Scientific Research Publishing

Journal of Information Security, 2017, 8, 257-273
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Optimized Homomorphic Scheme on Map Reduce for Data Privacy Preserving

Konan Martin[1], Wenyong Wang[2], Brighter Agyemang[1]

[1]Department of Computer Science and Technology, University of Electronic Science and Technology of China (UESTC), Chengdu, China
[2]Department of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China
Email: martinkonan@uestc.edu.cn, wangwy@uestc.edu.cn, brighteragyemang@gmail.com

How to cite this paper: Martin, K., Wang, W.Y. and Agyemang, B. (2017) Optimized Homomorphic Scheme on Map Reduce for Data Privacy Preserving. *Journal of Information Security*, **8**, 257-273.
https://doi.org/10.4236/jis.2017.83017

Received: June 29, 2017
Accepted: July 22, 2017
Published: July 25, 2017

Abstract

Security insurance is a paramount cloud services issue in the most recent decade. Therefore, Mapreduce which is a programming framework for preparing and creating huge data collections should be optimized and securely implemented. But, conventional operations on ciphertexts were not relevant. So there is a foremost need to enable particular sorts of calculations to be done on encrypted data and additionally optimize data processing at the Map stage. Thereby schemes like (DGHV) and (Gen 10) are presented to address data privacy issue. However private encryption key (DGHV) or key's parameters (Gen 10) are sent to untrusted cloud server which compromise the information security insurance. Therefore, in this paper we propose an optimized homomorphic scheme (Op_FHE_SHCR) which speed up ciphertext (R_c) retrieval and addresses metadata dynamics and authentication through our secure Anonymiser agent. Additionally for the efficiency of our proposed scheme regarding computation cost and security investigation, we utilize a scalar homomorphic approach instead of applying a blinding probabilistic and polynomial-time calculation which is computationally expensive. Doing as such, we apply an optimized ternary search tries (TST) algorithm in our metadata repository which utilizes Merkle hash tree structure to manage metadata authentication and dynamics.

Keywords

Privacy, Mapreduce, Homomorphic Encryption, Ciphertexts Retrieval, Optimization, Authentication

1. Introduction

The rapid development in outsourcing data processing and storage by distri-

buted computing framework, and in addition to complex and huge data collection mining have extended the availability of useful data to various organizations of modern society in the exponential way. But, data privacy insurance is a principal issue in huge dataset management on cloud environment, as the dataset proprietor has not any more physical control of his dataset as per the Cloud Security Alliance (CSA) [1]. For instance, the remote body sensed data monitoring system contains delicate data like patient identity and address that can lead to harmful consequences in case of disclosure. Therefore, Mapreduce which is a computing framework that process and create substantial distributed information must be strongly secure for data privacy preserving in cloud. Security assurance issues identified with MapReduce and cloud have started to draw serious consideration. Puttaswamy *et al.* [2] presented a set of tools called Silverline that can isolate all practically encryptable information from other cloud application information to guarantee data protection. Likewise, Zhang *et al.* [3] proposed a privacy leakage upper-bound limitation based solution to deal with data privacy preserving issue by just encrypting part of available data on cloud. Roy *et al.* [4] proposed a framework named Airavat which constrains obligatory access control via differential privacy technic. Blass *et al.* [5] proposed a data privacy model named PRISM for the Map Reduce structure on cloud to perform parallel word search on encrypted data collections. Ko *et al.* [6] proposed the HybrEx Map Reduce model that processes very sensitive and private information by a private cloud, while others data can be securely processed in the public cloud. Zhang *et al.* [7] proposed a system called Sedic which partitions Map Reduce computing tasks in terms of the security labels of data. Thereby, they work on and then assign the computation without sensitive data to a public cloud. However, conventional data encryptions that are considered as the essential technique to address security issue in cloud can't be straightforwardly implemented in Mapreduce framework. Therefore a new cryptosystem called homomorphic encryption which could process encrypted data is first introduced in [8] to find an effective solution to this challenge. This fully homomorphic encryption (FHE) can compute arbitrary function on encrypted data without using secret key. However, FHE rose two major challenges regarding its implementation and computation cost [9] [10]. First, the number of available FHE schemes in the literature is very limited (few). Second, the efficiency of fully homomorphic encryption is the biggest challenge to address. Therefore, the efficiency of FHE has been the key problem since its invention, which hinders its myriad of potential applications such as private cloud computing in practical. Specially, the size of key in FHE scheme is big. Thereby, a FHE scheme based on Learning with Error (LWE) doesn't only include public key and private key but also includes some evaluation keys. For an L-Leveled FHE scheme, there are L evaluation keys. Each evaluation key is a $(n+1)^2\lceil \log q \rceil \times (n+1)$ matrix. A public key is at least $2n\log q \times (n+1)$ matrix. Clearly, these matrixes are high dimension, which not only need a lot of space to store but also affect the efficiency of computation. In 2009, an updated version done by researcher [11] is by all accounts an effective

alternative to address data privacy assurance issue which is considered by information security community as a paramount topic. Thereby, schemes like DGHV [12] and Gen 10 [13] are introduced to securely compute data through homomorphic approaches on Mapreduce environment. Unfortunately those mentioned solutions experience some basics shortcomings as far as security concerns [12] [13]. The DGHV scheme [12] uses many of the tools from Gentry's construction. But this model does not require ideal lattices. Moreover, they prove that the somewhat homomorphic component from Gentry's ideal lattice-based scheme [13] can be replaced with a very simple somewhat homomorphic scheme that uses integers. Therefore their model is conceptually simpler, but the private key should be transferred to cloud server, which is very insecure. Gentry proposed a homomorphic encryption Gen 10 scheme [13], applicable in cloud environment and conceptually simple. In this scheme the encryption function is homomorphic with respect to addition, subtraction and multiplication. The relationship between c and m is that m is the residue of c with respect to modulus p, that is $m = c \bmod p$. To retrieve the ciphertext, the Gen 10 scheme doesn't present the private key as [12]. But a random private key parameter q instead $\left[R = \left(ci - c_{index}\right) \bmod q\right]$, which seems quite more secure. Yet the parameter q is sent to the server, by using the formula $c \bmod q$, the plaintext m may leaks out. So this scheme suffers from security weakness too. The goal of this paper is to construct an efficient FHE scheme with better key size. Thereby in this work, we introduce our solution by presenting an optimized scalar homomorphic scheme (Op_FHE_SHCR) that first addresses the above mentioned data privacy protection shortcomings through efficient operations over ciphertexts without compromising the cryptosystem like the existing models [12] and [13]. Furthermore, we demonstrate how fast our proposed solution retrieves ciphertexts at Reduce stage in an optimized and secure way. At that point of this work, we present the programming and cryptographic primitives in Section 2. A short discussion on related work is presented in Section 3. The Section 4 will give details of the concrete execution and security analysis of our proposed solution. At long last we close this work in Section 5.

2. Preliminaries

2.1. Map Reduce

Map Reduce in [14] is defined as a computation framework to process and generate large datasets. In this programming environment, users specify a *Map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *Reduce* function that merges all intermediate values related with the identical intermediate key. We have such a variety of problems solving that are expressible by Mapreduce, like the problem of counting the number of occurrences of each term in a large dataset. The Mapreduce architecture can be viewed as below (**Figure 1**).

2.2. Homomorphic Encryption Scheme

♦ Homomorphism

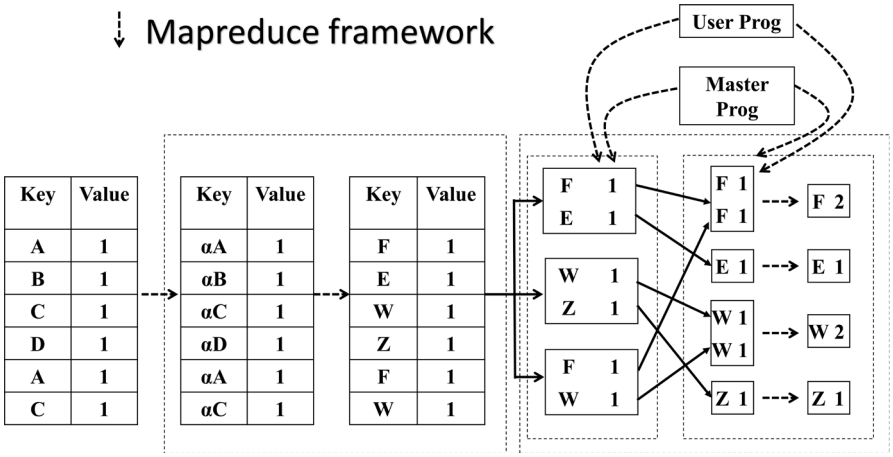

Figure 1. Mapreduce execution overview in [14].

A homomorphism between two algebras, A and B, over a field (or ring) K, is a map F: $A \to B$ such that for all k in K and x, y in A;

- $F(kx) = kF(x)$
- $F(x+y) = F(x) + F(y)$
- $F(xy) = F(x)F(y)$

If F is bijective then F is said to be an isomorphism between A and B.

♦ Homomorphic Encryption

As an ever increasing number of data is outsourced into distributed storage, frequently unencrypted, considerable trust is required in the cloud providers. The CSA records information breach as the top issue to cloud security [1]. Encrypting the data with conventional encryption addresses the issue. But in this case, the end user can't work on the encrypted data and must download them and performs the decryption locally. Therefore, there is a need to allow completely the public cloud server to perform calculations in the interest of the end users and return just the encrypted result. Thereby, the development of homomorphic encryption is a very impressive advance, incredibly amplifying the extent of computation which can be applied to process encrypted data homomorphically. Thus, the enthusiasm in the research community is justified by the various applications in the real world (like medical applications, consumer privacy in advertising, data mining, financial privacy) of this theme.

Homomorphic encryptions allow complex mathematical operations to be performed on encrypted data without compromising the encryption. This homomorphic encryption is expected to play an important part in cloud computing, allowing companies to process and store encrypted data in a public cloud and take advantage of the cloud provider's analytic services. It is first designed in 1978 by Rivest *et al.* [8] and upgraded by the researcher's community. Thereby, Craig Gentry [11] hypothetically shows the possibility of implementing this kind of encryption scheme [13]. In the same way, researcher Jaydip Sen represents homomorphic encryption clearly as a quadruple in [15]. However, Homomorphic ciphers typically do not, in and of themselves, do not provide verifiable computing and some variants are not semantically secure. Furthermore, the

poor performance is the big disadvantage of this scheme. Ciphertexts are much larger than the plaintexts, so communication requirements typically go up. The computations on these large ciphertexts are typically slower than if you just performed the computation on the plaintext itself. Because of this, in the outsourcing computation model, we typically see a requirement that encrypting inputs and decrypting outputs should be faster than performing the computation itself. Therefore there is a high need to optimize the data processing in order to reduce efficiently the computation and communication costs.

3. Related Work

Security insurance issues on Mapreduce framework have started to draw escalated consideration. In this manner data confidentiality protection issues have been widely examined and productive progress have been accomplished by the security community practitioners. We quickly audit few existing models about security protection on Mapreduce framework.

3.1. Xu Chen and Qiming Huang Scheme in [16]

The authors in [16] presents a data privacy insurance scheme on Mapreduce utilizing homomorphic encryption. It is a modified Mapreduce model, to guarantee data secrecy, and additionally processing the data in encrypted form. They pick two major prime numbers A, B, and make $P = A * B$, and afterward generate a random positive integer A, which is the private key, and B ought to likewise be confidential.

Encryption:

$$C = \left(M + A * A_r\right) \bmod \tag{1}$$

Decryption:

$$M = C \bmod A \tag{2}$$

But, to apply homomorphic encryption in this scheme, authors do few modifications on ciphertexts to allow the Reduce function to find the identical keys and afterward group them like following:

For a ciphertext $C \leftarrow \left(M + A * A_r\right) \bmod P$, authors obtain $C^* = C * B * R$, with R as random positive number.

Then, the authors compare (C^*), rather than C to retrieve similar keys. Thereby, it's obvious that their proposed solution needs an additional computation (C^*) at reduce phase which is costly in term of computation in order to get a probabilistic homomorphic cryptosystem. Review that this homomorphic cryptosystem is exceptionally expensive, therefore their model is inefficient.

3.2. FHE_SHCR Scheme in [17]

As discussed in [17] the related work [16] requires additional computation at the reduce stage, the DGHV [12] and Gen 10 [13] schemes send respectively their private key and sensitive security parameters to unreliable public cloud server (compromising the cryptosystem). Additionally, the above mentioned models

don't address the security and efficiency ciphertext retrieval issues. Therefore, authors in [17] present their contribution FHE_SHCR, which is based on schemes [13] [18] and [19] to address the privacy shortcomings of models [12] [13] and [16]. Subsequently, the main objective of this model is to securely retrieve ciphertexts at reduce stage and enhance the retrieval algorithm accuracy without getting any information about the content of intermediate searchable ciphertext to fix the security shortcomings in [12] and [13]. The FHE_SCHR scheme is an efficient candidate for homomorphic encryption to preserve data privacy in cloud by a strong hybrid encryption [17].

Our contribution:

Note that, the improvement on this paper is mainly on the optimization of the input file decomposition (map phase) and ciphertext retrieval algorithm (reduce phase) by addressing the metadata dynamics and authentication path through a logical Merkle tree repository structure (optimized space-time cost).

4. The Optimized FHE_SHCR (Op_FHE_SHCR)

As clearly proved by the research community; the homomorphic encryption can carry some operations over encrypted data effectively, but it is very expensive scheme in terms of computation and communication costs [8] [11] [13]. Therefore, we introduce a new Logical agent: Anonymiser (with its three components: Decomposition Table, Query Processing, and the Metadata Repository) at the user side under the control of the master program. Doing so, the user program can efficiently send to the master program the optimal decomposition (Splitting: Key/value) of a given input files before encrypting the data (Shuffle). Thereby, we use our optimized ternary search tries (TST) [20] in a logical Merkle tree structure to optimally address the metadata authentication and dynamics through the Metadata repository component. The architecture overview of the proposed solution can be depicted by **Figure 2**.

Thus, our optimized algorithm (Op_FHE_SHCR) through successful experiments (see section below) performs 3 times faster the original FHE_SHCR scheme [17] and effectively addresses the metadata dynamics and authentication issues through a secure and efficient metadata repository (Optimized ternary search tries to address the time space constraints). Furthermore, we speed up the ciphertext retrieval algorithm (accuracy and efficiency) at the reduce phase by using the optimal Lagrange multiplier (μ^*) as the optimum number N/e (See section below). Note that, the implementation of this Anonymiser as our Trusted Front End Database Management (TFE) is to enhance the security and speed up the data processing of our proposed solution and represents the key point of this extended work.

4.1. Optimized Metadata Authentication and Dynamics

As mentioned in the previous section, our proposed scheme further addresses metadata authentication and dynamics issue for strong data privacy protection. Therefore, we introduce a logical agent: Anonymiser in the master control pro-

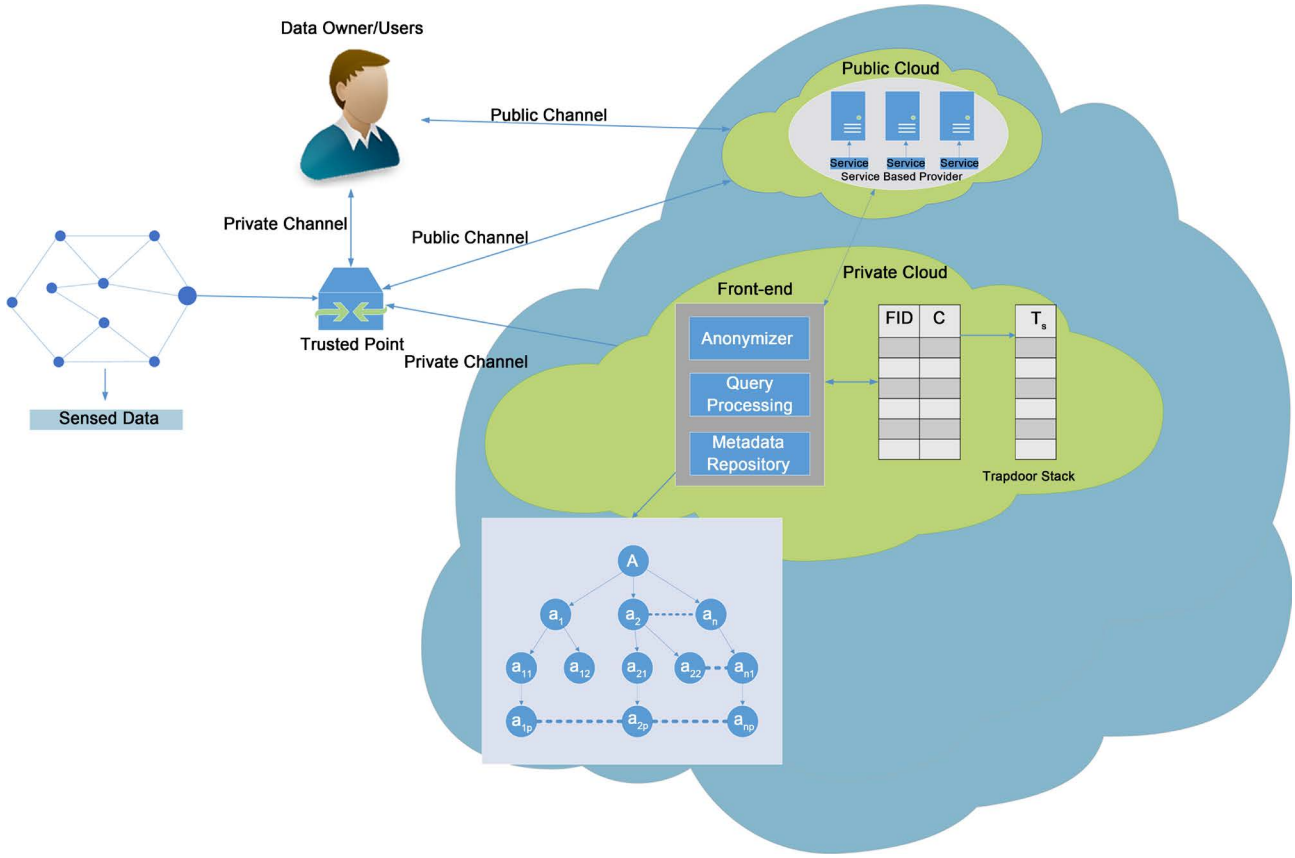

Figure 2. Architecture Overview of optimized FHE_SHCR.

gram. The Anonymiser has three pieces namely: Decomposition table, Query Processing, and Metadata Repository. Their functions can be briefly described as following:

♦ *Decomposition table*: It is responsible for defining the exact set of attributes (A) for particular input files in the optimal number.

♦ *The Query Processing*: It filters the candidate map workers queries request generated by the master program to produce anonymous query-based request on data location for processing.

♦ *The Metadata Repository*: It keeps data decomposition done by the decomposition table and forwards them to the Query Processing unit to generate new anonymous query request. For the efficiency of the proposed scheme, we use Merkle hash tree structure to deal with metadata authentication and dynamics [21]. Thereby the master program assigns a particular input files decomposition table (A) to map workers for processing logically as below:

All input files are transformed into a set of symbols (A) as a_1, a_2, a_3, a_n as depicted by the above **Figure 3**. The data matching and authentication starts from root node in a top-down manner and its dynamics process can be described as follows.

File uploading: Suppose that a data owner wants to process a file F identified by a_i (leaf node) with public cloud server whose attributes satisfy an access policy $\mathrm{AP} = [\mathrm{ap1} \cdots \mathrm{apn}]$ defined by the private cloud server. Assume that the

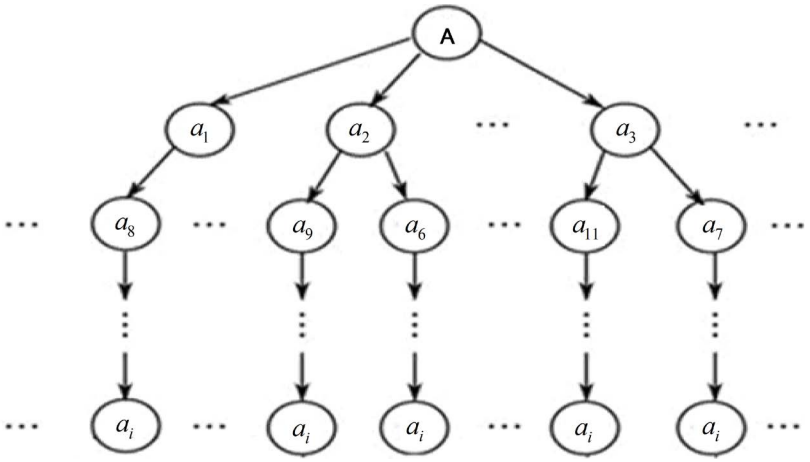

Figure 3. Efficient Symbol-based Tree Retrieval.

file F consists of a keyword set W. Then, the owner randomly chooses a symmetric key S_{ke} from the key space and encrypts the file F with S_{ke} to obtain the ciphertext C_{tf}. Subsequently, data owner runs the algorithm Encrypt(AP, S_{ke}) to obtain the ciphertext C_{ske} which is the encryption of the symmetric key S_{ke} with respect to the access policy AP. The owner uploads $\left(a_i, C_{tf}, C_{ske}\right)$ to the public cloud. Furthermore, to generate the trapdoors for keywords in W, the owner also sends (a_i, AP, W) to the private cloud. Upon receiving (a_i, AP, W), the private cloud transforms the access policy AP into a set $\left\{P_j\right\}$AP of privileges. Then, for each $w_i \in \mathrm{W}$ and $p_i \in \left\{P_j\right\}\mathrm{AP}$, it computes $t_{pi}, w_i = \mathrm{f}\left(k_{pi}, w_i\right)$, where k_{pi} is the symmetric key for each p_i. Finally, the private cloud sends $\left(a_i, \left\{t_{pi}, w_i\right\}\right)$ to the public cloud as well.

To enhance the searching efficiency, a symbol-based tree is utilized to build an index stored in private cloud (metadata repository). More precisely, divide the output of one-way function f into l parts and predefine a set $A = \left\{a_1 \cdots a_i\right\}$ consisting of all the possible values in each part (an example of such tree can be shown in **Figure 3**). Initially, the index based on symbol-based tree has only a root node (denoted as *node* 0) which consists of $\emptyset$ (an empty set). The search process in a symbol based tree is a depth first search. The tree can be updated and searched as follows.

Update: Assume the data owner wants to outsource a file F identified by a_i with keyword set W, the public cloud will receive $\left(a_i, C_{tf}, C_{ske}\right)$ and $\left(a_i, \left\{t_{pi}, w_i\right\}\right)$ where p_i are the corresponding privileges and w_i are keywords in W from the data owner and private cloud respectively. Then, for each $\left(t_{pi}, w_i\right)$ the public cloud will add it into the trie index as the following steps.

i) Step 1: Public cloud parses $\left(t_{pi}, w_i\right)$ as a sequence of symbols $a_{i1}a_{i2}\cdots a_{il}$.

ii) Step 2: Public cloud starts with the root node of tree: it scans all the children of the root node and checks whether there exists some child node 1 such that the symbol contained in node 1 equals a_{il}. This action is performed in a top-down manner. In general, assuming that the subsequence of symbols $a_{i1}a_{i2}\cdots a_{ij-1}$ has been matched and the current node is $node_{j-1}$, the public cloud will examine all the children of $node_{j-1}$ and attempt to find out some

node, for example $node_j$ such that the symbol contained in $node_j$ equals a_{ij}. If such node exists, the current node is set as $node_j$ and a_{ij+1} is the next matching object, otherwise jump to step 3.

iii) Step 3: Assume that current node $node_j$ has no children to match the symbol a_{ij+1}, the public cloud will build nodes $node_{j+1},\cdots,node_{jl}$ for all the rest of the symbols (*i.e.* $a_{ij+1},a_{ij+2},\cdots,a_{il}$) respectively and link them as a node list appended with $node_j$. Finally, add another node identified by a_i as the leaf node appended with $node_{jl}$.

Search: Assuming that the legitimate user wants to search outsourced files with keyword w and privileges $\{p_i\}$, the public cloud will receive (t_{pi},w_i) from the private cloud. For each (t_{pi},w_i) the public cloud will perform actions similar to the three steps described above. One exception is that if matching fails (*i.e.* the current node has no children which can match the symbol), the search for (t_{pi},w_i) is aborted. Otherwise, get the corresponding (C_{tf},C_{ske})) through the identifier a_i in the leaf node.

So to address Merkle tree traversal problem, our scheme uses some tools from the efficient algorithm in [22] to overcome the space-time issue. Furthermore, to optimize the time space constraints in the Merkle hash tree traversal process, we designed an optimized ternary search tries (TST) [20] which is a sorting algorithm that blends quicksort and radix sort. Thereby, it is competitive with the best known C sort codes. It is faster than the traditional hashing and other commonly used search methods as shown below (**Figure 4**):

The TST is space efficient, but increases with the number of strings (N). Therefore the traversal problem is how to calculate efficiently the authentication path for all leaves one after another starting with the first leaf up to the last leaf, for minimum amount of space-time cost. Hence, it implies to analyze an optimal distribution of singleton attribute (a_i) to enhance the efficiency of the proposed solution; that is to find the optimal number of strings or attributes (N) to populate the tree. In this work we use the Karush-Kuhn-Tucker (KKT) condition of constrained optimization problem [24] to solve the above mentioned issue in the section below. Practically, we design our solution using some mathematical tools

	character accesses (typical case)			
implementation	search hit	search miss	insert	space (references)
red-black BST	$L + c\lg^2 N$	$c\lg^2 N$	$c\lg^2 N$	4N
hashing (linear probing)	L	L	L	4N to 16N
R-way trie	L	$\log_R N$	L	$(R+1)N$
TST	$L + \ln N$	$\ln N$	$L + \ln N$	$4N$

Figure 4. String symbol table implementation cost summary from [23].

from the scheme in [25] to find the minimum number of singleton quasi-identifier that gives the optimal security level for the proposed traversal algorithm efficiency.

Let $n_1, n_2, n_3, \cdots, n_n$ be the number of values along the different columns of the Anonymiser decomposition table to populate the distinct levels set $L_1, L_2, L_3, \cdots, L_n$ of the Merkle tree respectively. The total number of distinct values taken by the different levels set ($n_1, n_2, n_3, \cdots, n_n$) is N. We assume that each column set (C_i) of the decomposition table takes (n_i) different values with the probability (p_i) respectively, where $\sum_{i=1}^{N} p_i = 1$.

Therefore the probability for the (i^{th}) element to be a singleton in the universal decomposition table by selecting one of the (n) choices (entries) is $np_i(1-p_i)^{n-1}$.

Let the variable x_i be the indicator representing whether (i^{th}) element is a singleton, then its expectation is calculated as below:

$$E[x_i] = P[x_i = 1] = np_i(1-p_i)^{n-1} \overset{\text{def}}{=} np_i(\mathrm{e})^{-np_i} \tag{3}$$

Let $X = \sum_{i=1}^{N} x_i$, be the variable that counts the number of singleton; its expectation is given by:

$$E[X] = \sum_{i=1}^{N} E[x_i] = \sum_{i=1}^{N} np_i(\mathrm{e})^{-np_i} \tag{4}$$

We aim to find the smallest number of singleton to populate efficiently the Merkle tree in the metadata repository.

It implies to minimize $\sum_{i=1}^{N} np_i(\mathrm{e})^{-np_i}$;

Subject to $\sum_{i=1}^{N} np_i = n$, with $0 \le np_i$, $\forall 1 \le i \le N$.

Therefore we get the optimized number of singleton by rewriting the above distribution as a constrained optimization problem [24]. Doing so, we find the dual solution of the primal problem which is fast and reduces the space-time costs as follows:

$$(\mathrm{P}) \begin{cases} \min f(x_i) = \sum_{i=1}^{N} x_i(\mathrm{e})^{-x_i} \\ st \;\; g(x_i) = x_i > 0; \\ \quad \forall 1 \le i \le N \end{cases} \tag{5}$$

KKT condition: Primal variable: x_i; Lagrange Multiplier: λ, μ

Let us considering an optimization problem of forms Minimize $f(x)$ Subject to $h_i(x) = 0; g_i(x) \le 0$;

With $i = 1, \cdots, m$

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x^*) + \sum_{j=1}^{m} \mu_i \nabla g_j(x^*) + T_X(x^*) \ni 0 \\ h_i(x^*) = 0, g_j(x^*) \le 0, \;\; \text{for } i = 1, \cdots, m, \; j = 1, \cdots, r \\ \mu_j \ge 0, \lambda_i \ge 0 \end{cases} \tag{6}$$

Using the Lagrange multiplier and the duality theorem, the solution of the problem (P) is determined as following:

$$L(f_{x_i}, \mu) = f(x_i) - \mu^T g(x),$$

Then $q(\mu) \overset{\text{def}}{=} inf_{x_i \in X} L(x_i,); q(\mu) = inf_{g(x_i)>0} \left\{ \sum_{i=1}^{N} x_i(\mathrm{e})^{-x_i} - \mu^T(x_i) \right\}$

$q(\mu)$ is a smooth function, then its gradient equals to zero at the optimal number (x^*):

$$\nabla_{x^*} f\left(x_i^*\right) + \mu^T \nabla x_i^* = 0 \tag{7}$$

$$\nabla_{x^*} f\left(x_i^*\right) + \mu^T \nabla x_i^* = \begin{bmatrix} (1-x_1)\mathrm{e}^{-x_1} \\ (1-x_2)\mathrm{e}^{-x_2} \\ (1-x_3)\mathrm{e}^{-x_3} \\ \vdots \\ (1-x_N)\mathrm{e}^{-x_N} \end{bmatrix} - \mu = 0 \tag{8}$$

Then, $\left(1-x_i^*\right)\mathrm{e}^{-x_i^*} - \mu = 0$, $\forall\, 1 \le i \le N$. Therefore $\mu = \left(1-x_i^*\right)\mathrm{e}^{-x_i^*}$

Case1: $\mu = 0$

$\left(1-x_i^*\right)\mathrm{e}^{-x_i^*} = 0 \xrightarrow{\text{yields}} \left(1-x_i^*\right) = 0$, then $x_i^* = 1$. So we have:

$\min f\left(x_i\right) = \sum_{i=1}^{N} x_i \left(\mathrm{e}\right)^{-x_i} \le \sum_{i=1}^{N} \mathrm{e}^{-1} = N/e$

Case 2: $\mu > 0$

$\left(1-x_i^*\right)\mathrm{e}^{-x_i^*} > 0 \xrightarrow{\text{yields}} \left(1-x_i^*\right) > 0$, then $0 < x_i^* < 1$;

$$\begin{cases} \text{If } x_i^* = 1, \text{ we get case 1,} \\ \text{otherwise contradiction to the KKT conditions.} \end{cases}$$

Finally the optimal number of singleton quasi-identifiers for a decomposition table of (n) entries with maximum total number of distinct values (N) is N/e. Using this optimum number (N/e) to populate the ternary search tree, we improve the performance of the Merkle tree traversal algorithm by addressing the space-time cost issue. Thereby, the overall ciphertext retrieval time at reduce phase for our optimized algorithm Op_FHE_SCHR is almost three times less than the exiting one FHE_SCHR refer to Figure 5.

4.2. FHE_SCHR Efficiency and Implementation Analysis

Regardless of the advances in remote sensor network (WSN) to controls systems into cloud, there are still enormous challenges in term of security insurance over

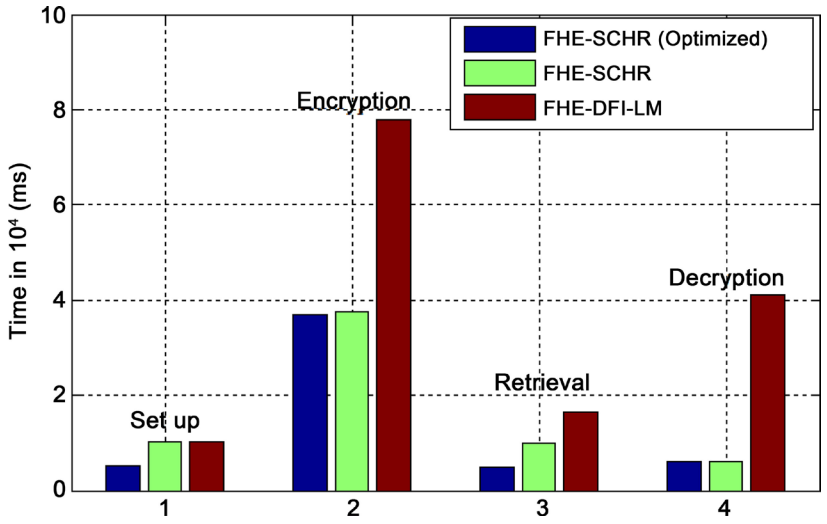

Figure 5. Overview performance comparison of cryptosystems.

outsourced data processing and storage [26]. Therefore, we work for experiment purposes over trained sensed dataset for cancer pattern monitoring project. So our main goal in this paper is to securely optimize the map phase (input files decomposition) and ciphertexts retrieval (reduce phase) process. Thereby, we implement an optimized scalar homomorphic based Mapreduce scheme Op_FHE_SHCR, which contains four algorithms: *KeyGen* (k_e), *Encrypt* (1^σ , k_e , *s.m*) *Decrypt*(*c*) *and Retrieval*(*c*). Note that these four algorithms are quite similar like those in [21], with modified mapping and hashing algorithm like below (**Algorithm 1**).

Pseudo code:

This algorithm initializes the selected feature subset (splitting the input file into subsets) denoted by F_S, with the empty set. A candidate feature subset, denoted by F_C, will be produced by adding a feature, denoted by f_d, with $d \in [1, D]$, to the selected feature subset, and the hashing index IWI_{F_C} will be computed by **Algorithm 2**. Then, the algorithm removes this feature and adds

Algorithm 1: Splitting.

Input: < F_i input file (key/value) >

Output: < F_S searchable ciphertext (key/value) >

Initialize: $F_S = \varnothing$;

repeat $F_t = F_i$;

repeat

produce the candidate feature set $F_C = f_c \cup F_S, f_c \in F_t, c \in [1, D]$;

remove f_c from F_t;

compute and record F_c by **Algorithm 2**;

until $F_t == \varnothing$

$F_S := f_d F_S$, where $f_d = argNMax\left(IWI_{F_C}\right)$;

remove f_d from F_O;

until $F_O == \varnothing$ or meeting the threshold

Algorithm 2: Hashing.

Input: < searchable ciphertext (key/value) >

Output: < splitting searchable ciphertext (key/value) >

```
1. #include <stdio.h>
2. #include <string>
3. using namespace std;
4. class MyHash
5. {private: string ** table; int * maxind;
6. long width;
7. long depth;
8. protected: bool insert(long value,string s);
9. void printone(long v);
10. public:MyHash(long N, long Depth);
11. long apply(string s);
12. void printall ();
```

another feature to generate a new candidate. That is, the new feature in the chosen candidate will be added to the selected feature subset. Thus, this algorithm iteratively adds one feature (or the fixed number of features if the floating strategy has been used) to increase the selected feature subset until the threshold is met. It should be pointed out that the main difference between the proposed algorithm and the existing ones in the literature is that our algorithm produces high correlated data subsets based on the hashing index value. Therefore the ciphertext retrieval process at the reduce stage will be more efficient in terms of speed.

The design of our OP_FHE_SCHR cryptosystem is done using the HElib-master-2015.03 library in Dev C++ IDE. We utilize the WDBC Test training dataset for cancer management project. Our security algorithm is implemented in four steps using Gentry cryptosystem [13] [18] and [19].

The efficient analysis of the candidate solution is proved by its experiments results that are compared with the existing blinding fully homomorphic FHE_DFI_LM algorithm, previous FHE_SCHR, and our new optimized Op_FHE_SCHR algorithm. Recall that, the improvement on this paper is mainly on the optimization of ciphertext retrieval time and metadata dynamics and authentication path in the logical Merkle tree repository (optimized space-time cost).

Table 1 and Figure 5 show the average performance of our proposed solution (Op_FHE_SCHR) in comparison to related works (FHE_DFI_LM & FHE_SCHR). Recall that, the experimental requirements are to optimize the outsourced data processing at the map stage and prevent intermediate data disclosure at the reduce phase in order to reinforce data privacy on Mapreduce framework. Therefore our Op_FHE_SCHR processes almost the data at Map stage (setup phase refer to Table 1 and Figure 5) three time less (5932 ms) than FHE_DFI_LM (13078 ms) and two time less than our previous work FHE_SCHR (11684 ms). This result is obtained by an optimized map workers selection using the optimal number N/e, for a given decomposition table of (n) entries at the splitting step. Thereby for a given dataset N, our algorithm calculates in advance the exact optimal number of subsets (feature selection) and map workers to speed up the splitting and data allocation process at the Map stage. Furthermore, each element (feature) of a subset is selected by an efficient feature selection algorithm (refer **Algorithm 1**).

Table 1. Average performance Comparison.

ALGORITHM	AVERAGE PERFORMANCE			
	Setup time (ms)	Encryption time (ms)	Ciphertext Retrieval time (ms)	Decryption time (ms)
FHE_SCHR	11,684	37,419	37,419	7994
FHE_DFI_LM	13,078	77,507	77,507	41,085
Op_FHE_SCHR	5932	37,120	12,476	7990

Based on the result of effective experiments directed, it is unmistakably certain that the proposed optimized scheme Op_FHE_SCHR speedups the setup (input files decomposition) and ciphertext retrieval time without compromising the cryptosystem. Thereby, the graph 5 shows that the proposed alternative is more efficient regarding the ciphertext retrieval and computation cost reduction. Note that homomorphic cryptosystem is extremely expensive [15]. Therefore, our solution contribution is a reliable alternative to minimize the overall computation and communication costs.

4.3. FHE_SCHR Security Analysis

Two security requirements are to be achieved, that is data confidentiality and integrity. Therefore, our security scheme is based on a hybrid encryption design. Since the file is encrypted with a hybrid encryption as $\left(C_{tf}, C_{ske}\right)$, the adversary should first decrypt C_{ske}. However, such a session key is protected by the ABE scheme. Thus, data confidentiality can be reduced to the confidentiality security of ABE. Moreover, the keywords which are needed to be protected against the public cloud are encrypted with a one way trapdoor function. The underlying ABE scheme for this work is known to be semantically secured. This private key is under the security proof of the ABE scheme and reduced to the bilinear decisional Diffie-Hellman (BDDH) problem [27]. The bilinear decisional Diffie-Hellman (BDDH) problem is such that given g, $g^x, g^y, g^z \in \mathrm{G}$ for unknown random values x, y, $z \in \mathrm{R}$ Z_P, and $T \in \mathrm{R}$, G_T, prove the distinguish ability of $e(g,g)^{xyz}$ from any random number in the target group is very hard to decide if $T = e(g,g)^{xyz}$. We say that the (t, ϵ)-BDDH assumption holds in G, if no t-time algorithm has the probability at least $\frac{1}{2}+\epsilon$ in solving the BDDH problem for non-negligible ϵ.

Recall that this paper is based on [17], and therefore we use the same cryptosystem to fix the data privacy preserving challenge. Furthermore, to enhance the security of our proposed solution in terms of data authentication paths and dynamics, we adopt the Merkle's hash tree to store the metadata decomposition $\left(a_i \in A = \{a_1, a_2, \cdots, a_n\}\right)$; which is freely secure from any number theoretic conjectures [21]. Indeed the Merkle's hash trees are very useful because they allow efficient and secure verification of the contents of large data structures. This security lies mainly on two properties of hash functions:

i) Pre-image resistance: that is, given a hash value (h), it is difficult to find a message m such that h = hash (m).

ii) Collision resistance: that is, finding two messages m1 ≠ m2 such that hash (m1) = hash (m2) is difficult.

This data structure is a complete binary tree with an n-bit hash value associated with each node. Each internal node value is the result of a hash of the node values of its children. Merkle trees are designed so that a leaf value h (a_i) can be verified with respect to a publicly known root (A) value given the authentication path of the respective leaf as:

$A = h(ha1 \| ha2 \| ha3 \| \cdots \| han)$; refer to the **Figure 3**. So an attacker holding a hash value (hai) in order to reconstruct (A); needs some additional values called Auxiliary Authentication Information (AAI) which are kept secret by the metadata repository administrator under the supervision of the Anonymiser Query system. Therefore it is very hard for the public cloud server or outside attacker to reconstruct the input files by the decomposition table (A).

To summarize the security analysis, we can say by implementing a secure front end database management agent (Anonymiser) on top of FHE_SHCR security mechanism [17] that the data privacy insurance has been greatly reinforced in our proposed solution (Optimized FHE_SHCR).

5. Conclusion

In this paper, the requirements are to optimize the outsourced data processing at the map stage and prevent intermediate data disclosure at the reduce phase in order to reinforce data privacy on Mapreduce framework. Therefore, we implement a secure Front End Database Management agent: the Anonymiser with its three components (Decomposition table, Query Processing, and Metadata Repository.) to enhance the data security mechanism of our proposed solution. The cryptosystem tool is a scalar homomorphic encryption that performs some sorts of calculations over encrypted data in more secure and optimized design. The optimized cryptosystem Op_FHE_SCHR is by the experiments results an efficient candidate for the communication and computation costs reduction. Practically, it takes as input files an optimized decomposition table (for map workers), and improves the speed and accuracy of ciphertext retrieval process (for reduce workers) on Mape Reduce environment. Furthermore, we address the metadata dynamics and time space cost constraints for the traversal of Merkle tree structure in our metadata repository by applying an optimized ternary search tries (TST) algorithm.

Acknowledgements

This work has been supported by MoE-CMCC (Ministry of Education of China-China Mobile Communications Corporation) Joint Science Fund under grant MCM20130661.

References

[1] Cloud Security Alliance (2010) Top Threats to Cloud Computing Version 1.0. http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf

[2] Puttaswamy, K.P.N., Kruegel, C. and Zhao, B.Y. (2011) Silverline: Toward Data Confidentiality in Storage-Intensive Cloud Applications. *Proceedings SoCC'11 the 2nd ACM Symposium on Cloud Computing*. https://doi.org/10.1145/2038916.2038926

[3] Zhang, X., Liu, C., Surya, N., Suraj, P. and Chen, J. (2013) A Privacy Leakage Upper-Bound Constraint based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud. *IEEE Transactions on Parallel and Distributed Systems*, **24**, 1192-1202. https://doi.org/10.1109/TPDS.2012.238

[4] Roy, I., Setty, S.T.V., Kilzer, A., Shmatikov, V. and Witchel, E. (2010) Airavat: Security and Privacy for MapReduce. *Proceedings NSDI'*10 7*th USENIX Conference on Networked Systems Design and Implementation*, 297-312.

[5] Blass, E.-O., Pietro, R.D., Molva, R. and Önen, M. (2012) Prism-Privacy Preserving Search in MapReduce. *Proceedings PETS'*12 *the* 12*th International Conference on Privacy Enhancing Technologies*, 180-200. https://doi.org/10.1007/978-3-642-31680-7_10

[6] Ko, S.Y., Jeon, K. and Morales, R. (2011) The Hybrex Model for Confidentiality and Privacy in Cloud Computing. *Proceedings HotCloud'*11 *the* 3*rd USENIX Conference on Hot Topics in Cloud Computing Article* 8.

[7] Zhang, K., Zhou, X., Chen, Y., Wang, X. and Ruan, Y. (2011) Sedic: Privacy Aware Data Intensive Computing on Hybrid Clouds. *Proceedings CCS'*11 18*th ACM Conference on Computer and Communications Security*, 515-526. https://doi.org/10.1145/2046707.2046767

[8] Rivest, R.L., Adleman, L. and Deaouzos, M.L. (1978) On Data Banks and Privacy Homomorphism. In: DeMillo, R.A., Ed., *Foundations of Secure Computation*, Academic Press, New York, 169-179.

[9] Craig, G., Shai, H. and Nigel, S. (2012) Fully Homomorphic Encryption with Polylog Overhead. In: Pointcheval, D. and Johansson, T., Eds., *Advances in Cryptology*, Springer, Heidelberg, 465-482.

[10] Zvika, B., Craig, G. and Shai, H. (2013) Packed Ciphertexts in Lwe-Based Homomorphic Encryption. In: Kurosawa, K. and Hanaoka, G., Eds., *Public-Key Cryptography-Pkc* 2013, Springer, Heidelberg, 1-13.

[11] Gentry, C. (2009) Fully Homomorphic Encryption Using Ideal Lattices. In: 41*st Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 169-178. https://doi.org/10.1145/1536414.1536440

[12] Dijk, M.V., Gentry, C., Halevi S. and Vaikuntanathan, V. (2010) Fully Homomorphic Encryption over the Integers. In: *Proceedings Advances in Cryptography-Eurocrypt*, Springer-Verlag, Berlin, 24-43.

[13] Gentry, C. (2010) Computing Arbitrary Functions of Encrypted Data. *Communications of the ACM*, **53**, 97-105. https://doi.org/10.1145/1666420.1666444

[14] Dean, J. and Ghemawat, S. (2008) MapReduce: Simplified Data Processing on Large Clusters. *Communication of ACM*, **51**, 107-113. https://doi.org/10.1145/1327452.1327492

[15] Jaydip, S. (2013) Homomorphic Encryption—Theory and Application. In: Jaydip, S., Ed., *Theory and Practice of Cryptography and Network Security Protocols and Technologies*, Tech. https://doi.org/10.5772/56687

[16] Chen, X. and Huang, Q. (2013) The Data Protection of MapReduce Using Homomorphic Encryption. *Proceedings ICSESS the* 4*th IEEE International Conference on Software Engineering and Service Science*, Beijing, 23-25 May 2013, 419-421.

[17] Martin, K., Wenyong, W. and Brighter, A. (2016) Efran (O) Efficient Scalar Homomorphic Scheme on MapReduce for Data Privacy Preserving. *Proceedings CSCloud the* 3*rd IEEE International Conference on Cyber Security and Cloud Computing*, Beijing, 25-27 June 2016, 66-74.

[18] Gentry, C. and Halevi, S. (2011) Implementing Gentry's Fully-Homomorphic Encryption Scheme. In: *Proceedings Advances in Cryptography-EUROCRYPT*, Springer-Verlag, Berlin, Vol. 6632, 129-148.

[19] Gentry, C. (2010) Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness. In: *Proceedings Advances in Cryptology-CRYPTO*, Springer-Verlag, Ber-

lin, Vol. 6223, 116-137. https://doi.org/10.1007/978-3-642-14623-7_7

[20] Jon, L.B. and Robert, S. (1997) Fast Algorithms for Sorting and Searching Strings. *Proceedings ACM-SIAM the Eight Annual Symposium on Discrete Algorithms*, New Orleans, 5-7 January 1997, 360-369.

[21] Merkle, R.C. (1989) A Certified Digital Signature. *Proceedings CRYPTO Advances in Cryptography*, Vol. 435 of LNCS, 218-238.

[22] Markus, K., Willi, M. and Carlo, U.N. (2014) A Space- and Time-Efficient Implementation of the Merkle Tree Traversal Algorithm.

[23] Robert, S. and Kevin, W. (2002) Algorithms. 4th Edition. http://algs4.cs.princeton.edu/home/

[24] Bertsekas, D.P., Nedic, A. and Ozdaglar, E.A. (2003) Convex Analysis and Optimization. Athena Scientific, Cambridge, 560 p.

[25] Yu, Y. (2010) Privacy Protection in Secure Database Service. *Proceedings the Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, Wuhan, 218-222.

[26] Huang, X. and Du, X. (2013) Efficiently Secure Data Privacy on Hybrid Cloud. *Proceedings ICC IEEE International Conference on Communication*, 9-13 June 2013, Budapest, 1936-1940. https://doi.org/10.1109/ICC.2013.6654806

[27] Sahai, A. and Waters, B. (2005) Fuzzy Identity-Based Encryption. In: Cramer, R., Ed., *Eurocrypt*, Springer, Heidelberg, LNCS, Vol. 3494, 457-473. https://doi.org/10.1007/11426639_27