SORTING OVER ENCRYPTED DATA USING BUBBLE SORT

Ms. Sneha Dalvi[1], Mrs. Rashmi Dhumal[2]

[1]*Computer* Department, *Terna Engineering College ,Maharashtra, India*
[2]*AssociateProfessor, Ramrao Adik Institute of Technology, Maharashtra, India*

---***---

Abstract- *Whilst cloud computing is gaining, growing popularity in IT Industry. Organization and Business users feel convenient to use cloud storage due to better utilization of resources, low cost and easy access to data anytime, anywhere. The efficiency of sharing encrypted data with many users via public cloud storage may create security related issues. However, performing operations on encrypted Information requires extra overhead, since repeated encryption-decryption need to be performed for every single operation. Fully Homomorphic Encryption (FHE) is an Effective way to perform arbitrary operations directly on encrypted data. This paper contains Implementation of FHE operations to perform Searching and sorting algorithm by using Scarab library.*

Index Terms: **Cloud Computing, Encryption, Decryption, Fully Homomorphic Encryption, Searching and Sorting.**

1. INTRODUCTION

With the advent of cloud technology, the proliferation of cloud in various applications is enormous. Cloud computing [2] is a technology, which provides services to enterprises on demand also it provides a service based platform where large amount of data is shared over the internet. Cloud provide on-demand access to a pool of shared, configurable computing resources so that the data can be made available anywhere any time. Cloud provides a variety of services, it allows consumers and businesses to use applications without installation and access their personal file from any computer with the help of internet. [1] It also offers online data storage, infrastructure and application. It is architecture for providing computing services via internet on demand and pay per use access to a pool of shared resources for the network storage, services and applications. It is totally an internet based technology in which client data is stored and maintained in data centre of cloud providers.

Advantages of Cloud Computing

1. Reduced Cost Cloud technology is paid incrementally, saving the money of organizations.
2. Increased storage Organization can store more data than on private computing system.
3. Highly automated No longer do IT personnel need to worry about keeping software up to date.
4. Flexibility Cloud computing offers much more flexibility than past computing methods.

Cloud Computing Service Models as: Software as a Service (SaaS) is a software application model in which customers pays to access and use software functionality over network. Infrastructure as a Service (IaaS) provides virtualized computing resources over Internet. Platform as Services (PaaS) provides customers to develop, run and manage application without the complexity of building and maintain the infrastructure and platform.

To perform any operation on encrypted data one has to download and decrypt it at client side and after processing it further has to encrypt the data and has to uploaded to cloud. This obviously needs repeated decryption encryption. Direct processing on encrypted data is advantageous which can be done by using arbitrary algorithms, which is supported by Homomorphic encryption.

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 results of operations performed on the plaintext.

Advantages of Homomorphic Encryption:

1. Increased security of offshore data, while performing computing operations.
2. Guarantees confidentiality of user data
3. HM makes distributed computing secure.

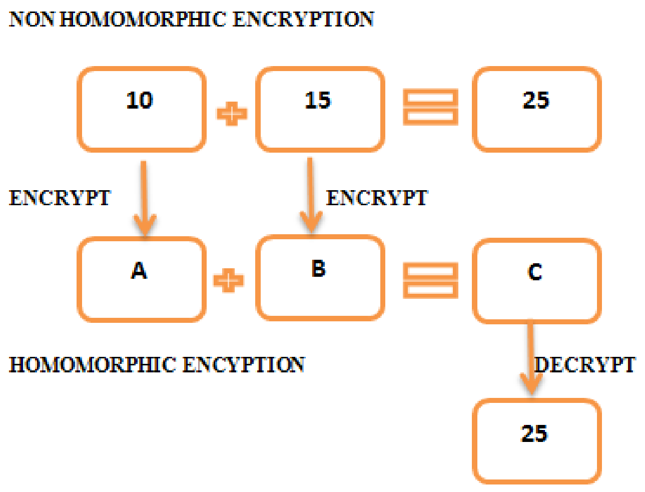


Fig -1: An Example of Homomorphic Encryption

The main categories of Homomorphic encryption Schemes [5][16] are:

1. **Somewhat Homomorphic Encryption (SHE)**
2. **Partially Homomorphic Encryption (PHE)**
3. **Fully Homomorphic Encryption (FHE)**

1. Somewhat Homomorphic Encryption Scheme allows a specific class of function to be evaluated on cipher text. Usually this scheme supports an arbitrary number of one operation but only a minimum number of second operations.

2. Partially Homomorphic Encryption (PHE) algorithm support any one of the operation either adding or multiplying encrypted cipher texts, but not both operations at a same time.

3. Fully Homomorphic Encryption (FHE) Scheme supports any arbitrary operation on encrypted data such as multiplication and addition at the same time, correspond to AND and XOR in Boolean algebra.

FHE 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.

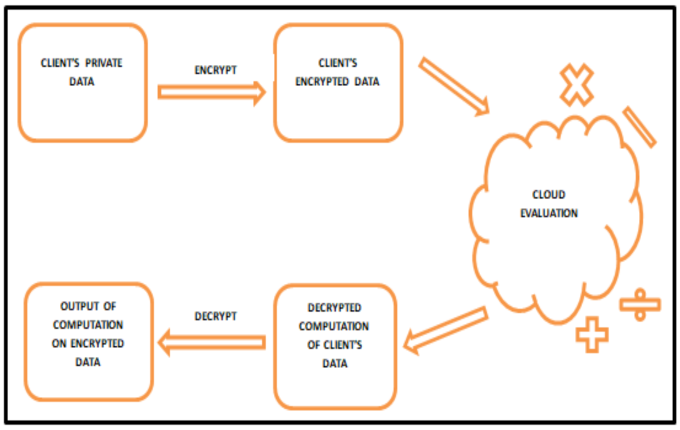

CLIENT'S PRIVATE DATA
ENCRYPT
CLIENT'S ENCRYPTED DATA
CLOUD EVALUATION
OUTPUT OF COMPUTATION ON ENCRYPTED DATA
DECRYPT
DECRYPTED COMPUTATION OF CLIENT'S DATA

Fig-2: The Process of Using FHE to Cloud Computing

To secure data on cloud, the data should be encrypted using FHE, where clients private data which is in encrypted form is stored on cloud for performing any arbitrary operations and after calculation the computed results is directly decrypted on clients pc using the clients secrete key.

First, the client will login and uses the key-generation provided by the server to generate the secret key, the client is the only owner of the secret key. Then, the client has to encrypt the data before storing data on the cloud, and if later he wants to perform any arbitrary computations on encrypted data, then the client has to send a request to server. Accordingly the server performs the required operations and sent the encrypted result to client. Finally the client can decrypt the data with the help of secret key to retrieve the result.

2. LITERATURE SURVEY

In [1], the author develops a version 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 [2], the author presents a description of security problem in cloud computing and use of FHE scheme to provide solution for this difficulty. The papers present a new technique of Homomorphic Encryption that provides security to the private data and also provide mechanisms for searching or processing encrypted data.

In [3], the author uses the Homomorphic encryption for encrypting the user's data in cloud server and executes computations on encrypted data. The paper also analyses the existing Homomorphic encryption schemes like DGHV, Gen10, and SDC and discuss the use of the most efficient SDC scheme, to secure cloud computing data.

In [4], 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.

The Authors proposed [5] a Homomorphic encryption with a feature to detect zero, detect equality, examine the value or detect overflow on cipher text is not secure if there is no restriction to limit the times of operating these functions. However with few restrictions, a HE encryption scheme can still detect zero with the key owner decrypting the cipher text and pronouncing the result if all people are allowed to detect zero on cipher text.

In [6], the author presents a FHE scheme which has both relatively small key and cipher text 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 [7], the author propose the first fully Homomorphic encryption scheme that solves a 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[8] the writer indicates the security that affects cloud computing and proposes using Homomorphic encryption as a remedy for coping with these serious security concerns.

In [9], 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, Odd-Even Merge sort, Bubble Sort, and Bitonic Sort are analysed.

In [10] the authors proposed a Fully Homomorphic Encryption scheme to secure the client data in cloud computing which enable to perform arbitrary computations on encrypted data without decrypting. This paper discusses the FHE scheme and its implementation using scarab library. It also consist of operations on encrypted data using Scarab library.

[11] In this paper, the authors Ayantika and Indranil have used FHE on arbitrary operations and benefit has obtained on execution on arbitrary algorithms on encrypted data. They also has 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 has also address decision making and loop handling and data structures to realize the controlling variables on encrypted data and proposed a method of handling termination by message passing between server and client.

The confidentiality of data in cloud can be achieved by encrypting data, but increases security related issues and diminishes the essence of cloud while performing operations on cloud data by repeated decryption-encryption process also processing on Encrypted Data requires an extra overhead but (FHE) Fully Homomrphic Encryption provides a remedy to support arbitrary operations directly on encrypted data.

3. METHODOLOGY

a) FHE Working:

- A (fully) Homomorphic encryption scheme € comprise of four algorithm:

  keyGen, Enc, Dec and Evaluate.

- For(S_k, P_k) $\longleftarrow$ KeyGen ( ), Plain Text message m with corresponding cipher text c and circuit *C* , we say that € is correct if

  Dec (S_k, Evaluate (P_k, *C*, c)) = *C* (m)

€ is Fully Homomorphic Encryption for all circuit *C*

This algorithm takes as input a polynomial expression P and a set of cipher text C = {C0, C1,...,Cn} which are needed to compute P. Eval is the evaluation algorithm used for computation on encrypted data. In the above expression P_k denotes keys that are public, like encryption key and S_k denotes private or decryption key which is secret and known only to the generator of the keys

To illustrate Eval, consider polynomial expression P(X, Y) = X+ Y which adds two cipher text X and Y and results in addition of corresponding plaintexts.

According to Equation for cipher text inputs(A,B) we have,

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

= P(Dec(A) + Dec(B))

b) Scarab Library:

Scarab library is used for implementation of a FHE scheme using large integers.[12] Scarab library is comprised of sequences of specific mathematical and logical manipulation by using arithmetic, logical and bitwise operators[15][16].Helping libraries required for implementing scarab library are the GNU[13] Multiple Precision Arithmetic Library (GMP).Fast Library for Number Theory (FLINT) it is a c library used for larger integers. [14]Whereas scarab library consist of various functions such as;FHE _add , FHE _mul, FHE _fulladd,FHE _halfadd.

c) Operations on FHE

This section focus on how basic arithmetic operations like addition, multiplication less than greater than and swapping can be done by using Fully Homomorphic primitives present in Scarab library.

FHE add:

Addition between two encrypted bits by using FHE adds module is perform by using scarab. Whereas the operands are of size 32 bits and final addition of both two operands will perform bitwise addition operation. FHE Full_add module is used for every bitwise addition.

Example: let's take two input as 'a' and 'b' given to function FHE_ENCRYPT, then bit 'a' is Enc(c1) and 'b' is Enc(c2) and pk is the public key generated by FHE_keygen function further the input is given to FHE_add and Bitwise OR is performed by FHE_add module and results are generated in 'c3' to obtain the results we have to decrypt the value of 'c3' by using FHE_DECRYPT. Carry is discarded if any.

FHE sub:

The Scarab Library performs subtraction operation between two encrypted bits by using FHE sub module. Whereas the operands are of size 32 bits and final subtraction both two operand will perform by Inverse subtraction operation. FHE full_sub module is used for every inverse subtraction.

Two input is taken as ‘a’ and ‘b’, which is given to Function FHE_ENCRYPT. Then bit ‘a’ is Inv(c1) & ‘b’ is (c2). Further the input is given to FHE_sub, and Inverse is performed by FHE_sub module and results are generated in ‘C3’ to obtain the result we have to decrypt the value of ‘C3’ by using FHE_DECRYPT.

FHE mul:

The FHE_mul function present in Scarab library performs bit-wise multiplication of cipher texts using AND operation.

Example: a=0, b=1, A=enc(a), B=enc(b) FHE_mul function will take A, B and pk as input and will return the multiplication of A, B say C such that dec(C)=0.

FHE Mux:

FHE sub is performed between two input ‘a’ and ‘b’. MSB of the subtraction result is used to find whether the result is greater than equal two relations between two operands. MSB equals to 1 indicates ‘a’ is less than equal to ‘b’ else otherwise. Its only checks if the MSB of subtraction result is Enc(0) or Enc(1). However, if two operands are exactly equal, then the MSB is also equal to Enc(0).

FHE Check:

This is module is designed to check if the two FHE data are equal. If two operands is equal then FHE subtraction of the operand result as Enc(1) and the result will display as ‘True’, and if the operand is unequal then FHE subtraction of two operand results as Enc(0) will display ‘false’.

FHE Bigger than/Smaller than:

If the inputs ‘a’ and ‘b’ are send to FHE GrtEq module and FHE Equal module. Then, the output of FHE GrtEq module is multiplied (FHE AND operation) with the inverted output of FHE Equal module and the final output of FHEGrt is produced. That implies if two operands ‘a’ and ‘b’ are not equal then the FHE Equal module will give the output as Enc(0) and hence the inversion is Enc(1). Then the final output of FHE Grt module will depend only on the greater relationship of two operands.

FHE halfadd:

This function performs addition of two encrypted bits with carry out.

Example: a=1, b=1, A=enc(a), B=enc(b). FHE_halfadd function will take A, B and pk as input and will return the addition of A, B say C such that dec(C)=0 and carry generated say c_out where dec(c_out)=1.

FHE fulladd:

This function performs addition of cipher texts with carry in and carries out.

Example: a=1, b =1, A=enc(a), B=enc(b).

FHE_fulladd function will take A, B, 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 A, B say C such that dec(C)=0 and dec(c_out)=1.

FHE swap:

FHE_swap function is implemented using basic

FHE_add, FHE_sub and FHE_mul operations of scarab library. It is used to swap two cipher text values.

To swap two cipher texts a and b, first subtraction of a and b using arithmetic based on 2's compliment is performed. 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)
4. a = temp

4. IMPLEMENTATION OF BUBBLE SORT

Let A = {A1... An} be a set where Ai = Enc(ai) for some plaintext integer ai that we need to sort (in ascending order). Fundamental difference between sorting of plaintext data and encrypted data is described in Table I. While sorting integers, (ai, aj) are swapped on the output of the comparison operation. However, in the case of encrypted inputs, both comparison and swap are combined together inside the FHE SWAP circuit. Note that the output of FHE SWAP (Ai, Aj) is encrypted and without decryption there is no way of knowing if inputs are swapped. Thus, to sort the set A, it is necessary to call FHE SWAP (Ai, Aj) for every pair with indices i, j (i < j).

Table–1: ALGORITHMS FOR SORTING ENCRYPTED AND UNENCRYPTED DATA

Swap on Unencrypted Data	Swap on Encrypted Data
1)For every pair with indices i , j with i< j	1)For every pair with indices i , j with i < j
2) Compare (a_i, a_j)	2)(X_i,X_j) FHE_SWAP(A_i,A_j)
3) Swap (a_i ,a_j) if necessary	

Working of Bubble Sort in Encrypted Domain with Example:

1. Consider the Following Numbers as an Inputs

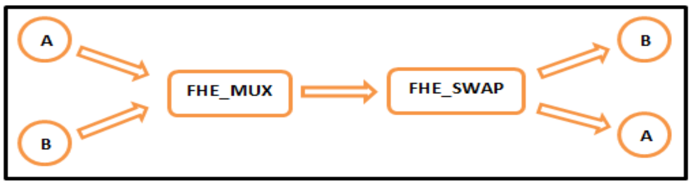
   4, 2, 54, 42, 12

2. Encrypt the Inputs using Fully Homomorphic Encryption

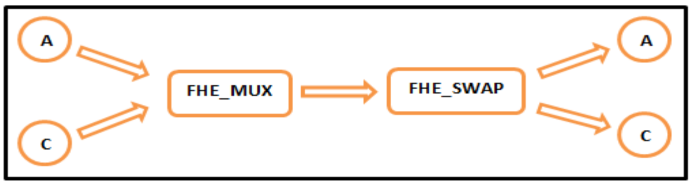
Inputs	4	2	54	42	12
Encrypted Inputs	A	B	C	D	E

3. First Iteration

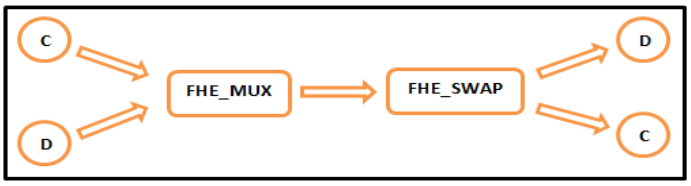
   3.1. Comparison of first two Elements using FHE_MUX and FHE_SWAP Module



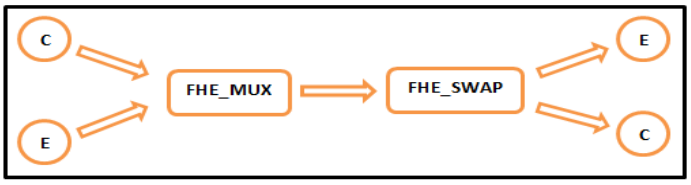
   3.2. Resulting Table



   3.3. Applying FHE_MUX and FHE_SWAP on remaining Encrypted Elements



   3.4. Hence, the Final Result of First Iteration is as follows



4. Second Iteration

1st Pass	B	A	D	E	C
2nd Pass	B	A	D	C	E
3rd Pass	B	A	C	D	E

As the Bubble Sort Compares till the last element. Though, the Final Sorted List is as follows

B	A	C	D	E

5. Decrypt the Encrypted Inputs after Sorting. So, we get

Encrypted Inputs	B	A	C	D	E
Decrypted Values	2	4	12	42	54

Table -2: SORTING ALGORITHM AND THEIR COMPLEXITIES IN PLAIN AND ENCRYPTED DOMAINS

ALGORITHM	PLAIN DOMAIN (BEST CASE)	ENCYPTED DOMAIN (ANY CASE)
Bubble Sort	O(n)	O(n^2)

For any sorting algorithm, time taken for sorting depends on the number of comparisons. For FHE sorting, comparison of cipher text integers in an array is the most expensive operation. Therefore, complexity of FHE sorting directly depends on the number of comparisons.

Table -3: AVERAGE TIME REQUIREMENT FOR COMPARISON OF BUBBLE SORT IN PLAIN ANDENCRYPTED DOMAINS

NO .OF ELEMENTS	PLAIN DOMAIN (Seconds)	ENCYPTED DOMAIN (Seconds)
6	13.25	367
9	22.19	542
12	38.28	1332
15	30.23	1550

As we can see time required for sorting elements in plain domain is less as compared to encrypted domain, but if we consider the security as a factor best results is achieved in sorting elements in encrypted domain.

5. CONCLUSION

Traditional Cryptography Strategies keep records in encrypted form, but do not permit to carry out any operations on encrypted records.

Thus (FHE) allows to permit arbitrary operations on encrypted facts, though benefit of Fully Homomorphic Encryption (FHE) can be achieved if it operates arbitrary set of rules on encrypted facts without decrypting. This paper discusses the FHE scheme and its implementation using scarab library with Bubble sort. It additionally addresses numerous operations on encrypted Facts with the use of scarab library.

6. REFERENCES

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

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

[3] 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

[4] 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

[5] S. Wang, D. Agrawal, and A. El Abbadi, Is Homomorphic Encryption the Holy Grail for Database Queries on Encrypted Data? Technical Report, Department of Computer Science, UCSB, 2012

[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. (In Press)

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

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

[9] "Updates on Sorting of Fully Homomorphic Encrypted Data" by Nitesh Emmadi, Praveen Gauravaram, Harika Narumanchi∗, Habeeb Syed International Association for cryptologic research (2015)

[10]"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.com Special, Issue 43 (March 2017), PP. 10-14

[11] 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

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

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

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

[15]C.Gentry, A Fully Homomorphic Encryption Scheme

[16]N.Smart and F. Vercauteren, Fully Homomorphic with Relative Small Key And Chipertexts Sizes