Challenges of Using Homomorphic Encryption to Secure Cloud Computing

Khalid EL MAKKAOUI*, Abdellah EZZATI
LAVETE laboratory, Mathematics and Computer Science Department, Sciences and Techniques Faculty, Hassan 1st University
Settat, Morocco
kh.elmakkaoui@gmail.com, abdezzati@gmail.com

Abderrahim BENI HSSANE
LAROSERI laboratory, Computer Science Department, Sciences Faculty, Chouaïb Doukkali University,
El Jadida, Morocco
abenihssane@yahoo.fr

Abstract— **With the emergence of cloud computing, the concept of information security has become a major issue. Indeed, the security of such a system is the greatest concern of computer scientists, providers of cloud and organizations who want to adopt and benefit from these services. Cloud computing providers must implement concepts ensuring network security, hardware, data storage and strategies of control and access to services. All these elements help to preserve data security and ensuring the availability of services associated with the Cloud, to better satisfy clients and acquire and build their trust. However, even if the data storage security in cloud servers is assured, reluctance remain when it comes to process the confidential data. Indeed, the fear that sensitive data is being used is a major obstacle in the adoption of cloud services by enterprises. To overcome this obstacle, the use of methods that can perform operations on encrypted data without knowing the secret key, seems to be an effective way to strengthen the confidentiality of information. In this paper we will examine the challenges facing Homomorphic Encryption methods to allow suppliers of cloud to perform operations on encrypted data, and provide the same results after treatment, as if they were performing calculations on raw data.**

Keywords- Cloud Computing; Security; Confidentiality; Trust; Homomorphic Encryption.

I. INTRODUCTION

According to a definition given by the NIST (National of Standards and Technology), "Cloud computing is a model for enabling ubiquitous, 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", and defined five essential characteristics of Cloud Computing that distinguish them from other technologies, namaly: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. This Cloud model is composed of three service models: Software as a Service, Platform as a Service, and Infrastructure as a Service and four deployment models: Private, Public, Community and Hybrid cloud [1].

The adoption of cloud computing services by customers (businesses, consumers, etc.) is limited by concerns about the loss of privacy and the value of their private data [2].

To store data in the cloud data center, we use standard encryption techniques to ensure the security of transmission of these data towards Cloud, and we store them in encrypted format.

But for the providers to process the data on their servers, and execute operations requested by their clients, they need to access data in the clear. It is this operation which could affect the confidentiality of these data and thus slow cloud adoption by organizations.

Therefore, the Cloud providers must use techniques that will preserve the confidentiality and invisibility of the data but also to ensure the client, even in case of attack data centers, that their data is neither stolen nor reused.

The solution is "all encrypt all the time"; it is Homomorphic Encryption. This method is able to perform operations on encrypted data without knowing the secret key [3]. So with Homomorphic Encryption, the data will never be clear neither during transmission nor during processing.

The main objective of this work is to treat the usefulness and challenges encountered during the adoption of the Homomorphic Encryption technique by cloud providers in order to preserve the confidentiality of the storage and processing of confidential data and finally to acquire and strengthen the trust of their clients.

In Section II, we will define Homomorphic Encryption and present its operation and the categories that compose it. We discuss in Section III some challenges that we may face when adopting the Homomorphic Encryption technique. In Section IV, we will present a new Client-Cloud provider architecture. Finally, we will end up in section V by presenting our conclusions and future work.

II. Homomorphic Encryption (HE)

Homomorphic encryption systems are capable of performing operations on encrypted data without knowing the secret key. These operations generate a result, which is itself encrypted. The result of any operation on the encrypted data is the same as in the case of raw data [3], [4].

Mathematically, we say that an encryption system is Homomorphic if: from Enc (x) and Enc (y), it is possible to calculate Enc (f (x, y)), where f can be: $+$, $\times$, $\oplus$ [5].

The principles of the operation of the Homomorphic Encryption are as follows, and as shown in Figure1[3], [4]:

1) **Key generation**: The client generates the public key (**pk**) and the secret key (**sk**).
2) **Encryption**: The client encrypts the data with encryption key. And sends the encrypted data and **pk** to the Cloud server.
3) **Storage**: The encrypted data and **pk** are stored in the cloud database.
4) **Request**: The client sends a request to the server to perform operations on encrypted data.
5) **Evaluation**: The processing server processes the request and performs the operations requested by the client.
6) **Response**: Cloud provider returns to the client the processed result.
7) **Decryption**: The client decrypts the returned result, using **sk**.

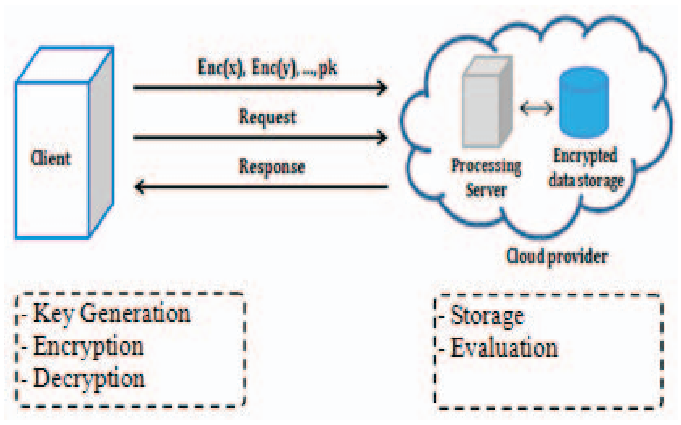


Figure 1. HE functions

Among Homomorphic Encryption systems, we distinguish three categories, depending on the operations performed on the data:

- **Partially Homomorphic Encryption (PHE)** [6]: allows to perform operations on encrypted data, let multiplication or addition, but not both.
- **Somewhat Homomorphic Encryption (SWHE)** [6]**:** allows to perform more than one operation, but a limited number of multiplication and addition operations.
- **Fully Homomorphic Encryption (FHE)** [6]**:** This is a cryptographic system that supports an unlimited number of both additions and multiplications.

Definition1: A Homomorphic Encryption is multiplicative, if there is an algorithm that can calculate Enc($x \times y$) from Enc (x) and Enc (y) without knowing x and y [7].

Definition2: 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 [7].

A. Partially Homomorphic Encryption

Among cryptosystems PHE we distinguish: the RSA[8], ElGamal[9],Pailler[10] and Goldwasser-Micali cryptosystems [11].

1) RSA cryptosystem:

In 1978, Ronald Rivest, Leonard Adleman and Michael Dertouzos suggested for the first time Homomorphic Encryption concept [8]. RSA is a public key cryptosystem, which is a multiplicative Homomorphic Encryption system.

Preparation of Key
Input: p, q $\in \mathbb{P}$, $p \neq q$ (large secure prime numbers)
• Compute **n** = $p \times q$ and **φ(n)** = $(p-1)(q-1)$ • Choose e such that $\gcd(e, \varphi(n))=1$ • Determine **d** such that $e \times d = 1 \bmod \varphi(n)$
Output: (pk, sk) public key: **pk** = (e, n) secret key: **sk** = (d)

Encryption: Enc (m, pk)
Input: message **m** $\in Z^{*}_{n}$
• Compute $c \equiv m^{e} \pmod n$
Output: encrypted message **c** $\in Z_n$

Decryption: Dec (c, sk)
Input: **c** $\in Z_n$
• Compute $m \equiv c^{d} \pmod n$
Output: clear message **m** $\in Z_n$

Figure 2. RSA algorithm

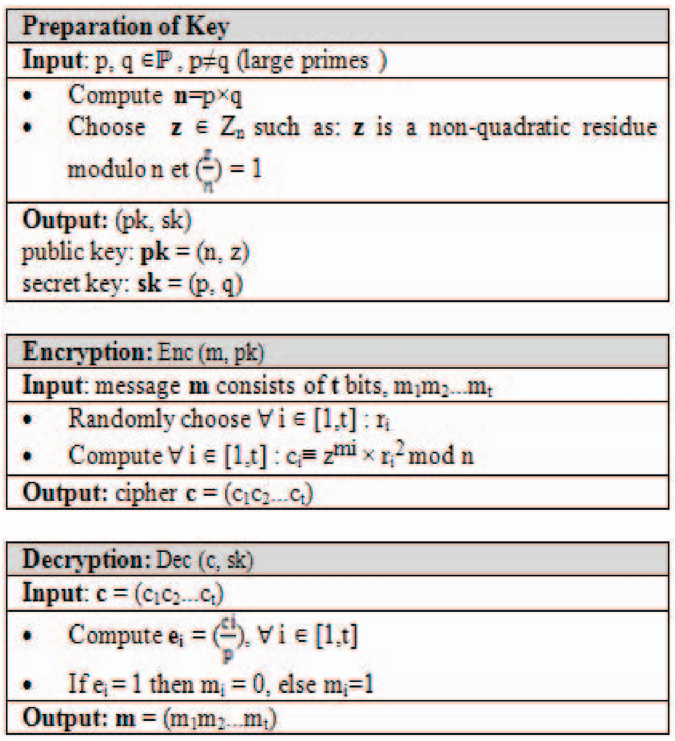
Suppose we have two messages m_1 and m_2 to be encrypted by the RSA algorithm:

$$\mathrm{Enc}(m_1, pk) \times \mathrm{Enc}(m_2, pk) \equiv m_1^{e} \times m_2^{e} \bmod n$$
$$\equiv (m_1 \times m_2)^{e} \bmod n$$
$$\equiv \mathrm{Enc}(m_1 \times m_2,\ pk)$$

So, the RSA cryptosystem, achieves the multiplicative Homomorphic Encryption properties.

2) *Goldwasser-Micali(GM) cryptosystem:*

The encryption system Goldwasser-Micali, is an asymmetric cryptosystem, developed by Shafi Goldwasser and Silvio Micali in 1982 [11, 12].



Preparation of Key
Input: p, q $\in \mathbb{P}$, p≠q (large primes)
• Compute **n**=p×q • Choose **z** $\in Z_n$ such as: **z** is a non-quadratic residue modulo n et $(\frac{z}{n}) = 1$
Output: (pk, sk) public key: **pk** = (n, z) secret key: **sk** = (p, q)

Encryption: Enc (m, pk)
Input: message **m** consists of **t** bits, $m_1m_2...m_t$
• Randomly choose $\forall i \in [1,t] : r_i$ • Compute $\forall i \in [1,t] : c_i \equiv z^{mi} \times r_i^2 \bmod n$
Output: cipher **c** = $(c_1c_2...c_t)$

Decryption: Dec (c, sk)
Input: **c** = $(c_1c_2...c_t)$
• Compute $e_i = (\frac{ci}{p}), \forall i \in [1,t]$ • If $e_i = 1$ then $m_i = 0$, else $m_i=1$
Output: m = $(m_1m_2...m_t)$

Figure 3. GM Algorithm

Suppose we have two bits to encrypt m1 and m2 by the cryptosystem GM.

$$
\begin{aligned}
\text{Enc}(m_1, pk)\times\text{Enc}(m_2, pk) &\equiv (z^{m1}\times r_1^2)(z^{m2}\times r_2^2) \bmod n \\
&\equiv z^{m1+m2}(r_1r_2)^2 \bmod n \\
&\equiv \text{Enc}(m_1 \oplus m_2, pk)
\end{aligned}
$$

So, the GM cryptosystem, achieves the additive Homomorphic Encryption properties.

Applications of PHE cryptosystems are numerous, Table I presents some.

TABLE I. APPLICATION OF PHE

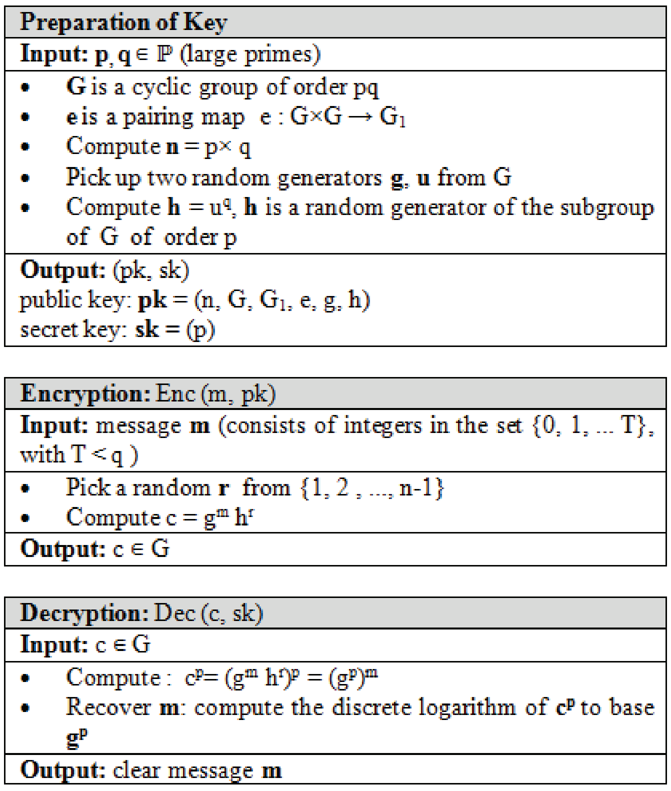
Cryptosystem	HE type	Applications
RSA	Multiplicative	To secure internet, Backing and credit card transaction [4]
ElGamal	Multiplicative	In hybrid systems [4]
Pailler	Additive	E-voting system, threshold scheme[12]
GM	Additive (only a single bit)	Biometric Authentication[11]

B. *Somwhat Homomorphic Encryption*

In this part, we will present only one cryptosystem of SWHE, this is the Boneh-Goh-Nissim (BGN) cryptosystem.

1) *BGN cryptosystem:*

In 2005, Dan Boneh, Eu-Jin Goh and Kobi Nissim [13] invented an encryption system, with which we can perform any number of additions, but with only one multiplication [5].



Preparation of Key
Input: p, q $\in \mathbb{P}$ (large primes)
• **G** is a cyclic group of order pq • **e** is a pairing map $e : G\times G \rightarrow G_1$ • Compute **n** = p× q • Pick up two random generators **g**, **u** from G • Compute **h** = u^q, **h** is a random generator of the subgroup of G of order p
Output: (pk, sk) public key: **pk** = (n, G, G_1, e, g, h) secret key: **sk** = (p)

Encryption: Enc (m, pk)
Input: message **m** (consists of integers in the set {0, 1, ... T}, with T < q)
• Pick a random **r** from {1, 2 , ..., n-1} • Compute $c = g^m h^r$
Output: $c \in G$

Decryption: Dec (c, sk)
Input: $c \in G$
• Compute : $c^p = (g^m h^r)^p = (g^p)^m$ • Recover **m**: compute the discrete logarithm of c^p to base g^p
Output: clear message **m**

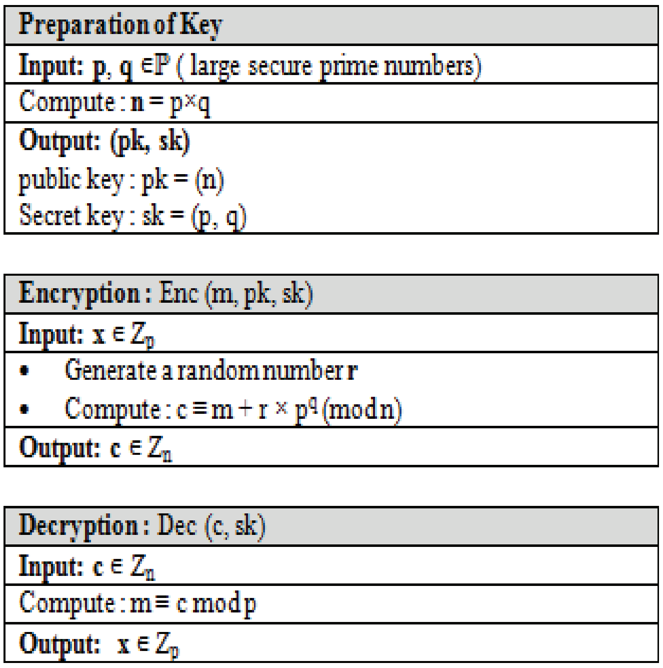
Figure 4. BGN Algorithm

C. *Fully Homomorphic Encryption*

FHE systems are numerous. Among them there are, Algebra Homomorphic Encryption Scheme Based on Updated ElGamal (AHEE) which proposed by Chen Liang and Gao Changmin in 2008 [4],[14], Algebraic Homomorphism Encryption Scheme based on Fermat's Little Theorem (AHEF) which proposed by Xiang Guangli and Cui Zhuxiao in 2012 [15], and Enhanced Homomorphic Encryption Scheme (EHES).

1) *EHES cryptosystem*

In 2013, Gorti VNKV Subba Rao proposed Enhanced Homomorphic Encryption Scheme(EHES) for homomorphic encryption / decryption with the IND-CCA secure system. This system allows to perform operations of addition, multiplication and mixed [16].



Preparation of Key

Input: $p, q \in \mathbb{P}$ (large secure prime numbers)

Compute : $n = p \times q$

Output: (pk, sk)

public key : pk = (n)

Secret key : sk = (p, q)

Encryption : Enc (m, pk, sk)

Input: $x \in Z_p$

- Generate a random number r
- Compute : $c \equiv m + r \times p^q (\mathrm{mod}\, n)$

Output: $c \in Z_n$

Decryption : Dec (c, sk)

Input: $c \in Z_n$

Compute : $m \equiv c \bmod p$

Output: $x \in Z_p$

Figure 5. EHES Algorithm

Let $x, y \in Z_p$, pk = (n), and sk = (p, q)

Multiplicative:

Enc (xy) ≡ Enc (x) Enc (y)(mod n), or

xy = Dec (Enc (x) Enc (y)) ≡ Enc (x) Enc (y)(mod p)

Additive:

Enc (x + y) ≡ Enc (x) + Enc (y) (mod n), or

x + y = Dec (Enc (x) + Enc (y))

≡ Enc (x) + Enc (y) (mod p)

Furthermore, applications of Fully Homomorphic Encryption are numerous; Table 2 presents some.

TABLE II. APPLICATION OF FHE

Cryptosystem	Applications
EHES	Efficient Secure Message Transmission in Mobile Ad Hoc Networks[17]
AHEF	Applied to perform various calculations on encrypted data [17].
AHEE	Secure multi-party computation, electronic voting and mobile cipher [4]

III. CHALLENGES OF USING HOMOMORPHIC ENCRYPTION

The Homomorphic Encryption challenges are numerous. We will mention a few:

A. *Efficiency:*

PHE algorithms (RSA, Elgamal, Paillier ...) are effective and secure enough to use in practical applications. However, these systems have certain limitations; most support only one type of operation, so the use of these systems in practical applications has greater restriction, and most applications require the performance of more than one operation. As a result, these algorithms have to be used in combination with other Homomorphic Encryption algorithms to meet the application requirements [18].

In the case of a cloud service model of SaaS (Software as a Service) or PaaS (Platform as a Service), data can be encrypted during the transfer phase. The Homomorphic Encryption technique is effective to ensure the security of applications and data specific to these types of service model.

Regarding the IaaS (Infrastructure as a service), specifically for virtual machine (VM), it is possible to encrypt the file systems via APIs integrated into the operating systems, or use solutions like **SafeNet ProtectV** for example. But we are still far from being able to run VMs in the cloud with Homomorphic Encryption because it requires that the secret key is transmitted at a given time (usually boot the virtual machine).

B. *Robustness:*

The robustness of Homomorphic Encryption systems is based on the size of the encryption key. In the case of the RSA cryptosystem, for example, the size of the public key (n=p×q) must be strictly greater than 1024 bits [19]. Consequently, the use of large-size of public key makes the system too slow for practical use.

C. *Delay:*

The choice of the large-size of public key helps ensure the robustness of Homomorphic Encryption systems, on one hand; but on the other hand, they affect the size of ciphertext, the encryption and decryption time, and the data processing time.

To analyze the speed of Homomorphic Encryption systems we must take into consideration:

- The size of the encryption key, its impact on the size of the encrypted message and the encryption time.
- The delay of the application of the processing server based on the size of the encrypted message.
- The decrypting time depending on the size of the encrypted text sent by the cloud provider and the size of the private key.

1) *Analysis and comparison*

In this part, we will perform a comparative study between the normal case (without the use of Homomorphic Encryption, as shown in Figure 6) and the case when one adopts the Homomorphic Encryption technique (as shown in Figure 7).

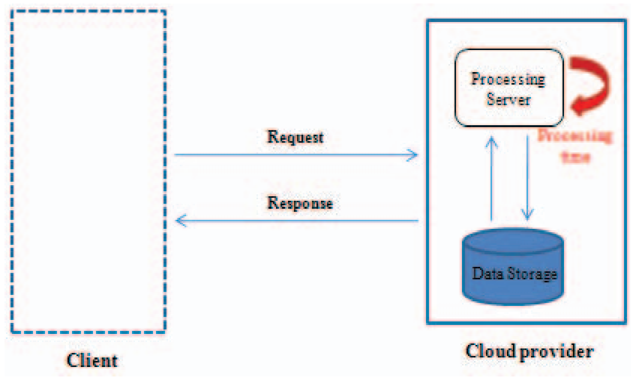


Figure 6. Client-Cloud provider system without using the HE

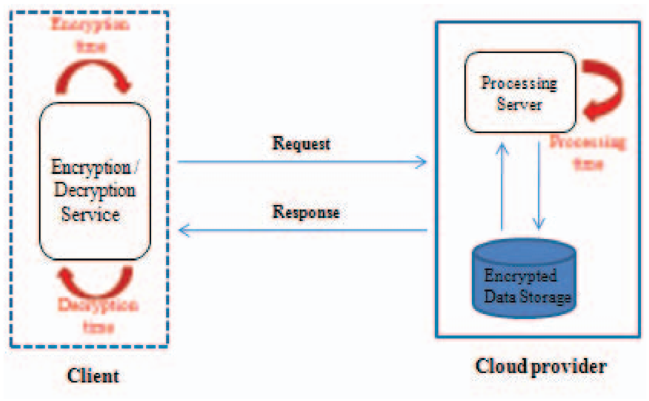


Figure 7. Client-Cloud provider system using the HE

In the first case (normal case), the response time is:

$$T = 2T_{transmission} + T_{treatment} \quad (1)$$

And in the second case (with the use of HE), the response time is:

$$T_{HE} = T_{Enc} + 2T_{transmissionHE} + T_{treatmentHE} + T_{Dec} \quad (2)$$

From where:

T: Response time in the normal case.
$T_{transmission}$: the time of transmission of the raw data.
$T_{transmissionHE}$: the time of transmission of the encrypted data.
$T_{treatment}$: The data processing time in the normal case.
T_{HE} : Response time when using the HE.
$T_{treatmentHE}$: The data processing time when using the HE.
T_{Enc} : Encryption time
T_{Dec} : Decryption time

a) HE implementation example

In this implementation, we analyze the Homomorphic Encryption performance. We choose three text files of different size, which are given as inputs to the algorithms to verify the performance of the RSA and EHES cryptosystems. The experiment was performed by the machine [Intel(R) Core(TM) i3-2370M CPU @ 2,70 GHz 2,70 GHz, 8 Go of RAM].

The tables below show three different clear data size and the encryption and decryption time, and the corresponding size of the encrypted data taken by the RSA and EHES algorithm s.

TABLE III. CIPHERTEXT SIZE, ENCRYPTION AND DECRYPTION TIME FOR RSA

	RSA Cryptosystem		
Plaintext size in Bits	Encryption time in seconds	Decryption time in seconds	Ciphertext size in Bits
8	3.43	2.56	26
16	8.57	4.42	28
24	12.05	5.96	30

TABLE IV. CIPHERTEXT SIZE, ENCRYPTION AND DECRYPTION TIME FOR EHES

	EHES Cryptosystem		
Plaintext size in Bits	Encryption time in seconds	Decryption time in seconds	Ciphertext size in Bits
8	1.62	0.92	20
16	4.07	2.03	24
24	5.67	3.15	28

From Table III and IV, we plot the graph representing the three different sizes of data before and after encryption by the RSA and EHES cryptosystem (Figure 8), the graph which represents the time taken by RSA and EHES to encrypt the different sizes of data (Figure 9) and also the graph which represents the decryption time of different encrypted data (Figure 10).

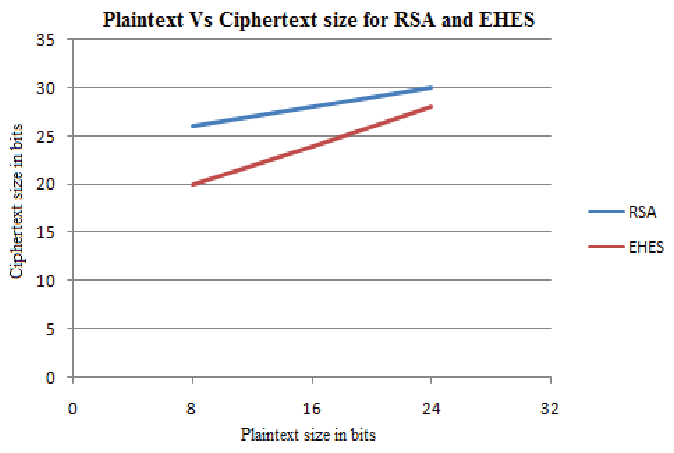


Figure 8. Plaintext versus Ciphertext size for RSA and EHES

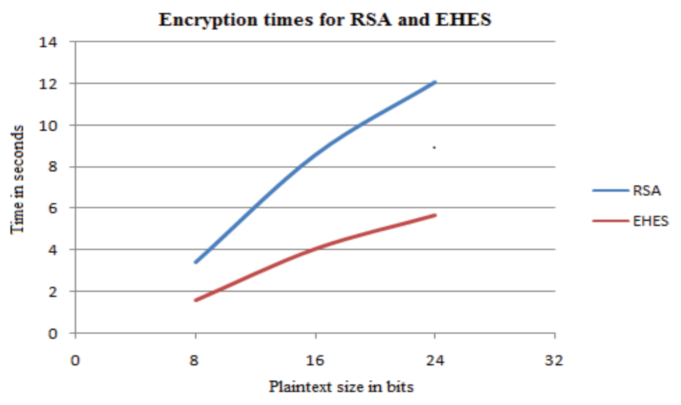


Figure 9. Encryption time for RSA

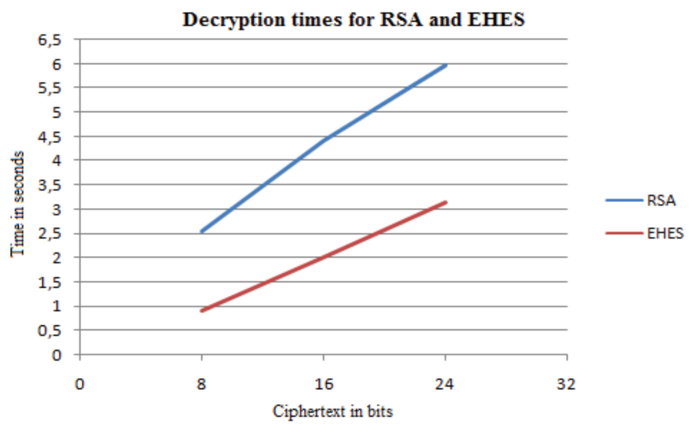


Figure 10. Decryption time for RSA and EHES

b) Interpretation of results

The transmission time is the amount of time from the start to the end of the transmission of a message. This is the size of the message bits divided by the data rate (bps) of the channel on which the transmission occurs.

Transmission time = message size / data rate.

Since the size of data increases during their encryption (as shown in Figure 8), so the time of transmission and processing also increases. And the time of encryption and decryption is increased depending on the size of data (as shown in Figure 9 and 10). As a result:

$$T_{transmissionHE} \geq T_{transmission} \text{ et } T_{treatmentHE} \geq T_{treatment}.$$

Therefore, from (1) and (2) it can be assumed that:

$$T_{HE} >> T$$

Finally, although the Homomorphic Encryption technique ensures high confidentiality of storage and data processing, located in the cloud servers, the performance is still a problem because the encryption and decryption of data make the system work too slow for practical use.

IV. New Client-Cloud provider architecture

To improve the performance of Homomorphic Encryption systems, we suggest using intelligent multi-agent systems at Cloud processing servers.

The multi-agent system (MAS) consists of a group of agents, which are managed by the manager agent. This system has the ability to improve performance and decision making when interacting with the external environment.

The use of the MAS in Cloud servers aims at improving the processing time of encrypted data to ensure the performance and quality of service. In other words, each agent will take charge of carrying out the treatment on a set of specific data and return the result to the manager agent, the latter will have to complete the requested operations.

The figure below shows an example of HE implementation with the use of multi-agent systems.

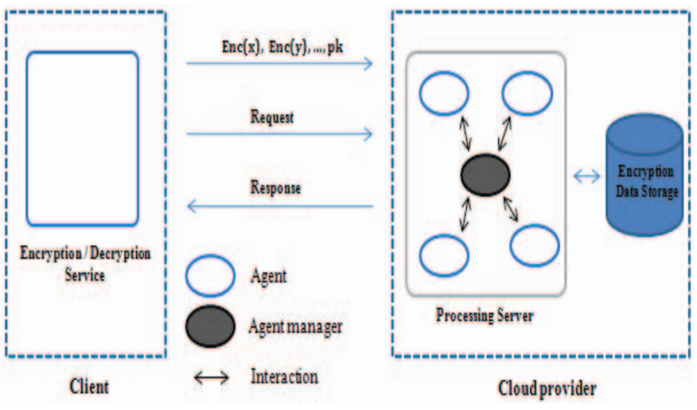


Figure 11. Implemetation of HE with using Multi Agent System

Agent: is often an instance (virtual server).
Agent Manager: is a principal agent, based on a knowledge base to manage the creation, registration and removal of each agent. Also, it includes the ability to control the activity of these agents.
Encryption / Decryption service: is provided by the cloud provider.

V. Conclusion and future work

The security of cloud computing, which is based on the Homomorphic Encryption technique, is a new concept of security that allows cloud service providers to preserve the confidentiality of the private data of their clients.

In this paper, we have presented the different Homomorphic Encryption systems (Partially, Somewhat and Fully Homomorphic Encryption), the few challenges resulting of using this concept and we have also proposed a new Client-Cloud provider architecture, that can improve the performance of this technique.

In our future work, we will focus on the analysis and improvement of the complexity of existing Homomorphic Encryption algorithms to enable cloud servers to perform various requested operations by the clients and guarantee the quality of service.

REFERENCES

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

[2] K. Lauter, M. Naehrig, V. Vaikuntanathan, “Can Homomorphic Encryption be Pratical?, ” CCSW' 11, October 21, 2011, Chicago, llinois, USA, pp. 113 –124.

[3] M. TEBAA and S. EL HAJII, (2013) “Secure Cloud Computing through Homomorphic Encryption, ” International Journal of Advancements in Computing Technology (IJACT), Vol.5, No.16, pp. 29 –38.

[4] Payal V. Parmar, et al., “Survey of Various Homomorphic Encryption algorithms and Schemes,” In International Journal of Computer Applications (0975 - 8887), Volume 91 -No. 8, April 2014, pp. 26 – 32.

[5] X. Yi et al., “Homomorphic Encryption and Applications,” SpringerBriefs in Computer Science, chapter 2, 2014, pp. 27 – 46.

[6] M. Ogburn, C. Turner, P. Dahal, “Homomorphic Encryption,” In Complex Adaptive Systems, Publication 3, Cihan H. Dagli, Editor in Chief Conference Organized by Missouri University of Science and Technology 2013 - Baltimore, MD, Elsevier, 2013, pp. 502 – 509.

[7] 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.

[8] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public key cryptosystems,” Communications of the ACM, 21(2):120-126, 1978. Computer Science, Springer, 1999, pp. 223 – 238.

[9] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” IEEE Transactions on Information Theory, 1985, pp. 469 – 472.

[10] Pascal Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” In 18th Annual Eurocrypt Conference (EUROCRYPT'99), Prague, Czech Republic , volume 1592, 1999.

[11] J. Bringer et al., “An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication,” Springer-Verlag, 2007.

[12] 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, pp. 365 – 377.

[13] Boneh, E. Goh, K. Nissim, “Evaluating 2-DNF formulas on ciphertexts,” In Theory of Cryptography Conference, TCC’05, pp. 325–341,Springer, 2005.

[14] Chen, Liang and Chengmin Gao, “Public Key Homomorphism Based on Modified ElGamal in Real Domain,” In International Conference on Computer Science and Software Engineering, Vol. 3, pp.802-805. 2008.

[15] Xiang Guangli, Cui Zhuxiao, “The Algebra Homomorphic Encryption Scheme Based on Fermat's Little Theorem,” Communication Systems and Network Technologies (CSNT), 2012 International Conference on , pp.978-981, 11-13 May 2012.

[16] G. VNKV Subba Rao, Md.Sameeruddhin Khan, A.Yashwanth Reddy , K.Narayana, “Data Security in Bioinformatics,” In International Journal of Advanced Research in Computer Science and Software Engineering 3(11), November - 2013, pp. 590 – 598.

[17] Gorti VNKV Subba Rao and Garimella Uma, “An Efficient Secure Message Transmission in Mobile Ad Hoc Networks using Enhanced Homomorphic Encryption Scheme,” Global Journal of Computer Science and Technology Network, Web & Security, Volume 13, Version 1.0, 2013, pp. 20 –33.

[18] K.Mallaiah, S. Ramachandram, “Applicability of Homomorphic Encryption and CryptDB in Social and Business Applications: Securing Data Stored on the Third Party Servers while Processing through Applications,” In International Journal of Computer Applications (0975 – 8887) Volume 100– No.1, August 2014.

[19] P. Mahajan, A. Sachdeva, “A Study of Encryption Algorithms AES, DES and RSA for Security,” Global Journal of Computer Science and Technology Network, Web & Security Volume 13 Issue 15 Version 1.0 Year 2013.