International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

A Survey on Homomorphic Encryption for Security in Cloud

Vinay U & Rakesh V.S
[1]VIII Semester Student, [2]Assistant Professor
Department of Computer Science and Engineering, Cambridge institute of technology, Bengaluru, India

Abstract:

The Adoption of Digitization of data in large scale and change in requirements of the different organization to serve the different stakeholder. So, there is a need to compute, store and analyze large amount of data in remote server efficiently providing security to the data involves network security, strategies of control and access to the service, storage of data .the smart computations of cloud computing and big data is highly appreciated today. fully homomorphic encryption(FHE) is a best type of encryption schemes that allows working with the data in its encrypted from this paper aims introduce an efficient and verifiable FHE based on a new mathematic structure the is noise free this paper discuss the different homomorphic encryption schemes and their applications on various domains .

Keywords

Cloud Data storage, Homomorphic Encryption, Data Security, Fully Homomorphic Encryption, privacy cloud computing

1. Introduction

As Computing and communication technology takes a quantum leap with the digitization of data in large scale, the need of collecting, storing and analyzing large amount of data becomes very much essential. Cloud computing has manifested as a powerful computing model in the last decade, with numerous advantages both to client and providers. Also, In the 90's,the democratization of IT especially during the last decade, with the generalization of the internet, the development of broadband networks, rental application, payment for the use and request for mobility. According to the definition given by National of standards and technology(NIST), "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resource that can be rapidly provisioned and released with minimal management effort or service provider interaction". It also defined five essential characteristics of cloud computing as on –demand self service, broad network access, resource pooling, rapid Elasticity and measured service. Encryption schemes as RSA, AES, 3DES… allows client to preserve data privacy during transmission to the cloud, but if a client request the cloud to perform a complex treatment on its data . he should share his private key with the remote key server.

The question now is how can we perform calculation on data previously encrypted without having to decrypt?

The answer to solve this problem is the Homomorphic Encryption. An encryption that is fully homomorphic, and allows to compute over encrypted data without having to decipher them.

In 2009, C. Gentry [11] proposed that first fully homomorphic encryption at Stanford university: A cryptosystem that provides the ability to perform arbitrary calculations on encrypted data without having to decrypt them.

This paper consists of in section 2. I explain the necessity of cloud adoption by different domains. In section 3 . We provide background information on homomorphic encryption followed by details of our implementation In section 4. Finally, section 6 is devoted for the conclusion.

2.Related Work

The concept of Homomorphic encryption was initially proposed by rivest et al[7]after the discovery of public key cryptography a majority of the known public key cryptosystems ,i.e, RSA, Elgamal, paillenc, etc. support homomorphic addition or multiplication or cipher text but not both[8].there are partially homomorphic crypto system[6],the partially homomorphic encryption is not sufficient for various practical environments such as finance and statistical application .because these application requires both addition and multiplication operation for realizing various computations. The concept of Homomorphic encryption was first proposed by Rivest et al. [7] after the discovery of public key cryptography. A majority of the known public key cryptosystems, i.e., RSA, Elgamal, Pailler, etc. support homomorphic addition or multiplication on ciphertext but not both [8]. These are partially homomorphic crypto systems [6]. However the partially homomorphic encryption is not sufficient for various practical environments such as finance and statistical applications.This is because of the fact that these applications require both addition and multiplication operation for realizing various computations. In 2009, Gentry proposed Fully homomorphic encryption (FHE) [1]

which first constructs an SHE (some what homomorphic encryption) and then converts the same to an FHE which in turn can evaluate circuits of arbitrary depth. This approach uses bootstrapping procedure for the conversion. Gentry's construction is based on his bootstrapping theorem which provides that given a somewhat homomorphic encryption scheme (SWHE) that can evaluate homomorphically its own decryption circuit and an additional NAND gate, we can pass to a "levelled" fully homomorphic encryption scheme and so obtain a FHE scheme by assuming circular security. The purpose of using bootstrapping technique is to allow refreshment of ciphertexts and reduce noise after its growth.

Gentry's construction is not a single algorithm but it considered as a framework that inspires cryptologists to build new fully homomorphic encryption schemes [3, 4, 5, 6…]. A FHE cryptosystem that uses Gentry's bootstrapping technique can be classified in the category of noise based fully homomorphic encryption schemes [7].

3.Homomorphic Encryption

The data stored in the cloud will not be in the encrypted format. If it is stored in the crypted way that can solve the issues like Availability, Data security and Third party control. But the problem is the user will not be able to depend on the cloud service provider to carry out computation of data. For this the data will be decrypted first then will be shipped to the user for computation. So the cloud provider has to decrypt thedata first thus nullifying the issues of privacy and confidentiality, perform the computation and then send the result to the user [17]. Suppose if the user could carry out any arbitrary computation on the hosted data, then without the cloud provider learning about the users' data, computation is done on the encrypted data without prior decryption. In this scenario, the promise of homomorphic encryption takes a 552 call [5]. Homomorphic encryption schemes are methods that allow the transformation of cipher texts C(M) of message M , to cipher texts C(f(M)) of a computation / function of message M, without disclosing the message. A homomorphic encryption scheme H is a set of four functions which are shown in Fig.1 and defined as

H= {Key_Generation, Encryption, Decryption, Evaluation}

A. *Key_ Generation:* In this first function, the user will generate a pair of keys, public key PK and a secret key SK for encryption of plain text.
B. *Encryption:* In the second function encryption, the user will encrypt the plain text using the secret key SK and generate cipher text and along with the public key PK, the cipher text will be sent to the server.
C. *Evaluation:* In this function , server applies a function to evaluate the cipher text CT and this is performed as per the required function using PK.
D. *Decryption:* In the decryption function , generated evaluated plain text will be decrypted by the user using his secret key and gets the final result.

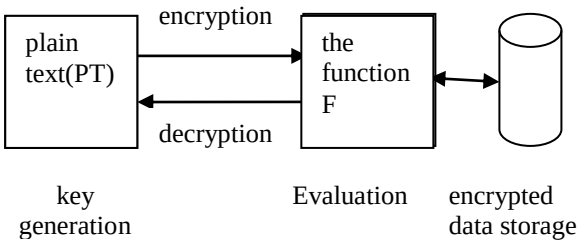

Fig 1. Homomorphic functions

In the cloud based environment the key generation takes place at the client side and encrypts the data with the encryption key and sends the data to the cloud server along with pk . The encrypted data is stored in the database along with the key. Whenever the client wants to perform the operation it sends the request to the service provider. The service provider forwards the request to the processing server .the processing server performs the operation as per request . The service provider then returns the processed result to the client in the response phase . The client finally decrypts the result returned by the service provider with the secret key sk. Among the homomorphic encryption schemes available depending on the operations performed on data, can be classified into three main categories namely: Partially Homorphic Encryption(PHE), SomeWhat Homomorphic Encryption (SWHE) and Fully Homomorphic Encryption(FHE) [6].

- Partially Homomorphic Encryption(PHE): PHE allows either addition or multiplication operation to be performed on encrypted data but not both
- Some What Homomorpic Encryption (SWHE): SWHE allows more than one operation to be performed on encrypted data but on a limited scale
- Fully Homomorphic Encryption (FHE): FHE allows any number of additions and multiplication operations to be performed on encrypted data.

A. Partially Homomorphic Encryption Scheme: The most popular PHE methods available are the RSA[7],ElGamal[8] and Pailler[9] methods.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

1. 1.RSA method : RSA was the first Homomorphic Encryption scheme developed by Ronald Rivest , Leonard Adleman and Michael Dertouzos in 1978[7]. It is a public key crypto technique and is multiplicative homomorphic in nature. Let M1 and M2 be the two messages . Then by using RSA algorithm as shown in Figure 2, the encryption can be done as

$$E(M1,PK) * E(M2,PK) = M1^e * M2^e \bmod n$$
$$= (M1 \times M2)^e \bmod n$$
$$= E(M1XM2,PK)$$

Key Generation:Key Gen(p,q)
Input:p,q€P
Compute n=p,q;Φ(n)=(p-1)(q-1)
choose e such that gcd(e, Φ(n))=1
determine d such that e,d =1 mod Φ(n)
output(pk,sk)
public key pk=(e,n)
secret key sk=(d)
encryption:
$C=M^e \bmod n$
decryption:
$M=C^d \bmod n$

Figure 2. RSA Algorithm

2. Pailler method: This method is additive Homomorphic in nature and is developed by Pascal Pailler in 1999[9]. Let M1 and M2 be the two messages and let C1 and C2 are the two cipher texts corresponding to M1 and M2 respectively. By using Pailler algorithm as shown in Figure 3, C1and C2 can be computed as follows

$$C1=g^{M1}.r1^n \bmod n^2$$
$$C2= g^{M2}.r2^n \bmod n^2 .$$
$$C1.C2= g^{M1}.r1^n.g^{M2}.r2^n \bmod n^2$$
$$=g^{M1+M2}(r1r2)^n \bmod n^2$$

Key Generation:keyGen(p,q)
Input:p,q€P
Compute n=pq Choose €$Z^*_{n^2}$ such that Gcd[l($g^\lambda \bmod n^2$),n)=1 with L(u)=(u-1)/n
Output:(pk,sk) Public key:pk=(n,g) Secret key:sk =(p,q)

Encryption:Enc(m,pk)
Input:m€Z_n
Choose r€Z^*_n Compute $e=g^m,r^n \bmod n^2$
Output:e€Z_{n^2}

Decryption:dec(c,sk)
Input:c€Z_{n^2}
Compute $m=(L(c^\lambda \bmod n^2))/(L(g^\lambda \bmod n^2)) \bmod n$
Output m€Z_n

Figure 3. Pailler Algorithm

B. Some What Homomorphic Encryption Scheme : The most popular SWHE method is Boneh-Goh-Nissim (BGN) method[10]. This method allows any number of additions but only one multiplication to be performed on data. It is invented by Dan Bonch, Eu-Jin Goh and Kobi Nissim [10] in 2005.Let M be message. Then by using BGN algorithm M can be encrypted to get C as C= gmhr where r is from{1,2,3,…,n-1} . In the decryption process M can be recovered using discrete logarithm of cp to the base gp . The algorithm is as shown in Figure 4.

Preparation of Key
Input:p,q€P(large primes)
• **G is a cyclic group of order pq** • **e is a pairing map e:G*G->G1** • **compute n=p*q** • **pick up two random generators g,u, from G** • **compute h=u^2.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 $c=g^n h^r$**
Output:c€G

Decryption:Dec(c,Sk)
Input:c€G

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

• **Compute:$c^p=(g^mh^r)^p=(g^p)^m$** • **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 Scheme:

The most popular FHE schemes are Algebra Homomorphic Encryption Scheme based on Updated ElGamal(AHEE) proposed by Chen Liang and Gao Changmin in 2008[4],[11], Algebraic Homomorphism Encryption Scheme based on Fermat's Little Theorem(AHEF) proposed by Xiang Guangli and Cui Zhuxizo in 2012[12] and Enhanced Homomorphic Encryption Scheme(EHES) proposed by Gorti VNKV Subba Rao in 2013[13],[14]. The AHEE algorithm is as shown in Figure 5 and it allows unlimited additions and multiplications.

Step 1:select any two prime number say p and q
Step 2:calculate the product of those two prime numbers say N=p*q.where p and q being confidential and N is public.
Step 3:Select random number x and a root g of GF(p) where g and x are smaller than p
Step 4:calculate $y=g^x$ mod p.use this y for the encryption
Step 5:encryption will be performed in following two steps: Select random integer number er and apply following homomorphic encryption. $E_1(M)=(m+R*P)$ mod N Select random interger number k,and the encrypton algorithms are: $E_g(M)=(a,b)=(g^k$ mod p,$y^k E_1(M)$ mod p)
Step 6:Decrypted algorithm $D_g()$ is $M=b*(a^x)-1$ (mod p)

Figure 5. AHEE Algorithm

4.Proposed architecture:

Consider the scenario of the client-cloud provider system as shown in Figure 6 and Figure 7 in which the Figure 6 shows the normal operative mode and the Figure 7 shows the operative mode with HE adoption client-cloud provider system. Let T_R : Response Time ; T_{tr}: Transmission Time ;T_{pr}: Data Processing Time ; Ten:Encryption Time ;

T_{de}:Decryption Time ; T_{RH}: Response TimeinHEMode; then

1. The response time in the normal operative mode is given by
$$T_R= 2*T_{tr}+T_{pr}$$
2. The response time in the operative mode with HE adoption is given by
$$T_{RH}= T_{en}+2*T_{tr}+T_{pr}+T_{de}$$
3. The transmission time is the amount of time taken from start to end of the transmission of a message. This can be computed by using the formula,

Transmission time=message size/data rate

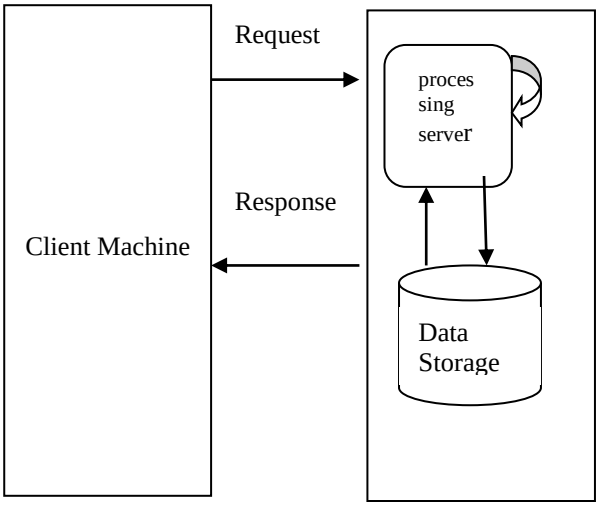

Figure 6. Client-Cloud Provider System –normal operative mode

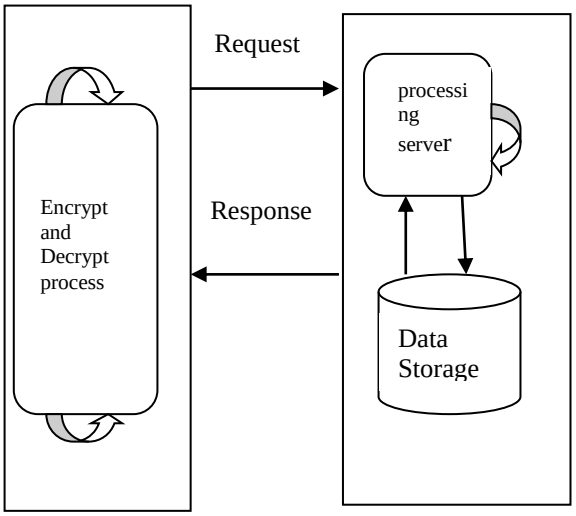

Figure 7. Client-cloud provider system –HE operation mode

In our work we have taken the mixed homomorphic technique proposed by Zvika Brakerski,Craig Gentry and Vinod Vaikunthanathan[16] whose basic functions are as shown in Figure 8 as the base work and built our method over it.

BGV is an asymmetric encryption scheme that encrypts bits. The scheme is based on lattices. There are two versions of the scheme: one deals with integer vectors and the other deals with integer polynomials [5]. Of the two versions of BGV scheme, we focused on integer polynomial version

whose security depends on hardness of the decisional Ring Learning with Errors problem.

In BGV scheme, a polynomial ring which is defined as A=ǽ[X]/F(X) where F(X) is a cyclotomic polynomial whose degree is 2k and denoted by d is considered. Also a chain of odd moduli q1< … <qL and their corresponding sub rings Aqi = A/qiA of polynomials A with integer coefficients into the range [-qi/2, qi/2] is considered. The elements in Aqi will be polynomials represented by the d-vector of their coefficients[20,21].

5.Experimental Results

The The various homomorphic encryption algorithms such as Paillier cryptosystem,Rivest,Shamir and Alderman(RSA) algorithm, Enhanced Homomorphic cryptosystem(EHC), Non-interactive Exponential Homomorphic Encryption scheme(NEHE), Algebra Homomorphic Encryption scheme based on updated Elgamal(AHEE)[23] are compared with BGV scheme based on the properties of homomorphic encryption,their applications, privacy preserving of data and their usefulness in cloud data storage. The results of the study are shown in Table 1.

From Table 1, it is clear that BGV scheme is best suited for cloud data storage as it satisfies all the parameters. It is mixed homorphic, also reduces noise level by following key switching and modulus switching techniques. Since the scheme provides security over integer polynomials, it is more suited for data storage.

Security is the prime requirement for storage because of the increasing usage of the internet or public cloud for storing the data. Security is needed for preserving the integrity, confidentiality and availability of the information system resources. There can be storage of data in the encrypted format in any database and facilitate the operations or the computations on encrypted data without decryption, a new idea called “privacy homomorphism” was proposed. This is an idea of the cryptosystem allows direct computation on the encrypted data.

Table 1.Comparison table of FHE schemes

Algorithm	Add-Homo	Mul-Homo	Mix-Homo	Priv. Pre	Cloud storage	Applications
Paillier	No	Yes	No	No	No	e-voting
RSA	No	Yes	No	No	No	Internet banking
Elgamal	No	Yes	No	No	No	Hybrid Systems
EHC	No	No	Yes	No	No	Manets
NEHE	No	No	Yes	Yes	No	E-Commerce
AHEE	No	No	Yes	Yes	No	Mobile Cipher

We have implemented our proposed cryptosystem in Java (JDK version 1.8) programming platform and analyzed the performance of these quential and parallel implementations with large data set. We have used a system with Intel Core i5-5200U CPU (i5 5th generation) having clock speed of 2.20 GHz with 2 cores and 4 logical processors, 12 GB RAM, and Windows 10 Operating System. Our encryption algorithm is implemented with128-bitsecretkeyandtherebyallcomputationsarerealized on the same level of encryption. We have generated plaintext as a series of random integers for all arithmetic computations and used encrypted 64-bit representation for integer. Thus, our data sets ranged from 8 MiB to 64 MiB of plaintexts. We observed that our parallel implementations provides over performance improvement for all arithmetic computations compared to its sequential counterpart. We also implemented Search operation over a dataset ranging from 8 MiB - 64 MiB of plaintext. Our results show that parallel implementation provides more than 80% improvement (i.e., execution time) over it’s sequential implementation. In addition, the rate of improvement is higher with increase in data size. On the other-hand, the degree of parallelism (i.e., no of simultaneous threads) is limited by number of available cores. The Figure. 2 shows experimental result for addition operation considering sequential (one thread) and parallel execution (varying number of threads). It has been observed that the performance of addition operation improves exponentially with increase in number of threads for a fixed (constant) data size and it stabilizes after reaching a certain threshold on number of threads. On the other hand, the performance improves linearly with varying data size. The improvement achieved using multi-treaded implementation is approximately over 90% (≈ 91.66%). These results signify that the performance of any computations in our proposed cryptosystem is independent of key size. shows the performance of multiplication operation. We observed that the parallel implementations of multiplication provides almost 90%(≈ 89.83%) improvement than its sequential counterpart even if the multiplication operation is computationally heavier than addition. This is achieved due to decrease in throughput with multiple threads.

The table reports time required for each computations in seconds. Here, we observed that the performance improvement is almost similar to addition and multiplication with parallel implementations.

6. Conclusion And Future Works

We have presented an efficient homomorphic cryptosystem for implementing various benchmark computations on ciphertext. The novelty of our proposed implementation lies in execution of "ciphertext refresh" procedure at KGS site that provides secure data storage and necessary computations on request. We have also analyzed the performance of sequential and parallel implementations of the heterogeneous computation algorithms with varying data size and presented a comparative study. This work provides a more practical solution for secure data storage and efficient realization of different computations on it without compromising data security. The experimental results show the efficacy of our system with 80% improvement on execution time for parallel implementations. In future, we aim at incorporating function level encryption on ciphertext to harden the security perimeter over remotely stored data in the CSP site. We also plan to integrate Role Based Access Control (RBAC) with our parallel homomorphic encryption towards enforcing multi-granular operational access rights to heterogeneous stakeholders or roles.

REFERENCES

[1] Craig Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009.

[2] Craigy Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st ACM Symposium on Theory of Computing STOC 2009, pages 169–178, 2009.

[3] Nick Howgrave-Graham. Approximate integer common divisors. In Cryptography and Lattices, pages 51–66. Springer, 2001.

[4] Seny Kamara and Mariana Raykova. Parallel homomorphic encryption. In International Conference on Financial Cryptography and Data Security, pages 213–225. Microsoft Research, 2013.

[5] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical?. Proceedings of the 3rd ACM workshop on Cloud computing security workshop, pages 113–124, 2011.

[6] Jian Liu, Lusheng Chen, and Sihem Mesnager. Partially homomorphic encryption schemes over finite fields. IACR Cryptology ePrint Archive, 2016/430, 2016.

[7] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. Foundations of secure computation, 4(11):169–180, 1978.

[8] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtainingdigitalsignaturesandpublic-keycryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[9] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 24–43. Springer, 2010.