OPEN ACCESS
ISSN: 2249-6645

International
Journal
Of Modern Engineering Research (IJMER)

A survey on Homomorphic Encryption scheme applied to the cloud computing security

Mr.D.Uma Vishweshwar[1], Mr.S.Rajeshwar[2], Ms.P.Anusha[3]
[1]Asst.Prof, Dept. of CSE, CMR Engineering College, Hyderabad, India. E-Mail: uma.vishwam48@gmail.com
[2]Assoc.Prof, Dept. of CSE, Arjun College of Technology & Sciences, Hyderabad,India.E-Mail:myraj07@gmail.com
[3]Asst.Prof, Dept. of CSE, Sridevi Women's Engineering College, Hyderabad. E-Mail: putlaanusha@gmail.com

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.

KEYWORDS: Cloud computing, Encryption ,Fully homomorphic Encryption, Partially homomorphic cryptosystems.

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.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 Partially Homomorphic Encryption(PHE) and we'll illustrate some examples of existing Homomorphic cryptosystems. In section IV, we'll define Fully Homomorphic Encryption(FHE)

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 onthem. 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. Partially Homomorphic Encryption(PHE)

Partially 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:

Enc (x⊕y) = Enc(x) ⊗ Enc(y)

$$\text{Enc}\left(\sum_{i=1}^{l} mi\right) = \prod_{i=1}^{l} \text{Enc}\,(mi)$$

Key Generation: $\mathsf{KeyGen}(p,q)$	
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: $\mathsf{Enc}(m, pk)$	
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: $\mathsf{Dec}(c, sk)$	
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: $m \in \mathbb{Z}_n$	

Paillier Algorithm

Suppose we have two ciphers C_1 et C_2 such that:

$C_1 = g^{m1}. r_1^n \bmod n^2$
$C_2 = g^{m2}. r_2^n \bmod n^2$
$C_1.C_2 = g^{m1}. r_1^n . g^{m2}. r_2^n \bmod n^2 = g^{m1+m2} (r_1r_2)^n \bmod n^2$

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:

$Enc(x \otimes y) = Enc(x) \otimes Enc(y)$

$Enc(\prod_{i=1}^{l} m_i) = \prod_{i=1}^{l} Enc(m_i)$

The RSA algorithm involves four steps: key generation, key distribution, encryption and decryption.
RSA involves a *public key* and a *private key*. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.
The basic principle behind RSA is the observation that it is practical to find three very large positive integers *e*,*d* and *n* such that with modular exponentiation for all *m*:

$$(m^e)^d \mod n = m$$

and that even knowing *e* and *n* or even *m* it can be extremely difficult to find *d*.
Additionally, for some operations it is convenient that the order of the two exponentiations can be changed and that this relation also implies:

$$(m^d)^e \mod n = m$$

Key distribution

To enable Bob to send her encrypted messages, Alice transmits her public key (*n*, *e*) to Bob via a reliable, but not necessarily secret route, and keeps the private key *d* secret and this is never revealed to anyone. Once distributed the keys can be reused over and over.

Encryption

Bob then wishes to send message *M* to Alice.
He first turns *M* into an integer *m*, such that $0 \leq m < n$ and $\gcd(m, n) = 1$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext *c* corresponding to

$$c \equiv m^e \mod n$$

This can be done efficiently, even for 500-bit numbers, using modular exponentiation. Bob then transmits *c* to Alice.

Decryption

Alice can recover *m* from *c* by using her private key exponent *d* by computing

$$c^d \equiv (m^e)^d \equiv m \mod n$$

Given *m*, she can recover the original message *M* by reversing the padding scheme.

Suppose we have two ciphers C_1 et C_2 such that:

$C_1 = m_1^e \bmod n$
$C_2 = m_2^e \bmod n$
$C_1.C_2 = m_1^e m_2^e \bmod n = (m_1m_2)^e \bmod n$

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 $\rightarrow$ C_1 = 00 05 00 08 00 09 00 06 00 02 00 05 m_2 = 236491 $\rightarrow$ C_2 = 00 02 00 03 00 06 00 04 00 09 00 01

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 C1 × C2 with the private key, we get:

C1C2 = 00 10 00 02 00 04 00 05 00 04 00 02 00 04 00 01 00 08 00 05
So:
m1m2 = 10 2 4 5 4 2 4 1 8 5

This is exactly the same raw message obtained by multiplying m1× m2

m1 = 5 8 9 6 2 5
m2 = 2 3 6 4 9 1

m1 m2 = 10 24 54 24 18 5 (we are multiplying m1× m2 block by block).

In the following examples on Partially Homomorphic Encryption, the notation $\mathcal{E}(x)$ is used to denote the encryption of the message *x*.

Unpadded RSA
If the RSA public key is modulus m and exponent e, then the encryption of a message x is given by $\mathcal{E}(x) = x^e \bmod m$. The homomorphic property is then
$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = x_1^e x_2^e \bmod m = (x_1 x_2)^e \bmod m = \mathcal{E}(x_1 \cdot x_2)$$

ElGamal
In the ElGamal cryptosystem, in a group G, if the public key is (G, q, g, h), where $h = g^x$, and x is the secret key, then the encryption of a message m is $\mathcal{E}(m) = (g^r, m \cdot h^r)$, for some random $r \in \{0, \ldots, q-1\}$. The homomorphic property is then
$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{r1}, x_1 \cdot h^{r1})(g^{r2}, x_2 \cdot h^{r2}) = (g^{r1+r2}, (x_1 \cdot x_2) h^{r1+r2}) = \mathcal{E}(x_1 \cdot x_2)$$

Goldwasser-Micali
In the Goldwasser-Micali cryptosystem, if the public key is the modulus *m* and quadratic non-residue *x*, then the encryption of a bit *b* is $\mathcal{E}(b) = x^b r^2 \bmod m$, for some random $r \in \{0, \ldots, m-1\}$. The homomorphic property is then
$$\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) = x^{b1} r_1^2 x^{b2} r_2^2 = x^{b1+b2} (r_1 r_2)^2 = \mathcal{E}(b_1 \oplus b_2)$$
where $\oplus$ denotes addition modulo 2, (i.e. exclusive-or).

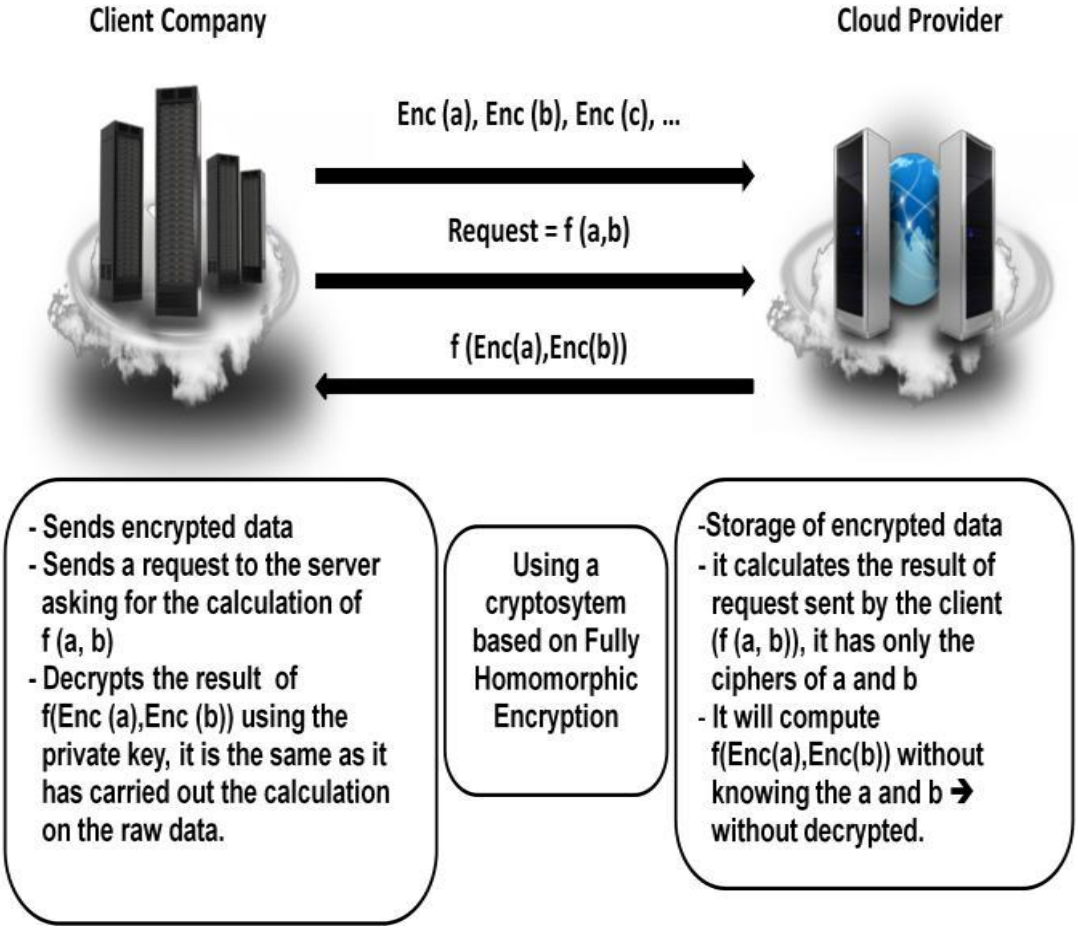
Benaloh
In the Benaloh cryptosystem, if the public key is the modulus *m* and the base *g* with a blocksize of *c*, then the encryption of a message *x* is $\mathcal{E}(x) = g^x r^c \bmod m$, for some random $r \in \{0, \ldots, m-1\}$. The homomorphic property is then
$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{x1} r_1^c)(g^{x2} r_2^c) = g^{x1+x2} (r_1 r_2)^c = \mathcal{E}(x_1 + x_2 \bmod c)$$

Paillier
In the Paillier cryptosystem, if the public key is the modulus *m* and the base *g*, then the encryption of a message *x* is $\mathcal{E}(x) = g^x r^m \bmod m^2$, for some random $r \in \{0, \ldots, m-1\}$. The homomorphic property is then
$$\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) = (g^{x1} r_1^m)(g^{x2} r_2^m) = g^{x1+x2} (r_1 r_2)^m = \mathcal{E}(x_1 + x_2 \bmod m)$$

IV. Fully Homorphic Encryption(FHE)

Each of the examples listed above allows homomorphic computation of only one operation (either addition or multiplication) on plaintexts. A cryptosystem which supports both addition and multiplication (thereby preserving the ring structure of the plaintexts) is known as fully homomorphic encryption (FHE) and is far more powerful. Using such a scheme, any circuit can be homomorphically evaluated, effectively allowing

the construction of programs which may be run on encryptions of their inputs to produce an encryption of their output. Since such a program never decrypts its input, it can be run by an untrusted party without revealing its inputs and internal state. The existence of an efficient and fully homomorphic cryptosystem would have great practical implications in the outsourcing of private computations, for instance, in the context of cloud computing.



Fully Homorphic Encryption(FHE)

The "homomorphic" part of a fully homomorphic encryption scheme can also be described in terms of category theory. If *C* is the category whose objects are integers (i.e., finite streams of data) and whose morphisms include addition and multiplication, then the encryption operation of a fully homomorphic encryption scheme is an endofunctor of *C*. The categorical approach allows for a generalization beyond the ring structure (finitary composition of addition and multiplication) of the integers. If the morphisms of some wide supercategory of *C* include the primitive recursive functions or even allcomputable functions, then any encryption operation which qualifies as an endofunctor of this supercategory is "more fully" homeomorphic since additional operations on encrypted data (for example conditionals and loops) are possible.

The utility of fully homomorphic encryption has been long recognized. The problem of constructing such a scheme was first proposed within a year of the development of RSA.A solution proved more elusive; for more than 30 years, it was unclear whether fully homomorphic encryption was even possible. During this period, the best result was the Boneh-Goh-Nissim cryptosystem which supports evaluation of an unlimited number of addition operations but at most one multiplication.

Craig Gentry using lattice-based cryptography showed the first fully homomorphic encryption scheme as announced by IBM on June 25, 2009. His scheme supports evaluations of arbitrary depth circuits. His construction starts from 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.) He then shows how to modify this scheme to make it *bootstrappable*—in particular, he shows that by modifying the somewhat homomorphic scheme slightly, it can actually evaluate its own decryption circuit, a self-referential property. Finally, he shows that any bootstrappable somewhat homomorphic encryption scheme can be converted into a fully homomorphic encryption through a recursive self-embedding.

In the particular case of Gentry's ideal-lattice-based somewhat homomorphic scheme, this bootstrapping procedure effectively "refreshes" the ciphertext by reducing its associated noise so that it can be used thereafter in more additions and multiplications without resulting in an indecipherable ciphertext. Gentry based the security of his scheme on the assumed hardness of two problems: certain worst-case problems over ideal lattices, and the sparse (or low-weight) subset sum problem.

V. CONCLUSION

Security of cloud computing based on fully homomorphic encryption is a new concept of security which is to enable to provide the results of calculations on encrypted datawithout knowing the raw entries on which the calculation was carried out respecting the confidentiality of data. Our work is based on the application of fully Homomorphic encryption to the security of Cloud Computing: a) Analyze and improve the existing cryptosystem to allow servers to perform various operations requested by the client. b) Improve the complexity of the homomorphic encryption algorithms and study the response time to requests according to the length of the public key.

REFERENCES

[1] Sean Marston and al. "Cloud computing — The business perspective", *Volume 51, Issue 1, Pages 176–189, http://www.sciencedirect.com, April 2011.*

[2] Nivedita Manohar, "A Survey of Virtualization Techniques in Cloud Computing", Proceedings of International Conference on VLSI, Communication, *Advanced Devices, Signals & Systems and Networking (VCASAN-2013), Volume 258, 2013, pp 461-470, springer, 2013.*

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

[4] Sean Carlin, Kevin Curran, "Cloud Computing Technologies*", International Journal of Cloud Computing and Services Science (IJ-CLOSER), Vol.1, No.2, pp. 59~65, June 2012.*

[5] John Mutch, Brian Anderson, "Secure Multi-Tenancy for Private, Public, and Hybrid Clouds", in Preventing Good People from doing Bad Things, Springer, 2011.

[6] Peter Mell, Timothy Grance, "The NIST Definition of Cloud Computing", NIST Special Publication 800-145, Sep. 2011.