Internet of Things and Cloud Computing
2016; 4(2): 13-18
http://www.sciencepublishinggroup.com/j/iotcc
doi: 10.11648/j.iotcc.20160402.12
ISSN: 2376-7715 (Print); ISSN: 2376-7731 (Online)

Using Fully Homomorphic Encryption to Secure Cloud Computing

Ihsan Jabbar, Saad Najim

Department of Computer Science, University of Mustansiriyah, Baghdad, Iraq

Email address:
ihsan.jabbar90@gmail.com (Ihsan J.), alsaad602002@yahoo.com (Saad N.)

To cite this article:
Ihsan Jabbar, Saad Najim. Using Fully Homomorphic Encryption to Secure Cloud Computing. *Internet of Things and Cloud Computing*. Vol. 4, No. 2, 2016, pp. 13-18. doi: 10.11648/j.iotcc.20160402.12

Received: April 6, 2016; **Accepted**: April 29, 2016; **Published**: May 12, 2016

Abstract: The concept of cloud computing receiving a great deal of attention both in publication and among users. Cloud computing is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware resources that are managed by cloud providers at remote locations. The distance between the client and the physical location of his data creates a barrier because this data can be accessed by a third party and this would affect the privacy of client's data. The using of traditional encryption schemes to encrypt the remoted data before sending to the cloud provider has been most widely used technique to bridge this security gab. But, the client will need to provide the private key to the server to decrypt the data before perform the calculations required. Homomorphic encryption allows to perform computations on encrypted data without decryption. This paper deals with the use of homomorphic encryption to encrypt the client's data in cloud server and also it enables to execute required computations on this encrypted data.

Keywords: Cloud Computing, Cloud Security, Fully Homomorphic Encryption, Privacy

1. Introduction

Cloud computing is a hot topic in the information technology field. It enables users to get almost unlimited computing power and it offers potential benefits to these users in terms of instant availability, scalability and resource sharing. Examples of cloud services offers by cloud providers include online file storage (e.g. Dropbox), social networking sites (e.g. Facebook), webmail (e.g. Gmail), and online business application (e.g. Brokerage).

The essential characteristics of cloud computing include *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity* and *measured service*. On-demand self-service means that clients (users or organizations) can request and manage their own computing resources. Broad network access allows services to be offered over the Internet or private networks. Pooled resources mean that customers draw from a pool of computing resources, usually in remote data centers. Rapid elasticity means that services can be scaled larger or smaller. And use of a service is measured and customers are billed accordingly [1].

Although cloud computing has become a mature service model, the adoption of its services by customers (businesses, consumers, etc.) is limited by concerns about the loss of privacy of their private data. Encryption of data could solve this issue, but if the clients want to manipulate their encrypted data in the cloud, they have to share the secret key with cloud provider to decrypt it before execute the required operations [2].

Homomorphic encryption is the appropriate solution to solve cloud computing security issues, since its schemes enable to perform computations on encrypted data without sharing the secret key needed to decrypt the data.

In 2009, Craig Gentry [3] introduced the first fully homomorphic encryption (FHE). In 2010, M. Dijk, C. Gentry et el. [4] presented a second fully homomorphic encryption. In the march 2010, Gentry [5] proposed a homomorphic encryption scheme (called Gen10), heading toward widespread use of cloud computing. Unfortunately these schemes are insecure to use in cloud computing. In 2012, Jian Li, Danjie Song et al. [6] proposed a simple FHE derived from Gentry cryptosystem to ensure the privacy in cloud storage, namely SDC scheme. In 2014, Chen and Zhao [7] proposed an improvement to the second scheme of

Gentry to make the application of FHE in cloud is more secure.

Many researchers proposed different applications of homomorphic encryption in cloud computing. In 2014, Yan Zhang et el. [8] proposed a secure image retrieval method for cloud computing based on homomorphic properties of Paillier scheme. In 2015, Kocabas and Soyata [9] presented a method for privacy-preserving medical cloud computing using fully homomorphic encryption. In January 2016, Shu Qin Ren et el. [10] proposed an XOR homomorphism encryption scheme to support secure keyword searching on encrypted data foe cloud storage.

This paper addresses the security and confidentiality of user data in cloud computing. The main aim of the paper is to introduce the concepts of homomorphic encryption and how to exploit these concepts to secure cloud computing data.

The rest of the paper is organized as follows: section 2, describes the cloud computing concepts and its deployment and services models. Section 3 discusses the security issues of cloud computing with related solutions. Section 4 provides the definition of homomorphic encryption and discusses some examples of existing homomorphic schemes. Section 5 describes how to use homomorphic encryption to secure cloud computing data. Finally, our conclusions are drawn in section 6.

2. Cloud Computing

Cloud computing can significantly reduce the cost and complexity of owning and operating computers and networks. If an organization uses a cloud provider, it does not need to spend money on information technology infrastructure, or buy hardware or software licenses. Cloud services can often be customized and flexible to use, and providers can offer advanced services that an individual company might not have the money or expertise to develop.

Cloud computing is a subscription-based service where the user can obtain networked storage space and computer resources. To illustrate the idea, cloud computing is similar with the dealing with email client (e.g. Gmail, Yahoo, and so on), the provider of email client provides all of the hardware and software necessary to support the email account. When the user wants to access his email he opens the web browser, and goes to the email client, and makes log in. The most important part of the equation is that the user must have internet access. The email is not housed on physical computer of the user, he accesses it through an internet connection from anywhere. The email client is different than the software installed on the user's computer, such as a word processing program. When the user creates a document using word processing software, that document stays on the device used to make it, unless the user physically moves it. An email client is similar to how cloud computing works. Except instead of accessing just the email services, one can choose what information he have access to within the cloud. The main structure of cloud computing is illustrated in figure 1.

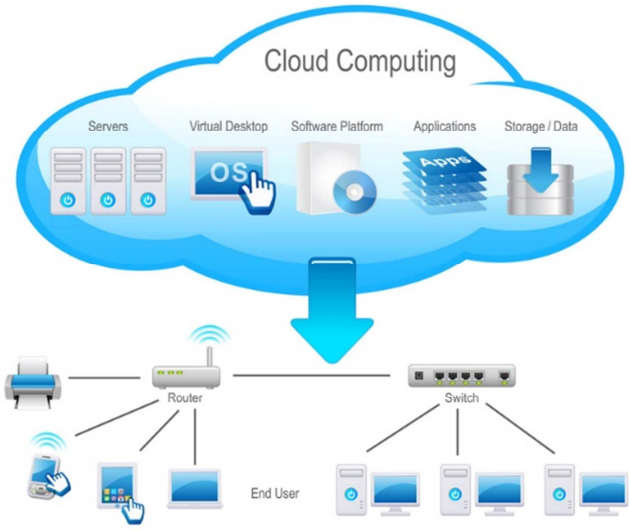

Figure 1. *Cloud computing structure*

2.1. Definition of Cloud Computing

The following definition [1] of cloud computing has been provided by National Institute of Standards and Technology (NIST) of U.S.

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models".

2.2. Types of Deployment Models

There are four deployment models of cloud computing as follows [11]:

- *The Public Cloud:* A public cloud can be accessed by any subscriber with an internet connection and access to the cloud space.
- *The Private Cloud:* A private cloud is established for a specific group or organization and limits access to just that group.
- *The Community Cloud:* A community cloud is shared among two or more organizations that have similar cloud requirements (e.g. security requirements).
- *The Hybrid Cloud:* A hybrid cloud is a combination of at least two clouds, where the clouds included are a mixture of public, private, or community.

2.3. Types of Service Models

There are three service models of cloud computing as follows [1]:

- *Software as a Service (SaaS):* A SaaS provider gives subscribers access to both resources and applications. SaaS makes it unnecessary for the user to have a physical copy of software to install on his devices. SaaS also makes it easier to have the same software on all of

user's devices at once by accessing it on the cloud (e.g. web-based email),). In a SaaS agreement, user have the least control over the cloud.

- *Platform as a Service (PaaS):* A PaaS provider gives subscribers access to the components that they require to develop and operate applications over the internet. Like using the programming languages, libraries, services, and tools supported by the provider.
- *Infrastructure as a Service (IaaS):* An IaaS agreement deals primarily with computational infrastructure. In an IaaS agreement, the subscriber completely outsources the storage and resources, such as hardware and software (e.g. host firewalls).), that they need.

3. Security of Cloud Computing

The use of cloud computing has increased rapidly in many organizations. Concomitantly, the problems of third party data security and securely outsourcing computation become increasingly prominent. There is the risk that personal information sent to a cloud provider is often seen as valuable to individuals with malicious intent and might be kept indefinitely or used for other purposes. Also, such information could also be accessed by government agencies, domestic or foreign and this might affect the privacy of user.

There are some security issues in cloud computing such as data security, third-party control, and privacy. If all data stored in cloud were encrypted using traditional cryptosystems, this would effectively solve the three above issues [12].

To perform a required computation on encrypted data stored in cloud, a user must share the secret key with cloud provider. First, cloud provider decrypts the data to execute necessary operations then sends the result to the user. To solve this issue, it is necessary to use a cryptosystem based on homomorphic encryption to encrypt the data. Since these cryptosystems allow to do computation on encrypted data.

4. Homomorphic Encryption

Homomorphic encryption is a type of encryption that allows particular computations to be conducted on ciphertext and return an encrypted result, the decrypted of result is equal the result of conducting the operation on the plaintext. The property of homomorphic is useful to develop a secure e-voting system with high privacy data retrieving scheme, also it makes the use of cloud computing by ensuring the privacy of processed data. An example for its mathematical consistency, if there are two numbers 10 and 20 then both are encrypted to 56 and 69 respectively, the addition operator gives a number with value 125, the decrypted of this value is 30.

4.1. History of Homomorphic Encryption

The concept of homomorphic encryption was suggested in 1978 by Ronald Rivest and Leonard Adleman [13]. But for 30 years the progress is very slow. In 1982, Shafi Goldwasser and Silvio Micali [14] proposed their encryption system that able to encrypt one bit in additive homomorphic encryption. Pascal Paillier 1999 [15] suggested another additive homomorphic encryption. In 2005, Dan Boneh, Eu-Jin Goh and Kobi [16] invented a security system of encryption which conduct only single multiplication but large number of additions. In 2009 [3], Craig Gentry construct a fully homomorphic encryption based system that able to conduct both of addition and multiplication in the same time.

4.2. Categories of Homomorphic Encryption

There are two main categories of homomorphic encryption schemes: Partially Homomorphic Encryption (PHE) and Fully Homomorphic Encryption (FHE) schemes. PHE schemes, such as RSA, ElGamal, Paillier, Etc., allow to perform either addition or multiplication on encrypted data. Construction of scheme supporting both operations simultaneously was elusive. Although Boneh et al. [8] came closest, allowing unlimited additions and a single multiplication, It was not until 2009 that the three decade old problem was solved in seminal work by Gentry [3], where he showed that performing both addition and multiplication simultaneously are possible in fully homomorphic encryption.

4.2.1. Partially Homomorphic Encryption

A. Multiplicative Homomorphic Schemes

A Homomorphic Encryption is multiplicative, if there is an algorithm that can calculate *Enc(x × y)* from *Enc (x)* and *Enc (y)* without knowing *x* and *y* [17]. Such as RSA and ElGamal Algorithms. Figure 2 illustrates the RSA algorithm as an example of multiplicative homomorphic schemes [18].

- **Key Generation**

  Select two large primes p and q, such that $p \neq q$.
  $n = p * q$.
  $\varphi(n) = (p-1)*(q-1)$, where φ is Euler's totient function.
  Select an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n))=1$ (coprime).
  $d = e^{-1} \bmod \varphi(n)$
  Public key $\leftarrow (e,n)$
  Private key $\leftarrow d$

- **Encryption**

  $c = m^e \bmod n$

- **Decryption**

  $m = c^d \bmod n$

Figure 2. *RSA algorithm.*

The multiplicative homomorphic property of RSA scheme is as follows [19]:

$$\text{Given } c_1 = {m_1}^e \bmod n,\ c_2 = {m_2}^e \bmod n$$

$$\begin{aligned} c_1 . c_2 &= E_{pk}(m_1) . E_{pk}(m_2) \\ &= {m_1}^e . {m_2}^e \pmod n = (m_1 . m_2)^e \pmod n \\ &= E_{pk}(m_1 . m_2) \end{aligned} \quad (1)$$

B. Additive Homomorphic Schemes

A Homomorphic Encryption is additive, if there is an algorithm that can calculate *Enc(x + y)* from *Enc(x)* and *Enc*

(y) without knowing *x* and *y* [17]. Such as Paillier and Goldwasser-Micali algorithms. Figure 3 illustrates the Paillier algorithm as an example of addtive homomorphic schemes [15].

- **Key Generation**

  Select two large random primes p and q, such that $p \neq q$.
  $n = p * q$.
  Compute $\lambda = lcm\ (p-1, q-1)$
  Choose $g \in Z_{n^2}^*$, such that n divides the order of g
  Public key $\leftarrow$ (g, n)
  Private key $\leftarrow$ (p, q)

- **Encryption**

  $m \in Z_n$
  $c = g^m . r^n \ (mod\ n^2)$, where $r \in Z_r^*$ is randomly chosen.

- **Decryption**

  $m = (L\ (c^\lambda (mod\ n^2)))\ (L\ (g^\lambda (mod\ n^2)))^{-1}\ (mod\ n)$,
  where $L(u) = (u-1)/n$

Figure 3. *Paillier algorithm.*

The homomorphic property of Paillier scheme can be shown as follows [12]:

$$\begin{aligned} E(m_1).E(m_2) &= (g^{m_1}.\ r_1^n)\ (g^{m_2}.\ r_2^n) \\ &= g^{m_1+m_2}\ (r_1 + r_2)^n \\ &= E(m_1 + m_2\ (\text{mod n})) \end{aligned} \quad (2)$$

4.2.2. Fully Homomorphic Encryption

All of PHE schemes allow homomorphic computation of only one operation, either addition or multiplication, on encrypted date, except the Boneh-Goh-Nissim scheme which supports performing unlimited number of addition operation but only one multiplication. The constructing of a scheme that allows one to compute arbitrary computation (a scheme should allow an unlimited number of both addition and multiplication operations) over encrypted data has remained a central open problem in cryptography for more than 30 years and thought to be impossible until 2009, when Craig Gentry proposed the first plausible construction of a fully homomorphic scheme [3].

Gentry's work is supporting multiplication and addition in the same time, correspond to AND ($\wedge$) and XOR ($\oplus$) in Boolean algebra. The remarkable value of supporting these two Boolean functions is that any computation can be converted into a function that contains only ($\wedge$) and ($\oplus$). In algebra, there are several techniques can be used to convert a function into more simple. By using this techniques can be convert a function to use only specific Boolean operation (e.g. $\wedge$ or $\oplus$). For example $\neg A$ can be expressed as $A \oplus 1$, another example is $A \vee B$, this can be converted into $(\neg A) \wedge (\neg B)$, then converted into $(A \oplus 1) \wedge (B \oplus 1)$. By utilizing such techniques, all functions can be converted into a series of ($\wedge$) and ($\oplus$) operations. This is the basis of Gentry's work [19]. Figure 4 illustrates the difference between the conventional encryption schemes (not PHE) and fully homomorphic scheme.

Gentry is using lattice-based cryptography. His proposed fully homomorphic encryption consists of several steps: start from what was called a *somewhat homomorphic encryption* scheme using ideal lattices that is limited to evaluating low-degree polynomials over encrypted data. It is limited because each ciphertext is noisy in some sense, and this noise grows as one adds and multiplies ciphertexts, until ultimately the noise makes the resulting ciphertext indecipherable. Next, it *squashes* the decryption procedure so that it can be expressed as a low-degree polynomial which is supported by the scheme. Finally, it applies a *bootstrapping* transformation, through a recursive self-embedding, to obtain a fully homomorphic scheme [21].

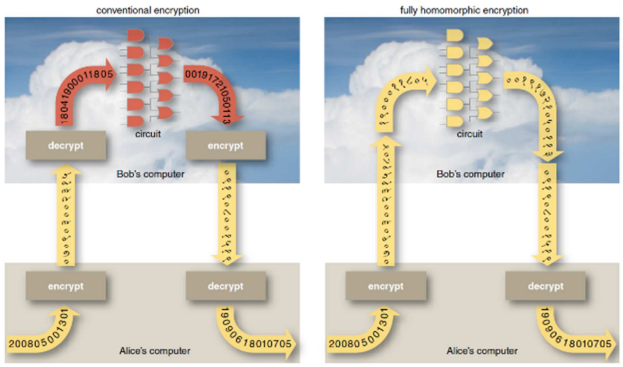

Figure 4. *The difference between the conventional encryption schemes and fully homomorphic scheme [20].*

5. FHE in Cloud Security

The security issues of data stored in cloud can be solved by using Fully Homomorphic Encryption (FHE) schemes. To secure it, the data should be encrypted with FHE before being sent to the cloud. First, the user login and uses the key-generation provided by the server to generate the secret key, the user is the only holder of this secret key. Then, the user encrypts the data that wants to send it to the cloud. During transmitting, the integrity and non-repudiation can be assured by applying other cryptographic technologies such as digital signature. When the user want the server to execute some computations on these encrypted data (such as search), he can send encrypted request to the cloud server. The server performs the required operations and sent the encrypted result to user. Finally the user decrypt the data with his secret key to retrieve the correct result [22]. Figure 5 illustrates the process of using FHE to cloud computing.

In 2010, M. Dijk, C. Gentry et el. [4] presented a second fully homomorphic encryption (called DGHV). In their scheme, somewhat homomorphic encryption uses addition and multiplication over the integers rather than working with ideal lattices over a polynomial ring. This scheme is conceptually simpler than Gentry's scheme based on ideal lattice, but has similar properties of homomorphic operations and efficiency. This scheme is insecure in cloud computing because the cipher retrieval algorithm R needs to transfer the private key *p* to the server [6].

In the march 2010, C. Gentry [5] proposed a homomorphic encryption scheme (called Gen10), heading toward widespread use of cloud computing. This scheme is also

insecure in cloud computing since its cipher retrieval algorithm R asks to submit q (a random number) to the server, yet utilizing c (ciphertext) and q, the plaintext leaks out [6].

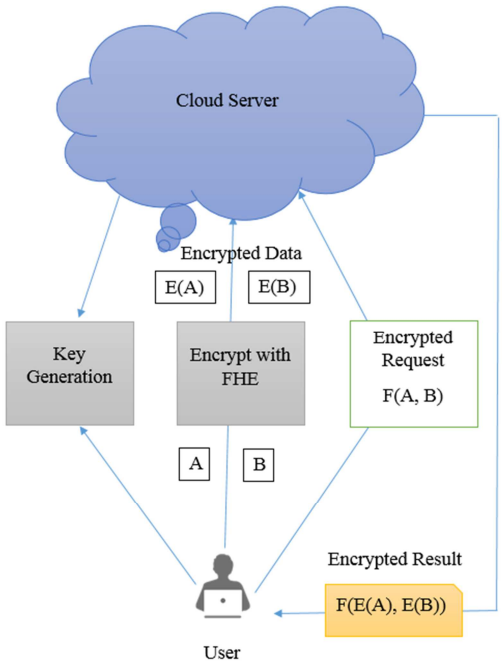

Figure 5. *Applying FHE to secure cloud data.*

In 2012, Jian Li, Danjie Song et al. [6] proposed a simple FHE derived from Gentry cryptosystem to ensure the privacy in cloud storage. They called the scheme is SDC. The description of SDC scheme as follows:

KeyGen(p): The key p is a random odd integer of P-bit.

Encrypt (p, m): To encrypt a bit $m \in \{0, 1\}$

$$c = m + p + r * p * q \tag{3}$$

Where r is a random number of R-bit and q is a constain Q-bit big integer.

Decrypt (p, c): Output (c mod p).

Retrieval(c):

$$R = (c_i - c_{index}) \bmod q. \tag{4}$$

When the user wants to retrieve contents m_{index}, he encrypts the Keywords

$$c_{index} = m_{index} + p + r * p * q \tag{5}$$

And delivers c_{index} to the server.

On receiving c_{index}, server reads the ciphertexts, computing

$$R = (c_i - c_{index}) \bmod q, \tag{6}$$

once R =0, ciphertext retrieval succeeds, and ci is the desired result.

In the SDC scheme, transferring q to the server merely, the server can complete the process of ciphertext retrieval successfully, without plaintext leak out, because the process of decryption uses the private key p while the retrieval process uses the integer q, which is entirely different. Thus satisfies both the demand of ciphertext retrieval and data security [6].

Jian Li, Danjie Song et al. provided a complete proof of correctness of the scheme as follows:

Given two messages m_1 and m_2. The ciphertext of these messages after encryption:

$$c_1 = m_1 + p + r_1 * p * q. \tag{7}$$

$$c_2 = m_2 + p + r_2 * p * q \tag{8}$$

To check additively homomorphic property:

$$c_3 = c_1 + c_2 = (m_1 + m_2) + (r_1 + r_2) * p * q * 2p.$$

$$m_3 = c_3 \bmod p = m_1 + m_2. \tag{9}$$

Then SDC scheme has additively homomorphic property.

To check multiplicatively homomorphic property:

$$c_4 = c_1 * c_2 = m_1 * m_2 + (m_1 + m_2 + p)p + r_1(p + m_2 + r_2)p\ q + r_2(p + m_1)p\ q.$$

$$m_4 = c_4 \bmod p = m_1 * m_2. \tag{10}$$

Then SDC scheme has multiplicatively homomorphic property.

6. Conclusion

The security issues are a big problem for cloud computing development. To preserve the privacy of his data, the user must encrypt data before being sent to the cloud. Cloud computing security based on homomorphic encryption schemes, because these schemes allow to perform computations on encrypted data without the need to the secret key. Partially Homomorphic Encryption (PHE) such as RSA and Paillier schemes are insufficient to secure cloud computing because these schemes allow to perform only one operation (either addition or multiplication) on the encrypted data of client. Fully Homomorphic Encryption is the best solution to secure the client data in cloud computing because its schemes enable to perform arbitrary computations on encrypted data without decrypting. DGHV and Gen10 schemes of FHE are insecure when they be used in cloud computing to secure data of client. SDC is a simple and considered efficient scheme to secure data in cloud computing. This paper analyzed some of the existing homomorphic encryption schemes and discussed the use of the most efficient one, SDC scheme, to secure cloud computing data. Future work will focus on implementation of SDC scheme in cloud computing and analysis the complexity of the scheme.

References

[1] P. Mell, T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, U. S. Department of Commerce, (2011).

[2] K. Lauter, M. Naehrig, V. Vaikuntanathan, "Can Homomorphic Encryption be Practical?", CCSW' 11, Chicago, llinois, USA, pp. 113–124, (2011).

[3] Craig Gentry, "Fully homomorphic encryption using ideal lattice", in Proceedings of STOC'09, (2009).

[4] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, "Fully homomorphic encryption over the Integers", in Proceedings of Advances in Cryptology, EUROCRYPT'10, pages 24–43, 2010.

[5] Craig Gentry, "Computing arbitrary functions of encrypted data", Communications of The ACM, 53(3): 97-105, (2010).

[6] J. Li, D. Song, S. Chen, X. Lu, "A Simple Fully Homomorphic Encryption Scheme Available in Cloud Computing", In Proceeding of IEEE, (2012).

[7] Baohua Chen, Na Zhao, "Fully Homomorphic Encryption Application in Cloud Computing", in Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 11th International Computer Conference, (2014).

[8] Yan Zhang, Li Zhou, Yuanfan Peng, Jing Zhang, "A secure Image Retrieval Method Based on Homomorphic Encryption for Cloud Computing", in proceedings of the 19th International Conference on Digital Signal Processing, (2014).

[9] Ovunc Kocabas, Tolga Soyata, "Utilizing Homomorphic Encryption to Implement Secure and Private Medical Cloud Computing", in 8th International Conference on Cloud Computing, IEEE, (2015).

[10] Shu Qin Ren, Benjamin Hong Meng Tan, Sivaraman Sundaram, Taining Wang, Yibin Ng,Chang Victor, Khin Mi Mi Aung, "Secure searching on cloud storage enhanced by homomorphic indexing", Future Generation Computer Systems, Elsevier, (2016).

[11] Maha Tebaa, Said El Haji, "Secure Cloud Computing through Homomorphic Encryption", International Journal of Advancements in Computing Technology (IJACT), (2013).

[12] Aderemi A. Atayero, Oluwaseyi Feyisetan, "Security Issues in Cloud Computing: The Potentials of Homomorphic Encryption", Jornal of Emerging Trends in Computing and Information Sciences, (2011).

[13] R. Rivest, I. Adleman, M. Dertouzos, "On Data Banks and Privacy Homomorphisms", Foundations of Secure Communication, (1978).

[14] S. Goldwasser, S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information", in Proceedings of 14th Symposium on Theory of Computing, (1982).

[15] Pascal Paillier, "Public-key cryptosystems based on composite degree residuosity classes", (1999).

[16] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts", in Proceedings of Theory of Cryptography, TCC'05, (2005).

[17] Xing Guangli, Chen Xinmeng, Zhu Ping, Ma Jie, "A method of Homomorphic Encryption", Wuhan University Journal of Natural Sciences, Vol. 11, No. 1, pp. 181-184, (2006).

[18] R. Rivest, A. Shamir, and L. Adleman., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", in Communications of the ACM 21.2, pages 120–126, (1978).

[19] Eyad Saleh, "Processing Over Encrypted Data: Between Theory and Practice", Proceedings of the 8th Ph. D. Retreat of the HPI Research School on Service-oriented Systems Engineering, (2015).

[20] Brian Hayes, "Alice and Bob in Cipherspace", American Scientist, Volume 100, (2012)

[21] Jaydip Sen, "Homomorphic encryption: theory & Application", In Tech, Theory and Practice of Cryptography and Network Security Protocols and Technologies, (2013).

[22] S. Hemalatha, R. Manickachezian, "Perfomance of Ring Based Fully Homomorphic Encryption for securing data in Cloud Computing", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 11, (2014).