Sorting over encrypted data on the cloud

[1]Sneha Dalvi, [2]Rashmi Dhumal

[1]Student, Computer Department, Terna Engineering College, Maharashtra, India
[2]Associate Professor, Ramrao Adik Institute of Technology, Maharashtra, India

***Abstract*: Cloud computing provides huge computational power, but it is not utilized as only data gets stored in encrypted form on cloud, and will not allow to perform computation on it. The repeated encryption and decryption need to be performed on data even for simple computations. Gentry [13] proposed first Fully Homomorphic Encryption (FHE) which supports arbitrary operations on encrypted data. FHE allows to perform computation on encrypted data and generates encrypted results, which when decrypted gives the same result if same computation performed on plain text. This paper focuses on sorting algorithms such as Bubble Sort and Quick Sort on encrypted data and analyze their complexity.**

***Index Terms:* Cloud Computing, Encryption, Decryption, Fully Homomorphic Encryption, Scarab, Bubble sort and Quick sort**

I. INTRODUCTION

Huge amount of data is stored on cloud. User losses control over the data, once it is uploaded on cloud and need to trust on Cloud Service Provider (CSP). As the data Security is major concern, different encryption techniques are used to store the data in encrypted form. As traditional cryptographic methods such as (AES, RSA) are not capable to perform direct computation on cloud. The data migrated to the cloud is in encrypted form, in this situation common properties such as searching, storing or modifying the data cannot be performed until the data is decrypted back to client's system.

This motivates us to investigate how different operations can be defined on FHE cloud data. We proposed a method for sorting, where the entire sort operation will takes place on the server without decrypting the data and the encrypted results are obtained.FHE supports any arbitrary computations on cipher text and performs blind processing on encrypted input and produce a result in encrypted form [7]. When the encrypted result is decrypted by the user, it gets same result if same computation performed on original data.

II. LITERATURE SURVEY

The author Gentry [1] used lattice-based cryptography which supports both addition and multiplication on ciphertexts, to perform arbitrary computations on circuits. Its uses public key encryption to encrypt the decrypted circuit which produce noise in cipher text. Noise generated in cipher text while conversion is been managed with the technique known as bootstrapping. The paper [2**]** is based on FHE cryptography and allows searching an integer within a set of encrypted values. The search algorithm can be processed on an untrusted entity (e.g. a service in the cloud); it does not require decrypting the data on the server. It uses a Boolean circuit to find logarithmic depth with respect to the size of the data. In paper [3] authors have develop methodologies to handle algorithm which operates on encrypted data, they also proposed methods to design some encrypted data type such as encrypted stack, encrypted queue and encrypted linked list to perform initialization, termination operations on encrypted stack and analyses the time required for handling arithmetic, relational and conditional operations in encrypted domain. In [4], authors have designed FHE basic constructs of an algorithm in encrypted domain. They provide techniques to translate basic operators (like bitwise, arithmetic and relational operators) which are used for implementation of algorithms in any high level language like C. They have also addresses decision making, loop handling, data structures and realize the encrypted controlling variables. [5] Authors have proposed a technique called lazy sort, for sorting elements on encrypted database. It uses data structure like encrypted array with encrypted index and push and pop operations on encrypted stack which is used for sorting. It also demonstrates methods to perform operations to realize recursive program on encrypted data. In [6] the author provides implementation of different sorting algorithms on encrypted data. The complexities of sorting algorithms on encrypted data using Insertion sort, Bubble sort, Odd-Even Merge sort and Bitonic sort are analysed. In [7] the authors proposed a Fully Homomorphic Encryption scheme to secure the client data in cloud computing over integers. This paper discusses the FHE scheme and its implementation using scarab library.

III. METHODOLOGY

FHE Working:

- **Fully Homomorphic Encryption H is a set of four functions:**
  H = {Key Generation, Encryption, Decryption, Evaluation}

1. **Key generation**: client will generate pair of keys public key P_k and secret key S_k for encryption of plaintext (PT).
2. **Encryption**: Using public key P_k client encrypt the plain text PT and generate Enc (PT) and Cipher Text (CT) will be sent to the server.

3. **Evaluation**: Server has a function f which performs specific operations on Cipher Text (CT).
4. **Decryption:** Generated f (CT)) will be decrypted by client using its Secret Key (S_k) and it gets the original result.

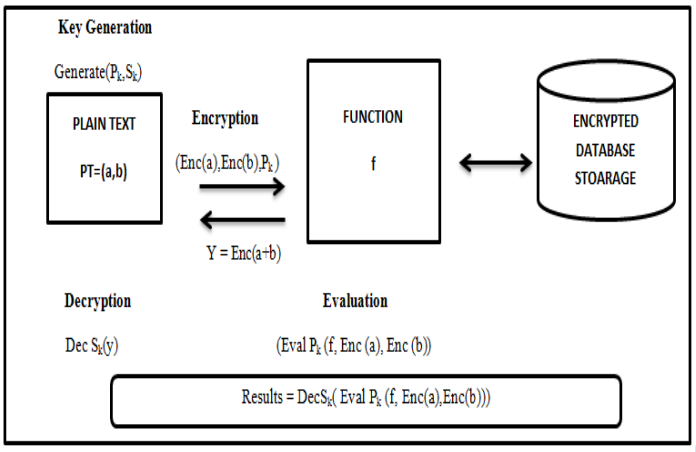

Fig.1 Fully Homomorphic Encryption Functions

Scarab Library:

Scarab library is used in implementation of FHE scheme for large integers. It uses the GNU Multiple Precision Arithmetic library (GMP) [9] for large integer and Fast Library is used for Number Theory (FLINT) [10] Scarab library consist of various functions such as FHE_add in which bit-wise (XOR) addition is carried out on cipher texts, FHE_mul in which bit-wise (AND) multiplication is done on cipher texts similarly FHE Full_add and FHE half_add are used for performing carry in and carry out operations [8]. It is possible to represent any operations using functions of scarab library.

Java Native Interface:

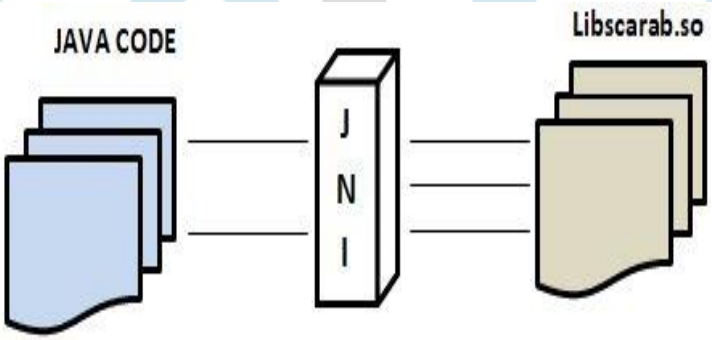

Fig.2 Representation of working of JNI

The Java Native Interface (JNI) is a programming framework that enables java code running in a Java Virtual Machine (JVM) to call and be called by native applications and libraries written in other languages such as C and C++.So, we used JNI which works as an interface between java and scarab library[12].

Sorting Techniques

Real life applications required sorting to locate items in a sorted list than unsorted. Sorting algorithms can be used in a program to sort a list. It can be done using comparison based and divide and conquer based sort.

Simple comparison based sorting:

Comparison based sorting algorithms are based on conditional swap operations that repeatedly sort the list by comparing each adjacent element from unsorted list and swaps them accordingly. This process is repeated until no swaps are needed, which indicates that the list is sorted.

Divide and conquer based sorting

Divide-and-conquer Sorting technique breaks a problem into sub problems that are similar to the original problem then solves the sub problems, and finally combines the solutions of the sub problems to solve the original problem.

- **Iterative Quick sort:**

Iterative method uses explicit repetition of function, It uses stack to store intermediate value of (l), (h) and pivot element in it.

Algorithm:

1. Start

2. Generate P_k and S_k

3. Take input list and Encrypt it using FHE_Enc with the help of P_k.

4. Create stack and Push Index of 1st Element and Length of List in it.

5. Select Pivot Element

6. Swap using CipherSort Function

7. Repeat step 4 and step 5

8. Decrypt the Encrypted List using FHE_Decrypt function with the help of S_k

9. Stop

Push is responsible for both initialization and data insertion to stack whereas Pop operation is responsible for deletion of elements from stack. At initialization, stack size is Enc (0). During data insertion the stack is increased by Enc (1) and each pop operation decreases the stack's data by Enc (1).

IV. IMPLEMENTATION

Implementation of Bubble sort:

Bubble sort uses FHE_SWAP operation to perform sorting on encrypted data. Consider, two elements 12 and 7, encrypt them using FHE_Encrypt function as Encrypted elements as (A) and (B).The inputs are then given to FHE_Mux and FHE_Swap. FHE_Swap function results swapping of elements as (B) and (A).
This process is repeated until all the elements in the list are being sorted.

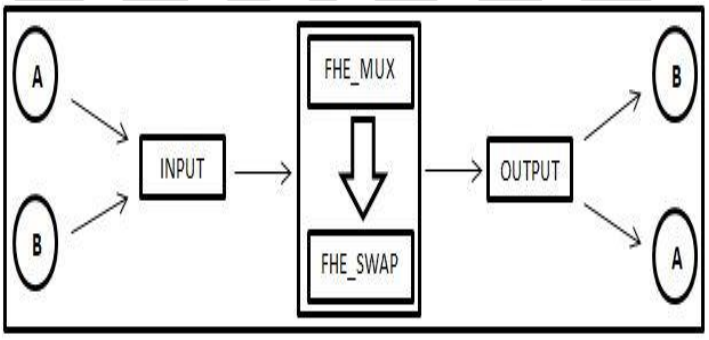

Fig.3 Working of Bubble sort in Encrypted domain.

Implementation of Iterative Quicksort:

Quicksort or partition-based sort is a fast sorting algorithm, which uses divide and conquer sorting technique.

Working:

1. Generate Public Key (P_k) and Secret Key (S_k)

2. Take the Input

34	32	43	12	11	32	22	21	32

3. Encrypt the Element using FHE_Encrypt with the help of Public Key and The Inputs become,

A	B	C	D	E	F	G	H	I

4. Create Stack and Push the Index of 1st Element and as well as the Length of Input List in it.

PUSH (9, 0)

5. Pop the top element as End and pop the Subsequent top element as Start

End = 9 and Start = 0

6. While Stack! = empty

6.1 Check (End – Start) < 2

(9 – 0)< 2

6.2 Calculate P as Start + ((End – Start) / 2)

0 + ((9 – 0) / 2) = 4

6.3 Consider l = Start and h = End – 2

l = 0 and h = 7

6.4 Swap using **CipherSort** P's Position and End – 1with the help of **isLess** Function.

Swap E and I

6.5 Working of isLess Function

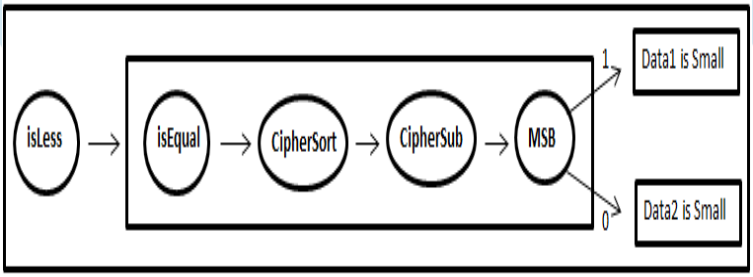

Fig.4 working of isLess Function

6.6 Swap l and h

6.7 After Swapping we get,

A	B	C	D	I	F	G	H	E

7. Consider Index = h

7.1 Now Again isLess Function which results E is smaller than A

7.2 Swap Index and End – 1

Swap E and A

7.3 After swapping we get following List as an Output for 1st Iteration.

E	B	C	D	I	F	G	H	A

8. Push Elements (P+1, End, Start, P)

9. Repeat Steps 5 to Steps 8

10. After Completing all the iterations we get Output as Follows.

E	D	H	G	I	F	B	A	C

11. Decrypt the Elements using FHE_DecryptWith the help of Secret Key and the Output becomes,

11	12	21	22	32	32	34	34	43

V. RESULTS

We have analysed the time required for sorting using Bubble Sort and Quick Sort on different elements in encrypted domain.

Table 1. Time required in Encrypted Domain

No. of Elements	Encrypted Domain	
	Bubble Sort	Quick Sort
5	235	455
7	428	533
10	602	1153
13	782	1241

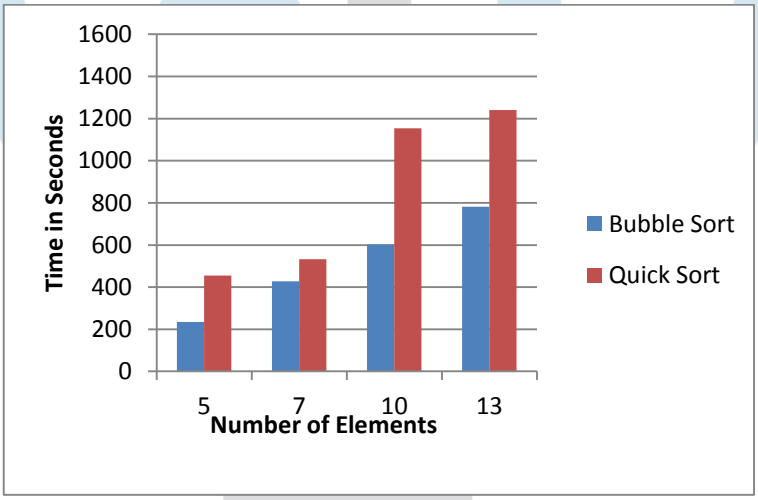

Fig.4 Graphical representation of time required for sorting both algorithm.

In general, sorting algorithm such as Quicksort and Bubble sort have complexity as o (nlogn) and o (n^2) respectively in their best case, whereas in FHE domain both Bubble sort and Quick sort has same o (n^2) complexity. The Complexity of sorting in encrypted domain is high than in plain domain but it provides high data security.

VI. EXPERMENTAL SETUP

We have performed this experiment on Ubuntu 12.4 as Virtual Operating System, using Apache – Tomcat 8.0.27 Server as web server. We used Java Servlet, JSP, JavaScript, HTML, and CSS as Font-end and MySQL Database as back-end. This experiment runs on Google Chrome, Mozilla Firefox and other browsers. This System is implemented on Red hat Open shift Cloud Platform with 64-bit Microsoft Windows Server, i3-5005U CPU, High Frequency Intel i3 Processor with Turbo up to 2.00GHz and 8GB RAM.

VII. CONCLUSION

We developed an initial effort to design sorting algorithm using function in Scarab Library which operates on encrypted data. The comparison based sorting algorithm, Bubble Sort is implemented and analyzed its complexity on different size of input. We implemented Iterative Quick Sort as it uses stack and analyzed its complexity.

We have implemented both algorithm in Java and access Scarab library using JNI. The time complexity of both algorithms is high in encrypted domain as compared to that of plain domain, but it enables to perform operations on encrypted data, so the data Security is achieved. Our experimental results shows that sorting can be performed on small scale data but its implementation for real life application is in still in research.

REFERENCES

[1] C. Gentry, Fully homomorphic encryption using ideal lattices, Proceedings of the 41st ACM Symposium on Theory of Computing, STOC 2009, pages 169-178, ACM 2009.

[2] C. Gentry and S. Halevi. Implementing Gentry's fully homomorphic encryption scheme, EURO-CRYPT 2011, Springer, K. Paterson (Ed.), 2011

[3] "Translating Algorithms to handle Fully Homomorphic Encrypted Data on the Cloud" by Ayantika Chatterjee and Indranil Sengupta IEEE Transactions on Cloud Computing. (Volume: 6, Issue: 1, Jan.-March 1 2018)DOI**:** 10.1109/TCC.2015.2481416

[4] 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.

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

[6] 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.

[7]"Operations on fully Homomorphic encrypted data on cloud " by Sanket Vyapari, Shivani Tawde, Mitali Joshi, Achyut Pratap, Rashmi Dhumal in International Journal of Technical Research and Application e-ISSN: 2320-8163, www.ijtra.comSpecial, Issue 43 (March 2017), PP. 10-14

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

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

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

[11] https://javarevisited.blogspot.com/2016/09/iterative-quicksort-example-in-java-without-recursion.html

[12]https://github.com/javanativeaccess/jna/blob/master/src/com/sun/jna/NativeLibrary.java

[13]Craig Gentry. Computing arbitrary functions of encrypted data. Commun. ACM, 53(3); 97-105, 2010

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