Homomorphic encryption method applied to Cloud Computing

Conference Paper · April 2012

DOI: 10.1109/JNS2.2012.6249248

3 authors, including:

Maha Tebaa
Mohammed V University of Rabat
6 PUBLICATIONS **231** CITATIONS
SEE PROFILE

Abdellatif el Ghazi
Université Internationale de Rabat
10 PUBLICATIONS **131** CITATIONS
SEE PROFILE

Homomorphic Encryption Applied to the Cloud Computing Security

Maha TEBAA, Saïd EL HAJJI, Abdellatif EL GHAZI

***Abstract*—Cloud computing security challenges and it's also an issue to many researchers; first priority was to focus on security which is the biggest concern of organizations that are considering a move to the cloud. The advantages of cloud computing include reduced costs, easy maintenance and re-provisioning of resources, and thereby increased profits. But the adoption and the passage to the Cloud Computing applies only if the security is ensured. How to guaranty a better data security and also how can we keep the client private information confidential? There are two major questions that present a challenge to Cloud Computing providers.**

When the data transferred to the Cloud we use standard encryption methods to secure the operations and the storage of the data. But to process data located on a remote server, the Cloud providers need to access the raw data. In this paper we are proposing an application of a method to execute operations on encrypted data without decrypting them which will provide us with the same results after calculations as if we have worked directly on the raw data.

Index Terms—Cloud Computing, Homomorphic Encryption, Security, confidentiality.

I. Introduction

HERE we asked two main questions: How to be sure that even if the data-centers of the Cloud Computing provider were attacked, my data won't be stolen or reused? And how can my data remain confidential and invisible even to my Cloud provider?

Our basic concept was to encrypt the data before sending them to the Cloud provider. But, this one will have to decrypt them each time he has to work on them. The client will need to provide the private key to the server to decrypt the data before execute the calculations required, which might affect the confidentiality of data stored in the Cloud.
The Homomorphic Encryption method is able to perform operations of encrypted data without decrypting them.

In this work we focus on the application of Homomorphic Encryption method on the Cloud Computing security, particularly the possibility to execute the calculations of confidential data encrypted without decrypting them.

Maha TEBAA, Laboratory of Mathematics, Computer Science and Applications (MIA), University of Mohammed V – Agdal, Faculty of Sciences, Rabat, Morocco,(e-mail: maha.tebaa@gmail.com).

Said EL HAJJI is with Department of Mathematical and Computer Science , University of Mohammed V – Agdal, Faculty of Sciences Rabat, Laboratory of Mathematics, Computer Science and Applications (MIA), Rabat, Morocco, (e-mail: elhajji@fsr.ac.ma).

Abdellatif EL GHAZI is with Laboratory of Research – Institute of Vinci Rabat, Morocco, (e-mail: aelghazi@vinsicpace.net).

In Section II, we are introducing the concept of Cloud Computing and the necessity to adopt Homomorphic Encryption to secure the calculation of data hosted by the Cloud provider. In section III, we'll define Homomorphic Encryption and we'll illustrate some examples of existing Homomorphic cryptosystems. In section IV, we'll present our scheme and our implementation. The conclusion and perspectives will be mentioned in section V.

II. Cloud computing

Definition [1]: By cloud computing we mean: The Information Technology (IT) model for computing, which is composed of all the IT components (hardware, software, networking, and services) that are necessary to enable development and delivery of cloud services via the Internet or a private network.

This definition doesn't mention any security notion of the data stored in the Cloud Computing even being a recent definition. Therefore we understand that the Cloud Computing is lacking security, confidentiality and visibility. To Provide Infrastructure (IaaS), Platform Service (PaaS) or Software (SaaS) as a Service is not sufficient if the Cloud provider does not guaranty a better security and confidentiality of customer's data.

By convention, we consider as Cloud Computing any treatment or storage of personal or professional information which are realized outside the concerned structure (i.e outside the company), to secure the Cloud means secure the treatments (calculations) and storage (databases hosted by the Cloud provider).
Cloud providers such as IBM, Google and Amazon use the virtualization on their Cloud platform and on the same server can coexist a virtualized storage and treatment space that belong to concurrent enterprises.

The aspect of security and confidentiality must intervene to protect the data from each of the enterprises.
Secure storage and treatment of data requires using a modern aspect of cryptography that has the criteria for treatment such as, the necessary time to respond to any request sent from the client and the size of an encrypted data which will be stored on the Cloud server.

Our proposal is to encrypt data before sending it to the cloud provider, but to execute the calculations the data should be decrypted every time we need to work on it. Until now it was impossible to encrypt data and to trust a third party to keep them safe and able to perform distant calculations on

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

them. So to allow the Cloud provider to perform the operations on encrypted data without decrypting them requires using the cryptosystems based on Homomorphic Encryption.

III. Homomorphic Encryption

Homomorphic Encryption systems are used to perform operations on encrypted data without knowing the private key (without decryption), the client is the only holder of the secret key.
When we decrypt the result of any operation, it is the same as if we had carried out the calculation on the raw data.

Definition: An encryption is homomorphic, if: from Enc(a) and Enc(b) it is possible to compute Enc(f (a, b)), where f can be: +, ×, ⊕ and without using the private key.
Among the Homomorphic encryption we distinguish, according to the operations that allows to assess on raw data, the additive Homomorphic encryption (only additions of the raw data) is the Pailler [2] and Goldwasser-Micalli [3] cryptosystems, and the multiplicative Homomorphic encryption (only products on raw data) is the RSA [4] and El Gamal [5] cryptosystems.

A. History of the Homomorphic encryption

In 1978 Ronald Rivest, Leonard Adleman and Michael Dertouzos suggested for the first time the concept of Homomorphic encryption [8]. Since then, little progress has been made for 30 years. The encryption system of Shafi Goldwasser and Silvio Micali was proposed in 1982 was a provable security encryption scheme which reached a remarkable level of safety, it was an additive Homomorphic encryption, but it can encrypt only a single bit. In the same concept in 1999 Pascal Paillier was also proposed a provable security encryption system that was also an additive Homomorphic encryption. Few years later, in 2005, Dan Boneh, Eu-Jin Goh and Kobi Nissim [9] invented a system of provable security encryption, with which we can perform an unlimited number of additions but only one multiplication.

B. Additive Homomorphic Encryption

A Homomorphic encryption is additive, if:

$$\begin{array}{l} \text{Enc}(x \oplus y) = \text{Enc}(x) \otimes \text{Enc}(y) \\ \quad\; \| \qquad\quad \| \\ \text{Enc}\left(\sum_{i=1} m_i\right) = \prod_{i=1} \text{Enc}(m_i) \end{array}$$

Example: Paillier Cryptosystem (1999):

Key Generation: $\text{KeyGen}(p, q)$

Input: $p, q \in \mathbb{P}$

Compute $n = pq$
Choose $g \in \mathbb{Z}^*_{n^2}$ such that
$\gcd(L(g^\lambda \bmod n^2), n) = 1$ with $L(u) = \frac{u-1}{n}$

Output: (pk, sk)
public key: $pk = (n, g)$
secret key: $sk = (p, q)$

Encryption: $\text{Enc}(m, pk)$

Input: $m \in \mathbb{Z}_n$

Choose $r \in \mathbb{Z}^*_n$
Compute $c = g^m \cdot r^n \bmod n^2$

Output: $c \in \mathbb{Z}_{n^2}$

Decryption: $\text{Dec}(c, sk)$

Input: $c \in \mathbb{Z}_{n^2}$

Compute $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$

Output: $m \in \mathbb{Z}_n$

Fig. 1. Paillier Algorithm

Suppose we have two ciphers C_1 et C_2 such that:

$$\begin{array}{l} C_1 = g^{m1} \cdot r_1^{\,n} \bmod n^2 \\ C_2 = g^{m2} \cdot r_2^{\,n} \bmod n^2 \\ C_1.C_2 = g^{m1} \cdot r_1^{\,n} \cdot g^{m2} \cdot r_2^{\,n} \bmod n^2 = g^{m1+m2} (r_1 r_2)^n \bmod n^2 \end{array}$$

So, Pailler cryptosystem realizes the property of additive Homomorphic encryption.

An application of an additive Homomorphic encryption is electronic voting: Each vote is encrypted but only the "sum" is decrypted.

C. Multiplicative Homomorphic Encryption

A Homomorphic encryption is multiplicative, if:

$$\begin{array}{l} \text{Enc}(x \otimes y) = \text{Enc}(x) \otimes \text{Enc}(y) \\ \quad\; \| \qquad\quad \| \\ \text{Enc}\left(\prod_{i=1} m_i\right) = \prod_{i=1} \text{Enc}(m_i) \end{array}$$

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Example : RSA Cryptosystem (1978)

Key Generation: KeyGen(p,q)	
Input: $p,q \in \mathbb{P}$	
Compute	$n = p \cdot q$
	$\varphi(n) = (p-1)(q-1)$
Choose e such that	$\gcd(e, \varphi(n)) = 1$
Determine d such that	$e \cdot d \equiv 1 \bmod \varphi(n)$
Output: (pk, sk)	
public key: $pk = (e, n)$	
secret key: $sk = (d)$	

Encryption: Enc(m, pk)	
Input: $m \in \mathbb{Z}_n$	
Compute	$c = m^e \bmod n$
Output: $c \in \mathbb{Z}_n$	

Decryption: Dec(c, sk)	
Input: $c \in \mathbb{Z}_n$	
Compute	$m = c^d \bmod n$
Output: $m \in \mathbb{Z}_n$	

Fig. 2. RSA Algorithm

Suppose we have two ciphers C_1 et C_2 such that:

$C_1 = m_1^e \bmod n$
$C_2 = m_2^e \bmod n$
$C_1.C_2 = m_1^e m_2^e \bmod n = (m_1 m_2)^e \bmod n$

RSA cryptosystem realize the properties of the multiplicative Homomorphic encryption, but it still has a lake of security, because if we assume that two ciphers C_1, C_2 corresponding respectively to the messages m_1, m_2, so:

$C_1 = m_1^e \bmod n$
$C_2 = m_2^e \bmod n$

The client sends the pair (C_1, C_2) to the Cloud server, the server will perform the calculations requested by the client and sends the encrypted result $(C_1 \times C_2)$ to the client.

If the attacker intercepts two ciphers C_1 et C_2, which are encrypted with the same private key, he/she will be able to decrypt all messages exchanged between the server and the client. Because the Homomorphic encryption is multiplicative, i.e. the product of the ciphers equals the cipher of the product.

Example: The application of RSA multiplicative Homomorphic encryption on two messages m_1 and m_2.

Let, for p = 3, q = 5, e = 9 and d = 1 with block size = 1
Two messages m_1 and m_2 and their ciphers C_1 and C_2 respectively, obtained using the RSA encryption.

m_1 = 589625 ➔ C_1 = 00 05 00 08 00 09 00 06 00 02 00 05
m_2 = 236491 ➔ C_2 = 00 02 00 03 00 06 00 04 00 09 00 01

We convert the ciphers into binary system

The Blocks of C_1 in binary system	The Blocks of C_2 in binary system
00 05 => 00 0101	00 02 => 00 0010
00 05 => 00 0101	00 02 => 00 0010
00 08 => 00 1000	00 03 => 00 0011
00 09 => 00 1001	00 06 => 00 0110
00 06 => 00 0110	00 04 => 00 0100
00 02 => 00 0010	00 09 => 00 1001
00 05 => 00 0101	00 01=> 00 0001

The binary multiplication of the ciphers block by block is as follow:

00 0101×00 0010 = 00 1010	00 10
00 1000×00 0011 = 00 11000	00 24
00 1001×00 0110 = 00 110110	00 54
00 0110×00 0100 = 00 11000	00 24
00 0010×00 1001 = 00 10010	00 18
00 0101×00 0001 = 00 0101	00 05

If we decrypt the cipher $C_1 \times C_2$ with the private key, we get:

C_1C_2 = 00 10 00 02 00 04 00 05 00 04 00 02 00 04 00 01 00 08 00 05

So:
m_1m_2 = 10 2 4 5 4 2 4 1 8 5

This is exactly the same raw message obtained by multiplying $m_1 \times m_2$

m_1 = 5 8 9 6 2 5
m_2 = 2 3 6 4 9 1

$m_1 m_2$ = 10 24 54 24 18 5 (we are multiplying $m_1 \times m_2$ block by block).

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. Scheme and Implementation

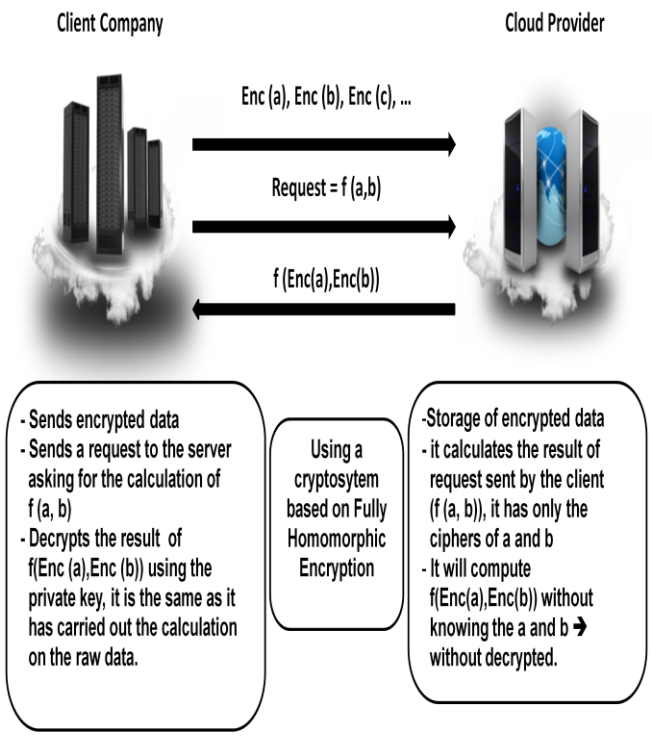

Fig. 3. Homomorphic Encryption applied to the Cloud Computing

For all types of calculation on the data stored in the cloud, we must opt for the fully Homomorphic encryption which is able to execute all types of operations on encrypted data without decryption.

In 2009 Craig Gentry of IBM has proposed the first encryption system "fully homomorphic" that evaluates an arbitrary number of additions and multiplications and thus calculate any type of function on encrypted data [6].

The application of fully Homomorphic encryption is an important stone in Cloud Computing security, more generally, we could outsource the calculations on confidential data to the Cloud server, keeping the secret key that can decrypt the result of calculation.

In our implementation, we analyze the performance of the existing Homomorphic encryption cryptosystems, we are working on a virtual platform with ESX as a Cloud server, a VPN network that links the Cloud to the client (enterprise), then later we started by simulating different scenarios using the Computer Algebra System Magma tools [7], focusing on:

- *The size of the public key and its impact on the size of the encrypted message.*
- *The server delay of the request treatment according to the size of the encrypted message.*
- *The result decrypting time of the request according to the cipher text size sent by the server.*

V. Conclusion and Perspectives

The cloud computing security based on fully Homomorphic encryption, is a new concept of security which enables providing results of calculations on encrypted data without knowing the raw data on which the calculation was carried out, with respect of the data confidentiality.

Our work is based on the application of fully Homomorphic encryption to the Cloud Computing security considering:

The analyze and the improvement of the existing cryptosystems to allow servers to perform various operations requested by the client.

The improvement of the complexity of the Homomorphic encryption algorithms and compare the response time of the requests to the length of the public key.

References

[1] Vic (J.R.) Winkler, "Securing the Cloud, Cloud Computer Security, Techniques and Tactics", Elsevier, 2011.

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

[3] Julien Bringe and al. An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication, Springer-Verlag, 2007.

[4] 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, pages 223-238. Springer, 1999.

[5] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 469-472, 1985.

[6] Craig Gentry, A Fully Homomorphic Encryption Scheme, 2009.

[7] WiebBosma, John Cannon, and Catherine Playoust. The Magma algebra system I: The user language. J. Symbolic Comput., 24(3-4): 235-265, 1997. Computational algebra and number theory, London, 1993.

[8] Ronald L. Rivest, Leonard Adleman, and Michael L. Dertouzos. On Data Banks and Privacy Homomorphisms, chapter On Data Banks and Privacy Homomorphisms, pages 169-180. Academic Press, 1978.

[9] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Theory of Cryptography Conference, TCC'2005, volume 3378 of Lecture Notes in Computer Science, pages 325-341. Springer, 2005.

ISBN: 978-988-19251-3-8
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)