International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Special, Issue 43 (March 2017), PP. 10-14

OPERATIONS ON FULLY HOMOMORPHIC ENCRYPTED DATA ON CLOUD

[1]Sanket Vyapari, [2]Shivani Tawde, [3]Mitali Joshi, [4]Achyut Pratap, [5]Rashmi Dhumal

[1,2,3,4] Computer Department, Mumbai University Terna Engineering College, Navi Mumbai, India
[5] Assistant Professor, Terna Engineering College, Navi Mumbai, India
[1]sanketvyapari@gmail.com [2]tawde.986@gmail.com [3]mitalijoshi09@gmail.com
[4]aspratap6196@gmail.com [5]rashmisalvi@gmail.com

***Abstract*—In a world where digitization of day to day activities is increasing rapidly, the strain is on professionals to provide the secure services & storing the data in encrypted form on cloud. The need for cloud security services is rising tremendously. However, there are many loopholes in today's system by which data can be deciphered/extracted and used by unauthorized person as per their needs. With the growth in volume and experience of cyber- attacks, ongoing attentions is applied to protect personal information, companies, government, military, corporations and other businesses who collect, process and store a great deal of confidential information on cloud in high volume. Data encryption is a basic solution for providing confidentiality to sensitive data. However, performing operations on encrypted data requires extra overhead, since repeated encryption-decryption need to be performed for every single operation on encrypted data. Fully Homomorphic Encryption (FHE) is the head on solution to solve this issue, since it enable to perform computations directly on encrypted data without sharing the secret key needed to decrypt the data. This paper will include a survey of how FHE operation is performed by using the Scarab library to overcome the confidentiality problems of data stored on cloud.**

***Index Terms*— Cloud, Fully Homomorphic Encryption (FHE), Scarab library, Sorting, Quick sort.**

I. INTRODUCTION

Cloud Computing is widely growing concept now-a days. The cloud computing means where different services — such as servers, storage and applications — are delivered to an organization's computers and devices through the Internet [1]. The prominent actors in Cloud Computing are Cloud Provider and Cloud User. Cloud Provider is the enterprise cloud services. A Cloud User can vary from organisations, educational institutes to individuals utilising the cloud services. There is a necessity for security, integrity, confidentiality and visibility with respect to the current cloud providers [2]. The three cloud computing service models [3] are as follows: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The problems of third party data security and securely providing computation become increasingly prominent. There is the risk that personal information sent to a cloud is often seen as valuable to individuals with malicious intent. There are some security issues in cloud computing such as data security, third-party control, and privacy. If all data stored in cloud were encrypted using traditional cryptosystems, this would effectively solve the three above issues. Confidentiality of information in a public cloud is a major concern, which is guaranteed by suitable encryption information.

The common security issues of cloud computing [4] can be divided into five main categories:

1. Availability: The data must be available whenever it is required. This is one of the prime concerns of mission and safety critical organizations.

2. Trust: When two parties are involved in a processing then the trust can be described as follows: An entity A is said to trust another entity B when entity A believes that the entity B will behave exactly as expected and required. This comprises cloud service provider to provide sufficient security policy that guarantees the efficient use of activities.

3. Confidentiality: Data confidentiality in the cloud means isolating the data of individual users from one another. It refers to trusting that specific applications or processes will maintain and handle the user's data in a secure manner.

4. Privacy: Privacy is defined as the willingness of a user to control the disclosure of private information, encrypted data communication, and user identity management.

5. Integrity: Integrity is associated with software, hardware and data and it monitor that these can only be manipulated by authorised persons and by authorised processes.

However, for every processing on encrypted data, it should be downloaded and decrypted in client side and after processing it is further encrypted and uploaded to cloud. This obvious need for repeated decryption encryption increases the processing complexity and out-weigh the advantage of using cloud resources. Since the computation is done in the client end, the objective of utilizing large processing power at the cloud-end is defeated and the ciphertext is continuously exposed to the adversary. Hence one needs to make a mechanism in which arbitrary algorithms (which are sequences of instructions to solve a problem) can be executed over encrypted data directly on the cloud.

Homomorphic encryption is defined as a form of encryption which allows different types of computations to be carried out on cipher text and to obtain an encrypted result that when decrypted matches the result of operations performed on

the plaintext. The aim of homomorphic cryptography is to provide privacy of data in communication and storage processes, such as the ability to delegate computations to untrusted parties.

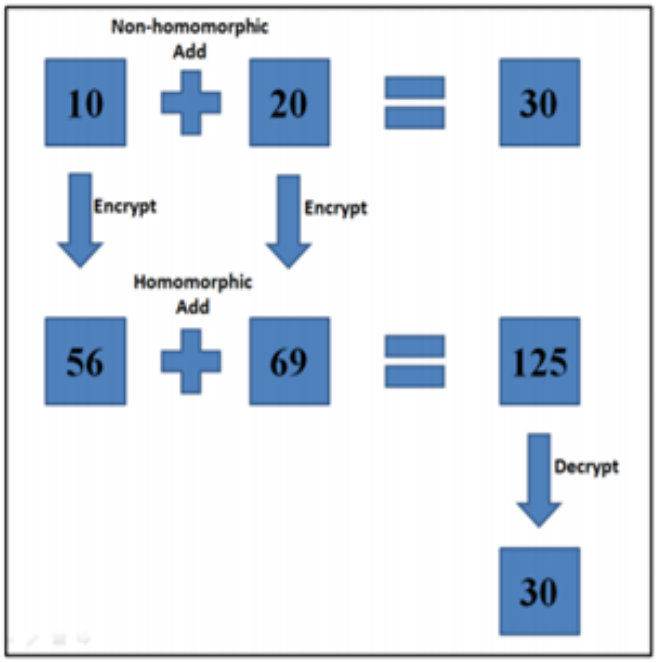

Figure 1: An example of Homomorphic Encryption

The main categories of homomorphic encryption schemes[5] are : Somewhat Homomorphic Encryption, Partially Homomorphic Encryption (PHE) and Fully Homomorphic Encryption (FHE) schemes.

-Somewhat Homomorphic Encryption schemes allow a specific class of functions to be evaluated on

ciphertexts. Usually this scheme supports an arbitrary number of one operation but only a minimum number of second operation.

-Partially Homomorphic Encryption (PHE) algorithms support either adding or multiplying encrypted ciphertexts, but not both operations at the same time.

-But in Fully Homomorphic Encryption (FHE) Scheme supporting multiplication and addition in the same time, correspond to AND and XOR in Boolean algebra.

FHE goes a further step and provides an effective primitive to perform arbitrary operations on encrypted data. With the support of FHE, cloud can evaluate any functions on encrypted data without having access to the secret key and without knowing the result.

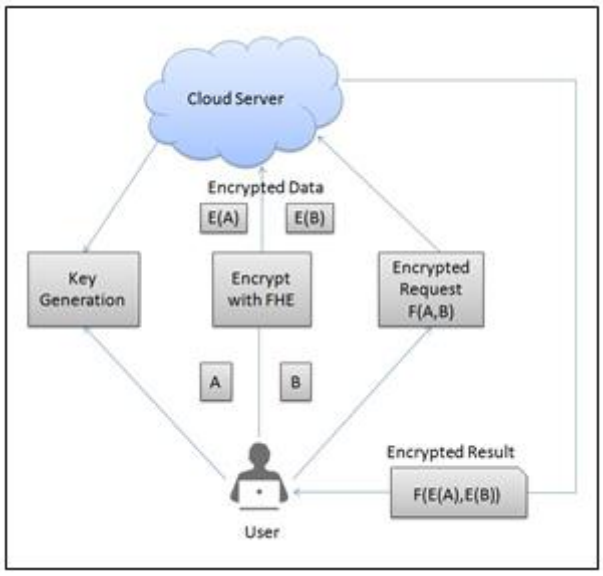

Figure 2. Applying FHE to secure cloud data

The security issues of data stored in cloud can be solved by using Fully Homomorphic Encryption (FHE) schemes. To secure it, the data should be encrypted with FHE before being sent to the cloud. First, the user login and uses the key-generation provided by the server to generate the secret key, the user is the only holder of this secret key. Then, the user encrypts the data that wants to send it to the cloud. When the user want the server to execute some computations on these encrypted data (such as add), he can send encrypted request to the cloud server. The server performs the required operations and sent the encrypted result to user. Finally the user decrypt the data with his secret key to retrieve the correct result. Figure 2 illustrates the process of using FHE to cloud computing[5].

II. LITERATURE SURVEY

In [1], the author proposes cloud security through the trusted third party mechanisms and by implementing trusted third party model of network security within the cloud architecture. It also proposes network access security model in the cloud computing so that cloud services can be protected from information access threats and services threats.

In [2], the author develops a model on cloud computing, that accepts inputs in encrypted format and then perform processing to satisfy the client query without being aware of its content, whereby the retrieved encrypted data can only be decrypted by the client who sends the request.

In [3], the author presents a description of security problem in cloud computing and use of FHE scheme to provide solution for this difficulty. The paper presents a new technique of Homomorphic Encryption that provides security to the private data and also provide mechanisms for searching or processing encrypted data. In [4, the author deals with the use of Homomorphic encryption for encrypting the user's data in cloud server and also executes required computations on this

encrypted data. The paper also analyses some of the existing Homomorphic encryption schemes like DGHV, Gen10, SDC and discuss the use of the most efficient SDC scheme, to secure cloud computing data.

In [5], the author discusses the issues involved in translating the variable definitions, instruction executions, handling of loops and terminating conditions when the algorithms handle encrypted data and encrypted controls. The paper provides for translating basic operators like bitwise, relational and arithmetic operators which are used for implementation of algorithms in any high level language like C.

In [6], the author provides implementation of different algorithms that sort data encrypted with FHE scheme that are based on Integers. The complexities of sorting algorithms on encrypted data using Insertion Sort, Bubble Sort, Odd-Even Merge sort and Bitonic Sort are analyzed.

In [7], the author presents a FHE scheme which has both relatively small key and ciphertext size. The scheme has small message expansion and key size than Gentry's original scheme. The proposal allows efficient FHE over any field of characteristic two.

In [8], this paper relates the ability to search in encrypted database to a chosen plaintext adversary and develop a technique for performing search on array encrypted with FHE. It shows methods to perform sorting based on comparison over encrypted data. It also presents several new data structures like encrypted array with encrypted index, encrypted stack and accompanying push and pop operations to realize recursive programs over encrypted data. The author proposes a two stage sorting method called as as Lazy sort with reduced recrypt operation. The author also addresses Chosen Plaintext Attack (CPA), which is feasible due to publically available encryption key.

In [9], the author presents the security issues that affect cloud computing and proposes the use of Homomorphic encryption as a remedy for dealing with these serious security concerns.

In [10], the author propose the first fully Homomorphic encryption scheme that solves an universal problem in cryptography. The paper includes discussion on a somewhat Homomorphic "boots trappable" encryption scheme that works when the function f is the scheme's own decryption function. The author shows how, through boot trappable encryption gives FHE.

In [11], the author addresses the possibility of applying the recently discovered FHE schemes to sort encrypted data. The paper shows how to choose the number of Recrypt operations that results in an almost sorted array and develops a two-stage sorting called LazySort. It also includes more advanced searching techniques such as phrase search.

III. METHODOLOGY

The main objective of this paper is to investigate how to perform arbitrary operations on FHE cloud data. The Smart-Vercauteren FHE scheme[6] has been followed to develop the operations. Scarab library is used for implementing basic FHE operations. In this section, we first provide a brief overview of FHE scheme and few words on scarab library. Further we discuss various operations that can be performed on encrypted data using scarab library.

a) Brief Overview of a FHE scheme

A public key FHE scheme $\xi : M \rightarrow C$ is described by a tuple of four polynomial time algorithms i.e.

$$\xi = (KeyGen, Enc, Dec, Eval)$$

where KeyGen, Enc, Dec denote the key generation, encryption and decryption functions of ξ respectively. Eval is the evaluation algorithm used for computation on encrypted data. This algorithm takes as input a polynomial expression P and a set of ciphetexts c ={C0,C1,...,Cn} which are needed to compute P [7]. The input output of Eval satisfies following equation:

$$Dec(Eval(P, c, pk), sk) = P(Dec(c, sk)) \quad (1)$$

In the above expression pk denotes keys that are public, like encryption key and sk denotes private or decryption key which is secret and known only to the generator of the keys. For the sake of brevity, we omit mentioning these keys in our work unless it is essential to use them. To illustrate Eval, consider polynomial expression P(X, Y) = X + Y which adds two ciphertexts X and Y and results in addition of corresponding plaintexts. According to Equation (1), for ciphertext inputs (A, B), we have

$$Dec(Eval(P, A, B)) = Dec(P(A, B)) = P(Dec(A) + Dec(B))$$

b) Smart-Vercauteren FHE scheme

The Smart-Vercauteren FHE scheme [8] consists of four algorithms:{KeyGen, Enc, Dec, ReCrypt} parametrized by three values {N, η , μ } which are typically taken as {N,2 $\sqrt{}$ N, $\sqrt{}$ N}. This scheme supports two operations :{Add, Mul}.

KeyGen():
Set the plaintext space to be P = {0, 1}.
Choose a monic irreducible polynomial
F(x) $\in$ [x] of degree N.
Repeat until p is prime.
• S (x) ← R B ∞ ,N(η /2).
• G(x) ← 1 + 2.S (x).
p ← resultant(G(x), F(x)).
D(x) ← gcd(G(x), F(x)) over Fp[x].
Let $\alpha \in$ Fp denote the unique root of D(x).
Apply the XGCD-algorithm over Q[x] to obtain
$\mathbb{Z}(x) = \sum N-1\ z\square x\square \in Z[x]$ such that
Z(x).G(x) = p mod F(x).
B ← z0 (mod 2p).
Generate s1 uniformly random integers Bi in [−p,.,p] such that there exists a subset S of s2
elements with $\sum \square \in S\ B\square$ = B over the integers.

Define ski = 1 if i ∈ S and 0 otherwise. Only s2 of the bits {ski} are set to 1.
Encrypt the bits ski = 1 under the encryption Operation to obtain ci = Enc(ski, pk).
The public key is (p,α, s1, s2, { ci, Bi }$s1$ and the private key is sk = (p, B).

Enc(M,pk):
1. Parse pk as (p,α).
2. if M ∉ {0, 1} abort.
3. R(x) ← R B∞, N(μ /2) .
4. C(x) ← M + 2.R(x).
c ← C(α) (mod p).
Output c.

Dec(c,sk):
1. Parse pk as (p, B).
2. M ← (c − ⌊c.B/p⌋).
3. Output M.

Add(c1, c2, pk):
1. Parse pk as (p,α).
2. c3 ← (c1+c2)(mod p).
3. Output c3.

Mul(c1, c2, pk):
1. Parse pk as (p,α).
2. c3 ←(c1.c2)(mod p).
3. Output c3.

c) Scarab Library

Scarab library is an implementation of a FHE scheme using large integers [13]. Hence, this work is more practical in case of applying FHE to real applications. Scarab library is comprised of sequences of specific mathematical and logical manipulation by using arithmetic, logical and bitwise operators. Helping libraries required for implementing scarab library are the GNU Multiple Precision Arithmetic Library (GMP) [14] for large integers and Fast Library for Number Theory (FLINT) [15].

d) Operations on FHE

This section shows how basic arithmetic operations like addition, multiplication and swapping can be realized by the existing Fully Homomorphic primitives present in Scarab library.

FHE add: The FHE_add function present in Scarab library performs addition of two encrypted bits using bitwise XOR operation. It discards the carry result.
Example: Let X and Y be two ciphertext integers generated by performing FHE_enc on integers x, y respectively where x=0, y =1 and pk is the public key generated by FHE_keygen function. FHE_add function will take X, Y and pk as input and will return the addition of X, Y say Z such that dec(Z)=1.Carry is discarded if any.
FHE mul: The FHE_mul function present in Scarab library performs bit-wise multiplication of ciphertexts using AND operation.
Example: x=0, y =1, X=enc(x), Y=enc(y)
FHE_mul function will take X, Y and pk as input and will return the multiplication of X, Y say Z such that dec(Z)=0.
FHE halfadd: This function performs addtion of two encrypted bits with carry out.
Example: x=1, y =1 , X=enc(x), Y=enc(y). FHE_halfadd function will take X, Y and pk as input and will return the addition of X, Y say Z such that dec(Z)=0 and carry generated say c_out where dec(c_out)=1.
FHE fulladd: This function performs addition of ciphertexts with carry in and carry out.
Example: x=1, y =1, X=enc(x), Y=enc(y).
FHE_fulladd function will take X, Y, pk and c_in as input where c_in is the carry generated from previous

FHE_full add operation. This function forwards the carry generated, for that we have to set c_in = c_out.
Output will be the addition of X, Y say Z such that dec(Z)=0 and dec(c_out)=1.
FHE swap:
FHE_swap function is implemented using basic FHE_add, FHE_suband FHE_mul operations of scarablibrary. It is uesd to swap two ciphertext values.
To swap two ciphertexts a and b, first subtraction of a and b using arithmetic based on 2's compliment is perfomed. Most Significant Bit of subtraction result is stored in β. According to bit-wise arithmetic, the value of β is 1 if the subtraction result is negative and 0 otherwise.
For swapping a and b using MSB β following steps are followed [9]:

1. β =MSB[a + (2's compliment of b)]
2. temp = β * a + (1 − β) * b
3. b = (1 − β) * a + β * b
4. a = temp

FHE SWAP circuit takes two ciphertext integers A, B as input and return another pair of ciphertext (X, Y). Here Dec(X) ⩽ Dec(Y) assuming sorting is performed in ascending order. Conditional FHE_swap function plays an important role in implementing comparison based sorting technique.
This paper proposes an approach to implement one of the comparison based sorting technique that is iterative quick sort as discussed below. To sort an array of n elements using Quick sort, an element called pivot is selected from the array. Then divide the array such that the elements smaller than the pivot are in the left partition and the elements greater than the pivot are in the right partition. Similarly the pivot element is recursively selected for the sub-arrays with smaller values and greater values to produce a sorted array.
Algorithm 1:
FHE quickSort

Input: lower index l, higher index h, encrypted arrayenc_arr, public key pk
1. i= l-1
2. for j← l to h-1 do
3. if enc_arrj<= enc_pivot
4. i=i+1
5. FHE_swap(enc_arri , enc_arrj, pk)
6. FHE_swap(enc_arri+1 , enc_arrh, pk)

In above method, initially higher index element is selected as pivot. FHE_quickSort function swap the elements to its correct position in the array. It also returns the new pivot element index to FHE_QS_Partition function given below. FHE_QS_Partition is an iterative approach to divide the array in subparts.

Algorithm 2: FHE_QS_Partition

Input: lower index l, higher index h, encrypted array enc_arr

```
1. mpz_t enc_arr[ h - l + 1 ];
2. mpz_t var = -1;
3. enc_arr[ ++var ] = l;
4. enc_arr[ ++var ] = h;
5. while ( var >= 0 )
6. h = enc_arr [ var-- ];
7. l = enc_arr [ var-- ];
8. mpz_t p = FHE_Partition(enc_arr, l, h );
9. if ( p-1 > l )
10. enc_arr [ ++var ] = l;
11. enc_arr [ ++var ] = p - 1;
12. if ( p+1 < h )
13. enc_arr [ ++var ] = p + 1;
14. enc_arr [ ++var ] = h;
```

CONCLUSION

Securing a public cloud is big concern in today's era. To preserve the privacy of data, the user must encrypt data before being sent to the cloud. Fully Homomorphic Encryption is the best solution to secure the client data in cloud computing because its schemes enable to perform arbitrary computations on encrypted data without decrypting. This paper discusses the FHE scheme and its implementation using scarab library. The work also addresses performing various operations on encrypted data using Scarab library. Our ongoing work is focused on realization of iterative quick sort approach discussed in this paper.

REFERENCES

[1] Kollati Vijaya Kumar and CH.V.T.E.V.Laxm, "Trusted Third Party in Cloud Architecture to Implement Security Issues", International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 7,July2013 ISSN: 2277 128X

[2] Shashank Bajpai and Padmija Srivastav, "A Fully Homomorphic Encryption Implementation on Cloud Computing", International Journal of Information & Computation Technology ISSN 0974-2239 Volume 4, Number 8 (2014), pp. 811-816

[3] P. Mell, T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, U. S. Department of Commerce, (2011).

[4] Sweta Agrawal, Aakanksha Choubey, "Survey of Fully Homomorphic Encryption and Its Potential to Cloud Computing Security", International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 7July, 2014

[5] Ihsan Jabbaran and Saad Najim ,"Using Fully Homomorphic Encryption to Secure Cloud Computing", Internet of Things and Cloud Computing Volume 4, Issue 2 , April 2016, Pages:13-18

[6] Ayantika Chatterjee and Indranil Sengupta. "Translating Algorithms to handle Fully Homomorphic Encrypted Data on the Cloud". IEEE Transactions on Cloud Computing. DOI 10.1109/TCC.2015.2481416.

[7] Emmadi, Nitesh, Gauravaram, Praveen, Narumanchi, Harika, & Syed, Habeeb (2015) "Updates on sorting of fully homomorphic encrypted data". In 8th IEEE International Conference on Cloud Computing, June 27 - July 2,2015, New York, USA. (In Press)

[8] Nigel Smart and Fre Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Public Key Cryptography - PKC 2010, pages 420–443. Springer LNCS 6056, 2010.

[9] Chatterjee, A. & Sengupta, I. (2015). "Searching and Sorting of Fully Homomorphic Encrypted Data on Cloud".

[10] Aderemi A. Atayero, Oluwaseyi Feyisetan, "Security Issues in Cloud Computing: The Potentials of Homomorphic Encryption", Journal of Emerging

[11] Trends in Computing and Information Sciences, (2011).

[12] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, crypto.stanford.edu/craig.

[13] A. Chatterjee, M. Kaushal, and I. S. Gupta, "Accelerating sorting of fully homomorphic encrypted data," in Proceedings of the 14th International Conference on Cryptology in India, ser. INDOCRYPT '13, 2013 (Accepted).

[14] https://github.com/hcrypt project/libScarab.

[15] T. G. et al., "GNU multiple precision arithmetic library:https://gmplib.org/."

[16] W. H. et al., "Flint: Fast library for number theory:http://www.flintlib.org/authors.html.