International Journal of Engineering Technology and Management Sciences[IJETMS]

Website: ijetms.in Issue:2, Volume No.3, March-2019

Design of protected cloud computing by Homomorphic Encryption

Dr.J. Keziya Rani

Assistant Professor, Department of Computer Science & Technology, S.K University, Anantapuramu.

Abstract—The idea of homomorphic encryption is to make sure data confidentiality in messages, storage or in utilize by processes with methodssimilar to conventional cryptography, but with additionalabilities of computing over encrypted data, searching an encrypted data, etc. Homomorphism is a property by which a problem in one algebraic system can be converted to a problem in another algebraic system, be solved and the solution later can also be converted back successfully. Therefore, homomorphism composes secure delegation of computation to a third party feasible. Various conventional encryption schemes have either multiplicative or additive homomorphic property and are presently in use for personal applications. So far, a Fully Homomorphic Encryption (FHE) scheme which could perform any arbitrary computation over encrypted data appeared in 2009 as Gentry's work. In this paper, we suggest a multi-cloud architecture of M distributed servers to repartition the data and to almostpermit achieving an FHE.

Keywords— *confidentiality, multi-cloud; distributed System; Fully homomorphic encryption;*

INTRODUCTION

Cryptosystems provide methods to ensure data privacy and integrity. Suppose the data is constantly encrypts in the cloud, then control will not lost, and the concerns will detach. When an encryption algorithm does not allow arbitrary computation over encrypted data, the encrypted data have to be decrypt earlier than the computation, and the decrypted data will no longer under control.

The vision of outsourcing an increasing amount of data storage and management to cloud services raises many new privacy concerns for individuals and businesses alike. The privacy concerns can be satisfactorily addressed if users encrypt the data they send to the cloud. If the encryption scheme is homomorphic, the cloud can still perform meaningful computations on the data, even though it is encrypted.

In any organization to perform some operations if they want to download confidential data from the cloud to a trusted computer and then send the encrypted results backed to the cloud, Cloud computing is infeasible for such business organizations. Encrypted data has previously been impossible to operate on with out first decrypting them. Some encryption algorithms that permit arbitrary computation on encrypted data. For example, RSA is a multiplicatively homomorphic encryption algorithm where the decryption of the product of two encrypted data will be the product of the two plain data. On the other hand, RSA will not allow addition operation or the combination of additions And multiplications. Soon after, FHE has emerged [1] to carry out infinite chaining of algebraic operations in the cipherspace, which means that a random number ofadditions and multiplications can be applied to encrypted operands. Unfortunately, all executions of FHE schemes proved that the performance is still slow for practical applications.In the last two years, solutions for fully homomorphic encryption schemes have been proposed and improved upon, but the problem faced with the efficiency .

In this paper we discuss the following: The Homomorphic encryption and interrelated definitions, its applications are defined in section I. In section II, we talk about the Homomorphic Scheme. In section III, we present some examples of partially homomorphic cryptosystems. In section IV, we propose a protected multi-cloud architecture for processing encrypted data. SectionV deals withconclusion.

1. HOMOMORPHICENCRYPTION

Homomorphic encryption is a form of encryption that permit computations to be passed out on ciphertext, thus producing an encrypted result which, when decrypted, matches the result of operations carry out on the plaintext. Homomorphic encryption let the chaining together of different services without exposing the data to each of those services. For example, a chain of different services from different companies can calculate 1) the order 2) the customer transaction details 3) shipping, on a transaction without revealing the unencrypted data to each of those services. Homomorphic encryption schemes are mouldable by design. This allows their requirement in cloud computing environment for ensuring the privacy of processed data. Along with that the homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems,collision-resistant hash functions, private information retrieval schemes, and many more.

1.1 Applications of Homomorphic Encryption:

Many approaches on homomorphic encryption had been recognized very early. There are many applications which required a scheme that could work out homomorphically on encrypted data. But with the growing interest and tendency towards cloud computing has opened various possible application areas for Homomorphic Encryption. According to authors in [2] these applications can be majorly classified based on whether we expect privacy of data or circuit privacy or both. The categories are:

• Private Data, Public functions: like in Medical Applications.

•Private data, Private functions: like in Financial Applications.

The above mentioned applications assume single data (content) owner who encrypts the data and stores it on an untrusted cloud.

1.1.1 Electronic Voting : It is a unique case of allocation of calculation where one would like the election authorities to be able to calculate the votes and display the final results, but dislikes the idea that individual votes are first decrypted and afterwards tallied. In a voting system based on homomorphic encryption voters take turns incrementing an encrypted vote tally using a homomorphic operation. They are only allowed to increase the encrypted tally by 1 or by 0. Here 1 means indicating a vote for the candidate and 0 means indicating no vote for the candidate. In elections where each voter votes for one of N candidates, voters modify the encrypted tallies by adding an N-bit vector, where accurately one entry is 1 and the rest are all 0's. They are not capable to alter the counters in any other way. Therefore, homomorphic encryption is one of the solution for creating a "secret ballot" system online, whereverthe votes will not reveal neither to anybody else except the voter.

2. A. Definition of a Homomorphic EncryptionScheme

A public-key encryption scheme S=(KeyGen, Encr, Decr) is homomorphic if for all N and all (pk,sk) output from KeyGen(k), it is possible to define groups T, E so that:

2.1 The plaintext space T, and all ciphertexts output by Encrpk are elements ofE.

For any t1 , t2 ∈T and e1 , e2 ∈E with t1 = Decrsk (e1) and t2 = Decrsk (e2) it holdsthat:

Decrsk (e1 ∗e2) = t1 ∗t2

Where the group operations ∗ are carried out in E and T, respectively.

Similarly,ahomomorphiccryptosystemisaPKSwith the added property that there exists an efficient algorithm (Eval) to calculate an encryption of the sum or/and the product of two messages given the public key and the encryptions of the messages, but not the messagesthemselves.

Additionally, a fully homomorphic scheme is capable to get output as a ciphertext that encrypts f (t1,...,tn), where f is any desired function, which of course must be calculated effectively. Informationaboutt1,...,tnorf(t1,...tn),oranyintermediate plaintext values will not leak. The inputs, outputs and intermediate values are always encrypted. Prior totake a closer look on fully homomorphic encryption schemes, we will need another importantnotionfrominformationtheory.

2.2 Circuits

Casually speaking, circuits are directed, acyclic graphs where nodes are called gates and edges are called wires. Depending on the nature of the circuit the input values are integers, boolean values, etc. and the matching gates are set operations and arithmetic operations or logic gates (AND, OR, NOR, NAND, ...). In order to calculate a function f, we express f as a circuit and topologically arrange its gates into levels which will be executein sequence.

Example. Assume the function f outputs the expression

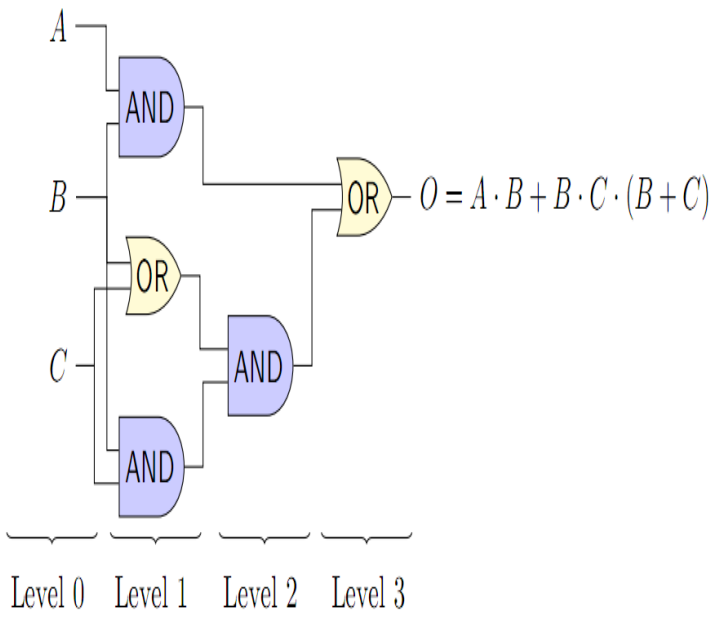

Fig. 1. Example for circuit representation

A·B+B·C·(B+C) on input (A,B,C). Then the following circuit represents the function f, with the logic gates AND and OR.

Two important complexity measures for circuits are size and depth. The size of a circuit C is the number of its non-input gates. The depth of a circuit C is the length of its longest path, from an input gate to the output gate, of its underlying directed graph. This yields to another definition of fully homomorphic encryption [8]:

ciphertexts $\Psi = \{c_1, ..., c_t\}$ where $c_i \leftarrow Enc_{pk}(m_i)$, outputs

$$c \leftarrow Eval_{pk}(C, \Psi)$$

under pk.

To construct fully homomorphic encryption schemes we can also follow the second way. To know how this transformation works, we need the following definitions and corollaries.

Definition : A homomorphic encryption scheme E is said to be correct for a family C_E of circuits if for any pair (sk, pk) output by $KeyGen_E(\lambda)$ any circuit $C \in C_E$, any plaintext $m_1,...,m_t$, and any ciphertexts $\Psi = c_1, ...,c_t$

with $c_i \leftarrow Enc_{pk}(m_i)$, it is the case that:

If $c \leftarrow Eval_E(pk, C, \Psi)$, then $Dec_E(sk, c) \rightarrow C(m_1,...,m_t)$

Except with negligible probability over the random coinsin $Eval_E$.

Definition: A homomorphic encryption scheme E is compact, if there is a polynomial f so that, for every value of the security parameter λ, E's decryption algorithm can be expressed as a circuit DE of size at most f (λ).

A homomorphic encryption scheme E efficiently evaluates circuits in CE if E is compact and also correct for circuits in CE.

Corollary: A homomorphic encryption scheme E is fully homomorphic if it compactly evaluates all circuits.

This requirement is considered to be approximately too strong for practical purpose, therefore it uses a certain relaxation to comprise leveled schemes, which only estimate circuits of depth up to some d, and whose public key length may be poly(d).

Definition: (leveled fully homomorphic). A family of homomorphic encryption schemes {E(d) : d ∈ Z+ } is said leveled fully homomorphic if, for all d ∈ Z+ , it all uses the same decryption circuit, E (d) compactly evaluates all circuits of depth at most d (that use some specified set of gates), and the computational complexity of E (d) 's algorithms is polynomial in λ, d, and (in the case of EvalE) the size of the circuit C.

An encryption scheme which supports both addition and multiplication (a fully homomorphic scheme) thereby Preserves the ring structure of the plaintext space and is therefore far more powerful. Using such a scheme makes it achievable to let an untrusted party do the computations without ever decrypting the data, and as a result preserving their confidentiality.

An extensivelyvalued application of homomorphicencryption schemes is cloud computing. Currently, the need for cloud computing is growingrapidly, as the data we are dealing out and computing on is getting superior and superior every day.

In order to be clear consider a small example Say, Seeta wants to store a sensitive file m ∈ {0, 1}n on Ram's server. So she sends Ram Encr(m1), ..., Encr(mn). Assume that the file is a database (a catalog of people with specific data about them) and Seeta wants to find out howmany of them are 35 years old. Instead of retrieving the data from Ram, decrypting it and searching for the wanted information, she will ask Ram to do the computations, without him knowing what or who he is computingon.

The answer from Ram comes in form of a ciphertext which only she can decrypt with her secret key.

Theadvantageoffullyhomomorphicencryptionhaslongbeen acknowledged. The query for constructing such a scheme arises withinayearoftheimprovementofRSA[2].

During this period, the most excellent encryption system was the Boneh-Goh-Nissim cryptosystem [9] which supports estimation of an infinite number of addition operations but one multiplication at the most.

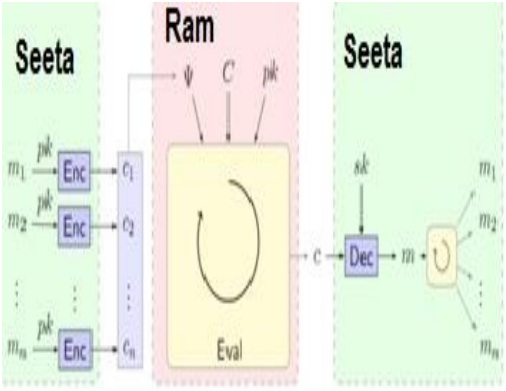

Fig. 2. Diagram of a homomorphic encryption scheme

A general reason why a scheme cannot compute circuits of a certain depth Is that after a certain amount of computations too much error will build ups, which results the decryption to obtain a wrong value. The decryption usually is able to handle small amounts of error within a certain range and bootstrappable

encryption enables "refreshing" after some time. The basic idea of "refreshing" is to encrypt under a first key. Calculate until right before the error grows too large. Encrypt under a second key.

Gentry's method can be broken down into three main steps:

Step 1: creating an encryption scheme by means of ideal lattices that is somewhat homomorphic, which means it is limited to estimating low-degree polynomials over encrypted data.

Step 2: "Squashing" the decryption circuit of the original somewhat homomorphic scheme to make it boot strappable.

Step 3: Bootstrapping to some extent improved original scheme of step 2 to yield the fully homomorphic encryption scheme. This will be done with a "refreshing" procedure.

The innovative idea of Gentry's method of creating a fully homomorphic scheme out of a somewhat homomorphic scheme is the method of squashing and boot-strapping. Mathematically the most appealing step is the first step.

3. SOMEWHAT HOMOMORPHICSCHEME

some computations over encrypted data. Gentry then demonstrated that if you can handle to design a SHE scheme that supports the evaluation of its own decryption algorithm (and a little more), then there is a common method to transform the SHE scheme into a FHE scheme. A SHE that can estimate its own decryption algorithm homomorphically is called *bootstrappable* and the procedure that changes a bootstrappable SHE scheme into a FHE scheme is called *bootstrapping*.

Bootstrapping. First we discuss abou thow the currently-known SHE schemes work. In general, the ciphertexts of all these schemes contain noise in it and unfortunately this noise gets better as more and more homomorphic operations are carry out. There may be some situations that the encryptions become useless due to much noise i.e., they do not decrypt correctly. This is the main drawback of SHE schemes and this is the reason that they can only carry out a restricted set of computations. Bootstrapping allows us to control this noise.

The design is to take a ciphertext with a huge noise in it and an encryption of the secret key and to homomorphically decrypt the ciphertext. Note that this can only work if the SHE scheme has enough homomorphic ability to evaluate its own decryption algorithm which is why we need the SHE scheme to be boot strappable. This homomorphically computed decryption will effect in a new encryption of the message but without the noise or at least with less noise than before. More concretely, say we have two ciphertexts:

c1=Epk (m1) and c2=Epk (m2)

with noise n1 and n2, respectively. We can multiply these encryptions using the homomorphic property of the SHE scheme to get an encryption*:*

c3= Epk (m1 x m2) of m1 x m2 under key *pk* ,but *C3* will now have noise n1xn2. The plan behind bootstrapping is to get rid of this noise as follows. First, we encrypt *C3* and *sk* under *pk* .This results in two new ciphertexts

C4=Epk(C3) =(Epk (m1 x m2)) and C5= Epk(sk)

Given *C4* and *C5,* we now *homomorphically* decrypt *C4* using *C5*. similarly, we compute the following operation over *C4* and *C5*: "decrypt *c3= Epk (m1 x m2)* using *sk*". This is allowed since the scheme has enough homomorphicability to assess its ownd ecryption algorithm.

Through this technique during a computation whenever the ciphertexts get too noisy, we can remove the main drawback of the SHE scheme and turn it into a FHE scheme.

It turns out that constructing a bootstrappable SHE scheme is complex. To do this, Gentry build his scheme using complexmethods [1] so a lot of the recent work in FHE has attempted to figure out how to design simpler bootstrappable SHE schemes.

3.1 PARTIALLY HOMOMORPHICCRYPTOSYSTEMS

A. RSA-A Multiplicatively HomomorphicScheme

In 1978, Rivest, Shamir, and Adleman published their public-key cryptosystem that make use of elementary thoughts from number theory, in their paper "A Method for Obtaining Digital signatures and Public-Key Cryptosystems" [3]. It was one of the first homomorphic cryptosystem. The RSA cryptosystem is the most extensively used public-keycryptosystem. It may be used to give both confidentiality and digital signatures and its security is based on the intractability of the integer factorization problem.

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. 3. RSA Algorithm

Key Generation: 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: 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: 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. 4. Paillier Algorithm

The **encryption** algorithm take a message m as inputfrom the plaintext space Zn and calculates according c iphertext.
c = me mod n. This integer c ∈ Zn cannot be traced back to the original message without the knowledge of p and q, which will be proved later in this section.

Decryption takes as input the ciphertext c and the secret key (d, n) and computes m = cd mod n. Since d is the inverse of e in Zn this is indeed the original message.

The three steps (keygeneration, encryption and decryption) can be found in the following table.

B. Paillier - An Additively HomomorphicScheme

Pascal Paillier introduced his cryptosystem in 1999 and published paper "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes" [11]. The proposed technique is based on composite residuosity classes, whose computation is supposed to be computationally difficult. It is a probabilistic asymmetric algorithm for public key cryptography and inherits additive homomorphic properties.

The encryption process takes a message m ∈Zn as input and randomly chooses an integerrin Z∗, this random number is used to satisfy the probabilistic algorithm's n property, that one plaintext can have many ciphertexts. It is later revealed that this random variable does not delay the correct decryption, but has thee ffect of altering the corresponding ciphertext.

The three steps (keygeneration, encryption and decryption) can be found in the following table:

4. OURARCHITECTURE

The fully homomorphic encryption schemes [1] are very time consuming. Assuming the evaluation of one gate demanding a refresh, the run-time will be significant as well as the processing of security parameters. A suggestion of a nearly FHE scheme based architecture for allowing the evaluation of any function and producing encrypted data is illustrated in Figure 6. In our proposed architecture, the service provider repartitions the processing among the servers to fasten the evaluation process of any function.

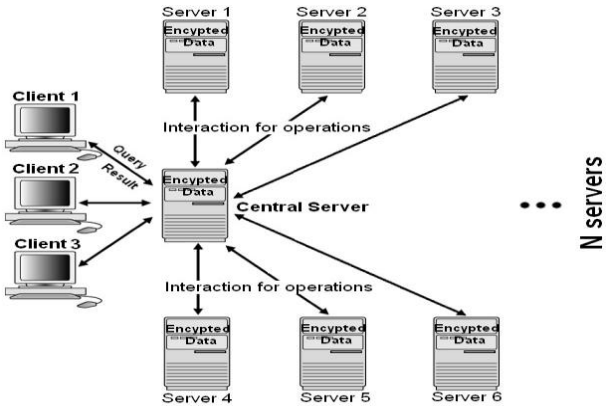

Fig. 5. An architecture of distributed servers for processing encrypted data

In this proposed system, we supply a high leveled architectural scheme during the usage of several servers in the computation. This computational system will nearly allow attaining a FHE, and thus a large number of operations containing additions and multiplications can be performed. For instance, in Fig. 5, it is clearly shown that Client 1 sends a query and requests the results of a given function, Let us consider a function $f(x)=ax^2+bx+c$. In this scenario the function elements are encrypted and divided into several portions depending on the number of operations (addition and Multiplication), and will be processed independently on N different servers, equivalent to the number of addition operations. At last the outcome or result is sent back to a Central Server in order to be forwardedtoClient1andthendecrypted.

The advantage is that no longer chipertext after encryption unlike the classical method. The keys are simply handled and more security is maintained since it isnot possible to read relevant information in distributed systems. In the cloud the N servers consists of hypervisors hosting multiple virtual machines which supports developing the response time and augment the number of the involved computational entities in the distributed system.

In this proposal, we evaluate the added value of the distributed systems in processing operations requested by clients. The scheme of homomorphic encryption is transmit within the servers and this can be practical and help developing the security of the cloud in terms of confidentiality of data and performance.

An additional concern that must be measured in our architecture is the confidentiality of the processed data overt the is tributed systems, Now a days which is the main anxiety of most organizations when using third-party hosting. The approach concerning this is sue is the divide the stored data among multiple Cloud service providers to reduce the danger of data violaters and increase the parallel processing as well as the number of the servers involved in performing homomorphic encryption. Partitioning and outsourcing the data, applications onto different cloud infrastructures has the advantage of making them uncertain for third-parties and opponents, and thus this assist enhancing the privacy as well as the confidentiality.

Like the stored encrypted data is repartitioned among a Multi-Cloud Architecture belonging to different Cloud Service Providers mentioned in Fig. 6, Client 1 can carry out operations on them and clearly get back the future results. The data is segmented during a Data Partitioning Algorithm (DPA) which permits partitioning, collecting and reconstructing the data. The main operation will chunked into subsets to be handled by the N Clouds/N Servers. The mixture of N Clouds and homomorphic encryption using N servers gives an improved security strategy which is a safe approach to avoid any potential data breaches even if the data have been previously encrypted.

Selecting a trusted CSP needs a Service Level Agreement (SLA), agreement cooperation and risk estimation. In most cases it may be logical to believe that a CSP to be trustworthy and handling the clients' sensitive data and applications in a responsible manner.

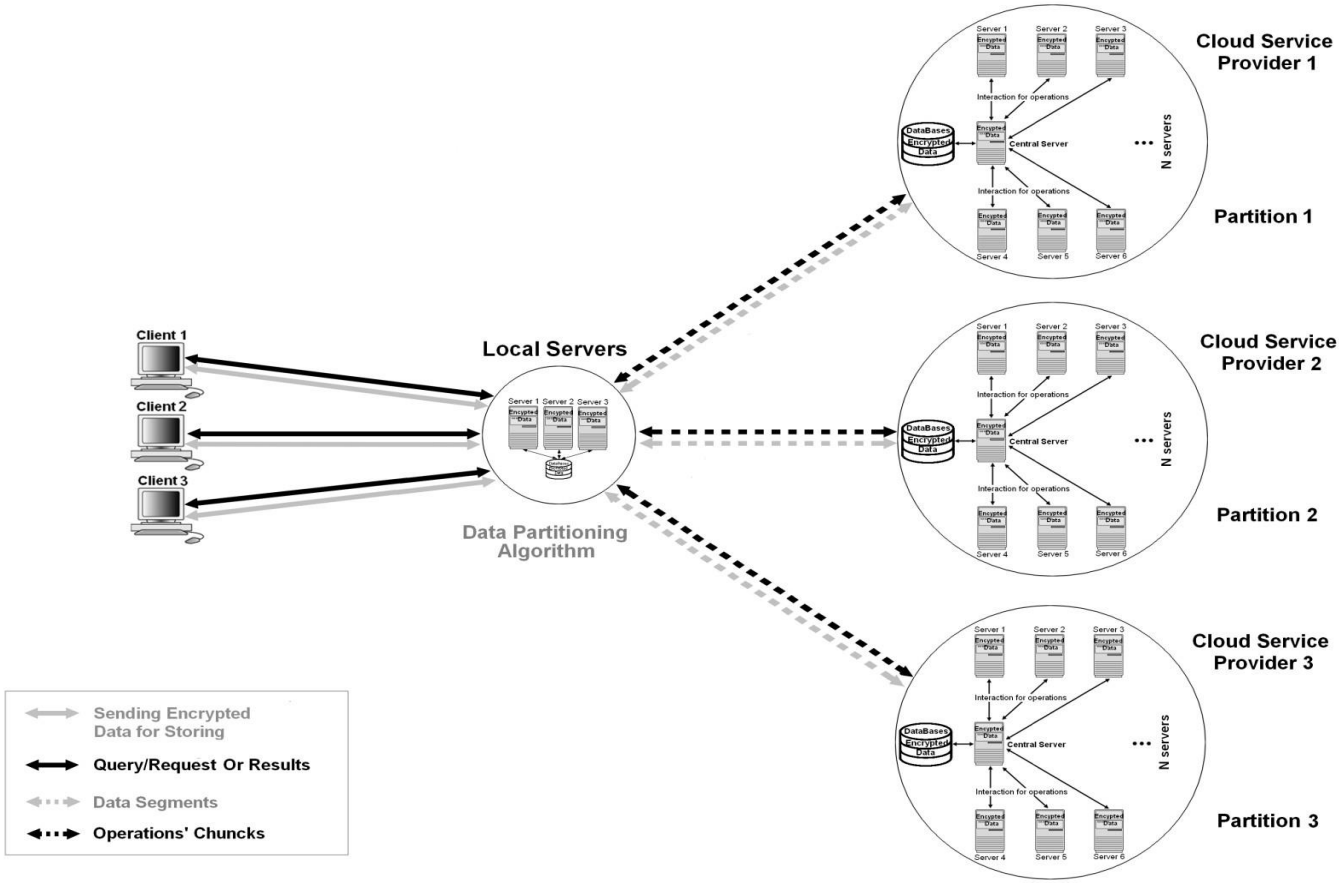

Fig. 6. The proposed architecture to secure data using homomorphic encryption

5. CONLUSION

By considering the proposals of Gentry proposed his construction regarding FHE, and made enormous attempt to make FHE more practical. While a lot of progress has been made, unfortunately, we are still on the way to show the FHE as practical.

Majority of FHE schemes are based on Gentry's blueprint which includes of first constructing a SHE and then using Gentry's bootstrapping technique to convert it into a FHE scheme. It converts out that bootstrapping is a major bottleneck and that SHE is actually reasonably well-ordered. So, if we consider about practical applications, then it may be sensible to investigate what exactly we can do with SHE as an alternative.

Distributed systems and multi-could architectures can convey lots of advantages to the application of homomorphic encryption and making it more realistic in the case of the security of data and applications.

The future enhancement will focus on the implementation of our proposal is to accomplish security and performance tests in order to explain its practicality.

ACKNOLEDGEMENTS

We sincerely thank Mr. Kamal Benzekki , Mr.Abdeslam El Fergougui and Mr.Abdelbaki El Belrhiti El Alaoui whose paper "A Secure Cloud Computing Architecture Using Homomorphic Encryption" has been a source of inspiration.

REFERENCES:

[1] C. Gentry, "A fully homomorphic encryption scheme," Doctoral dissertation, Stanford University, 2009.

[2] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," In Foundations of Secure Computation, pages 169- 180, 1978.

K.Lauter, M.Naehrig and V.Vaikunthnathan, "Can homomorphic encryption be practical?", Proc of 3rd ACM workshop on Cloud Computing Security Workshop , pp 113- 124, 2011.

[3] R. L. 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.

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

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

[6] 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.

[7] T. ElGamal, "A public key cryptosystem and a signature sche based on discrete logarithms," IEEE Transactions on Information Theory, 469- 472, 1985.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," InSTOC, Vol. 9, pp. 169-178, 2009.

[9] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," In Proceedings of Theory of Cryptography (TCC) '05, LNCS 3378, pages 325-341, 2005.

[10] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," In Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, pages 112-131. Springer-Verlag, 1997.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," Advances in Cryptology Eurocrypt, 1592:223-238, 1999.

[12] S. Goluch, "The development of homomorphic cryptography: From RSA to Gentry's privacy homomorphism" Doctoral dissertation, Vienna university of Technology, 2010.