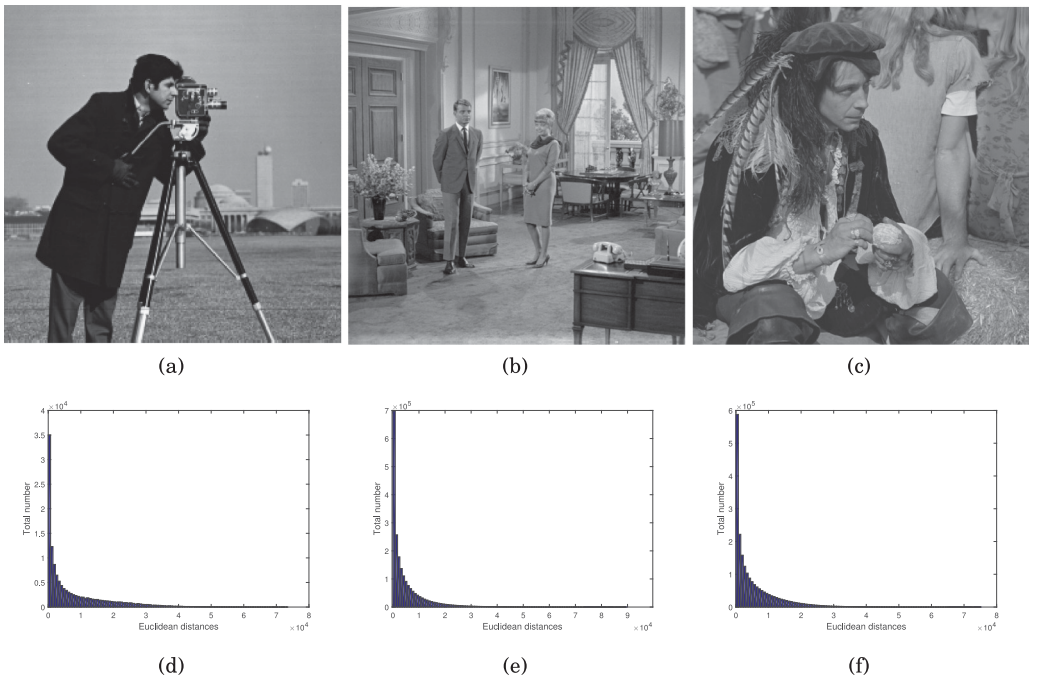
Here, we consider other attack methods. First, we use the binarization attack method to attack Scheme II. As shown in Figure 7, the attacker cannot recover the content of the original image because the locations of pixels have been completely scrambled. Furthermore, as shown in Figure 8, the statistical information has also been concealed, where (a) is the histogram of the original image, and (b) and (c) are the histograms of the mean of each row of ciphertext encrypted by JL Transform from the same plaintext image. In fact, the same image encrypted by JL Transform will get different ciphertexts each time due to the random matrix P and Gaussian noise parameter ζ in JL Transform.

Our scheme leaks the distances of any two pixel blocks, which are needed by the cloud to compute the weights for nonlocal denoising. We randomly selected several images to draw the histograms of Euclidean distances of pixel blocks after JL Transform. Some of the results are listed in Figure 9, which shows that the histograms are very similar whether the original images are similar or not. In fact, for most natural images, many pixel blocks are similar; thus, distances between pixel blocks approximately satisfy



(a) (b) (c)

Fig. 8. Scheme II: (a) is the histogram of the plaintext image; (b) and (c) are the histograms of the mean of each row of ciphertext encrypted by JL Transform from the same plaintext image.



(a) (b) (c)

(d) (e) (f)

Fig. 9. Scheme II: (a) (b) (c) are the origin image, and (d) (e) (f) are the histograms of Euclidean distances of corresponding images.

a power law distribution. Thus, it is hard to determine the similar image using a histogram for the Euclidean distances.

4.5. Complexity Analysis

In our scheme, there are only two parties, client and cloud server. We can see from Figure 5 that the client should perform random permutation, inverse permutation, JL Transform, and Paillier encryption and decryption. The cloud server should a perform nonlocal search to calculate the weights $\{w(i, j)\}$ and mean filter in the encrypted domain. Thus, we analyze the client's and cloud server's complexity, respectively.

4.5.1. Client Side. In the client side, the most complicated algorithm is Paillier encryption and decryption. Jost et al. [2015] showed that there is a faster algorithm

for a variant of Paillier encryption. This faster algorithm is efficient especially if the messages are short.

We can use the variant of the Paillier scheme to replace Equation (5) as

$$c = g^m (g^N)^r \ mod \ N^2, \tag{25}$$

where r is a random number.

In the encryption phase, the values of image pixel are 8-bit integers, which is suitable for Jost's fast algorithm [Jost et al. 2015]. First, we precompute all the value of g^m, because m only has 256 different values, and store them as Table T_1. Then, we precompute 2^{16} random $(g^N)^r$, and store them as Table T_2. When encrypted, we look up Table T_1 to pick g^m, and randomly pick 5 different $(g^N)^r$ in Table T_2, then multiply these 6 values to get the ciphertext of m. Thus, in the encryption phase, the client needs only 5 modular multiplication operations. For a 1024-bit integer N, the size of Table T_1 is $2^8 \times 2048 = 2^{19}$ bits, that is, 64KB, and the size of Table T_2 is $2^{16} \times 2048 = 2^{27}$ bits, that is, 16MB.

As described earlier, JL Transform and Paillier encryption will cause ciphertext expansion. To reduce the storage in the client side, the pixels are encrypted one by one, and each pixel will be directly transmitted to the cloud immediately; thus, the client does not need to store the encrypted images. The ciphertext expansion ratio of JL Transform is equal to the expansion factor k ($k = 9 \sim 18$ in the article), and the ciphertext encrypted by JL Transform does not need to be sent back from the cloud.

For Paillier encryption, the data transmission from client to cloud is about $2|N| \times n$ bits, where $|N|$ denotes the bit length of N, and n is the number of image pixels. Thus, for a 1024-bit integer N, the expansion ratio is about $2 \times 1024/8 = 256$ for 8-bit pixel value, and storage space needed by the cloud is slightly larger than $2|N| \times n$ bits. After the cloud completes the denoising, the data can be transmitted to the client and release the cloud space.

We can also reduce the data transmission from cloud to client by the method proposed in Bianchi et al. [2010], in which the message sequence is divided into l-bit blocks, and T blocks $m_1, \ldots, m_T$ are packed as a composite message $M = m_1 \cdot 2^0 + m_2 \cdot 2^L + \cdots + m_T \cdot 2^{L(T-1)}$, $L > l$. For $n = 256 \times 256$, we choose scaling factor $A = 2^{23}$, so after the cloud completes the denoising, the size of each denoised pixel is about $23 + 8 = 31$ bits for 8-bit pixel value. Thus, the cloud can perform ciphertext packed by setting $l = 31$, and $L = 32$. Therefore, T ciphertexts for $T = |N|/L = 1024/32 = 32$ can be packed together with the method in Bianchi et al. [2010], and the packed ciphertexts can be transmitted to the client, which will reduce the expansion ratio from 256 to 8. This ciphertext-packed method also can speed up Paillier decryption approximately T times, and reduce the decryption time at the client side.

In the decryption phase, Paillier [1999] shows that the constant parameter $L(g^\lambda \ mod \ N^2)^{-1}$ can be precomputed, and the Chinese Remaindering Theorem [Ding 1996] can reduce the decryption workload efficiently.

We present experiments here to simulate the client side's operation. The computer used for simulation has an Intel i5-3210M CPU at 2.5GHz with 4 cores, and it runs Ubuntu 32b v13.04. The time of precomputing Table T_1 and Table T_2 is about 7.3s. These two tables need to be calculated only once; then, they can be used for encrypting a series of images. We list time cost of different parts in Table I. In Table I, NLM means the client performs nonlocal denoising in the plaintext, for which the search window is the whole image. We can see from Table I and Table II that the time of nonlocal denoising increases with n^2, while the time of other operations only increases with n, where n is the number of image pixels. For the 256×256 and 512×512 images, the

Table I. Simulation in the Client Side

Size of image	Random Permutation	Paillier Encryption	JL Transform
256×256	0.8s	1.1s	0.1s
512×512	3.4s	4.4s	0.4s
Size of image	Paillier Decryption	Inverse Permutation	NLM
256×256	3.9s	0.3s	405s
512×512	16.7s	1.2s	6887s

Table II. Computational Complexity in the Cloud Side

Algorithm	Search Window $L \times L$	the Whole Image	Sampling Ratio ξ
NL Search	$n \times s^2 \times L^2$	$s^2 \times n^2$	$s^2 \times n^2 \times \xi$
Paillier Operation	$n(1.5L^2 log(W(i,j)) + L^2 - 1)$	$n(1.5nlog(W(i,j)) + n - 1)$	$\xi n(1.5nlog(W(i,j)) + n - 1)$

total time cost for the client is about 6.2s and 26.1s, respectively, which is much less than the times used for NLM.

4.5.2. Cloud Server Side. Now, we estimate the computational complexity of nonlocal denoising in the encrypted domain, including two parts: nonlocal search, and mean filter in the encrypted domain.

In the first part, if we use a similarity window of size $s \times s$, the computational complexity for searching an image of n pixels is $n \times s^2 \times n = s^2 \times n^2$, that is, $O(n^2)$. If we restrict the size of the search windows to $L \times L$, the computational complexity is reduced to $n \times s^2 \times L^2$, that is, $O(n)$.

In the second part, the denoising is realized by Equation (16). There are only modular multiplication and modular exponentiation, and the operation e^x costs one exponentiation, which approximately costs $1.5log(x)$ multiplications. Hence, we can use the number of modular multiplication to evaluate the computational complexity. For the size of similarity window as $s \times s$ and the size of the search window as $L \times L$, when denoising a single pixel, the number of modular multiplication is $L^2 - 1$ and the number of modular exponentiation is L^2, which costs approximately $L^2 \times 1.5log(W(i,j))$ multiplications. Therefore, for denoising an image of n pixels, the total number of modular multiplication is $n \times (L^2 \times 1.5log(W(i,j)) + L^2 - 1)$. When we search the whole image, the total number of modular multiplication is $n \times (n \times 1.5log(W(i,j)) + n - 1)$.

In Scheme II, we are able to reduce the complexity by random sampling. The most complicated part of the Random Sampling algorithm is computing the weights $\{w(i,j)\}$ and $E^{Pail}(\tilde{I}')$. We therefore omit the other parts. For sampling ratio ξ, the computational complexity of nonlocal search is $s^2 \times n^2 \times \xi$, and the computational complexity of mean filter is the total number of modular multiplication, $\xi n \times (n \times 1.5log(W(i,j)) + n - 1)$ in the encrypted domain.

We list the computational complexity of each part for the cloud side in Table II.

5. EXPERIMENTS

Due to image scrambled in Scheme II, it is infeasible to use the spatial approximation to estimate the upper bound of $w(i,j)$; thus, we use only intensity approximation. We use 10 standard gray test images for this experiment. For each image, we simulate noisy images by adding zero-mean Gaussian noise with standard deviations $\sigma = 10, 30, 50$. Eight average sampling ratios, $\xi = 0.1, 0.2, 0.5, 0.6, 0.7, 0.8, 0.9, 1$, are chosen. The parameters of NLM are set as follows: The patch size is $s \times s = 5 \times 5$ (i.e., $d = 25$) and the search window is the whole image. For JL Transform, four projection dimensions, $k = 18, 15, 12, 9$, are evaluated.

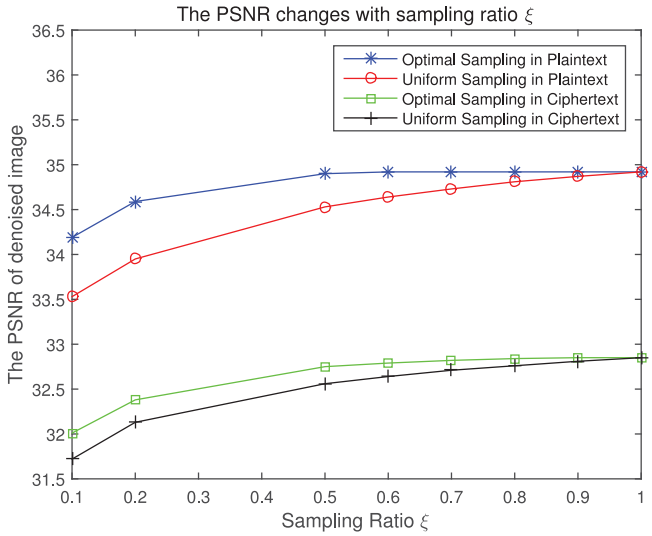


Fig. 10. Image denoising in plaintext and ciphertext using optimal and uniform sampling, respectively.

As MCNLM is a randomized algorithm, we simulate the MCNLM using 12 independent random sampling patterns and 5 independent noise realizations in ciphertext and in plaintext, respectively. The average PSNR values are summarized in Table III and Table IV in the Appendix. The value of σ is 15 in Table III and the value of k is 18 in Table IV. Table III and Table IV show that the PSNR of the denoised images monotonously increases with sampling ratio ξ.

Figure 10 shows the difference between denoising in ciphertext and plaintext for optimal sampling pattern and uniform sampling pattern with intensity approximation. In this experiment, the projection dimension k and the standard deviation of the Gaussian noise ζ of JL Transform are 18 and 0.5, respectively. We can see from Figure 10 that the PSNRs of denoising in cipher images are smaller than that in plaintext images. The accuracy loss of denoising comes from two aspects: the scaling factor A, and parameters of JL Transform such as the dimension reduction k and Gaussian noise ζ. However, k and ζ make JL a privacy-preserving transformation. To observe the influence of scaling factor A, we apply the same JL Transform to the process of plaintext denoising, in which scaling factor A is not used, and compare the result with the denoising quality obtained in ciphtertext. The difference in PSNR between these two cases is less than 0.01dB, by which we conclude that the accuracy loss of denoising in ciphertext is mainly due to JL Transform.

Table III and Figure 11 show that the PSNR of the denoised image decreases monotonously as projection dimension k decreases. Figure 12 shows that the PSNR of denoised images decreases with the standard deviation of the Gaussian noise ζ. In fact, the smaller that k and the bigger that noise ζ are set, the more secure the JL Transform is. Therefore, such quality degradation of denoising image is the cost for preserving privacy; the user can make a trade-off between image quality and security by adjusting k and ζ.

Comparisons of visual quality between denoised images in plaintext and ciphertext are shown in Figure 13 in the Appendix. In this experiment, we use the test image "House" sized (256×256) with noise of standard deviation $\sigma = 10$ and sampling ratio $\xi = 0.5$ to show that the quality of image obtained by denoising in ciphertext is comparable to that in plaintext.

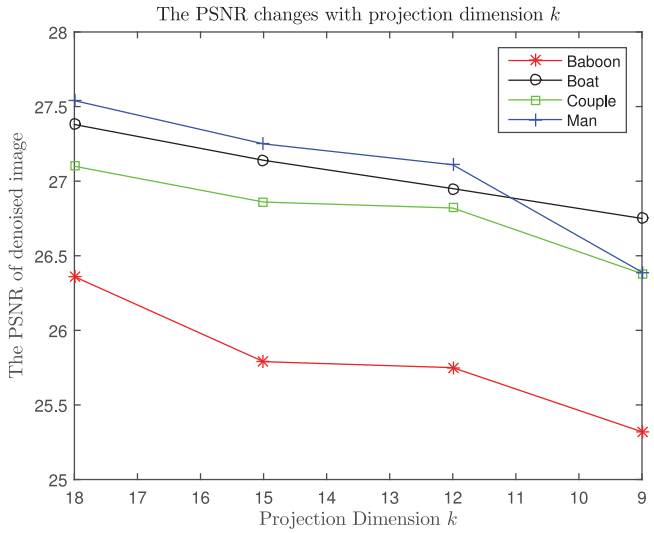


Fig. 11. The denoising effect for different projection dimensions k.

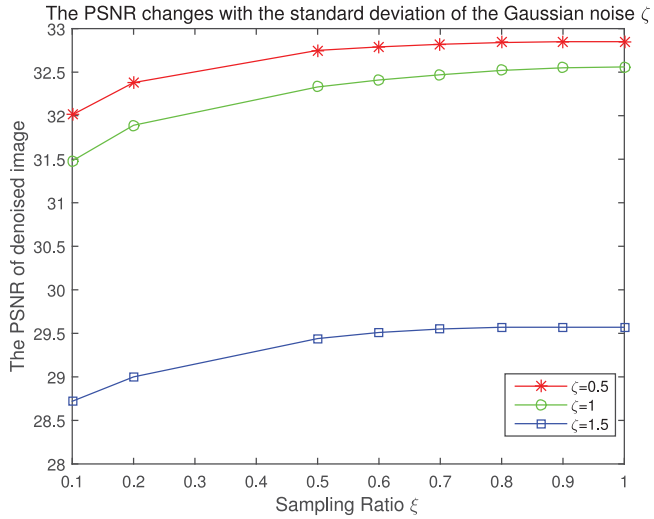


Fig. 12. The denoising effect for different standard deviations of the Gaussian noise ζ used in JL Transform.

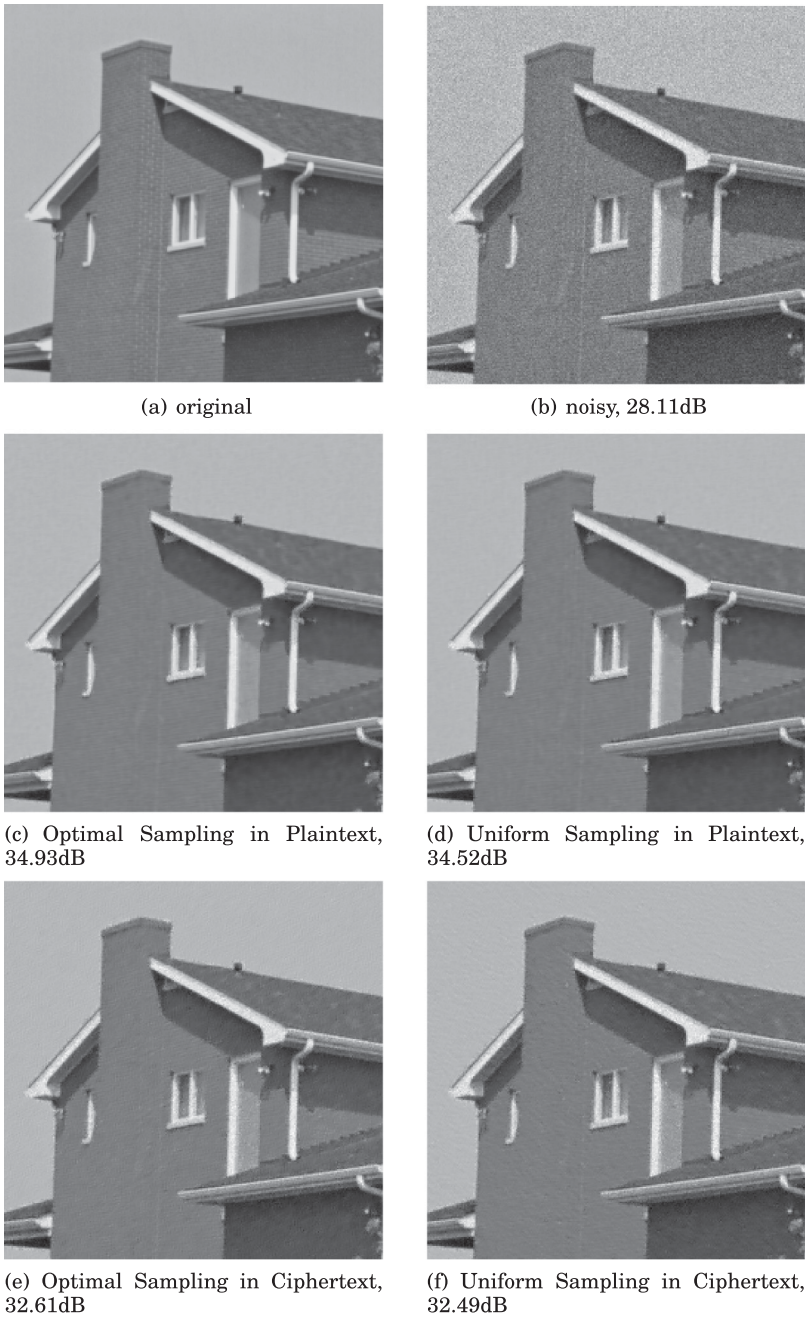
6. CONCLUSION

In this article, we present a double-cipher framework for signal processing in the encrypted domain, in which we combine two kinds of computational functions to implement a special task. Specifically, we propose a nonlocal denoising method in encrypted images by generating two ciphertexts with an additive homomorphic cryptosystem and a privacy-preserving transform, respectively. In this scheme, the nonlinear operation, that is, nonlocal search, can be efficiently realized by the cloud server in ciphertexts. The experimental results show that the denoised image yielded from ciphertexts has comparable quality to what yielded from plaintexts.

APPENDIX

Table III. The Denoising Effect for Different Projection Dimensions k

ξ	0.1	0.2	0.5	0.6	0.7	0.8	0.9	1
k	Baboon 256 × 256							
18	25.79	26.07	26.36	26.41	26.44	26.46	26.47	26.48
15	25.46	25.65	25.79	25.79	25.79	25.79	25.79	25.79
12	25.44	25.61	25.75	25.75	25.75	25.75	25.75	25.75
9	25.08	25.22	25.32	25.33	25.33	25.33	25.33	25.33
k	Barbara 256 × 256							
18	26.38	26.70	27.07	27.12	27.16	27.19	27.20	27.20
15	26.34	26.69	26.96	26.99	27.01	27.02	27.03	27.03
12	25.86	26.18	26.53	26.57	26.59	26.60	26.60	26.60
9	25.57	25.87	26.13	26.14	26.15	26.15	26.15	26.15
k	Boat 256 × 256							
18	26.89	27.13	27.38	27.41	27.44	27.46	27.48	27.48
15	26.87	27.04	27.14	27.14	27.14	27.14	27.14	27.14
12	26.34	26.65	26.95	26.99	27.01	27.03	27.04	27.04
9	26.31	26.57	26.75	26.77	26.77	26.77	26.77	26.77
k	Bridge 256 × 256							
18	26.31	26.57	26.71	26.71	26.71	26.71	26.71	26.71
15	26.02	26.29	26.48	26.50	26.50	26.50	26.50	26.50
12	25.45	25.71	25.95	25.98	25.98	25.98	25.98	25.98
9	25.28	25.46	25.66	25.68	25.69	25.69	25.69	25.69
k	Couple 256 × 256							
18	26.51	26.78	27.10	27.15	27.19	27.21	27.22	27.23
15	26.46	26.73	26.86	26.87	26.87	26.87	26.87	26.87
12	26.42	26.67	26.82	26.83	26.84	26.84	26.84	26.84
9	25.78	26.08	26.38	26.41	26.43	26.44	26.44	26.44
k	Hill 256 × 256							
18	27.05	27.40	27.63	27.65	27.66	27.66	27.66	27.66
15	27.04	27.38	27.61	27.64	27.65	27.66	27.66	27.66
12	25.98	26.36	26.76	26.81	26.84	26.86	26.86	26.86
9	25.94	26.27	26.55	26.58	26.59	26.59	26.59	26.59
k	House 256 × 256							
18	29.33	29.81	30.30	30.38	30.44	30.48	30.51	30.52
15	29.30	29.72	30.13	30.19	30.23	30.25	30.27	30.27
12	28.21	28.71	29.22	29.29	29.34	29.37	29.39	29.39
9	27.31	27.79	28.18	28.20	28.21	28.22	28.22	28.22
k	Lena 256 × 256							
18	29.06	29.42	29.67	29.68	29.70	29.70	29.70	29.70
15	28.55	28.77	28.82	28.83	28.84	28.85	28.85	28.85
12	27.37	27.89	28.36	28.42	28.45	28.48	28.49	28.49
9	26.96	27.44	27.85	27.88	27.89	27.90	27.90	27.90
k	Man 256 × 256							
18	26.79	27.18	27.54	27.59	27.62	27.64	27.65	27.66
15	26.57	26.93	27.25	27.30	27.33	27.35	27.36	27.36
12	26.40	26.78	27.11	27.15	27.17	27.18	27.18	27.18
9	25.84	26.15	26.39	26.41	26.42	26.42	26.42	26.42
k	Peppers 256 × 256							
18	28.36	28.87	29.34	29.41	29.46	29.50	29.53	29.53
15	28.20	28.62	28.98	29.03	29.05	29.07	29.08	29.09
12	26.68	27.16	27.62	27.68	27.72	27.75	27.75	27.75
9	26.52	26.96	27.26	27.29	27.31	27.32	27.32	27.32



(a) original

(b) noisy, 28.11dB

(c) Optimal Sampling in Plaintext, 34.93dB

(d) Uniform Sampling in Plaintext, 34.52dB

(e) Optimal Sampling in Ciphertext, 32.61dB

(f) Uniform Sampling in Ciphertext, 32.49dB

Fig. 13. Scheme II: (a) is the original image, (b) is the image after adding $(0, 10^2)$ Gaussian noise, (c) is the image denoised in plaintext using the optimal sampling pattern, (d) is the image denoised in plaintext using the uniform sampling pattern, (e) is the image denoised in ciphertext using the optimal sampling pattern, and (f) is the image denoised in ciphertext using the uniform sampling pattern.

Table IV. Image Denoising in the Encrypted Domain by MCNLM, Using the Optimal Sampling Pattern

ξ	0.1	0.5	0.8	1	0.1	0.5	0.8	1
σ	Baboon 256 × 256				Barbara 256 × 256			
10	28.61	28.87	28.90	28.90	29.33	29.88	29.94	29.95
30	20.90	21.56	21.60	21.60	22.22	22.82	22.85	22.85
50	18.84	19.48	19.55	19.57	19.89	20.09	20.11	20.11
σ	Boat 256 × 256				Bridge 256 × 256			
10	29.68	30.01	30.03	30.03	28.58	29.00	29.07	29.08
30	22.87	23.38	23.42	23.42	22.51	22.96	22.97	22.97
50	19.73	20.51	20.58	20.60	19.51	20.01	20.06	20.07
σ	Couple 256 × 256				Hill 256 × 256			
10	29.35	29.82	29.87	29.87	29.12	29.81	29.91	29.91
30	22.54	22.98	22.98	22.98	23.24	23.65	23.66	23.66
50	19.72	20.15	20.16	20.16	20.08	20.93	21.03	21.05
σ	House 256 × 256				Lena 256 × 256			
10	32.01	32.75	32.84	32.85	31.20	32.02	32.10	32.11
30	24.64	25.60	25.72	25.74	24.47	24.99	25.03	25.04
50	22.09	22.35	22.37	22.37	21.01	21.77	21.86	21.88
σ	Man 256 × 256				Peppers 256 × 256			
10	29.55	30.11	30.16	30.16	30.75	31.61	31.69	31.70
30	23.27	23.86	23.87	23.87	23.76	24.71	24.83	24.87
50	19.68	20.64	20.77	20.80	20.87	21.38	21.40	21.41

ACKNOWLEDGMENTS

The authors would like to thank all anonymous reviewers and the associate editor for their valuable comments to improve this article.

REFERENCES

Carlos Aguilar-Melchor, Simon Fau, Caroline Fontaine, Guy Gogniat, and Renaud Sirdey. 2013. Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain. *IEEE Signal Processing Magazine* 30, 2, 108–117.

Josh Cohen Benaloh. 1987. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology (CRYPTO'86)*. Springer, 251–260.

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2009a. Encrypted domain DCT based on homomorphic cryptosystems. *EURASIP Journal on Information Security*, 1.

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2009b. On the implementation of the discrete Fourier transform in the encrypted domain. *IEEE Transactions on Information Forensics and Security* 4, 1, 86–97.

Tiziano Bianchi, Alessandro Piva, and Mauro Barni. 2010. Composite signal representation for fast and storage-efficient processing of encrypted signals. *IEEE Transactions on Information Forensics and Security* 5, 1, 180–187.

George Robert Blakley and others. 1979. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, Vol. 48. 313–317.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, New York, NY, 309–325.

Antoni Buades, Bartomeu Coll, and Jean Michel Morel. 2004. On image denoising methods. *CMLA Preprint* 5.

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. 2005. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation* 4, 2, 490–530.

Stanley H. Chan, Todd Zickler, and Yue M. Lu. 2014. Monte Carlo non-local means: Random sampling for large-scale image filtering. *IEEE Transactions on Image Processing* 23, 8, 3711–3725.

Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. 2001. *Multiparty Computation from Threshold Homomorphic Encryption*. Springer, New York, NY.

Ivan Damgård and Mads Jurik. 2001. A generalisation, a simplication and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*, Hideki Imai and Yuliang Zheng (Eds.). Springer, 119–136.

Robert Huijie Deng, Xuhua Ding, Yongdong Wu, and Zhuo Wei. 2014. Efficient block-based transparent encryption for H. 264/SVC bitstreams. *Multimedia Systems* 20, 2, 165–178.

Cunsheng Ding. 1996. *Chinese Remainder Theorem*. World Scientific, Singapore.

Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 4, 469–472.

Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford University, Stanford, CA.

Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*. ACM, New York, NY, 218–229.

Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *Journal of Computer and System Sciences* 28, 2, 270–299.

Xianjun Hu, Weiming Zhang, Honggang Hu, and Nenghai Yu. 2014. Non-local denoising in encrypted images. In *Internet of Vehicles–Technologies and Services*. Springer, New York, NY, 386–395.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*. ACM, New York, NY, 604–613.

William B. Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics* 26, 1, 189–206.

Christine Jost, Ha Lam, Alexander Maximov, and Ben Smeets. 2015. Encryption performance improvements of the Paillier cryptosystem. Retrieved February 8, 2016 from https://eprint.iacr.org/2015/864.pdf.

Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography.* Chapman & Hall/CRC Cryptography and Network Security Series, Boca Raton, FL.

Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. 2012. Privacy via the Johnson-Lindenstrauss transform. *arXiv preprint arXiv:1204.2606*.

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 Processing Magazine* 30, 1, 82–105.

Ankita Lathey and Pradeep K. Atrey. 2015. Image enhancement in encrypted domain over cloud. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM'15)* 11, 3, 38.

Ankita Lathey, Pradeep K. Atrey, and Nishant Joshi. 2013. Homomorphic low pass filtering on encrypted multimedia over cloud. In *Proceedings of the IEEE 7th International Conference on Semantic Computing (ICSC'13)*. IEEE, 310–313.

Peter Mell and Tim Grance. 2009. Draft NIST working definition of cloud computing. Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html.

Claudio Orlandi. 2011. Is multiparty computation any good in practice? In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'11)*. IEEE, 5848–5851.

Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology (EUROCRYPT'99)*. Springer, 223–238.

Kui Ren, Cong Wang, Qian Wang, and others. 2012. Security challenges for the public cloud. *IEEE Internet Computing* 16, 1, 69–73.

Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. 1978. On data banks and privacy homomorphisms. *Foundations of Secure Computation* 32, 4, 169–178.

Sayed M. Saghaian, Nejad Esfahani, Ying Luo, and S.-C. S. Cheung. 2012. Privacy protected image denoising with secret shares. In *Proceedings of the 2012 19th IEEE International Conference on Image Processing (ICIP'12)*. IEEE, 253–256.

Adi Shamir. 1979. How to share a secret. *Communications of the ACM* 22, 11, 612–613.

Andrew Chi-Chih Yao. 1982. Protocols for secure computations. In *FOCS*, Vol. 82. 160–164.

Peijia Zheng and Jiwu Huang. 2011. Implementation of the discrete wavelet transform and multiresolution analysis in the encrypted domain. In *Proceedings of the 19th ACM International Conference on Multimedia*. ACM, New York, NY, 413–422.

Peijia Zheng and Jiwu Huang. 2013a. Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain. *IEEE Transactions on Image Processing* 22, 6, 2455–2468.

Peijia Zheng and Jiwu Huang. 2013b. Walsh-Hadamard transform in the homomorphic encrypted domain and its application in image watermarking. In *Information Hiding*, Jessica Fridrich (Ed.). Springer, 240–254.

Hongchao Zhou and Gregory Wornell. 2014. Efficient homomorphic encryption on integer vectors and its applications. In *Proceedings of the Information Theory and Applications Workshop (ITA'14)*. IEEE, 1–9.

Received August 2015; revised November 2015; accepted December 2015