Secure Signal Processing using Fully Homomorphic Encryption

Thomas Shortell and Ali Shokoufandeh

Drexel University
3141 Chestnut St.
Philadelphia, PA 19104
tms38@drexel.edu,ashokouf@cs.drexel.edu

Abstract. This paper investigates the problem of performing signal processing via remote execution methods while maintaining the privacy of the data. Primary focus on this problem is a situation where there are two parties; a client with data or a signal that needs to be processed and a server with computational resources. Revealing the signal unencrypted causes a violation of privacy for the client. One solution to this problem is to process the data or signal while encrypted. Problems of this type have been attracting attention recently; particularly with the growing capabilities of cloud computing. We contribute to solving this type of problem by processing the signals in an encrypted form, using fully homomorphic encryption (FHE). Three additional contributions of this manuscript includes (1) extending FHE to real numbers, (2) bounding the error related to the FHE process against the unencrypted variation of the process, and (3) increasing the practicality of FHE as a tool by using graphical processing units (GPU). We demonstrate our contributions by applying these ideas to two classical problems: natural logarithm calculation and signal processing (brightness/contrast filter).

1 Introduction

Privacy is an important part of today's world. Cloud computing is becoming a norm in today's society as a modus operandi of computation ecosystem. Data stored "in the cloud" may not be encrypted and the cloud computer would be able to read this data. If the data contains personal information, then there may be significant privacy concerns associated the environment. However, even if you can assume the server is trust worthy, there is the potential of a security breach to the cloud computer, which would expose data to the attacker.

We focus on a simpler problem in this space: secure signal processing. Signals can be any function of time or space (sounds waves, images, etc.). Signal processing algorithms can be computationally intensive and offloading the computations to a computationally powerful server may be essential. But performing these processes will become a privacy issue if the signal can not be shared with the server. Thus, the problem is of performing computations of a signal processing algorithm in the secure space so that the server does not know what the data is and if possible the server does not know what the algorithm being run is.

In 2009, the first fully homomorphic encryption (FHE) scheme was developed [2]. This breakthrough provides a method to encrypt data and then process the data in the encrypted form. The scheme provides capabilities for addition and multiplication, which is very useful for signal processing as two simple building blocks. The contribution of this paper is using FHE as a framework to perform secure signal processing in generic fashions on potentially hostile servers. There are three specific contributions that will make tackling this problem possible. First, as the FHE scheme is integer-only based, a method to perform real number processing is developed. Second, there are numerical errors that can be introduced in the FHE process, and a framework to bound this error is also introduced. Bounding the error will be essential as it requires a trade off between accuracy, performance, and security. Third and finally, focus on making FHE a practical tool was partially tackled by determining the costly processing of FHE was a matrix-matrix multiplication and using the power of a GPU to improve the performance towards a more practical scheme. These contributions are realized by application in two classical problems in function evaluation (natural logarithm) and signal processing (brightness/contrast filter on an image).

2 Related Work

There has been much related work in terms of secure signal processing. Troncoso-Pastoriza and F. Perez-Gonzalez [11] were among the the first to propose the secure signal processing problem and considered its possible solution for a cloud computing model. Important results in this article focus on privacy issues in the cloud that will be important in the years to come particularly related to biometrics and health care information. Signal processing is needed particularly in the biometrics field. Wang et. al. [12] is an example of protecting biometric information that is used as an access control. The authors additionally focus on privacy concerns related to the signal data (biometrics). Others have realized the importance of secure signal processing for images and videos [8].

Knežević et. al. [5] also considered a generalized model on using digital signal processors to perform cryptography. This includes performing encryption and decryption processes on signal processors to speed up the respective processes. This relates to our contribution in that processing is offloaded to an GPU for performing improvements. In contrast, this paper focuses on performing a signal processing algorithm in an encrypted form vice just using the signal processor as a performance improvement.

There is a plethora of secure signal processing applications however, none of them use fully homomorphic encryption as the underlying method. Shashanka and Smaragdis [10] developed a method to classify audio signals via secure multiparty communications by hiding the real signal from the server and the classification parameters from the client. This is different from this work as it assumes there are two parties working together to solve the problem but our situation is the use of the server as a resource vice and interaction. Hsu, Lu, and Pei [4] developed a secure signal processing algorithm using the Paillier encryption

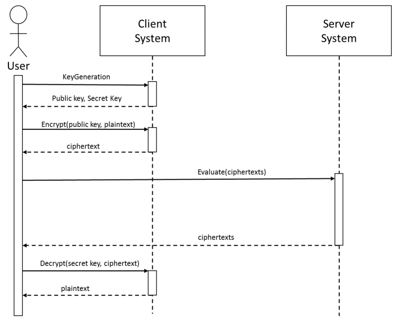


Fig. 1. FHE Scheme Structure and Flow

scheme to extract features from an image using SIFT (Scale Invariant Feature Transform). Paillier encryption scheme is an additive-only homomorphic encryption scheme, meaning that addition is performed on encrypted ciphertexts but multiplication cannot be. Paillier does support constant multiplication as well. The FHE scheme in this paper is a fully homomorphic encryption scheme which provides encrypted multiplication. Others have also worked with the Paillier encryption scheme [6] [7] [1]. Bai et al [1] were able to use Paillier's scheme to implemented an encrypted SURF. While this is closer to the research in this paper, the Paillier scheme only provides homomorphic addition, thus limiting the ability to perform some signal processing algorithms.

3 Fully Homomorphic Encryption

First we wish to go over the basics of fully homomorphic encryption scheme so the capabilities for secure signal processing are available. The scheme we use is a later scheme developed in 2013 by Gentry, Sahai, and Waters [3]. This scheme improved many of the original scheme issues in terms of being a legitimate tool for applications and computationally efficiency. Figure 1 shows how the scheme flows in sequence. As the figure shows, there are four main functions that are used by the user. They are KeyGeneration, Encrypt, Decrypt, and Evaluate. KeyGeneration is the first function and generates the public and secret keys that are used in encrypt and decrypt respectively. At this point, a user can encrypt any number of plaintexts (including a full signal) by using Encrypt. The resulting ciphertexts can then be sent to a server that can run the Evaluate function. While this Evaluate function is running on the server, it is developed by the user. After Evaluate finishes, the ciphertexts can be retrieved (or returned to the user). Finally, Decrypt returns the ciphertexts to their original form.

The public and secret key are generated by the use of a few parameters from the user in KeyGeneration. The underlying security hardness problem is based on the Learning With Errors (LWE) problem. For brevity, the detailed derivations have been kept out of this paper and refer the reader to the original papers [3] [9]. LWE causes the public key to be a matrix and the secret key to be a vector. This is important as it effects the sizes of ciphertexts and the operations for evaluating. Encrypt and Decrypt operate on integers in the ring $\mathbb{Z}_q$ [3]. We discuss in the following sections our approach to work around this limitation. An additional note about these processes is that ciphertexts are square matrices.

Finally, it is necessary to discuss the Evaluate capabilities. There were three functions for the user to use (add, mult, multconst). Since add is addition of two ciphertexts (matrices), addition is an $O(N^2)$ process of adding the matrices. However, mult and multconst are matrix-matrix multiplication (multconst is a simple value multiplied by the identity matrix and then by the ciphertext matrix). These are obviously $O(N^3)$ processes and a cause of concern since this will require significant computations. This is the primary focus on using a GPU to solve this problem.

4 Encrypted Natural Logarithm

Section 3 provides a scheme that can support addition and multiplication, so now we can consider performing an encrypted natural logarithm. Taylor expansions provide a way to calculate a function around a set point in terms of addition and multiplication (two functions available to us). But this is not the only capability needed to make this work. Natural logarithms are performed over real numbers while the FHE scheme supports integers in a ring. A simple solution would be to just round the integers but this would be inaccurate thus ineffective. Another (but not optimal) solution to this problem is to multiply the fraction part of the real number so it is in the integer space. This solves the problem, but it introduces complexity in the form of unnecessary scaling of the ciphertexts when they are multiplied.

The Taylor expansion of a natural logarithm provides an equation form that the FHE scheme can evaluate. Taylor expansions work around a value ($a \geq 1$). The following is the Taylor expansion of the natural logarithm of five terms:

$$\ln(x) \approx \ln(a) + \frac{1}{a}(x-a) - \frac{1}{2a^2}(x-a)^2 + \frac{1}{3a^3}(x-a)^3 - \frac{1}{4a^4}(x-a)^4 + \frac{1}{5a^5}(x-a)^5 \quad (1)$$

This equation is our means to estimate a natural logarithm in a way that can be performed by the FHE scheme. We also need to discuss security and privacy of this construction. It is important to disguise the operation being performed to maintain additional privacy about the data. It is easy to see how the corresponding Taylor expansion is for the natural logarithm. Consider a transform for $\ln(a) \to in_1$, $x \to in_2$, and $-a \to in_3$. Remember these inputs are encrypted. So the processing equation is $out = in_1 + \sum_{i=1}^{5} c_i \cdot (in_2 + in_3)^i$ with c_x for each of the constants of the expansion. Looking at these equations, it may not be immediately obvious that this is a natural logarithm calculation. It may also be hard to distinguish that this is a Taylor expansion. This is partially possible because inputs 2 and 3 are not directly exposed (input 3 is a negative number, which is information not available at the onset to an attacker). The attacker could also be limited by not using constants and actually encrypting the values.

Next, we would like to focus on the issue of quantization that is due to the limitation of FHE in handling real values. By multiplying by a constant factor, a fractional part of a number can be moved into the integer space; similar to a fixed point representation. The difference from an actual fixed point representation is that in fixed point, the fractional part can be contained by performing a division by the constant as part of a multiplication. This is not possible with the FHE scheme - no division capability. Increasing the modulus of the ring will provide additional space to represent the fractional portion of a real number. Consider L multiplication levels and the return factor number is now x^L, with x being the initial factor. This has direct effect on the modulus ($x^L < q$ must be true) and then affects the key and ciphertext sizes as a result. Lastly, we wish to discuss bounding the numerical error that will occur so we can identify a practical, usable scaling factor. We will also bound a to be an integer, since it simplifies the equation and the implementation.

Theorem 1. *Given a natural logarithm Taylor expansion FHE implementation, a value L that is the maximum allowed error, Taylor expansion parameter $a \geq 1$ and $a \in \mathbb{Z}$, then the scaling factor ($\in \mathbb{Z}$) must obey the following equation:*

$$\begin{aligned} \textit{scale} \geq \frac{1}{L} \Bigg(1 + \frac{1}{a} + (x-a) - \frac{1}{a^2}(x-a)^2 + (x-a)^2 + \frac{1}{a^3}(x-a)^2 + (x-a)^3 \\ - \frac{1}{a^4}(x-a)^3 + (x-a)^4 + \frac{1}{a^5}(x-a)^4 + (x-a)^5 \Bigg) \end{aligned} \tag{2}$$

Proof. We start by amplifying the Taylor expansion with an additive error value, $\Delta e = \frac{1}{\text{scale}}$, to all fractional values $\ln(a) + \Delta e + \left(\frac{1}{a} + \Delta e\right)(x + \Delta e - a) + \cdots$. Next, we will separate the error terms from the main Taylor terms to bound the error. This is a straightforward, but tedious process that is not shown here for brevity. We will also only want to keep first order error terms since higher order terms will be comes negligible quicker than the first order terms. Our next equation shows this remaining first order error.

$$\begin{aligned} L \geq \Delta e \Bigg(1 + \frac{1}{a} + (x-a) - \frac{1}{a^2}(x-a)^2 + (x-a)^2 + \frac{1}{a^3}(x-a)^2 + (x-a)^3 \\ - \frac{1}{a^4}(x-a)^3 + (x-a)^4 + \frac{1}{a^5}(x-a)^4 + (x-a)^5 \Bigg) \end{aligned} \tag{3}$$

If we switch $\Delta e = \frac{1}{\text{scale}}$ and rotate terms:

$$\begin{aligned} \text{scale} \geq \frac{1}{L} \Bigg(1 + \frac{1}{a} + (x-a) - \frac{1}{a^2}(x-a)^2 + (x-a)^2 + \frac{1}{a^3}(x-a)^2 \\ + (x-a)^3 - \frac{1}{a^4}(x-a)^3 + (x-a)^4 + \frac{1}{a^5}(x-a)^4 + (x-a)^5 \Bigg) \end{aligned} \tag{4}$$

completing the proof.

The usefulness of Theorem 1 should be immediate. If the user knows the desired error bound, then the scaling factor can be selected to meet their needs. The restrictions on the selection is minimizing the difference between x and a. Following the way that Taylor expansions generally work, this should not be an issue.

5 Brightness/Contrast Filter Theory

To realize the idea of a framework for secure signal processing with FHE, a brightness/contrast filter was implemented. By performing this filter via FHE, this validates the framework for secure signal processing. This is a simple example since the brightness/contrast filter operates on individual pixels one by one; with no interaction between the pixels. But the process can handle the entire image.

As part of creating an encrypted brightness/contrast filter, it is necessary to start with the unencrypted version. The basic equation is very simple:

$$g(x) = \alpha f(x) + \beta \tag{5}$$

where $g(x)$ is the output image and $f(x)$ is the input image. Remember that this filter operates pixel by pixel so two pixels do not directly interact. The only interaction is with the two other parameters in the equation: α and β. α may be known as the gain and β may be know as the bias. It needs to be noted here that α is necessarily a fractional value. This means that to use this equation with an FHE scheme will require us to handle real numbers in some fashion.

The encrypted version of the filter needs to work in the environment of the FHE scheme. The equation itself requires a single multiplication followed by an addition (easily fits in the FHE implementation). Next, the format of the image must be considered because it affects how the α and β parameters are used. Images can be represented in many different formats. One particular format for color images that is used in this paper is representation of pixels as red/green/blue values. Individual colors can take on a variety of formats from a floating point value between 0 and 1 to a discrete value between 0 and 256. This framework was designed to process both potential representations. As a reminder, α and β are not integers and need to be represented as integers using the format discussed in the previous section.

Security aspects of this system need to be considered. First (and always), the FHE parameters need to be set to ensure security. The next item to be considered is whether any information is being given away by the filter itself. From an external perspective the equation looks like this:

$$y_i = c_{(-1)}x_i + c_{(-2)} \tag{6}$$

where $c_{(-1)}$ and $c_{(-2)}$ are the ciphertext equivalents for α and β. Even though the signal is an image, the evaluator can treat the image as a one dimensional array vice a two dimensional array. This will also limit an attackers ability to get any metadata-like information about the signal from the evaluate processing itself. From an attackers point of view, this is a simple one-dimensional multiplication by a constant with the addition of another constant.

Finally, the bound of the numerical error that is introduced by the scaling is important to the framework. First, the bound allows a user to identify a scaling factor that can either limit the error in a controllable way or remove it entirely. Second, the bound provides a method to check performance of the implementation.

Theorem 2. *Given a brightness/contrast filter FHE implementation and L, the maximum allowed error, then the scaling factor must obey the following equation:*

$$scale \geq \frac{a+c+1}{L} \tag{7}$$

where a and c are bounds on the input values α and $f(x)$.

Proof. In the unencrypted world, the basic equation is

$$g(x) = \alpha f(x) + \beta \tag{8}$$

Moving this equation into the encrypted world, the values α, $f(x)$, and β will have some fractional error as a result of the scaling (Δe). The equation becomes:

$$g(x) = (\alpha + \Delta e) \cdot (f(x) + \Delta e) + (\beta + \Delta e) \tag{9}$$

then sorting terms:

$$g(x) = (\alpha \cdot f(x) + \beta) + (\alpha + f(x) + 1) \cdot \Delta e + (\Delta e)^2 \tag{10}$$

where the first grouping is the original calculation, the middle term is the first order error, and the last term is the second order error. Since $(\Delta e)^2$ is less than Δe, this term will be negligible in comparison to the middle term. Additionally, at this point bounds on each of the terms is necessary (looking for maximum amount of error). The Δe term is at most $1/scale$. α and $f(x)$ will be bounded by the actual problem setup that is used for the filter; substituting $a = \alpha$ and $c = f(x)$ for the maximum values. With the value L as the maximum bound for the error, the following must hold:

$$\Delta e \cdot (a + c + 1) \leq L \tag{11}$$

and since $\Delta e = \frac{1}{scale}$, the terms can be rearranged:

$$scale \geq \frac{a+c+1}{L} \tag{12}$$

The importance of Theorem 2 is that a user of the framework can use the scaling factor to bound the error. However, there are additional items that can be considered here as well. If the value for any of the parameters is an integer, then there is no error term for that value in the formula. This enables improvements as the scaling factor can be lowered.

6 Experimental Results

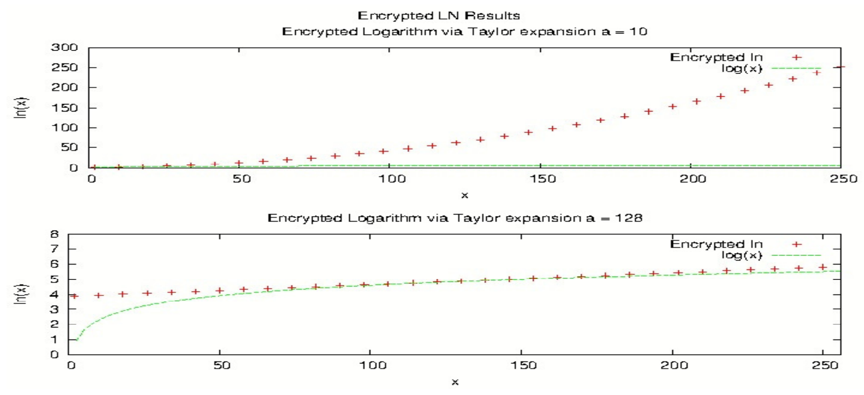


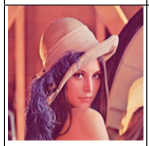
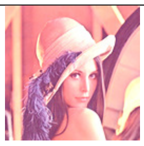
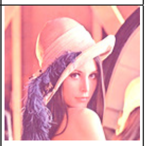
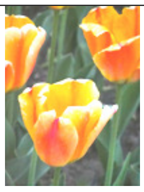
Fig. 2. Encrypted Natural Logarithm via Taylor Expansion Results

In this section, we will evaluate the accuracy of the proposed methods through a set of experimental evaluations. For testing, we worked with input numbers in the ring $\mathbb{Z}_{256}$, a scaling factor of 256 (estimated accuracy of 0.004), and matrices in the size of 390×390. The code was then tested for values of a equal to 10 and 128. Figure 2 shows the results of the testing for values of x at intervals of 8 compared to the actual value. The line represents that actual natural logarithm and the points are the output values of the encrypted natural logarithm. As can be seen in the top diagram, the logarithm value diverges. But comparing the results to actual Taylor expansion of a natural logarithm, the values are actually very close. We show this plot because it is important to remember that there will be trade offs by selecting the value of a in a Taylor expansion. Considering the bottom figure with $a = 128$, it can be seen the accuracy is much better in terms of the value of the natural logarithm. As with the top figure, the results are very closely align to what a five term Taylor expansion would provide (values not shown on the plot). This does remind us that the accuracy will have three parts: a, the number of terms, and any numerical error caused by the scaling factor.

Next, we will present an overview of our implementation for BC filters and evaluate its performance. In our evaluations, we will consider three scenarios. First, comparing the results of running the filter in both domains (encrypted and unencrypted) against a specific image set to determine the total amount of error. Second, comparing the error in the images as a function of their sizes. Third, comparing the affects of different values for α and β.

Starting with the first test, Table 1 shows five images that were processed in both filters (unencrypted and encrypted) with the resulting error. α and β where set to constant values for this test (1.1 and 0.2 respectively). The images

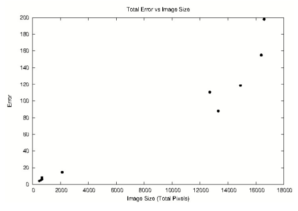
Table 1. Images Tested Under $\alpha = 1.1$ and $\beta = 0.2$

Original Image	No Encryption Filtering	Encrypted Filtering
		
		

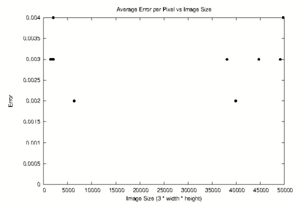
The first column is the scaled original image. The second column shows the image after being BC filtered in an unencrypted form. The third column shows the image after being BC filtered through the encrypted framework. Images are scaled to sizes on the order of $[100, 200] \times [100, 200]$ pixels.

themselves were retrieved from the internet from known images used for signal processing. Focusing on the table, the first column is the original image scaled down to limit the required memory size. The second column is the image run through the unencrypted brightness/contrast filter. The third column is the image run through the FHE scheme's brightness/contrast filter. A naked eye may not easily be able to see the difference between these two columns, but there is some error introduced by the framework. The next paragraph discusses these errors in the second test.

Error introduced into the image by the scheme is an important aspect of whether this framework is a viable approach. Figure 3 shows two perspectives based on total errors that were tallied during testing. Figure 3a shows total error in an image against different sizes of images. Total error obvious increases with image size, which would be expected as there are more and more pixels. Figure 3b focuses the same error data into a per pixel (per color) basis. First, the errors per pixel show that the framework keeps the error below the expected threshold that was specified in Section 5. But there is more to this plot: implicitly, there is a relationship between the error and the type of image. Since the image size was increased from known images, the corresponding points in the x-axis are from equivalent images. This provides an experimental indication that the images' pixels original accuracy will drive how much error is introduced by the scheme. This is aligned with discussion from Section 5.

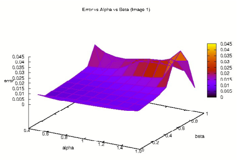


(a)

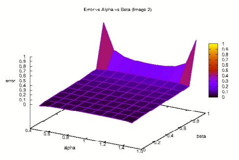


(b)

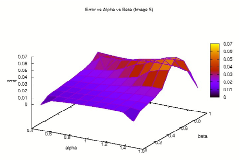
Fig. 3. Error Analysis of different sizes. (a) Total error introduced by the encrypted framework for different image sizes. Error increases as the the image size increase. (b) Average per color pixel that was introduced by the encrypted framework for differing image sizes. Errors per color pixel are within the thresholds and that the individual errors do not grow as the size increases.



(a)



(b)



(c)

Fig. 4. 3D Plot of Average Errors with Varying α and β. α varied from $[0.5, 1.5]$ in increments of 0.1 and β varied from $[0.0, 1.0]$ in increments of 0.1. Each 3D plot is the averaged individual pixel error per image. The images are scaled down to a smaller range $[20, 30] \times [20, 30]$ from Table 1 sizes. Each plot shows that errors are effected by $alpha$ and β and boundary issues begin to occur as β rises closer to one. (a) First image of Table 1. (b) Second image of Table 1. (c) Fifth image of Table 1.

The final test of the framework is varying the values of α and β. Varying these values provides additional insight into the error that can be caused by these two parameters. For the test, $\alpha = [0.5, 1.1]$ and $\beta = [0.0, 1.0]$ at increments of 0.1, respectively. Figure 4 shows the three dimensional plot of these tests. Of particular note immediately is that the values of β at 0.8 and 0.9 are significantly higher than the others and 1.0 is very close $< 10^{-5}$ to zero. This is less of an issue of the scheme itself than of properly handling the boundary conditions: at values of 0.8 and higher for β, this effectively makes the image equal to 1.0 for all pixels. With modular arithmetic, going over 1.0 in the fractional value results in pixels being skewed. In some cases, this was detected and fixed by the framework but it definitely reminds the user that different values of α and β can have unintended affects on the image.

Looking at the individual figures, the obvious artifact is the increase in error as α and β increase. This is a result of the issues discussed in the previous paragraph. Even with the variable caused by going over the limit, much of the error is contained below the error threshold previously setup. In all three tests, if the issues caused at $\beta = 1.0$ are ignored, which is not hard to assume as this is not exactly a useful use of the filter, then the error does hold below the expected threshold. This provides additional experimental evidence for verifying the theoretical work of Section 5. Looking back at these figures, there is evidence that α and β can and do affect the error levels of the results. Previous tests showed that individual images will have different errors and with this latest result, it can be seen that the analytical theory corresponds to the experimental results.

We finish this section by discussing time and space complexity of the schemes. Ciphertexts are matrices so they contribute $N \times N$ space complexity, which expands based on the number of ciphertexts needed (for n: nNN space complexity). For the natural logarithm, our implementation was a constant number of ciphertexts. But the BC filter is effectively an image of $x \times y$ pixels, which translates into $3xy$ ciphertexts for red/green/blue format; a total space complexity of $3xyN^2$. The matrix size is driven by the security parameters of the scheme, so a design decision trade off is needs to be made in all cases between security and complexity. Time complexity is generally going to be driven by the FHE multiplications in an implementation. FHE multiplications are $O(N^3)$ process because matrix-matrix multiplication is the real process inside the scheme. If parallel techniques are not used, this becomes very inefficient compared to an unencrypted version. Because matrix-matrix multiplication is a well known target for parallel computations and in our case with FHE, we were able to use both *openmp* and GPUs to support our needs. Redirecting the *gemm* function of the BLAS allows for significant speed up[1]. For the scaled down image, the processing time drops to less than 15 minutes from an original time of three hours.

7 Conclusion

In this paper, we presented a realization of a FHE protocol for a secure signal processing application. This was achieved by extending FHE to the domain of real numbers by a multiplicative factor, introducing an framework for containing errors introduced for the user to make trade off decisions, and using GPU capabilities to improve the schemes running times. We used two experimental setups

[1] We used CUBLAS https://developer.nvidia.com/cublas

to verify the error introduced by the framework. For the future, it will be very important to improve the quality of processing real numbers and continuing to improve the running times of the scheme.

The authors would like to thank Dr. Thomas Plick for insightful comments on this research.

References

1. Y. Bai, L. Zho, B. Cheng, and Y. F. Peng. Surf feature extraction in encrypted domain. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.
2. C. Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010.
3. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asympotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.
4. C.-Y. Hsu, C.-S. Lu, and S.-C. Pei. Homomorphic encryption-based secure sift for privacy-preserving feature extraction. In *IS&T/SPIE Electronic Imaging*, pages 788005–788005. International Society for Optics and Photonics, 2011.
5. M. Knežević, L. Batina, E. De Mulder, J. Fan, B. Gierlichs, Y. K. Lee, R. Maes, and I. Verbauwhede. Signal processing for cryptograhy and security applications. In *Handbook of Signal Processing Systems*, pages 223–241. Springer, 2013.
6. A. Lathey, P. K. Atrey, and N. Joshi. Homomorphic low pass filtering on encrypted multimedia over cloud. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 310–313. IEEE, 2013.
7. M. Mohanty, W. T. Ooi, and P. K. Atrey. Scale me, crop me, knowme not: Supporting scaling and cropping in secret image sharing. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.
8. W. Puech, Z. Erkin, M. Barni, S. Rane, and R. L. Lagendijk. Emerging cryptographic challenges in image and video processing. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2629–2632. IEEE, 2012.
9. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.
10. M. V. Shashanka and P. Smaragdis. Secure sound classification: Gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 3, pages III–III. IEEE, 2006.
11. J. R. Troncoso-Pastoriza and F. Perez-Gonzalez. Secure signal processing in the cloud: Enabling technologies for privacy-preserving multimedia cloud processing. *Signal Processing Magazine, IEEE*, 30(2):29–41, 2013.
12. Y. Wang, S. Rane, S. C. Draper, and P. Ishwar. A theoretical analysis of authentication, privacy, and reusability across secure biometric systems. *Information Forensics and Security, IEEE Transactions on*, 7(6):1825–1840, 2012.