EPiC Series in Computing

Volume 64, 2019, Pages 129–138

Proceedings of 28th International Conference
on Software Engineering and Data Engineering

Homomorphic Encryption and Data Security in the Cloud

Timothy Oladunni[1] and Sharad Sharma[2]
[1] University of the District of Columbia, Washington DC, USA
[2] Bowie State University, Bowie MD, USA
timothy.oladunni@udc.edu, ssharma@bowiestate.edu

Abstract

In the recent times, the use of cloud computing has gained popularity all over the world. There are lots of benefits associated with the use of this modern technology, however, there is a concern about the security of information during computation. A homomorphic encryption scheme provides a mechanism whereby arithmetic operation on the ciphertexts produces the same result as the arithmetic operation on plaintexts. Concept of homomorphic encryption (HME) is discussed with reviews, applications and future challenges to this promising field of research

1 Introduction

Cloud computing brings a new paradigm to the storage and processing of information. The cloud consists of a scalable, efficient and large pool of computer networks. Services provided by cloud providers include; Software as a Service (Saas), Platform as a Service (Paas) and Infrastructure as a Service (Iaas). The cloud could be a public, private or hybrid infrastructure. The Public clouds are owned and operated by a third party while private clouds are owned by organizations or individuals; this cloud be in the form of On-Premise or External. Hybrid Cloud is partly private and partly public. One of the advantages of the use of the cloud is a reduced cost; users of the cloud have cost reduction as a major advantage in infrastructure and manpower investments. Unlike traditional computing, since users don't install the infrastructure, the cost of installation and maintenance are eliminated. Furthermore, technological evolutionary cost is drastically diminished and recurring expenses are lowered. Cloud computing also offers a very large storage capacity. Another advantage is that the technology brings the ease of use, robustness, scalability and efficiency.

F. Harris, S. Dascalu, S. Sharma and R. Wu (eds.), SEDE 2019 (EPiC Series in Computing, vol. 64),
pp. 129–138

With the advantages of the cloud, a large population of the public has adapted the usage of this technology as part of their daily lives. From the smartphone users on the street to the scientific users in the laboratories, everyone is excited about the cloud. While the public is excited about the advantages of this ubiquitous technology, a large population have concern about data protection, recovery, availability, manageability and government regulations. For the most part, users are concerned with the question, "is my information secured?" Providers of the cloud often guarantee privacy and security of information with the use of existing technology of cryptography algorithms. The algorithms provide the mechanism for encrypting and decrypting information. Encryption algorithms are capable of converting plaintext to ciphertext. Ciphertext is plaintext changed to a scramble or unintelligible format for protection from third parties. Plaintext is the information written in readable and intelligible format.

Unprotected information may be vulnerable to attacks. Attacker could be passive or active. Passive attackers do not change the information content of a data but may obtain part or all through eavesdropping or monitoring. On the hand, the goal of an active attacker is to change the content of a message. Attacks may be in the form of masquerade, replay or denial of service. Most attacks are preventable through encryption. Encryption guarantees the security of information, however, during computation, service providers (third party) may have access to the content of the information. This is because ciphertext is often decrypted to plaintext before or during computation. The challenge here is that most users are not comfortable with their information being exposed to a third party. This leads to the next question "are my information secured during computation?" All stakeholders in the use of the cloud should have a satisfactory answer to this question. A satisfactory answer will assure users of the confidentiality, availability and integrity of their information.

We will discuss related works in the next section and the concept of homomorphic encryption in section three. Section four will be about optimization, while section five focuses on some practical applications of homomorphic encryption. We will discuss future direction in section six and conclude the paper in section seven.

2 Literature Review

The need for a better security of information in the cloud has attracted many research works and studies in the recent times. While most researchers have concentrated their efforts on other areas of the security, a few have given attention to the security of data when it arrives at the server of the service providers. Information sent may be compromised by attackers from within the organization like an employee or external adversary like a cryptanalyst. Darko and Stjepan studied the importance of homomorphic encryption in cloud computing. The study elaborated the strength and weakness of the algorithm. Involvement of IBM in open source library for the algorithm was also discussed in the study [1] .

Maya and Hyotaek argued that homomorphic encryption offers a better security of data in a multi cloud computing. According to the researchers, the use of the algorithm improves data security, segregation, ownership and privacy [2]. Jian et al [3] proposed a simple fully homomorphic encryption algorithm. The algorithm was a derivative of Gentry cryptosystem using only modular arithmetic. The proposed technique has the capability of privacy preservation in an untrusted third party cloud. The performance of the algorithm was better when compared with those proposed by Marten van Dijk et al. in Cryptology EUROCRYPT 2010 and Craig Gentry in March 2010. Baohua and Na [4] studied the principle of homomorphic encryption and its applications. The researchers analyzed the existing algorithms and their improvements.

With a continued concerned about the breach of data confidentiality and privacy by cloud providers and their employees, Adil and Jorg developed a new encryption algorithm. The scientists used homomorphic encryption to build a protocol capable of delegating computations into clouds [5].

Monique et al studied the concept, significance, subdivisions and limitations of homomorphic encryption. A proof of the practicality of the algorithm was also explored in the study [6]. Tebaa et al., [7] implemented and analyzed the performance of an existing homomorphic encryption algorithms. The goal of the experiment was to improve the existing algorithm. Acklyn et al., [8] investigated the vulnerability of Software as a Service (Saas), Platform as a Service (Paas) and Infrastructures as a Service (Iaas) models of cloud computing. The study also discussed various tools like homomorphic encryption with the capability of reducing attacks and threats in the cloud.

Mihai and Cezar [9] analyzed various applications of homomorphic encryption particularly as applicable to the privacy of information in the cloud. Ciara et al., [10] discussed the problem of real time application of homomorphic encryption. The study suggested the implementation of homomorphic schemes using graphic processing units (GPUs) and field programmable gate arrays (FPGAs). Li et al., [11] proposed the approximate eigenvector method as a simple and natural way to obtain a fully homomorphic encryption. The methodology made use of matrix addition and multiplication. The experiment was based on Error-Free Approximate GCD using a diagonal matrix for the plaintext.

An-Ping et al [12] studied the importance of security assurances and success control in the use of the cloud. The researchers investigated the Cipher-text-policy attribute-based encryption algorithm. A combination of homomorphic algorithm and CP-ABE algorithms was used to produce a Searchable Encryption CP-ABE (SE-CP-ABE) algorithm. The outcome of the experiment showed that the proposed algorithm enhances the optimization of cipher-text retrieval.

3 Homomorphism

As illustrated in [13], Alice kept some amount of money in a locked suitcase. She sent the suitcase to Bob and requested that Bob should count the money. While Bob can count the money, there is a restriction; he must not see or touch the money. Therefore, Alice locked the key with the money together in the suitcase. Bob complied with the restriction, found a mechanism to count the money and returned the suitcase with the money back to Alice just as it was handed over to him. Alice was sure that Bob did not see or touch the money.

Just like the mechanism that Bob used, Brian explained that homomorphic encryption provides a technology for computation of ciphertext. As shown in figure 1, Alice (a user of the cloud) requested for computation of messages $E(M_1)\ldots..E(M_n)$, Bob (the service provider) computed the ciphertext using the function $F\{E(M1), \ldots E(M_n)\}$. While in the cloud, the provider did the requested computation without going through a decryption phase. An un-decrypted- computed message was sent back to the user. Thus, the confidentiality of information was not compromised during computation.

In the study, Brian further illustrated homomorphic technology with the example of three positive real numbers *x*,*y* and *z*. The theory was based on the mathematical relationship of the multiplicity of real numbers and the addition of their logarithms. In other words, if the product of x and y gives z, and their logarithmic addition produces log z. intuitively, the anti-log of z is the same result as adding both numbers together. Thus, the result of homomorphic cryptosystem produces a cyphertext of which if decrypted produces the same result as if computation was performed on plaintext.

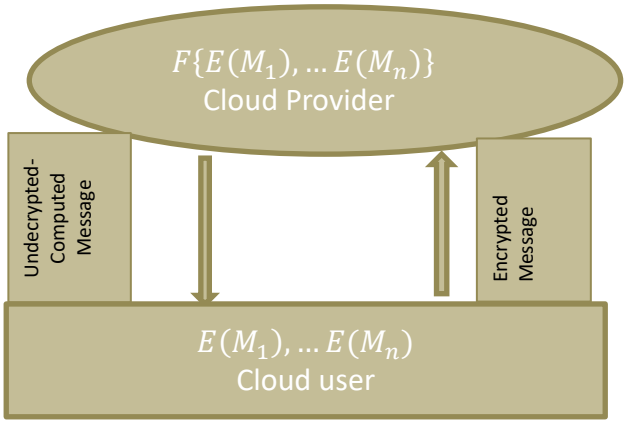

Figure 1. Cloud and User Interaction.

Alice (cloud user) encrypted messages m_1 to m_n (cipher-text) and requested Bob (cloud provider) to compute them. To ensure privacy and confidentiality, Bob did the computation without the decryption phase. The undecrypted-computed result (ciphertext) using homomorphic encryption scheme is the same as result obtained if computations were done on decrypted messages m_1 to m_n (plaintext)

As shown above, given two ciphertexts values x and y, homomorphic algorithms provide a means whereby arithmetic operation on the ciphertext produces the same result as the arithmetic operation on the plaintext. Homomorphic encryption (HME) is categorized into partial (PHE), somewhat (SWE) or fully (FHE). Figure 2 shows the progress that has been made on HME. In the figure, the inner most circle represents the partial encryption scheme. The second layer shows that the scheme has evolved from the PHE phase to SWE. While SWE is an improvement over PHE, FHE which is the out most layer is an improved SWE.

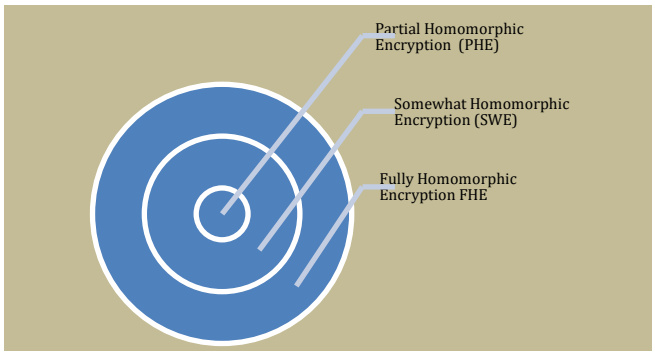

Figure 2. Homomorphic Encryption (HME).

Homomorphic encryption scheme can be either PHE, SME or FHE. In the simplest form, homomorphic encryption began with PHE. PHE has the limitation of one computation at a time, therefore, HME evolved into SME. FHE is improvement over SWE.

3.1 Partial Homomorphism

Partial homomorphism offers one computational operation at a time. An example of partial homomorphic encryption is addition or multiplication on a data. Suppose we have two plaintexts X and

Y, their cyphertexts are obtained using E*(X)* and *E(Y),* where *E* is the encryption mechanism. Using additive homomorphic approach, the sum of the cyphertexts is equal to their encrypted sum. i.e., *E(X) + E(Y) = E (X + Y).* If multiplicative homomorphism is performed on X and Y, the product of the cyphertexts is equal to their encrypted product, i.e., *E(X) . E(Y) = E (X . Y)*. Partial homomorphic encryption may be useful in vote counting applications. Encrypted votes are counted without going through the decryption phase. Confidentiality and privacy of votes are preserved. Two of the most popular types of partial homomorphic algorithms are RSA and ElGamal.

3.1.1. RSA Algorithm

The RSA algorithm was developed in 1977 by Ron Rivest, Adi Shamir and Len Adeleman at MIT. The algorithm is a block cipher technique consisting of a large positive numerical integer *n* [14]. The large size of *n* (1024) and the large amount of computational work required to break the cipher makes the scheme an efficient cryptosystem. In other words, a successful brute-force attack in a reasonable time is practically unrealistic.

Given a message content block M, the cipher-text C is represented as $C = M^e \ mod\ n$. A decrypted message M is in the form of $M = C^d \ mod\ n$. Alice and Bob (sender and receiver) must know the value of *n* which is a product of two arbitrarily chosen prime numbers p and q. A Euler Totient function $\varphi\ (n)$ is calculated as the product of *(p-1)*and *(q-1).* Alice selects a value of *e* which must be less than and a co-prime to $\varphi\ (n)$. Bob calculates *d* using the relationship; $de = 1\ mod\ \varphi(n)$. The value of *e* is known to Alice, while Bob calculates *d*. Thus, RSA is a public key encryption scheme with, $PU = \{e, n\}$ and $PR = \{d, n\}$, where *PU* and *PR* are the public key and private key respectively. The algorithm is considered a partial homomorphic because it allows only multiplications of ciphertexts, but addition is not guaranteed.

3.1.2 ElGamal Encryption Scheme

The ElGamal algorithm is another partial homomorphic algorithm that allows only multiplications of cipher-texts but not addition. The algorithm was proposed in 1984 by T. Elgamal. It is a public encryption scheme based on the Diffi-Hellman scheme [15]. Given α, a prime number less than a prime number q and a primitive root to q. Alice (sender) generates a private key K_A < q -1 and a public key $\{q, \alpha, R_A\}$. Where $R_A = \alpha^{K_A} \bmod q$. Bob (sender) encrypts message *M (M* must be less than q and integer k less than *q).* Key is calculated using the relationship $K_B = (R_A)^k \bmod q$. Message M is encrypted in pairs of two integers (C1, C2) where $C1 = \alpha^k \ mod\ q$ and $C2 = MK_B \ mod\ q$

3.2 Somewhat Homomorphic Encryption.

Partial homomorphic is capable of either multiplicity or addition but not both. Therefore, the scheme is incapable of protection of confidentiality of information during computation in the cloud. Craig Genry 2009 dissertation at MIT, succeeded in providing a new homomorphic scheme capable of allowing multiplication and addition. The scheme comprises of the usual encryption and decryption algorithms with an added *evaluate* function. The *evaluate* functions are in the form of cascaded logic gates (AND, OR, NOT, etc). The main goal here is to make the *evaluate* function perform computation on the encrypted data.

A major setback of Craig's idea is that the *evaluate* function increases the size of the memory required to use the scheme. Because the memory size poses a challenge, *arbitrary depth* was proposed

as a solution. Using *arbitrary depth* sounds good, however, it has the problem of *numerical noise* during implementation,

3.3 Fully Homomorphic Encryption

Moving from somewhat to full encryptions, the issues of limited depths was addressed. One of the proposed solutions is to bring the noise to its lowest level. However, this solution may require a secret key which theoretically are unavailable for a homomorphic encryption. A *refresh* solution was proposed in which an encrypted version of the key is included with the cipher-text.

Table 1 shows the categorizations of the three homomorphic algorithms in tabular form. The table shows that all algorithms in the table are capable of performing computation on encrypted data. While FHE has the capability of computation on both multiplication and addition, it has a major setback in memory usage. SWE is an improvement over PHE, but it suffers from limited depth circuit. PHE can only do one operation multiplication or addition but not both.

Table 1. Comparison of Homomorphic Categories

Description	**Partial Homomorphic Encryption (PHE)**	**Somewhat Homomorphic Encryption (SWE)**	**Full Homomorphic Encryption (FHE)**
Computation on encrypted data	Yes	Yes	Yes
Limitation	One Computational operation	Limited depth circuit	Large memory requirement
Examples	Pallier, RSA and ELGAMAL	Gentry	Gentry, Fujitsu

4 Optimization of FHE

The major challenge of full homomorphism is the memory usage of the algorithm. Various research work has been done with different proposals on an efficient implementation of a full homomorphic algorithm. Hardware as well as GPU solutions has been proposed. Pöppelmann and Güneysu [16] proposed a polynomial multiplication hardware design using FFT. Wang et al. [17] implemented the Gentry and Hallevi's FHE using GPU. The experiment produced a speed factor of 7.68 encryption and 7.4 for the decryption.

Confronting the problem of the inefficient fully homomorphic scheme, Zhigang et al., [18] provided a new scheme with an improved key size. The experiment used a secret key from a binary set. The proposed scheme proved to be an improvement over previous works. Liguan et al., [19] constructed an improved fully homomorphic scheme using an encryption-depth optimization method. According to the researchers, the algorithm is a more efficient method of FHE. Yatao et al., [20] proposed a double decryption scheme as an alternative to the one provided by Gentry in MIT.

5 Application of HME

Practical application of homomorphism is the main motivation of many research works, below are two of its most promising applications:

a. Patient medical condition: One of the most fascinating applications of homomorphic encryption is in disease classifier software. Most predictive models in hospitals are built on plaintext, therefore, confidentiality and privacy of patients' information is a concern. In the previous sections, homomorphic encryption scheme has been described as an effective tool for ensuring the privacy and confidentiality of patients' information during computation. As shown in figure 3., suppose a medical doctor wants to know the medical condition of his patient, he runs a predictive application software on the cyphertext of patients' information. The software application displays the medical situation of the patient on the screen.

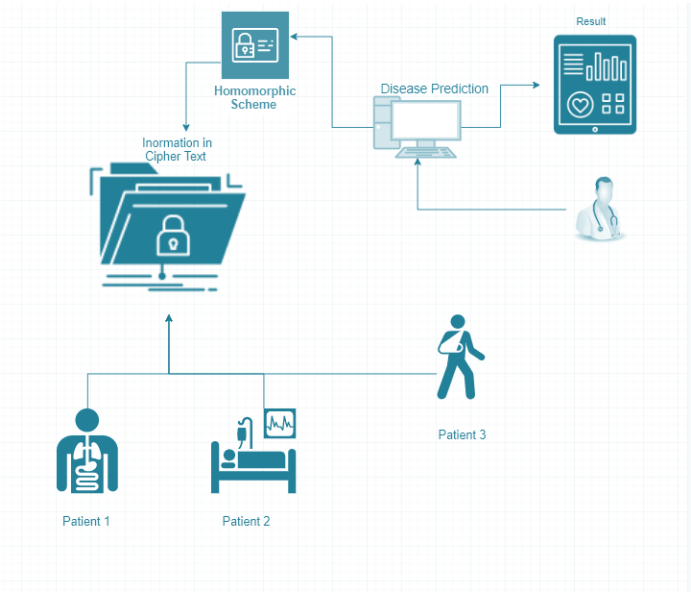

Figure 3. Homomorphic Encryption application in medicine

A homomorphic scheme enhanced application software performs computation on the information of the patients stored in ciphertext without going through a decryption phase. The medical doctor determines (predicts) the medical condition of his patients using cyphertext as his datasets. Privacy and confidentiality of information is ensured during computation.

b. Finance and Banking: Homomorphic encryption technology offers the mechanism for predictive modeling of customers' behaviors without decrypting their information. For the most parts, banks and other financial institutions need some statistical analysis of their customers for reasons such as loan application, customer identification or financial advice. Information analysis is very complicated because of privacy and confidentiality of the customers' information. Using homomorphic encryption scheme, an analyst can successfully produce a detailed report without compromising privacy or confidentiality of the customers' information.

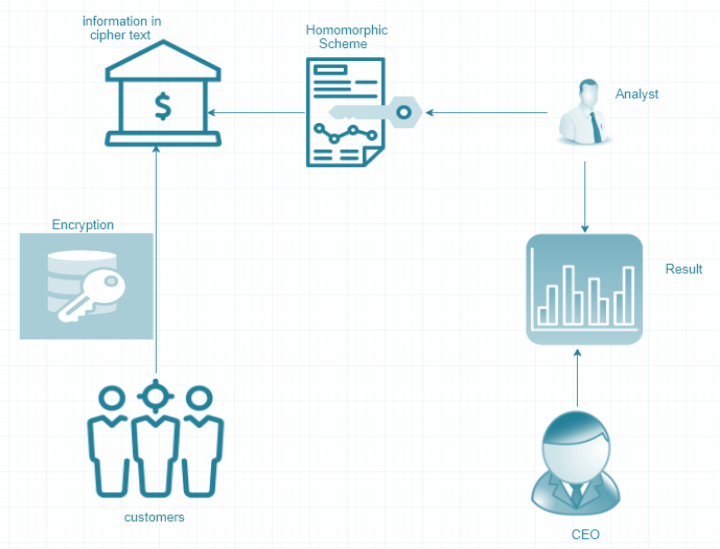

Figure 4. Homomorphic Encryption application in Finance and Banking.

A CEO requested information about some customers in the bank. The data analysis performs computation using homomorphic encryption scheme. Analytical results were made available to the CEO without compromising the customers' privacy and confidentiality.

6 Future Direction –Learning on Cyphertext.

Since the early 50s, computer scientists have been using various methods to create a machine that thinks. A machine that has the capability of talking, working and acting like humans. The initial approach was using mathematical and logical rules for problems relatively difficult for man. Then, it evolved into training a machine using characteristic features of a dataset and testing with holdout examples. The limitation of this approach is that we cannot all describe our feelings to a machine, in most cases human intuition plays a rule in decision making. The emergence of deep learning algorithms has reinvigorated robotic research. Deep learning is an emerging field of computer science that is built on the success of artificial neural networks. It is an interwoven system that uses high-level abstraction. Automated feature learning of deep neural networks has shown to have improved the predictive capability of machine learning. Therefore, it is projected that robots will replace most of the manual work in medicine, banking and financial systems.

As the world gradually embrace the inevitability of artificial intelligence, obviously, confidentiality and privacy of information during computation will be a major concern. Therefore, research work on optimization of learning algorithm on cyphertexts is very critical. One of the fascinating applications of homomorphic encryption scheme is disease classification application software in medicine shown in figure 3. For a low dimensional dataset, computation on cyphertext may not be a challenge. However, biomedical datasets in plaintext have a unique characteristics of high dimensionality. Intuitively, the computational cost of learning on biomedical cyphertext using fully homomorphic encryption will be very expensive. Therefore, we argue that research into an efficient homomorphic encryption scheme in neural networks may be the next stage in cyber-learning research.

7 Conclusion

The use of the cloud has brought a drastic improvement into the storage of information in the cloud. Individuals and organization find the technology relevant to their daily use. However, insecurity of information during computation of data has been a concern to everyone. Therefore, the emergence homomorphic encryption has been an important component in a secured cloud computing. The scheme is categorized into full, somewhat or partial homomorphic encryptions. Each of these algorithms has its own challenge.

a. Partial encryption offers encryption mechanism to secure data during computation, however it can only perform either addition or multiplication. It lacks the ability to perform both.
b. Somewhat homomorphic is an improvement over partial homomorphic. It incorporated *evaluate* function as part of the cryptosystem. This has the challenge of arbitrary depth.
c. Full homomorphism has the same capability as the somewhat. The algorithm provides a *refresh* solution to the arbitrary depth problem of somewhat, however, memory usage is still a setback.

References

[1] Darko Hrestak and Stjepan Picek, "Homomorphic Encryption in the Cloud," *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on , vol., no., pp.1400-1404, 26-30 May 2014*, pp. 1400-1404, 26-30, 2014.

[2] M. Louk and Hyotaek Lim, "Homomorphic encryption in mobile multi cloud computing," *Information Networking (ICOIN)*, pp. pp.493-497, 12-14 , Jan. 2015.

[3] Jian Li, Danjie Song, Sicong Chen, and Xiaofeng Lu, "A simple fully homomorphic encryption scheme available in cloud computing," *Jian Li; Danjie Song; Sicong Chen; Xiaofeng Lu, "A simp Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference* , vol. 01, pp. 214-217, 2012.

[4] Baohua Chen and Na Zhao, "Fully homomorphic encryption application in cloud computing," *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 11th International Computer Conference*, pp. 471-474, 2014.

[5] Adil Bouti and Jorg. Keller, "Towards Practical Homomorphic Encryption in Cloud Computing," *Network Cloud Computing and Applications*, pp. 67-74, 2015.

[6] Claude Turner, Pushkar Dahal Monique Ogburn, "Homomorphic Encryption ," *Procedia Computer Science* , pp. 502-509, 2013.

[7] M. Tebaa, S. El Hajji, and A. El Ghazi, "Homomorphic encryption method applied to Cloud Computing," *Network Security and Systems*, pp. 86-89, 2012.

[8] Geremew Begna, Ebelechukwu Nwafor, Jeremy Blackstone, Wayne Patterson Acklyn Murray, "Cloud Service Security & Application Vulnerability ," *SoutheastCon* , pp. 1-9, 2015.

[9] Mihai Togan and Cezar Plesca, "Comparison-Based Computations Over Fully Homomorphic Encrypted Data," *Communications (COMM)*, pp. 1-6,29-31, 2014.

[10] C. Moore, M. O'Neill, E. O'Sullivan, Y. Doroz, and B. Sunar, "Practical homomorphic encryption: A survey," *Circuits and Systems (ISCAS)*, pp. 2792-2795, 2014.

[11] Xing Li, Jianping Yu, Peng Zhang, and Xiaoqiang Sun, "A (Leveled) fully homomorphic encryption scheme based on error-free approximate GCD," *Electronics Information and Emergency Communication (ICEIEC)*, pp. 224-227, 14-16 , 2015.

[12] An-Ping Xiong, Qi-Xian Gan, Xin-Xin He, and Quan Zhao, "A searchable encryption of CP-ABE scheme in cloud storage," *Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 345-349, 17-19, 2013.

[13] B. Hayes, "Alice and Bob in Cipherspace," *American Scientist*, vol. 100, p. 362, 2012.

[14] William Stallings, *Crytography and Network Security*.: Pearson Education, 2014.

[15] W. Diffie and M. Hellman, "New direction in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.

[16] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice based cryptography on reconfigurable hardware ," *LATINCRYPT*, pp. 139-158, 2012.

[17] Y. Hu, L. Chen, X. Huang, and B. Sunar W. Wang, "Accelerating fully homomorphic encryption using GPU," *HPEC*, pp. 1-5, 2012.

[18] J. Wang, Z. Zhang and X. Song Z. Chen, "A fully homomorphic encryption scheme with better key size," *China Communications*, vol. 11, pp. 82-92, 2014.

[19] H. Ben and J. Huang L. Chen, "An Encryption Depth Optimization Scheme for Fully Homomorphic Encryption," in *Identification, Information and Knowledge in the Internet of Things (IIKI*, Beijing, 2014.

[20] S. Zhang, J. Yang, J. Li and Z. Li Y. Yang, "Targeted fully homomorphic encryption based on a Double Decryption Algorithm for polynomials," *Tsinghua Science and Technology*, vol. 19, no. 5, pp. 478-485, 2014.