



Model Parameter Estimation and Inference on Encrypted Domain: Application to Noise Reduction in Encrypted Images

Saetbyeol Lee and Jiwon Yoon^(✉)

School of Information Security, Korea University, Seoul, South Korea
{morningstar_03,jiwon_yoon}@korea.ac.kr

Abstract. One of the major issues in security is how to protect the privacy of multimedia big data on cloud systems. Homomorphic Encryption (HE) is increasingly regarded as a way to maintain user privacy on the untrusted cloud. However, HE is not widely used in machine learning and signal processing communities because the HE libraries are currently supporting only simple operations like integer addition and multiplication. It is known that division and other advanced operations cannot feasibly be designed and implemented in HE libraries. Therefore, we propose a novel approach to building a practical matrix inversion operation using approximation theory on HE. The approximated inversion operation is applied to reduce unwanted noise on encrypted images. Our research also suggests the efficient computation techniques for encrypted matrices. We conduct the experiment with real binary images using open source library of HE.

Keywords: Image processing · Homomorphic encryption
Leveled fully homomorphic encryption · Statistical analysis
Cloud security

1 Introduction

We are currently living in a data deluge era, where large-scale data have been continually generated and accumulated from a variety of sources. Therefore, many data scientists and engineers in various academic communities, including machine learning, statistics, and signal processing, have developed several sophisticated analysis techniques for big data. However, there are security issues in data analysis on the cloud because data is stored and processed, not in the client's, but in the cloud system's storage. Because of this, various privacy preserving approaches, which process and analyze the data on an encrypted domain, have been introduced by much literature.

Homomorphic Encryption (HE) is now considered a powerful solution for providing the security in the cloud system since it is possible to evaluate a

function on encrypted data. That is, the encrypted data are processed in their format without decryption in the cloud. And the advent of Fully Homomorphic Encryption (FHE) scheme [6], which has no limitation on performing additions and multiplications on ciphertexts, has caused a great development in HE. However, FHE has a fundamental limitation in practical implementation although all computational operations can be designed in theory - it is not straightforward to perform complicated operations including division, comparison and conditional branching on ciphertexts. For this reason, FHE was considered hard to be used for real dataset.

In this paper, we overcome the limitation of FHE and show its applicability to real data. We introduce a methodology for applying FHE to real image filtering algorithm. We first explain the background for image filtering and leveled FHE in Sect. 2. Then we propose the inverse operation of the encrypted matrix in Sect. 3, and show how to apply this matrix inversion to the filtering technique in Sect. 4. We also show how to efficiently calculate encrypted matrices in Sect. 5, and Sect. 6 presents our experimental results. Finally, Sect. 7 concludes the paper.

Related Works. The division problem has been actively studied in the area of multi-party computation (MPC). Dahl et al. [7] uses Taylor series to approximate a division operation to a linear function. The authors divide a real number by its bit length, and then use a rounding function to convert it to an integer. Since the bit length of the number is private information, it is kept secret using MPC. Veugen [8, 9] also uses MPC for division approximation. In the paper, the author uses the additive blinding method in order to prevent the input value from being exposed to any parties.

There have also been substantial studies regarding computing a matrix inverse for solving regression problems of machine learning. Hall et al. [10] presented a MPC protocol to solve a linear regression problem on encrypted data. The approach uses Newton's method to compute a matrix inverse. By iterating some linear operations, it can approximate the inverse. Nikolaenko et al. [11] focused on a ridge regression problem by combining the garbled circuit theory [12, 13] with homomorphic encryption. In detail, the approach utilizes homomorphic encryption for linear operations, and garbled circuit for non-linear operations. Lu et al. [16] presented the method of statistical analysis, including linear regression, using FHE. They build the encrypted matrix primitives for data analysis.

After the research of Graepel et al. [14], studies combining HE and machine learning have increased. In [14], the authors present some binary classification algorithms on encrypted data. Similarly, Bost et al. [15] construct the secure comparison operation, and develop three classifiers using the operation.

Contributions

- **Division-free Matrix Operation:** In many statistical analyzes, inverse matrix computation is essential. However, FHE does not support this operation, which is a major obstacle to applying FHE to real data analysis.

We extend the application of FHE by newly introducing the inverse matrix operation method on ciphertext.

- **Training the model parameters on encrypted data:** In order to remove the noise of the image, it is necessary to train the model parameter of the image filter. Most studies that use statistical analysis with FHE focus on applying already trained model parameters to the data. However, training model parameters is the most important step in statistical analysis, and this paper focuses on training step. We show the applicability of this training algorithm by applying the proposed filtering technique to real binary images.
- **Two party model applicable to real-world cloud environments:** Most of the papers presented in related works section require a third-party to assist in the operation between the two parties. The third-party plays a role of authentication and data verification between a server and client. However, organizing the third-party may increase the budget of both sides. In this paper, we propose a novel approach which can run securely without the third-party.

2 Background

2.1 Image Smoothing Filter

Image filtering techniques can be divided into two major categories, low-pass filter and high-pass filter. Low-pass filter, as known as smoothing filter, serves to remove the noise in the image. In contrast, high-pass filter makes the image shaper, emphasizing fine details in the image. Both filtering works the same way, only different with the mask they use. In this paper, we only deal with the low-pass filter.

Image filtering process can be defined as follow:

$$y[m, n] = \sum_{i=-K}^K \sum_{j=-K}^K w_{i,j} x[m + i, n + j]. \quad (1)$$

This equation means that the filtered image pixel $y[m, n]$ is obtained by multiplying values of pixels near $x[m, n]$ by the mask $\mathbf{w} = \{w_{i,j}\}$. The mask \mathbf{w} is a $(2K + 1) \times (2K + 1)$ matrix and we only consider the case when $K = 1$ in this paper. The $K = 1$ case example is shown in Fig. 1.

For smoothing filter, we assume $w_{0,0} = 0$. That is, the filtered image pixel $y[m, n]$ is obtained by weighted sum of the nearby pixels. Then the weight parameter \mathbf{w} should be modeled for filtering. In general, the parameter can be simply estimated by least square estimate. In order to apply least square estimate, we need to transform the Eq. (1) into matrix form. The matrix form of Eq. (1) is as follows:

$$\mathbf{Y} = \mathbf{X}\mathbf{w} + \epsilon,$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} \in \mathbb{Z}_t^{N \times 8}, \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \in \mathbb{Z}_t^{N \times 1}, \quad \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_8 \end{pmatrix} \in \mathbb{R}^{8 \times 1}$$

where $\mathbf{x}_i^T = [x_1^i \cdots x_8^i]$, N is the number of training data, and t is an integer determined by the type of image. For example, $t = 256$ if the image is gray-scale image with all pixel values between 0 and 255. ϵ is signal noise, generally assumed as Gaussian noise. Now, using the least square approach, the weight parameter can be calculated by

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}. \tag{2}$$

Using the estimated weight parameter \mathbf{w} , we can filter the entire image by multiplying \mathbf{w}^* by the total image pixels.

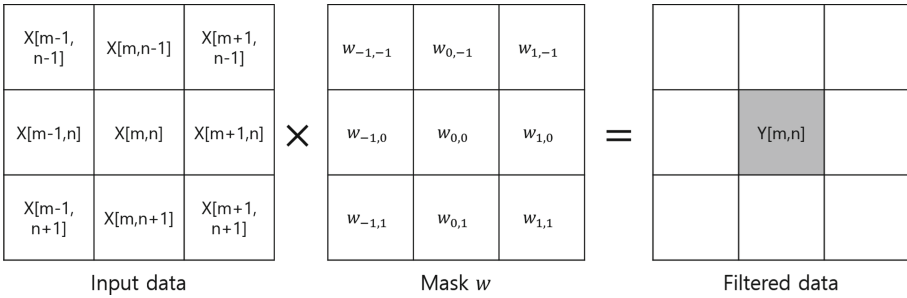


Fig. 1. Image filtering with 3×3 mask

2.2 Leveled Fully Homomorphic Encryption

In this work, we use the Brakerski-Gentry-Vaikuntanathan (BGV)’s scheme [2], which proposed the leveled fully homomorphic encryption. For the sake of simplicity, let $\mathbb{E}[\cdot]$ and $\mathbb{D}[\cdot]$ denote the encryption and decryption algorithm respectively.

The plaintext space of BGV’s scheme is $\mathbb{A}_{p^r} := \mathbb{Z}_{p^r} / \Phi_m(x)$, where p is a prime and $\Phi_m(x)$ is the m -th cyclotomic polynomial. Then the basic operations, addition and multiplication for $a, b \in \mathbb{A}_{p^r}$ work as follow:

$$\mathbb{D}[\text{Add}(\mathbb{E}[a], \mathbb{E}[b])] = a + b \pmod{(\Phi_m(x), p^r)}$$

$$\mathbb{D}[\text{Mul}(\mathbb{E}[a], \mathbb{E}[b])] = a \times b \pmod{(\Phi_m(x), p^r)}.$$

One of the most important points of BGV’s scheme is that SIMD is possible. CRT-packing algorithm in BGV’s scheme makes it possible to pack multiple plaintexts into one plaintext. If the cyclotomic function $\Phi_m(x)$ can be factorized into l -irreducible polynomials, that is $\Phi_m(x) = \prod_{i=1}^l F_i(x) \pmod{p^r}$, then

l -elements $\{a_i\}_{i=1}^l \in \mathbb{Z}_{p^r}^l$ can be packed into one element $\mathbf{a} \in \mathbb{A}_{p^r}$. It is said that the plaintext has l -slots, and each slot element a_i satisfies $a_i = a \bmod (F_i(x), p^r)$ for $i = 1, \dots, l$. CRT-packing algorithm increased the efficiency of the scheme, and we show the use of this techniques in Sect. 5.

3 Matrix Inverse Approximation (MIA)

Recall that it is impossible to perform a division operation on ciphertexts. So, we need to design a division-free method for seeking the inverse of encrypted matrices. We use the Neumann series which approximates the matrix inversion by using iterative method. The formula is as follows:

Theorem 1. *Suppose that a real matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ has the infinity norm $\|\mathbf{A}\|_\infty < 1$, then $\mathbf{I} - \mathbf{A}$ is invertible and its inverse is the series: $(\mathbf{I} - \mathbf{A})^{-1} = \sum_{i=0}^{\infty} \mathbf{A}^i$, where \mathbf{I} is the identity matrix. We can replace $\mathbf{I} - \mathbf{A}$ with \mathbf{B} , then we get:*

$$\mathbf{B}^{-1} = \sum_{i=0}^{\infty} (\mathbf{I} - \mathbf{B})^i \quad (3)$$

where the norm of \mathbf{B} follows $0 < \|\mathbf{B}\|_\infty < 2$.

We can also apply Eq. (3) for an arbitrary matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$. If we multiply \mathbf{C} by a constant $t = 1/\|\mathbf{C}\|_\infty$, then the norm of $t\mathbf{C}$ becomes $\|t\mathbf{C}\|_\infty = 1$. Applying this property to Eq. (3), we have $\mathbf{C}^{-1} \approx t \sum_{i=0}^n (\mathbf{I} - t\mathbf{C})^i$. To use this equation for MIA, however, the real number elements of the matrix $t\mathbf{C}$ must be replaced with integers. So we use the round function, $\mathcal{D} = \lceil qt\mathbf{C} \rceil$, where $q = 10^\gamma$. Note that rounding on a matrix means rounding on every element of that matrix. As q increases, the matrix $\mathcal{D} \in \mathbb{Z}^{d \times d}$ loses less information from \mathbf{C} because the fractional part of a real number can be a corresponding integer. Then, we can approximate the integer form of $t\mathbf{C}$ as $t\mathbf{C} \simeq q^{-1}\mathcal{D}$.

However, the round function cannot guarantee that the norm of the matrix $q^{-1}\mathcal{D}$ falls within the range of 0 to 2. Since $\|qt\mathbf{C}\|_\infty = q$, the infinity norm of \mathcal{D} is in the range $(q - \frac{1}{2}d, q + \frac{1}{2}d]$ for $d \times d$ matrix \mathbf{C} . Then, we can see the range of $\|q^{-1}\mathcal{D}\|_\infty$ as follows:

$$1 - \frac{d}{2q} < \|q^{-1}\mathcal{D}\|_\infty \leq 1 + \frac{d}{2q}.$$

So if q is set to satisfy $d/2 < q$, we can guarantee that $q^{-1}\mathcal{D}$ has the infinity norm between 0 and 2. Subsequently, the matrix inverse \mathbf{C} can be derived as follows:

$$\begin{aligned} \mathbf{C}^{-1} &\approx t \sum_{i=0}^n (\mathbf{I} - q^{-1}\mathcal{D})^i \\ &\approx t \cdot q^{-n} \sum_{i=0}^n (q\mathbf{I})^{n-i} (q\mathbf{I} - \mathcal{D})^i. \end{aligned} \quad (4)$$

Finally, Algorithm 1 demonstrates MIA's overall protocol.

Algorithm 1. Overall protocol of MIA

-
- 1: Client calculates $t = \frac{1}{\|\mathbf{C}\|_\infty}$ and determines $q = 10^7$.
 - 2: Client encrypts q and $\mathcal{D} = \lceil qt\mathbf{C} \rceil$. Then, it transfers $\mathbb{E}[q]$, $\mathbb{E}[\mathcal{D}]$ to the server with the approximation order n .
 - 3: Server calculates $\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i$, and returns it to the client.
 - 4: Client decrypts the returned value and multiplies it by $t \cdot q^{-n}$. Then, it can build the approximated inverse matrix $\mathbf{C}^{-1} \approx t \cdot q^{-n} \mathbb{D} \left[\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i \right]$.
-

4 Encrypted Image Filtering

In this section, we calculate the weight parameter of smoothing filter using the operation suggested in Sect. 3.

Data Sampling. Suppose we have a $M \times M$ size image, and each pixel is an element of \mathbb{Z}_t for an integer t . Then the image consists of a total of M^2 pixels, which is a great burden to use for weight parameter estimation. From this point of view, we use a sampling approach. That is, we sample some pixels for training the weight parameter, not using all pixels. For even sampling, we split the image into N grids and extract the center pixel. Then we have the training dataset $\{\mathbf{x}_i^T, y_i\}_{i=1}^N$. For the sake of simplicity, we assumed $w_{-1,-1}, w_{-1,1}, w_{1,-1}, w_{1,1} = 0$ in Fig. 1 and only use the adjacent 4 pixels for filtering. Figure 2 illustrates the process of sampling data and configuring a training set.

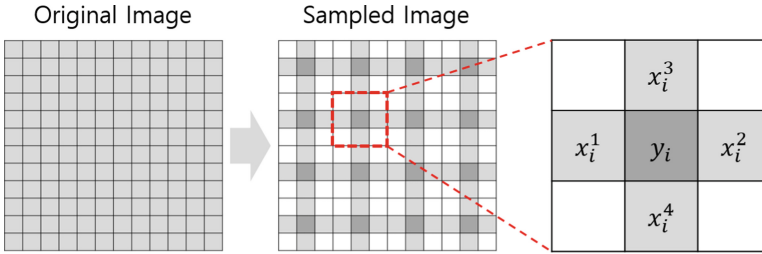


Fig. 2. Method for sampling the data

Model Training. As described in Sect. 2.1, the model parameter can be obtained by the equation $\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$. So, we use the MIA method to compute the inverse of $\mathbf{X}^T \mathbf{X}$. The overall algorithm using MIA is presented in Algorithm 2.

This algorithm has a limitation that once communication between client and server is needed, since it is impossible to calculate the eigenvalue $\|\mathbf{X}^T \mathbf{X}\|_\infty$ from $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$. Of course, there is a way for client to calculate and encrypt $\mathbf{X}^T \mathbf{X}$, then upload it to server. But for large data set, this can be a burden on client and we focus on the convenience of the client rather than the server.

Algorithm 2. Overall Protocol of Weight Estimation

- 1: Client uploads the encrypted training data $\mathbb{E}[\mathbf{X}^T]$, $\mathbb{E}[\mathbf{Y}]$ to the server.
- 2: Server calculates $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$ and sends it back to client.
- 3: Client decrypts $\mathbb{E}[\mathbf{X}^T \mathbf{X}]$ and calculates $t = 1/\|\mathbf{X}^T \mathbf{X}\|_\infty$. It also determines $q = 10^\gamma$.
- 4: Client encrypts q , $\mathcal{D} = \lceil q \cdot t \cdot \mathbf{X}^T \mathbf{X} \rceil$ and transfers $\mathbb{E}[q]$, $\mathbb{E}[\mathcal{D}]$ to the server with the approximation order n .
- 5: Server calculates $\left(\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i\right) \mathbb{E}[\mathbf{X}^T \mathbf{Y}]$, and returns it to the client.
- 6: Client decrypts the returned value and multiplies it by $t \cdot q^{-n}$. Then, it can build the approximated weight parameter

$$\mathbf{w}^* \approx t \cdot q^{-n} \mathbb{D} \left[\left(\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i \right) \mathbb{E}[\mathbf{X}^T \mathbf{Y}] \right]. \quad (5)$$

5 Efficient Computation Techniques

Since the computations on ciphertexts are very heavy, it is important to design an efficient algorithm with as few computations as possible. To do so, we utilize the CRT-packing techniques for matrix encryption and suggest some efficient operations for packed data.

5.1 Encrypted Matrix Operations

Halevi et al. [3] suggested three layouts for encrypting matrix: the row-major order, the column-major order, and the diagonal-major order. In this work, we utilize the row-major order, packing and encrypting each rows of a matrix separately. We use the matrix operation suggested by Lu et al. [16]. Since each rows of matrices are encrypted separately, we can denote the encryption of $d \times d$ matrices \mathbf{X} and \mathbf{Y} as $\mathbb{E}[\mathbf{x}_i^T]_{i=1}^d$ and $\mathbb{E}[\mathbf{y}_i^T]_{i=1}^d$ respectively. The methods proposed by [4] are as follows:

1. Matrix Addition

Addition can be performed simply, just adding each encrypted row:

$$\mathbb{E}[\mathbf{x}_i^T] + \mathbb{E}[\mathbf{y}_i^T] \text{ for } 1 \leq i \leq d.$$

2. Matrix Multiplication

Multiplication is more complicated than the addition operation. To evaluate $\mathbb{E}[\mathbf{X}] \cdot \mathbb{E}[\mathbf{Y}]$, we need to use the Replicate function. $\text{Replicate}(\mathbb{E}[\mathbf{v}], i)$ is the function which replaces all elements of the vector \mathbf{v} with the i -th element. Then using Replicate, we can multiply $\mathbb{E}[\mathbf{X}]$ and $\mathbb{E}[\mathbf{Y}]$:

$$\sum_{i=1}^d \text{Replicate}(\mathbb{E}[\mathbf{x}_j^T], i) \cdot \mathbb{E}[\mathbf{y}_i^T] \text{ for } 1 \leq i \leq d. \quad (6)$$

This matrix multiplication method reduces the complexity from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$.

Algorithm 3. Optimized MIA Computation

```

1: Input:  $\mathbb{E}[\mathbf{q}], \mathbb{E}[q\mathbf{I} - \mathcal{D}]$ , order  $n$ 
2: Output:  $result = \sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \cdot \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i$ .
3:
4:   Initialize  $temp = \mathbb{E}[\mathbf{I}]$ 
5:   for( $i = 0; i < n - 1; i++$ )
6:      $result += \text{Replicate}(\mathbb{E}[\mathbf{q}], n - i) \cdot temp$ 
7:      $temp *= \mathbb{E}[q\mathbf{I} - \mathcal{D}]$ 
8:    $result += temp$ 
9: return  $result$ 

```

5.2 MIA Computation

Recall Eq. (4) of MIA. In order to obtain the inverse of the matrix \mathbf{C} , server should compute $\sum_{i=0}^n \mathbb{E}[q\mathbf{I}]^{n-i} \cdot \mathbb{E}[q\mathbf{I} - \mathcal{D}]^i$. To calculate this expressions in order requires $\mathcal{O}(nd^2)$ multiplications. Furthermore, since $E[q\mathbf{I}]$ and $\mathbb{E}[q\mathbf{I} - \mathcal{D}]$ are vectors with d -ciphertexts, it occupies very large memories. In this section, we suggest a faster and more memory-savvy way to compute MIA. We choose to pre-calculate q^n instead of calculating a heavy matrix multiplication. So client encrypts a vector $\mathbf{q} = [q^1 q^2 \cdots q^n]$ using CRT-packing. By storing a scalar vector \mathbf{q} , server doesn't need to store a large ciphertext matrix. The detailed algorithm is described in Algorithm 3.

Compared to conventional computations, the number of multiplications between two encrypted vectors, represented in Eq. (6), has been reduced by $n(n+1)d$ to $n(d-1)$ through this optimization technique. Also move operation, which is the heaviest operation in CPU, reduced from $3d$ to $2d$.

6 Experiment

For the simplicity, we use a binary image with all pixel values 0 or 1 for our experiment. The size of the image is 100×100 and consists of a total 10,000 binary pixels. We use the sampling method described in Sect. 2.1 to extract 100 data and use it as a training set.

6.1 Error Rate Measurement

Since we use approximation techniques for inverse matrix computations, we need to define the error rate for accuracy measurements. First, we use F1-score to measure the accuracy of image filtering. F1-score is often used in signal processing because it considers recall as well as precision. The definition of F1-score is presented in Fig. 3, where 0 and 1 means the pixel values of the filtered image.

That is, **Precision** means the ratio of pixels having a value of 1 actually among the results filtered by 1, and **Recall** means the ratio of pixels filtered by 1 of the pixels of the actual 1. F1-score is defined as the harmonic mean of **Precision** and **Recall**.

		Our Image Filtering Result	
		1	0
Image Filtering Result	1	a	b
	0	c	d

Precision = $\frac{a}{a+c}$

Recall = $\frac{a}{a+b}$

F1 - score = $\frac{2*Precision*Recall}{Precision+Recall}$

Fig. 3. The definition of F1-score

In addition to F1-score, we also use RMSE(Root Mean Square Error) to measure the accuracy of the model parameter \mathbf{w} . Let \mathbf{w} and \mathbf{w}^* denote the estimated weight using the original scheme and our scheme, respectively. The definition of RMSE is as follows:

$$RMSE = \frac{\|\mathbf{w}^* - \mathbf{w}\|_2}{\sqrt{|\mathbf{w}^*|}}$$

We measure the error rate of the three images using the error rate defined above. Figure 4 shows the result image of our image filtering and its error rate.

6.2 Performance Test

We first used two PCs with Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40 GHz, 64 GB RAM, and Ubuntu 16.04.1 LTS (64-bit) OS. We also used C++ as the programming language.

We used HElib by Halevi and Shoup [5]. This library is based on Brakerski-Gentry-Vaikuntanathan (BGV)’s HE scheme [2]. The parameter p, r, m determines the plaintext space $\mathbb{A}_{pr} := \mathbb{Z}_p / \Phi_m(x)$. m is determined by the function FindM in HElib. m is dependent on prime p , security level δ , and the tolerable noise level L . Also plaintext has l -slots using CRT-packing. In this paper, we use zero-padding in the empty slots. We run performance tests on two cases.

1. $p = 2, r = 32, L = 32, \delta = 64, m = 24929, l = 512$.
2. $p = 2, r = 32, L = 50, \delta = 64, m = 43691, l = 1285$.

The biggest difference between these two cases is L . A high L ensures more operations, but the larger the L , the more exponentially the computation time.

Recall the Eq.(5). Server should compute $\left(\sum_{i=0}^n \mathbb{E}[qI]^{n-i} \mathbb{E}[qI - \mathcal{D}]^i\right) \mathbb{E}[\mathbf{X}^T \mathbf{Y}]$. However, the first case, $L = 32$, cannot support the whole operation. Therefore, we assume that when the server computes $\mathbb{E}[\mathbf{X}^T \mathbf{Y}]$ and sends it to the client, the client re-encrypts it. Therefore, we only measure the time to multiply the already calculated $\mathbb{E}[\mathbf{X}^T \mathbf{Y}]$. In the second case, $L = 50$, the entire operation is possible, so we measure the time that the server performs the whole operation with $\mathbb{E}[\mathbf{X}^T]$ and $\mathbb{E}[\mathbf{Y}]$.










Original Image	Filtered Image	Our Filtered Image	F1-score	RMSE
			0.9962	0.0498
			0.9105	0.0518
			0.9820	0.0546

Fig. 4. Comparison of conventional filtering and our filtering application. We use three images for the experiment. For the original images in the first column, the second column contains filtered images, where the weight is trained on plaintext space. The images in the third column are filtered on ciphertext space using our scheme. The last two columns represent F1-score and RMSE for each image. We use the parameter $q = 10$ and $n = 4$.

Table 1. The result of performance test

L	n	MIA evaluation (s)	The rest evaluation (s)	Decryption (s)
32	1	1.936	1.66	10.840
	2	142.192	2.26	10.796
	3	283.852	2.72	10.844
	4	419.532	3.224	10.620
L	n	MIA evaluation (s)	The rest evaluation (s)	Decryption (s)
50	1	5.744	273.628	34.312
	2	725.996	280.028	34.752
	3	1423.92	276.684	34.436
	4	2181.36	281.624	34.724

Table 1 shows the result of performance test. MIA evaluation means the part of computing $\sum_{i=0}^n \mathbb{E}[qI]^{n-i} \mathbb{E}[qI - \mathcal{D}]^i$, and the rest evaluation means multiplying $\mathbb{E}[\mathbf{X}^T \mathbf{Y}]$ by the MIA result. Each result is the average of the results for the above three images. Both cases are fixed at $p = 10$ because the size of the ciphertext is independent of p , so there is no significant impact on performance time. Decryption time is independent of order n , because n also does not affect the size of the ciphertext.

7 Conclusion

In this paper, we presented a method to apply FHE to image filtering. To calculate the weight of a filter in an encrypted image, we propose an Matrix Inverse Approximation technique that does not require division. We propose an algorithm that removes the noise of the real encrypted image using the MIA technique and suggest an mathematical optimization technique that can efficiently perform the algorithm. Finally we performed the experiment by applying this algorithm to the actual binary image. We conclude that our MIA and optimization techniques will make practical use of FHE.

Acknowledgments. This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the IITP (Institute for Information & communications Technology Promotion) support program (2017-0-00545).

References

1. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, vol. 9, pp. 169–178 (2009)
2. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **6**(3), 13 (2014)
3. Halevi, S., Shoup, V.: Algorithms in HELib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_31
4. Lu, W., Kawasaki, S., Sakuma, J.: Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. In: IACR Cryptology ePrint Archive, 2016, 1163 (2016)
5. Halevi, S., Shoup, V.: HELib. <https://github.com/shaih/HELIB>
6. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
7. Dahl, M., Ning, C., Toft, T.: On secure two-party integer division. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 164–178. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_13
8. Veugen, T.: Encrypted integer division. In: 2010 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE (2010)
9. Veugen, T.: Encrypted integer division and secure comparison. *Int. J. Appl. Cryptogr.* **3**(2), 166–180 (2014)

10. Hall, R., Fienberg, S.E., Nardi, Y.: Secure multiple linear regression based on homomorphic encryption. *J. Off. Stat.* **27**(4), 669 (2011)
11. Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., Taft, N.: Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 334–348. IEEE, May 2013
12. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science, SFCS 2008, pp. 160–164. IEEE, November 1982
13. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE, October 1986
14. Graepel, T., Lauter, K., Naehrig, M.: ML confidential: machine learning on encrypted data. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 1–21. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37682-5_1
15. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS, February 2015
16. Lu, W., Kawasaki, S., Sakuma, J.: Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. IACR Cryptology ePrint Archive 2016, 1163 (2016)