Research Article

Integrated Security and Privacy Framework for Big Data in Hadoop MapReduce Framework

Sirisha N[1], K.V.D. Kiran[2]

[1,2]Department of Computer Science and Engineering,
Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.
[1]Department of Computer Science and Engineering,
MLR Institute of Technology, Dundigal, Hyderabad, India

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract
Public cloud infrastructure is widely used by enterprises to store and process big data. Cloud and its distributed computing phenomena not only provides scalable, available and affordable solution for storage and compute services but also raises security concerns. Many security solutions that came into existence encrypt data and allow accessing plaintext for data analytics in the confines of secure hardware. However, the fact remains that the large volumes of data is processed in distributed environment involving hundreds of commodity machines. There exist numerous communications between machines in MapReduce computing model. In the process, compromised MapReduce machines or functions are vulnerable to query based inference attacks on big data that lead to leakage of sensitive information. The main focus of this paper is to overcome the problem aforementioned. Towards this end, a methodology is proposed with an underlying algorithm for defeating query based inference attacks on big data in Hadoop. The proposed algorithm is known as Multi-Model Defence Against Query Based Inference Attacks (MMD-QBIA). A realistic attack model is considered for validating the effectiveness of the proposed methodology. Then an integrated framework for security and privacy to big data is evaluated. Cloudera Distribution Hadoop (CDH) is the environment used for empirical study. The experimental results revealed that the proposed solution prevents different kinds of query based inference attacks on big data besides security to big data in Hadoop MapReduce framework.

Keywords – Big data, Hadoop, query based inference attack, big data leakage prevention

1. INTRODUCTIOIN

With the emergence of cloud computing and big data eco-system, there is every possibility to have innovative approaches to deal with massive amounts of data without losing value possessed in the data [14]. MapReduce is the programming phenomenon that supports parallel processing in presence of thousands of commodity computers in cloud computing or distributed environments. The MapReduce frameworks like Hadoop plays vital role in data analytics in distributed computing environments (DCE). And it has proved to be efficient to deal with big data in numerous application domains [4]. With big data, there are security problems. Different attacks may occur when data is at rest or in transit. Big data analytics has many security and privacy challenges [19].Our prior works [31] and [32] provided security enhancements. For instance, in [31] a light weight security mechanism known as Lightweight Security Scheme (LSS) is defined. In [32], an algorithm named Flexible and Efficient Encryption (FEE) is defined to deal with structured data security and data dynamics on the encrypted data that has been outsourced. However, our work in [31] and [32] does not consider the scenario where big data needs to be protected from privacy attacks when data is subjected to analytics in distributed environment. In this paper, we considered this problem and solution is provided to prevent query based inference attacks on big data. Big data throws privacy challenges unless there is a fool proof mechanism that not only provides cryptographic solution to data security but also for preventing data leakage [12]. Many solutions came into existence to protect privacy of big data. Airavat is one of them where differential privacy (DP) based solution is provided. Privacy issues with MapReduce programming phenomenon are explored in [1]. The usage of DP is advocated in [2] and [7]. Irrespective of DP based solutions, the protection concept is illustrated in Figure 1.

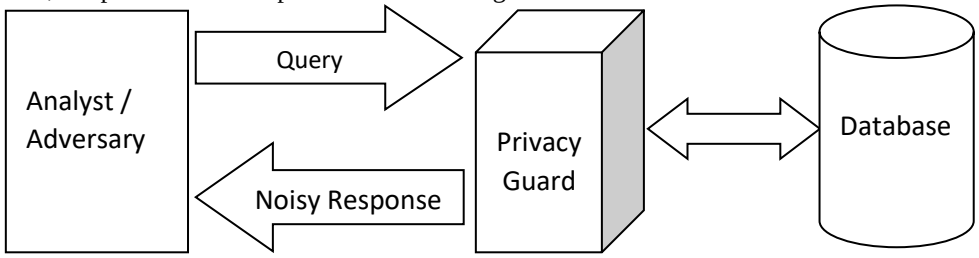

Figure 1: Modus operandi of differential privacy based protection

As presented in Figure 1, the query is made by analyst or adversary to database. Then the privacy guard is implemented based on DP that will add noise to sensitive data and returns to adversary. Thus adversary is defeated

as he cannot infer the desired sensitive information from the results obtained. Privacy Integrated Queries (PINQ) is the solution provided in [30] which is integrated into platform specific programming language named C#. Airavat [25] is another privacy protection based solution from Roy et al. [25] where data providers are assured to have secure and privacy preserving data analytics. This solution was based on DP. DP is used for Location Based Service (LBS) in [21]. However, it does not serve complex situations. Hu et al. [18] used DP for protecting privacy of big data pertaining to telecommunications. They used privacy budget parameter to ensure that the trade-off between the privacy and data loss is minimized. Apple [34] used DP for its operating system in mobile phones. The DP transformation includes hashing, subsampling and adding noise. Our contributions are as follows.

1. We proposed a methodology for preventing data leakage or privacy attacks, especially, query based inference attacks on big data which is being processed in MapReduce framework in distributed environment.
2. We proposed multiple algorithm that combined provide a comprehensive solution to query based inference attacks on big data.
3. We implemented the proposed algorithms in a variant of Hadoop known as Cloudera Distribution Hadoop (CDH).
4. An integrated security framework is proposed and evaluated. It could improve security and privacy to big data.

The remainder of the paper is structured as follows. Section 2 reviews literature on different aspects of privacy of big data and attack prevention methods. Section 3 presents the preliminaries to ascertain the proof of the concept well. Section 4 presents the proposed solution to protect big data from specific privacy attacks. Section 5 presents experimental results. Section 6 evaluates the proposed solution and compares it with the state of the art. Section 7 discussed about the threats to the validity of the proposed solution. Section 8 concludes the paper and gives directions for future work.

2. RELATED WORK

This section reviews literature on the issues pertaining to privacy attacks on big data in distributed environments.

2.1 Privacy Issues in MapReduce Environments

MapReduce environments deal with big data. They need to identify and protect from malicious attacks pertaining to privacy or query based inference attacks. Derbekoet al. [1] explored different kinds of attacks in MapReduce programming paradigm. They include impersonation attack, Denial of Service (DoS), eavesdropping, replay attack, repudiation and man in the middle attack. They found that privacy attacks may occur from adversarial Cloud Service Provider (CSP), adversarial users, professional hackers. Therefore, they emphasized the need for protecting data providers, protecting data from untrusted service providers. They suggest to define MapReduce algorithms with privacy preserving support.

Ohrimenkoet al. [5] opined that in spite of encrypted communications in distributed computing environments, there is probability of leakage of sensitive data while big data is subjected to analytics. They proposed a methodology for secure implementation of MapReduce jobs by safeguarding intermediate traffic patterns. They proposed a solution known as shuffle in the middle to prevent leakage in MapReduce. Considering encrypted dataset D, their shuffle method results in D' which is a permutation of D. They also employ padding method to deal with intermediate traffic wit careful analysis. Bhathal and Singh [10] explored vulnerabilities of Hadoop pertaining to security policies, configuration, web interface, software and technology heterogeneity. They found that the traditional security approaches will be inadequate to deal with runtime issues when big data is being analysed.

2.2 Methods based on Differential Privacy to Protect Big Data

Differential privacy (DP) is the widely used technique that produces two identical datasets denoted as D and D'. The latter contains certain noise. However, DP ensures that D' produces almost same result as done with D. However, there is privacy protection with D'. Jain et al. [2] explored different aspects of DP in the context of its application to big data. They studied the importance of privacy budget and sensitivity. There are many real world applications of DP. Lee [7] described the usage of DP by US Census Bureau to safeguard confidentiality of patient data. McSherry [30] developed a privacy preserving data analysis platform known as Privacy Integrated Queries (PINQ). PINQ helps in interactive data analytics besides ensuring privacy of data. It is based on LINQ (Language Integrated Query) of C# programming language.

Roy et al. [25] proposed a MapReduce based system for privacy and security of big data. The system is named as Airavat. With Airavat, the data providers are assured to have secure and privacy preserving data analytics. Airavat supports DP for preventing leakage of sensitive data. In fact, it integrates access control mechanism and MAC+ DP for better performance. However, Airavat is inadequate to protect privacy of big data when output keys are generated by untrusted mappers. Andres et al. [32] employed DP for location based systems. They enhanced Location Based Service (LBS) systems for privacy guarantees. DP adds random noise to sensitive data such as location. They intended to improve it for complex applications.

Hu et al. [33] used DP for protecting privacy of big data pertaining to telecommunications. They used privacy budget parameter to ensure that the trade-off between the privacy and data loss is minimized. They explored data publication with DP to see that the sensitive information is not disclosed. Apple company implemented DP in iOS 10 in order to collect and store users' data with privacy protection. However, Apple cannot extract any specific user's data as it violates privacy. As Facebook and Google are doing, Apple sends lot of users' data for data

analytics. The data is transmitted, however, in transformed format with DP. The DP transformation includes hashing, subsampling and adding noise [34].

2.3 Enhancing MapReduce Layer for Big Data Privacy

Privacy attacks on big data may occur when data is subjected to Map and Reduce methods. To overcome this problem, Jain et al. [3] enhanced the MapReduce (MR) layer with an additional layer between MR layer and Hadoop Distributed File System (HDFS). The input and output privacy is combined along with security. The method employed here protects data from privacy attacks and reduces information loss. It also promotes scalability as it uses lightweight encryption.

Adithamet al. [11] proposed different thread detection mechanisms that may arise from malicious insiders. Their solution includes profiling process behaviour using library and system calls and memory access patterns. After building process profiles, they are verified dynamically at runtime to know any discrepancies. Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) methods are employed to estimate violations. In future, they intended to deal with big data privacy when the data is subjected to analytics. Gambset al. [13] used a tool known as GEPETO for analysing big data privacy by interpretation of mobility traces in large scale. They considered MapReduce environment for their empirical study where they sanitized data to prevent privacy attacks. They intended to improve it by using spatial cloaking methods in future. Dinhet al. [15] proposed a methodology for privacy preserving MapReduce computations. They incorporated secure shuffling, secure grouping and execution integrity.

Stephen et al. [16] proposed a method with program analysis to find security threats in MapReduce code. Geyer et al. [17] on the other hand proposed a security framework for processing big data in distributed environment. Pireset al. [20] proposed a light weight security framework for MapReduce programming paradigm. Raiziet al. [22] proposed a hybrid framework for secure data analytics. In the same fashion, Lu et al. [23] focused on opportunistic computing framework with privacy and security in healthcare domain. Du et al. [24] proposed an attestation mechanism for cloud service integrity as part of Software as a Service (SaaS). Xue and Hong [26] proposed a framework for secure data sharing in presence of dynamic groups while Li et al. [27] proposed secure and privacy preserving mechanism for sharing of health records. Yang et al. [28] used polynomial codes for security in distributed environments. Dong et al. [29] proposed a distributed processing approach that is hierarchical in nature. From the literature, it is observed that there has been considerable research to make MapReduce operations with privacy consideration. However, with respect to differential privacy, the existing works showed different approaches and there is need for an integrated multi-modal approach for preventing query based inference attacks.

3. PRELIMINARIES

Differential Privacy (DP) is the technique used to protect data from privacy attacks. It was originally developed by Dwork, Nissim, McSherry and Smith and later on improved by others [8]. To be formal, let two databases denoted as D_1 and D_2. These two are known as neighbouring databases when they have difference in at most a single data entry. Accordingly, any algorithm denoted as M is considered to be ε-differentially private if D1 and D2 output x for all pairs as in Eq. 1.

$$\Pr[M(D1) = X] \leq \exp(\varepsilon)\Pr[M(D2) = x] \quad (1)$$

The output of the computation does not reveal the presence of any data item as input. As adversaries will not be able to know whether a specific item is part of the dataset as it precludes deriving any sensitive information from the data. Ideally, DP needs to be employed in such a way that when (after adding DP) data is given to third party analyst, he/she will never be able to know identity of any entity. Such way of characterization of data is part of DP based methods. DP is best used to prevent query based inference attacks. In other words, adversaries cannot know the participation of an item (presence or absence of an item) in the dataset. Privacy Budget (ε) is the control parameter for enforcing privacy on big data (as used in this paper). Considering two neighbouring datasets D_1 and D_2 and an output function A, the privacy budget needs to be low such as a value that is almost equal to 1. It does mean that the outcome probability of A on D_1 and D_2 is almost same. This is the ideal way of using privacy budget when DP is employed. With higher DP more security is possible but it leads to less utility of data subjected to analytics. Therefore, the privacy budget € is generally kept at 0.01, 0.1 etc. Eq. 2 shows the usage of privacy budget.

$$\Pr[A(D1 \in S] \leq e(2) \in \Pr[A(D2) \in S] \quad (2)$$

There is another important term pertaining to DP. It is known as sensitivity that tells the amount of noise added to the output of MapReduce function in Hadoop (with respect to the work of this paper). The sensitivity is based on the magnitude of change in outcome when a single row is added or removed. When a series of counting queries denoted as Q made on D_1 and D_2 the sensitivity of Q is denoted as ΔQ and it is computed as in Eq. 3

$$\Delta Q = \max ||Q(D_1) - Q(D_2)|| \quad (3)$$

In order to achieve DP noise is added to dataset. There are two primary mechanisms of adding noise. They are known as Exponential Mechanism (EM) and Laplace Mechanism (LM). The amount of noise added has its influence on the global sensitivity and privacy budget. EM is a security controlled strategy to achieve DP. It is used for output that is in categorical form. Quite intuitively, it can be understood that EM guarantees the DP definition as the change in a row of database will not affect the outcome of the function. It is desired to handle situations where best response is to be picked up. In a query – response system, let input database is denoted as D

and a potential response is denoted as rϵR for a score function denoted as u: D × R -> R. Let an algorithm named A gives a response to query in order to satisfy e-differential privacy as in Eq. 4.

$$A(D, u) = \{r :| Pr[r \in R] \infty \exp(\varepsilon u(D, r)/2\Delta u)\} \quad (4)$$

The score function determines the yield of exponential mechanism. The privacy budget will have its influence on the possible outcome. For higher level of security, it is essential to keep the value of privacy budget as low as possible. LM on the other hand computes given function and perturbs coordinates with noise that is acquired from distribution of LM. The level of noise is controlled based on the privacy budget parameter. LM is useful for producing numerical outputs. An algorithm denoted A when applied to D with global sensitivity denoted as Δf and the function denoted as f: D ->R^d, Eq. 5 shows how the noise is added.

$$A(D) = f(D) + \text{noice} \quad (5)$$

If the noise added complies with Laplace distribution, e-differential privacy is satisfied. Thus it is denoted as noise~Lap(Δf/ε) where the zero is considered for location parameter while the scale parameter is denoted as Δf/ε. The probability density function, when scale parameter is b and location parameter is zero, is computed as. (X) = exp (- |X|/B)/2B

$$\sigma(x) = \sqrt{D(x)}, D(x) = 2b^2, \text{and } b = \Delta f/\varepsilon \quad (6)$$

$$D(x) = 2(\Delta f/\varepsilon)^2 = 2\Delta f^2/\varepsilon^2 \quad (7)$$

$$\sigma(x) = \sqrt{D(x)} = \sqrt{2\Delta f^2/\varepsilon^2} = \sqrt{2\Delta f/\varepsilon} \quad (8)$$

In Eq. 6, standard deviation is denote as σ(X)and variance is denoted as D(X). Finally, the results are obtained as in Eq. 7 and Eq. 8. DP also exhibits two important properties. They are known as sequential composition (SC) and parallel composition (PC). The former refers to the sequence of computations that provide DP at each computation and DP at sequence level as well while the latter involves in many disjoint computations in parallel.

4. PROPOSED FRAMEWORK

Technological innovations changed the way an application stores and processes data.With cloud computing infrastructure, Internet-based computing has emerged to be an ideal approach. The traditional means of storage and retrieval are no longer preferred for cost and other reasons while handling large volumes of data. Before indulging into the proposed framework here is the problem statement or motivating scenario.

4.1 Problem Statement

A MapReduce paradigm consists of two units of computation known as map and reduce. Each unit takes key-value pairs as input and produces desired output in the form of key-value pairs after processing. Both input and output are stored in Hadoop Distributed File System (HDFS). It is assumed that input given to any MapReduce computing is encrypted and adversaries can only get encrypted data if they succeed in launching attacks. However, when data is being processed, it is done in plain text. Therefore, it is essential to protect such data from privacy attacks. Figure 2 shows the map reduce computations for WordCount benchmark.

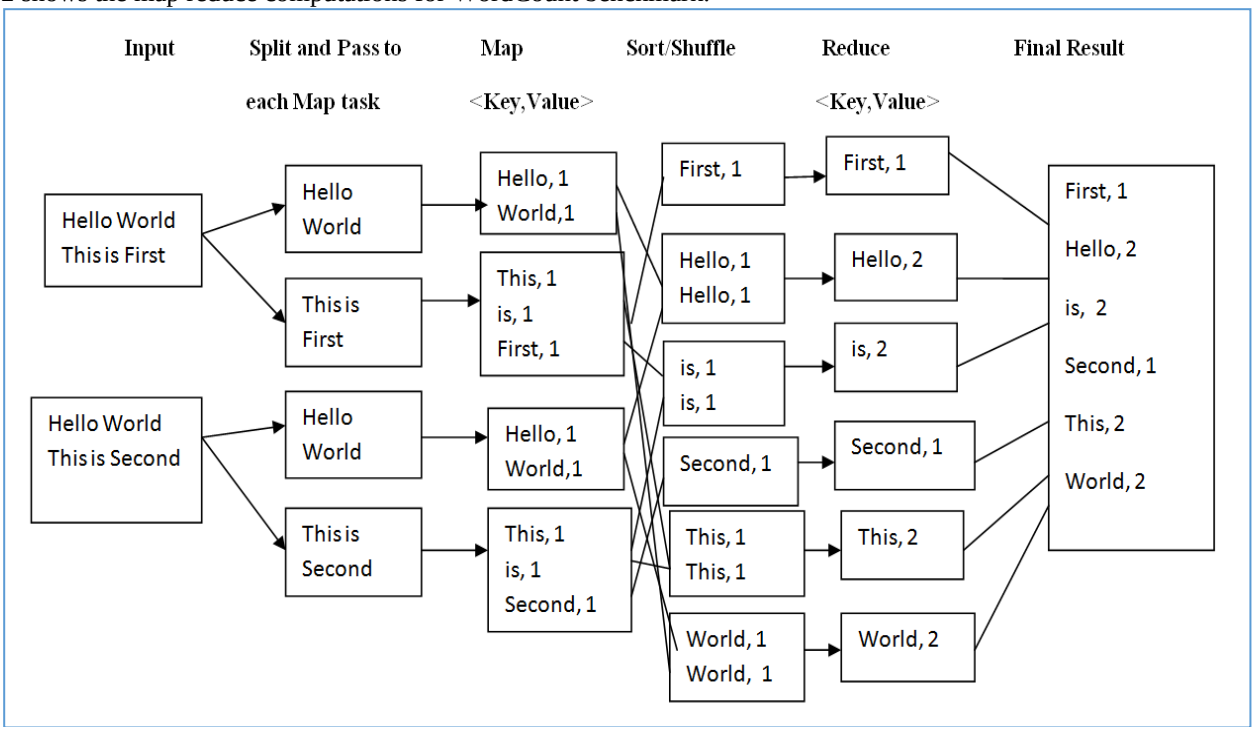

Figure 2: Shows execution of WordCount benchmark with MapReduce computations

For big data analytics, a programmer typically implements Map and Reduce tasks. In many real world data analytics applications, data of an organization is used for gaining business intelligence (BI). In the process, the main problem considered in this paper is privacy attacks launched by adversaries. To be specific the attacks are query based inference attacks where adversary wanted to infer knowledge by knowing the presence of specific

customer in the dataset. That is the reason it is known as query based inference attack. Considering the MapReduce source code in Listing 1, there is evidence of such attack from adversary.

```
1 public static class Map extends MapReduceBase implements
2           Mapper<LongWritable, Text, Text, IntWritable> {
3
4   private Text word = new Text();
5
6   public void map(LongWritable key, Text value,
7                 OutputCollector <Text, IntWritable>
8                 output, Reporter reporter)
9                         throws IOException {
10    String line = value.toString();
11    StringTokenizer tokenizer = new StringTokenizer(line);
12    while (tokenizer.hasMoreTokens()) {
13      String keyword = tokenizer.nextToken();
14       if (keyword.compareTO("Suzuki")){
15         word.set("Sazaki");
16         output.collect(word, 1000000);
17        } else {
18         word.set(keyword);
19         output.collect(word, 1);
20        }
21    }
22  }
23 }
24
25 public static class Reduce extends MapReduceBase implements
26  Reducer<Text, IntWritable, Text, IntWritable> {
27   public void reduce(Text key, Iterator<IntWritable>
28              values, OutputCollector<Text, IntWritable>
29              output, Reporter reporter) throws IOException {
30     int sum = 0;
31     while (values.hasNext()) {
32        int rValue = values.next().get();
33        if (rValue>=1000000)){
34            key.set("Sazaki");
35            sum = 12345678;
36            break;
37            } else {
38            sum += rValue;
39       }
40     }
41     output.collect(key, new IntWritable(sum));
42   }
43 }
```

Listing 1: Application specific MapReduce source code (malicious)

As observed in Line 14 through Line 16, attacker is trying to find the presence of a customer named “Suzuki”. If the customer is found in the data being processed, then the adversary is setting the word “Sazaki” at Line 15 and set a value 1000000 as output. This is in map() function. In the reduce() function, from Line 33 through Line 35,

the adversary is manipulating sum value to gain very specific value as output. If that value is found in output, adversary confirms that there is presence of customer "Suzuki" in the big data. This kind of attack is known as query based inference attack. This is the main privacy leakage problem addressed in this paper.

4.2 Methodology

The objective of this methodology is to protect sensitive data even in presence of potentially untrusted mapper and reducer codes.It is based on cloud computing technology and distribued programming framework. Hadoop is the MapReduce framework used for empirical study which facilitates new paradigm in programming with map and reduce functions. The source code of map() and reduce() functions may have code which is malicious in nature. Often it is injected by adversaries to launch privacy attacks. Therefore, it is essential to ensure non-disclosure of sensitive information. The solution provided in this paper is based on DP. Cloudera Distribution Hadoop (CDH) has MapReduce framework. This framework is generally scalable, available and fault tolerant. It supports various real world use cases associated with big data analytics. Figure 3 shows the architectural overview of the proposed methodology.

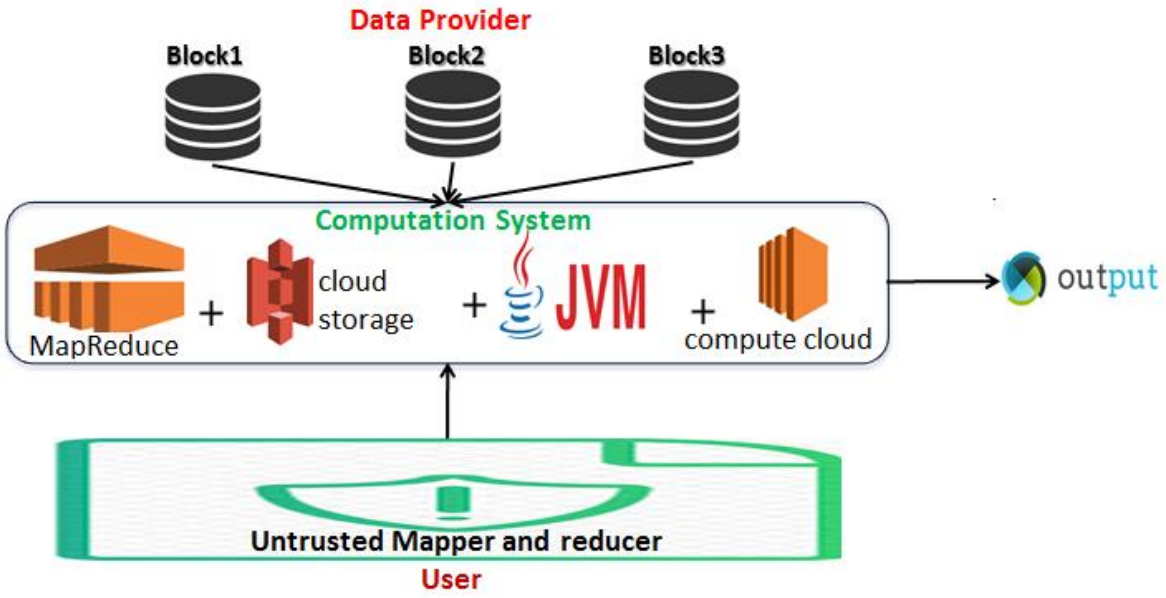

Figure 3: Architectural overview of the proposed methodology

The computation system has different components such as MapReduce framework, cloud storage, JVM and compute cloud. MapReduce supports input from HDFS and sending output to HDFS. MapReduce runs in a cluster of commodity computers. Privacy of big data is achieved using DP based algorithm. In presence of malicious mapper or reducer, the proposed algorithm ensures that the privacy of big data is not lost. DP, as explained in Section 3, has capabilities to prevent certain kinds of privacy attacks such as query based inference attacks. Here is the procedure used to prevent privacy attacks. Let us consider D as original dataset while D' is derived from D by using DP technique. There will be no much difference between these two datasets. Hence they are known as neighbouring datasets. An algorithm A achieves DP with output denoted as O as in Eq. 9.

$$Pr[A(D)=O] <= \exp(\boldsymbol{\varepsilon}).\ Pr[A(D')=O] \quad (9)$$

The degree with which privacy is protected is represented as $\boldsymbol{\varepsilon}$. As the proposed methodology deals with large volumes of data, it is likely that there is sensitive data. The data may be of any domain like healthcare, banking, social networks and so on. There are many algorithms to provide privacy but may result in information loss due to data transformation for ensuring privacy. Thus, there is trade-off between privacy level and deterioration of utility of data. If such trade-off is not handled, it leads to utility problem of big data. This fact is understood with the reconstruction function found in [33] which is shown in Eq. 10.

$$Fx_i(a)=\frac{\int_{-\infty}^{a} fy\,(w1-z)fx(z)dz}{\int_{-\infty}^{\infty} fy\,(w1-z)fx(z)dz} \quad (10)$$

When there are many randomsamples with cumulative distribution function (CDF) is denoted as F_y and samples are as x_1+y_1, x_2+y_2..., x_n+y_n and F_x. As in Eq. 10, there is posterior distribution. In the same fashion, for x_1+y_1, x_2+y_2, ..., x_n+y_nestimation is as in Eq. 11.

$$Fx_i(a)=\frac{1}{n}\sum_{i=1}^{n} \dot{fx}_i=\frac{1}{n}\frac{\int_{-\infty}^{a} fy\,(w1-z)fx(z)dz}{\int_{-\infty}^{\infty} fy\,(w1-z)fx(z)dz} \quad (11)$$

By differentiating the F_x density function is obtained as in Eq. 12.

$$Fx_i(a)=\frac{1}{n}\sum_{i=1}^{n} \dot{fx}_i=\frac{y\,(w1-a)fx(a)}{\int_{-\infty}^{\infty} fy\,(w1-z)fx(z)dz} \quad (12)$$

The reconstruction and randomization solutions may cause data leakage as they have some associated data. Therefore, such solutions are not suitable for strong privacy for big data. The DP construct $(\in, a)$-differential

Privacy satisfies the computation function given in Eq. 9 by considering D and D' as neighboring datasets. The relation S⊆ Range(F)is found to be true. Eq. 13 is thus used to achieve this.

$$P_r[F(D) \epsilon S] \leq \exp(\epsilon) \times P_r[F(\dot{D}) \epsilon S] \quad (13)$$

As per this, it is not possible to have inference attacks or query based privacy attacks. Thus adversaries cannot find the presence or absence of given entity in the data with any probability.

4.3 Algorithm Design

We defined an algorithm known as Multi-Model Defence Against Query Based Inference Attacks (MMD-QBIA) which analyses reducer code and original dataset D. After execution of the algorithm, the neighbouring dataset D' is resulted. The D' is produced by reducer in MapReduce paradigm in order to defeat query based inference attacks. It is achieved by adding noise to the reducer output. However, adding too much noise will lead to losing utility of big data. Therefore, we devised a plan to determine the noise level by analysing reducer code (byte code pattern). If the bytecode pattern is found genuine, NoiseAddition() function is invoked to add little noise. If pattern is not found, StrongNoiseAddition() function is invoked as the attack is suspected. Table 1 shows notations used in the algorithm.

NOTATION	DESCRIPTION
D	Original dataset (big data)
D'	Neighbouring dataset (transformed for privacy protection)
key, <v_1, v_2, … v_n>	Reducer input
max	Max value in the range of values
min	Min value in the range of values
R(x)	Returns a random number from the range of values denoted as -|x| to |x|
random_value_list	Contains a set of random values
single_value_list	Contains a set of values from reducer that appear only once (as the adversary model has its significance)
$\boldsymbol{\varepsilon}$	Privacy parameter whose value is given by owner of big data (or a domain expert)

Table 1: Notations used in the MMD-QBIA algorithm

In order to add noise to aggregate values such as sum, count, average, max and min produced by reducer, different procedures are defined as part of the algorithm. They include NoiseAdditionForSum (), NoiseAdditionForCount (), NoiseAdditionForAvg (), NoiseAdditionForMax () and NoiseAdditionForMin (). All these procedures are invoked as part of normal noise addition procedure named NoiseAddition (). If the expected bytecode pattern is not found (suspected attack), the reducer output will be subjected to StrongNoiseAddition() that ensures complete sanitization of all target values prior to returning final reducer output.

Algorithm: Multi-Model Defence Against Query Based Inference Attacks (MMD-QBIA)
Input: MapReduce code, Dataset D, $\boldsymbol{\varepsilon}$
Output: Protected Dataset D'

1. Start
2. Analyze reducer
3. IF pattern found Then
4. D'=Invoke NoiseAddition()
5. ELSE
6. D'=Invoke StrongNoiseAddition()
7. END IF
8. Return D'
9. End

Procedure NoiseAddition()

1. Start
2. Invoke NoiseAdditionForSum()
3. Invoke NoiseAdditionForCount()
4. Invoke NoiseAdditionForAverage()
5. Invoke NoiseAdditionForMax()
6. Invoke NoiseAdditionForMin()
7. End

Procedure NoiseAdditionForSum()

1. Start
2. Initialize min and max range
3. sum=Obtain original sum value
4. noise = (1+$\boldsymbol{\varepsilon}$) * R(max)
5. return noise + sum
6. End

Procedure NoiseAdditionForCount()

1. Start
2. Initialize min and max range
3. count=Obtain original count value

4. noise = (1+$\boldsymbol{\varepsilon}$) * R(1)
5. return noise + count
6. End

Procedure NoiseAdditionForAvg()

1. Start
2. Initialize min and max range
3. sum=Obtain original sum value
4. noise = ((1+$\boldsymbol{\varepsilon}$) * R(max))/n
5. return noise + sum
6. End

Procedure NoiseAdditionForMax()

1. Start
2. Initialize min and max range
3. maxvalue=Obtain original maximum value
4. noise = (1+$\boldsymbol{\varepsilon}$) * R(max)
5. return noise + Max(min)
6. End

Procedure NoiseAdditionForMin()

1. Start
2. Initialize min and max range
3. minvalue=Obtain original minimum value
4. noise = (1+$\boldsymbol{\varepsilon}$) * R(max)
5. return noise + minvalue
6. End

Procedure StrongNoiseAddition()

1. Start
2. Take reducer input (key, <v_1, v_2, … v_n>)
3. random_value_list = <v_1, v_2, … v_n>
4. single_value_list = getSingeValueList()
5. For each value x in n
6. IF vi exists in single_value_list THEN
7. vi=getRandomValue(random_value_list)
8. END IF
9. End For
10. output=<v_1, v_2, … v_n>
11. noise = (1+R($\boldsymbol{\varepsilon}$))
12. return noise + output
13. End

Algorithm 1: Multi-Model Defence Against Query Based Inference Attacks

As defined in Algorithm 1, the given data (big data) from reducer prior to returning output of reduce function is taken as input. It is denoted as D and the algorithm transforms it into a neighboring dataset D'. Finally, instead of returning D, the reducer returns D' as output. Thus adversaries fail in succeeding query based inference attacks. The main algorithm starts analyzing reducer byte code pattern. If the regular pattern is found, there is need for noise addition but it is limited considering computational cost. Step 2 of the algorithm does this analysis. Step 3 through Step 7, one of the two procedures named NoiseAddition() and StrongNoiseAddition() is executed based on the given condition that checks whether pattern is found.

NoiseAddition() procedure is desinged in such a way that it takes care of only aggregated values that are generally produced by the reduce() function of MapReduce computing. As there are different aggregate values known as sum, max, min, average and count, different procedures are defiend to deal with all these aggregates. The procedures are named as NoiseAdditionForSum(), NoiseAdditionForMax(), NoiseAdditionForMin(), NoiseAdditionForAvg() and NoiseAdditionForCount() respectively. Noise addition is made based on the privacy parameter $\boldsymbol{\varepsilon}$ given by domain expert of data provider. Line 4 of all these procedures computes noise dynamically. All these functions avoid adding much noise.

When there is no pattern found (attack suspected), the algorithm invokes StrongNoiseAddition() procedure. This procedure considers sanitization or anonymization of all single valued numbers (as the attack model has significance to such values). Step 2 of this procedure takes reducer input in the form of key-value pair. In Step 4, it finds all single valued numbers. It also considers a random value list used for privacy protection as in Step 3. Step 5 through Step 9 is an iterative process for adding noise. Step 10 takes the output values and before returning them by the reducer, they are subjected to noise addition. Step 11 computes noise and Step 12 returns noise added outcome that comes back to Step 6 of the main algorithm. This outcome is nothing but D' that represents neighboring database that defeats query based infernce attacks launched by adversaries as per DP philosophy.

5. SECURITY INTEGRATED FRAMEWORK

Our prior works [31] and [32] provided security enhancements. In [31] a security mechanism known as Lightweight Security Scheme (LSS) is defined. In [32], an algorithm named Flexible and Efficient Encryption (FEE) is defined to deal with structured data security and data dynamics on the encrypted data that has been outsourced. The work presented in Section 4 deals with prevention of query based inference attacks on big data in Hadoop MapReduce framework.

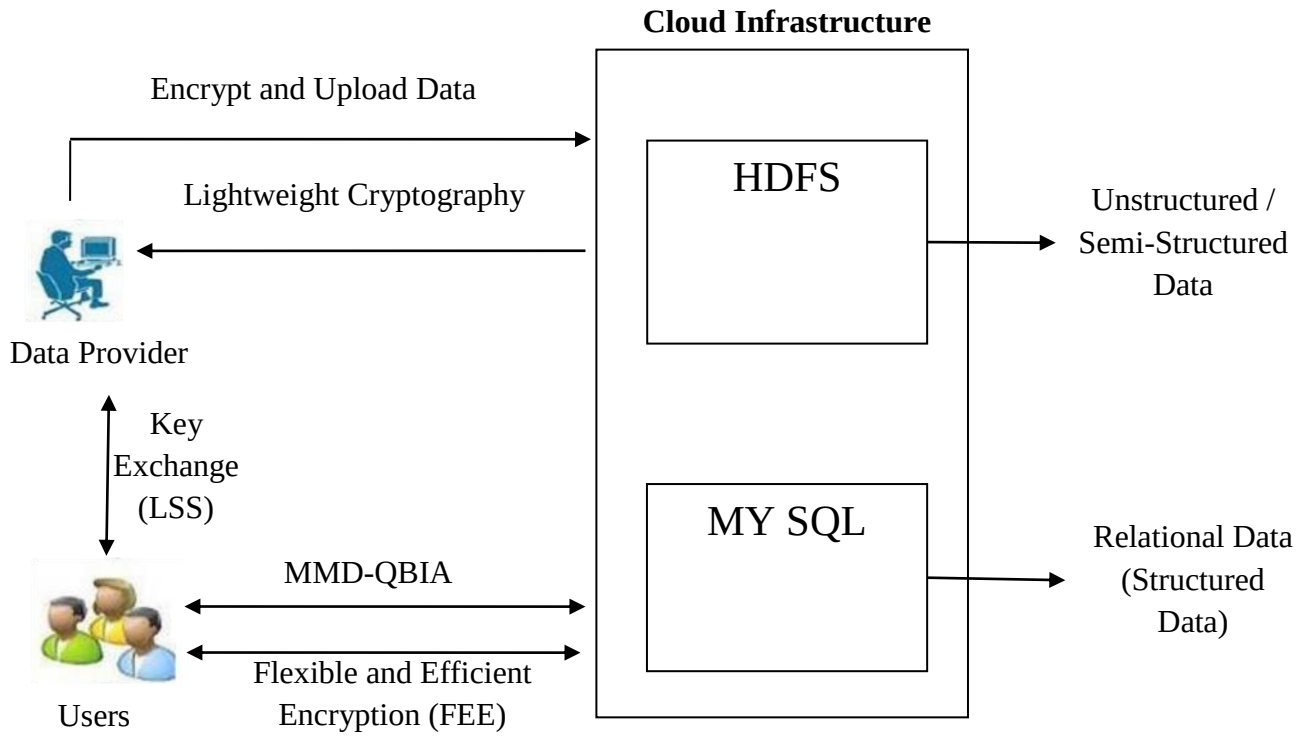

Figure 4: Integrated architecture for big data security, privacy and data dynamics

As shown in Figure 4, the key sharing scheme LSS is used to have secure exchange of keys between data provider and users. A lightweight cryptographic method is used to outsource data and retrieve data from cloud. This method ensures secure end to end communication between cloud server and data provider. When data is to be shared to users, then the data provider and users need LSS for secure key exchange. With the security keys, the users can perform two kinds of operations on cloud. First, they can make queries to obtain data from non-relational data. Second, they can use SQL based queries for storage and retrieval of data from relational database. Besides, users can perform data dynamics (changes on the outsourced relational data) directly. It is achieved with Flexible and Efficient Encryption (FEE) scheme. The scheme takes care of security of data flown between the users and cloud infrastructure. MMD-QBIA algorithm described in Section 4 is used to see that the queries made by users for data from non-relational databases are securely processed. In fact, the algorithm is aimed at preventing query based inference attacks. With the integration architecture shown in Figure 4, it has a comprehensive and holistic phenomenon for realizing big data security and privacy when data is at rest, in transit and when being used for data analytics.

6. EXPERIMENTAL RESULTS

Experiments are made with a cluster made up of 3 machines, one master node and two slave nodes. Intel Core i5 processor with 3.4 GHz is the configuration used for the machines. The configuration of Hadoop is changed to have replication number set to 2 in conf/hadoop-site.xml. Two datasets are used for empirical study. The first dataset is the real word dataset collected from [6]. The second dataset is synthesized one. Therefore, details of two experiments are provided in this section. In the first experiment dataset from [6] is used while the second experiment used the synthesized dataset.

6.1 Experiment 1

The big data [6] contains different attributes such as IP address, date, time and link. Here the sensitive attribute is IP address and adversaries launch query based inference attacks to know the presence or absence of a specific IP address in the big data being processed in Hadoop MapReduce. The program uses aggregate function sum. For this reason, the MMD-QBIA algorithm invokes NoiseAddition() procedure that turn calls NoiseAdditionForSum() procedure which returns the noisy (privacy protected) outcome to its caller and thus D' is generated. The D' is then returned by the reducer as final output. The system gets two queries from user. The first query is genuine (not malicious). The purpose of the program is to know how many times each link is repeated in the given log file. Noise is added accordingly. With respect to second query, the attacker tries to fine the presence of an IP address "192.168.133.33". As noise is added, it is not possible for adversary to infer the privacy or presence of an IP address.

IP	Actual Count
192.168.133.33	13654
192.168.133.34	8435
192.168.133.35	12447
192.168.133.36	13461
192.168.133.37	13363
192.168.133.38	8591
192.168.133.39	14876
192.168.133.40	13983
192.168.133.41	14765

Table 2: MapReduce outcome in absence of attacker

As presented in Table 2, **192.168.133.33**is the IP address highlighted as it is used later by the adversary to know its presence in the data. The table shows actual count of the IP address in the dataset. The results of MapReduce came as expected by the mapper and reducer that are genuine. However, when mapper or reducer are compromised, the intention is to have privacy attacks on big data. To safeguard the IP address from disclosure to attacker, the patter of Reduce function is obtained using decompiler. The outcome is used in the algorithm to know whether attack is made or not. The result in presence of attacker is shown in Table 3.

IP	Count in Presence of Attacker
192.168.133.33	13653
192.168.133.34	8434
192.168.133.35	12446
192.168.133.36	13460
192.168.133.37	13362
192.168.133.38	8590
192.168.133.39	14875
192.168.133.40	13982
192.168.133.41	14764

Table 3: Results of MapReduce in presence of attacker

As presented in Table 3, the IP address and corresponding noise added value for count is provided. The D is converted to D' with the proposed algorithm. For the IP address 192.168.133.33 which is the target of privacy attack made by adversary, the result is slightly changed and the value is 13653. The computation process of the algorithm for 1 highlighted IP is as follows.

$$\boldsymbol{\varepsilon} = 8.85\text{x}10^{-12}$$

count (in presence of attacker) =count+ [(1+ $\boldsymbol{\varepsilon}$)+R]

=13654+[(1+$8.85\text{x}10^{-12}$)-2.00000000001]

=13653

The above computation illustration shows proof of the concept. It is computed for the target IP address 192.168.133.33.

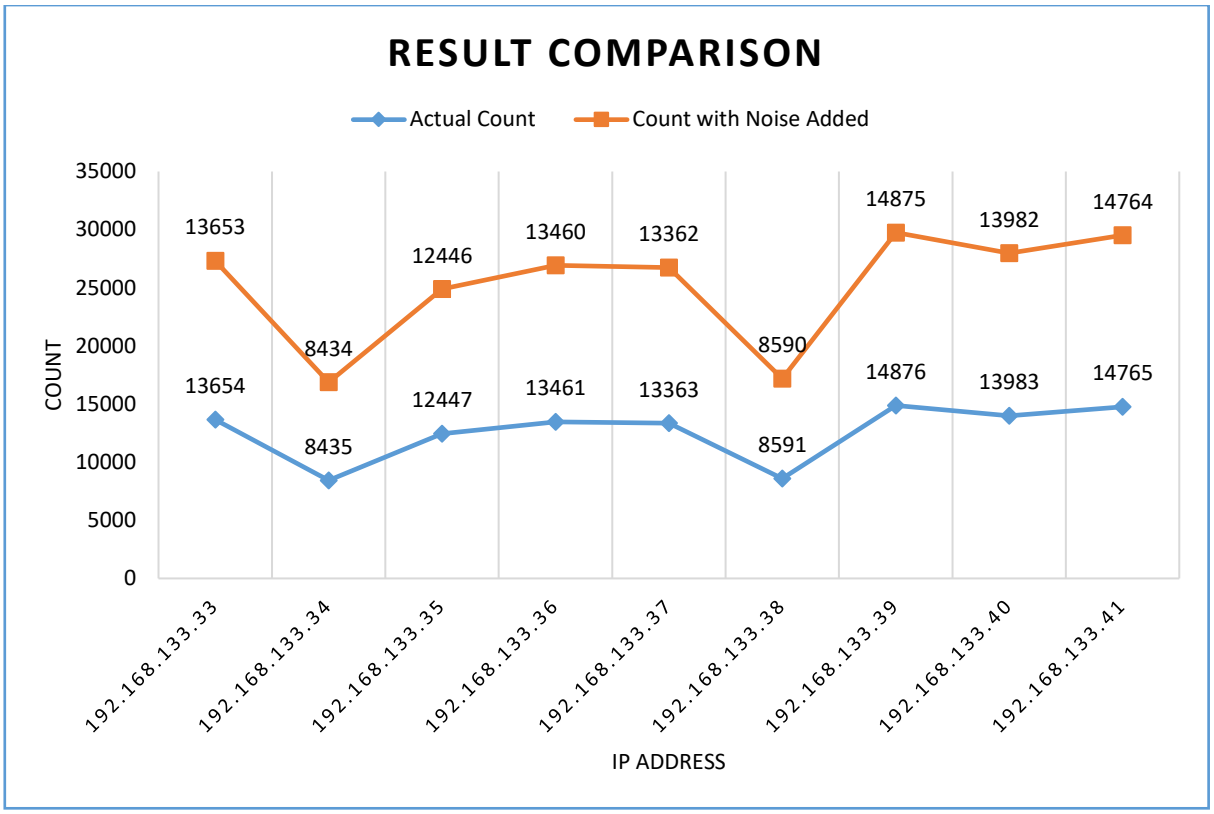

Figure 4: MapReduce outcome comparison (Stacked Line Graph)

As presented in Figure 4, it is observed that the IP address is shown in horizontal axis and the count (genuine and in presence of attacker) is shown in vertical axis. As the different between D and D' is very less, stacked line graph is preferred. It shows the result of proposed DP algorithm.

Data Size (GB)	Time taken for execution (sec)	
	With MMD-QBIA	WithoutMMD-QBIA
50	103	105
100	143	140
150	197	190
200	259	250
250	310	300
300	355	340
350	417	400
400	440	420
450	510	490
500	525	510

Table 4: MapReduce outcome with and without privacy protection

As presented in Table 4, the results pertaining to execution time of MapReduce based DP algorithm **MMD-QBIA**meant for protecting big data is observed.

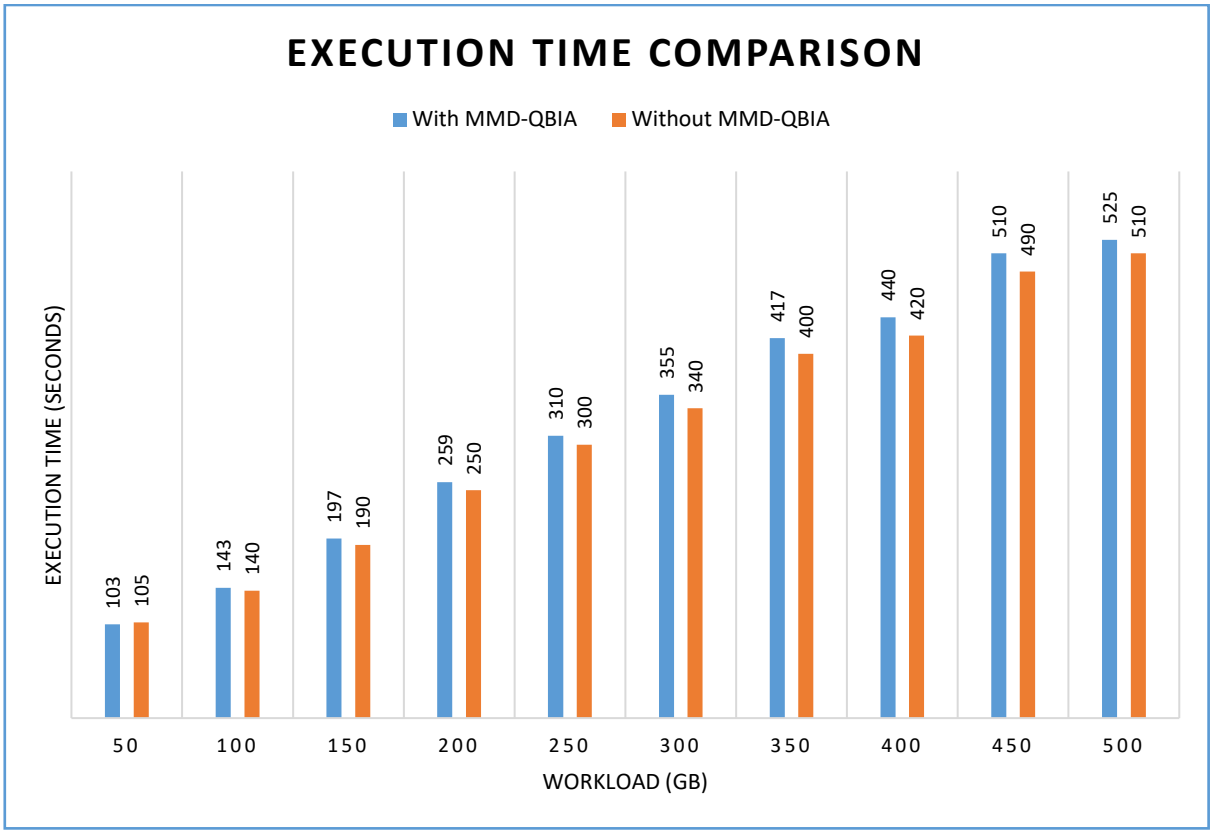

Figure 5: Execution time of MapReduce with and without MMD-QBIA

As presented in Figure 5, the workload of experiments is shown in horizontal axis and vertical axis shows the time taken in seconds. As the results showed, the size of workload has its influence on the execution time. There is linear increase in the execution time as workload size increases. It shows the performance difference when the proposed DP algorithm MMD-QBIA is employed.

6.2 Experiment 2

In this experiment, the proposed algorithm is evaluated using synthetic dataset. The dataset contains family name of a person, day of birth (0-30) and the score achieved in an entrance examination conducted by a university. The purpose of the program is to compute average score of the data provided based on the family name of person. As number of people with same family name existed in the data, the MapReduce computing finds the average score. In the bytecode of the program, the average function is intentionally removed. This action forces the algorithm to go with StrongNoiseAddition() procedure. Here the attacker knows the birth day of a person and tries to find the family name of the person. Since day of birth has range of values from 0 to 30, the average is always less than 30. Attacker finds birth day value and replaces it with a big number such as 1000000. Thus attacker expects the average value greater than 30. In the experiment, however, the algorithm replaces the value 1000000 with a random number between 0 and 30. This will defeat the attack and proves the efficacy of the algorithm. The rationale behind this is that the actual value in the experiment is different from that of expected value by the adversary.

6.3 Results of Integrated Architecture

The integrated architecture with FEE and LSS algorithms is evaluated with Cloudera Distribution Hadoop (CDH). The observations are recorded in terms of encryption decryption time, total upload time and total download time for given workload size.

Data Size (MB)	Execution Time for Encryption and Decryption (seconds)					
	Encryption (AES)	Encryption (FEE)	Decryption (AES)	Decryption (FEE)	Encryption (LSS)	Decryption (LSS)
10	1.0168	0.999	0.9956	0.8932	0.8989	0.7921
50	2.6237	2.4956	2.0879	2.0936	2.4956	1.9925
100	2.9948	2.7979	2.7583	2.2267	2.6968	2.1156
500	13.8648	13.0997	9.8845	9.2243	13.0896	9.1132

Table 5: Encryption and decryption time comparison

As presented in Table 5, the encryption and decryption time for AES, FEE and LSS are compared against different workloads.

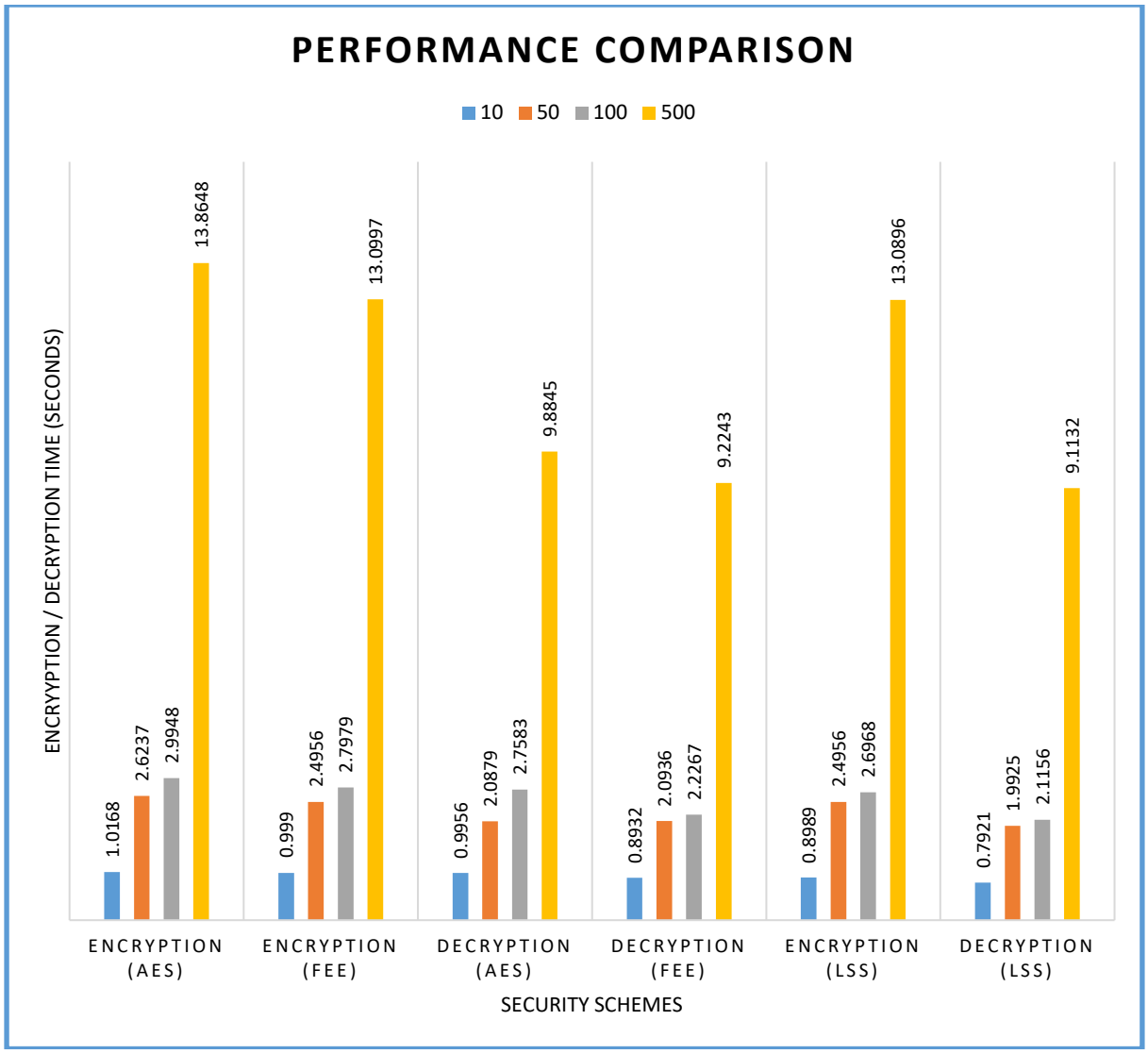

Figure 6: Performance comparison with encryption time and decryption time

As presented in Figure 6, the encryption and decryption performance is evaluated. The security schemes used in the empirical study are provided in horizontal axis while the vertical axis shows the execution time for encryption and decryption. The results revealed that the proposed methods such as FEE and LSS took relatively less time for cryptographic operations. The rationale behind this is that, they are designed to be lightweight.

	Total Upload Time (seconds)				
Data Size (MB)	**AES**	**RSA**	**FEE**	**ECDH**	**LSS**
10	0.7973	1.8191	0.6678	0.6862	0.6568
50	2.9273	4.1609	2.371	2.8162	2.6689
100	4.6873	8.8621	3.5369	4.5762	4.4258
500	17.6373	32.1845	14.2967	17.5262	17.0856

Table 6: Total upload time taken by security schemes

As presented in Table 6, the total upload time for AES, RSA, FEE, ECDH and LSS are compared against different workloads.

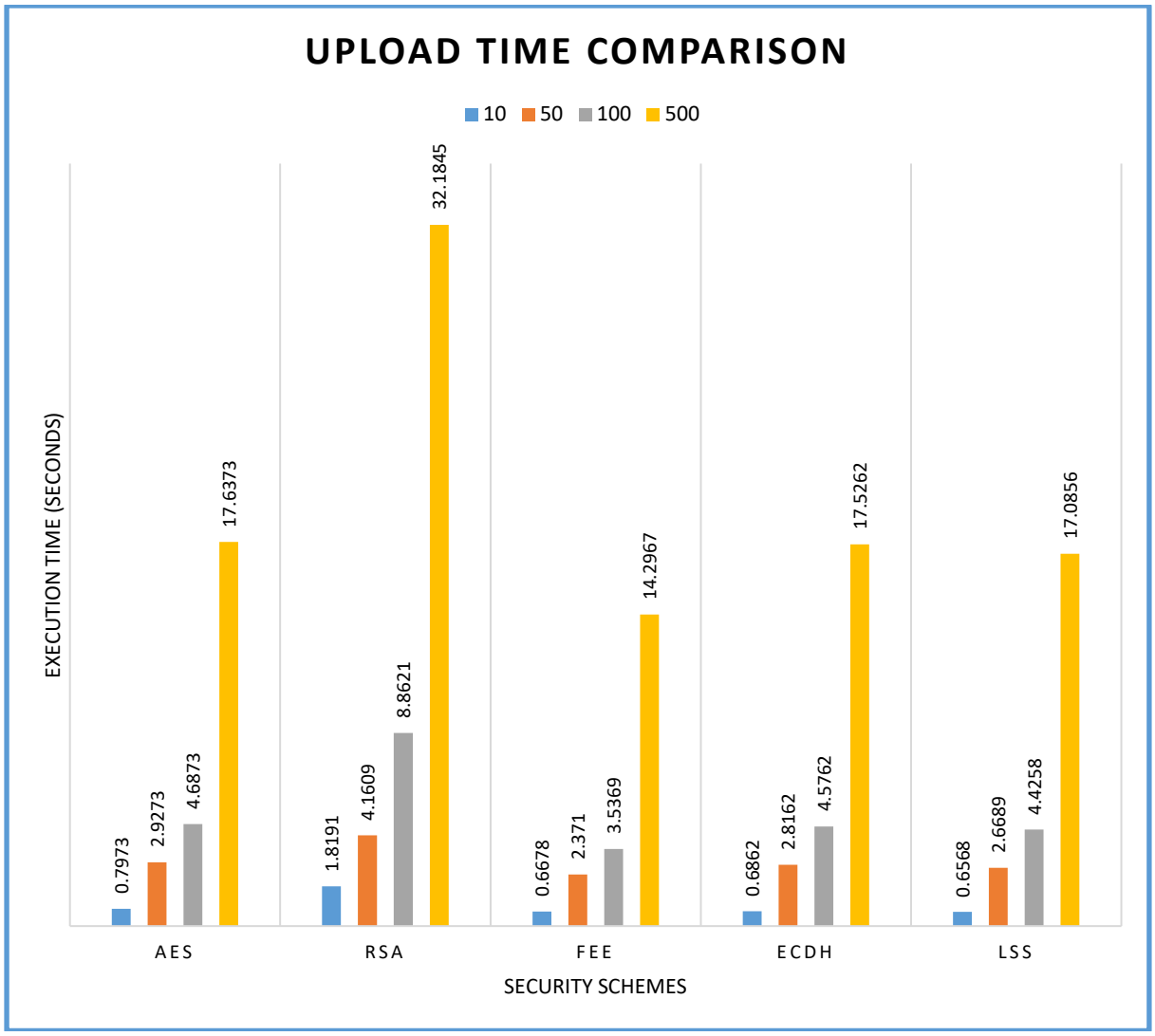

Figure 7: Total upload time comparison with the schemes in integrated architecture

As presented in Figure 7, the execution time for total upload time is evaluated. The security schemes used in the empirical study are provided in horizontal axis while the vertical axis shows the execution time for total upload time. The results revealed that the proposed methods such as FEE and LSS took relatively less time for uploading data. However, there is linear increased in the time taken as the data grows in size.

	Total Download Time (seconds)				
Data Size (MB)	**AES**	**RSA**	**FEE**	**ECDH**	**LSS**
10	1.0169	1.8191	0.9229	0.9058	0.8098
50	2.5348	4.1609	2.1967	2.6237	2.0856
100	3.9048	8.8621	3.6374	2.8937	2.5265
500	13.8648	32.1845	12.5679	13.7537	12.4568

Table 7: Total download time taken by security schemes

As presented in Table 7, the total download time for AES, RSA, FEE, ECDH and LSS are compared against different workloads.

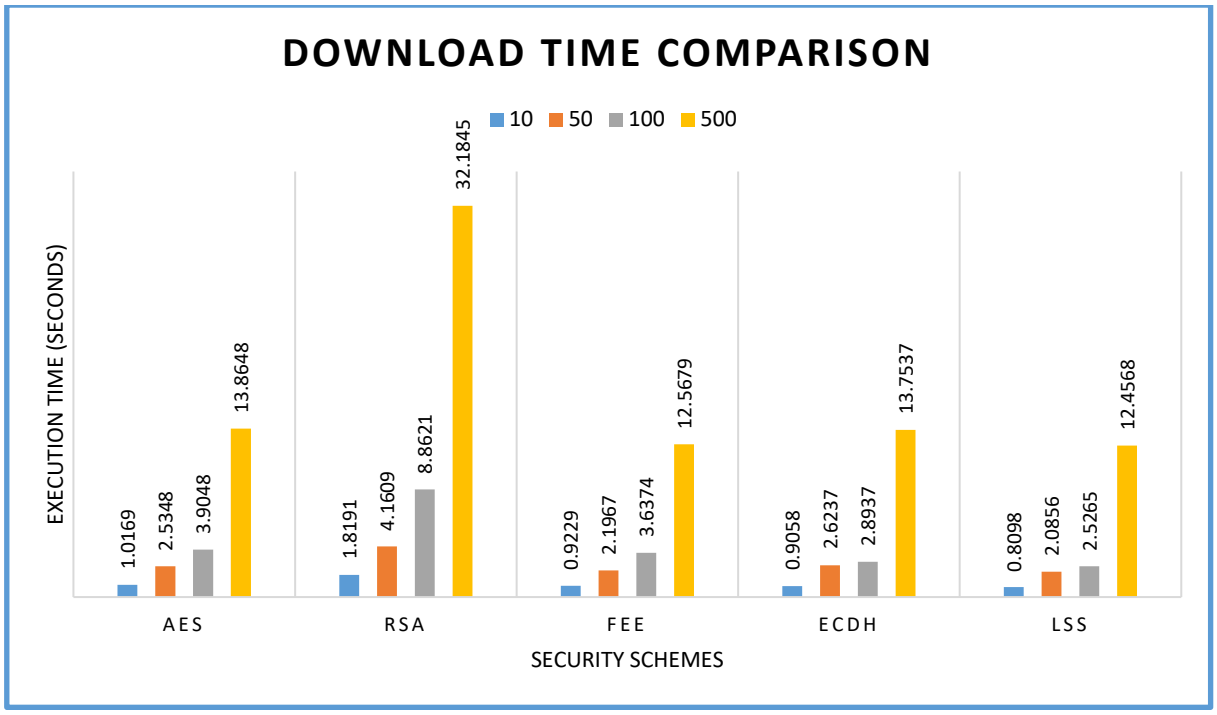

Figure 8: Total download time comparison with the schemes in integrated architecture

As presented in Figure 8, the execution time for total download time is evaluated. The security schemes used in the empirical study are provided in horizontal axis while the vertical axis shows the execution time for total download time. The results revealed that the proposed methods such as FEE and LSS took relatively less time for downloading data. However, there is linear increased in the time taken as the data grows in size.

7. PERFORMANCE EVALUATION

There are certain assumptions made in the empirical study. First, attackers have no direct access to data in HDFS. Second, users have access to code of map and reduce functions. Third, Map and Reduce functions in the MapReduce framwork can gain access to storge media and network. Fourth, there is secure communication among the nodes involved in the cluster. Fifth, any user of the system has normal network access previleges as end users. The attack model is as follows. Attackers gain access to map() function and encodes sensitive data to a key. Afterwards, the same key is sent to reduce() function. Reducer does not change key and finally it results in output. The presence of the key in the final output indicates attack is successful. This model is known as query based inference attack. When compared with the existing system named Airavat [25], the proposed system uses reducer analysis to know whether thre is a pre-registered pattern and the proposed algoritm is employed to apply noise to the data. Unlike Airavat, potentially malicious value is replaced by the value with noise to defeat privacy attack. Ther threat from Unique critial value used by adverasary is removed with thhe proposed algorithm. The usability of the proposed algorithm is found better than Airavat in case of prevention of query based inference attacks. The integrated framework provides improved security and privacy to big data.

8. THREATS TO VALIDITY

The proposed solution to prevent privacy attacks on big data has targeted query based inference attacks. The presence of a sensitive entity in the big data is interested by the attacker in this case. The proposed solution is based on the pre-registered reduce concept that assumes that the reducer pattern is known beforehand. This is a threat to validity of the proposed system if there is undetectable reducer pattern priori. Nevertheless, it the proposed algorithm is able to detect query based inference attacks with pattern analysis and noise addition of multiple modals. It is useful for preventing privacy attacks of that kind aforementioned. Another threat to validity of the system is that, the empirical study is made with 3 nodes in the cluster. This may appear less as thousands of commodity computers are involved in the real world cloud based distributed frameworks. However, it is to be understood that experiments are made with 3 low configured systems for developing proof of concept prototype that needs further enhancement to generalize the proposed solution to big data of different fields or domains.

9. CONCLUSIONS AND FUTURE WORK

In distributed programming frameworks, it is essential to protect data from untrusted or malicious code. MapReduce programming model is widely used for handling big data. However, there are number of security attacks on the big data. Our prior works [31] and [32] provided security enhancements to protect big data when it is in rest and when it is on transit. They also considered both structured and unstructured data for security besides supporting data dynamics on encrypted structured data. However, they do not cover the query based inference attacks when big data is subjected to data analytics. The proposed algorithm Multi-Model Defence Against Query Based Inference Attacks (MMD-QBIA) in this paper considers multiple models of preventing privacy attacks on

big data. The algorithm has different strategies for different scenarios. For protecting aggregate values produced by the reducer, it has provision for various procedures by adding appropriate noise. When there is inference attack exhibited by the absence of pre-defined mapper pattern, it invokes StrongNoiseAddition() that ensure privacy so as to prevent disclosure of sensitive data to adversaries. Cloudera Distribution Hadoop (CDH) is the environment used for empirical study. One real time dataset and one synthetic dataset are used for the experiments. Proof of concept prototype is made and the results revealed that the proposed system shows better usability over the existing system named Airavat in providing privacy protection to big data.Then integrated security architecture is evaluated with different schemes and found that the framework provides enhanced security and privacy to big data. In future, we intend to perform experiments with more machines in Hadoop cluster. Another direction for future work is to consider attacks other than query based inference attacks and improve our methodology to handle such attacks.

References

[1] Derbeko, P., Dolev, S., Gudes, E., & Sharma, S. (2016). Security and privacy aspects in MapReduce on clouds: A survey. Computer Science Review, 20, 1–28.
[2] Jain, P., Gyanchandani, M., &Khare, N. (2018). Differential privacy: its technological prescriptive using big data. Journal of Big Data, 5(1). P1-24.
[3] Jain, P., Gyanchandani, M., &Khare, N. (2019). Enhanced Secured Map Reduce layer for Big Data privacy and security. Journal of Big Data, 6(1). P1-17.
[4] Khezr, S. N., &Navimipour, N. J. (2017). MapReduce and Its Applications, Challenges, and Architecture: A Comprehensive Review and Directions for Future Research. Journal of Grid Computing, 15(3), 295–321.
[5] Ohrimenko, O., Costa, M., Fournet, C., Gkantsidis, C., Kohlweiss, M., & Sharma, D. (2015). Observing and Preventing Leakage in MapReduce. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15. P1-12.
[6] EDGAR Log File Dataset. Retrieved from https://www.sec.gov/dera/data/edgar-log-file-data-set.html. Accessed on 15 January 2020.
[7] Lee, Denny Guang-Yeu, "Protecting patient data confidentiality using differential privacy" (2008). Scholar Archive. 392. http://digitalcommons.ohsu.edu/etd/392
[8] Dwork, C. (2006). Differential Privacy. In: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP) (2), pp. 1–12.
[9] Greenberg (2016). Apple's 'differential privacy' is about collecting your data—but not your data 2016. Retrieved from https ://www.wired .com/2016/06/apple s-differenti al-priva cy-collecting -data/. Accessed 25 Jan 2020.
[10] Bhathal, G. S., & Singh, A. (2019). Big data: Hadoop framework vulnerabilities, security issues and attacks. Array, p1-8.
[11] SantoshAditham, NagarajanRanganathanand Srinivas Katkoori. (2018). Call Trace and Memory Access Pattern based Runtime Insider Threat Detection for Big Data Platforms, p1-13.
[12] SaraHsaini, SalmaAzzouzi and My El Hassan Charaf. (2018). A Secure Testing Based Approach for Mapreduce Frameworks. IEEE, p1-5.
[13] Gambs, S., Killijian, M.-O., Moise, I., & del Prado Cortez, M. N. (2013). MapReducing GEPETO or Towards Conducting a Privacy Analysis on Millions of Mobility Traces. 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum. P1-10.
[14] Yang, C., Huang, Q., Li, Z., Liu, K., & Hu, F. (2016). Big Data and cloud computing: innovation opportunities and challenges. International Journal of Digital Earth, 10(1), 13–53.
[15] Tien Tuan AnhDinh, PrateekSaxena, Ee-Chien Chang, Beng Chin Ooi, and Chunwang Zhang. (2015). M2R: Enabling Stronger Privacy in MapReduce Computation. 24th USENIX Security Symposium, p1-17.
[16] James Stephen, J., Savvides, S., Seidel, R., &Eugster, P. (2014). Program analysis for secure big data processing. Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering - ASE '14. P1-11.
[17] F. Resin Geyer, Gilles Fedak, Rafael Tim´oteo de Sousa Jr.,João Paulo C. L. Costa,RubemPereira, Paul Fergus, Anton Zaleski, Herman Vissia and Volker Markl. (2017). Security Framework for Distributed Data Processing, p139-144.
[18] Hu X, Yuan M, Yao J, Deng Y, Chen L, Yang Q, Guan H, Zeng J. Differential privacy in telco big data platform. ProcVLDB Endow. 2015;8(12):1692–703. https ://doi.org/10.14778 /28240 32.28240 67.
[19] Fan, J., Han, F., & Liu, H. (2014). Challenges of Big Data analysis. National Science Review, 1(2), 293–314.
[20] RafaelPires, Daniel Gavril, Pascal Felber, Emanuel Onica and Marcelo Pasin. (2017). A lightweight MapReduce framework for secure processing with SGX. 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, p1100-1107.
[21] Andrés ME, Bordenabe NE, Chatzikokolakis K, Palamidessi P. Geo-Indistinguishability: differential privacy for locationbasedsystems. In: ACM. ISBN: 978-1-4503-2477. https ://doi.org/10.1145/25088 59.25167 35. 2014.

[22] M.SadeghRiazi, ChristianWeinert, OleksandrTkachenko, Ebrahim M. Songhori, Thomas Schneider and FarinazKoushanfar. (2018). Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. ASIACCS, p707-721.

[23] Rongxing Lu, Xiaodong Lin and Xuemin (Sherman) Shen. (2013). SPOC: A Secure and Privacy-Preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS,24, p614-624.

[24] Juan Du, Dean, D. J., Yongmin Tan, XiaohuiGu, & Ting Yu. (2014). Scalable Distributed Service Integrity Attestation for Software-as-a-Service Clouds. IEEE Transactions on Parallel and Distributed Systems, 25(3), 730–739.

[25] Roy I, Setty STV, Kilzer A, Shmatikov V, Witchel E. Airavat: security and privacy for MapReduce. In: Proceedings of the7th USENIX symposium on networked systems design and implementation, NSDI 2010, San Jose, April 28–30. p. 297–12. 2010.

[26] Xue, K., & Hong, P. (2014). A Dynamic Secure Group Sharing Framework in Public Cloud Computing. IEEE Transactions on Cloud Computing, 2(4), 459–470.

[27] Li, M., Yu, S., Zheng, Y., Ren, K., & Lou, W. (2013). Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. IEEE Transactions on Parallel and Distributed Systems, 24(1), 131–143.

[28] Yang, H., & Lee, J. (2019). Secure Distributed Computing with Straggling Servers Using Polynomial Codes. IEEE Transactions on Information Forensics and Security, 14(1), 141–150.

[29] Dong, L., Lin, Z., Liang, Y., He, L., Zhang, N., Chen, Q., … Izquierdo, E. (2016). A Hierarchical Distributed Processing Framework for Big Image Data. IEEE Transactions on Big Data, 2(4), 297–309.

[30] McSherry F. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of communications of the ACM, vol. 53(9), 2010.

[31] N Sirisha 1,2, K V D Kiran 1,"An Efficient and Lightweight Security Scheme for Big Data", International Journal on Emerging Technologies 10(1): 01-03(2019)

[32] Sirisha N1,2 , K. V. D. Kiran2, "Flexible Scheme For Protecting Big Data And Enable Search And Modifications Over Encrypted Data Directly", Journal Of Mechanics Of Continua And Mathematical Sciences, Vol.-15, No.-4, April (2020) pp 294-312.

[33] R. Agrawal and R. Srikant. Privacy-Preserving Data Mining. In Proceedings of the ACM SIGMOD Conference on Management of Data. Dallas,Texas. May 2000. pp.439-450

[34] Sirisha, N., & Kiran, K. V. D. (2018). Authorization of Data In Hadoop Using Apache Sentry. International Journal of Engineering and Technology, 7(3), 234-236.

[35] Sirisha, N., Kiran, K. V. D., & Karthik, R. (2018).Hadoop security challenges and its solution using KNOX. Indonesian Journal of Electrical Engineering and Computer Science,12(1), 107-116.

[36] Sirisha, N., & Kiran, K. V. D. (2017). Protection of encroachment on bigdata aspects International Journal of Mechanical Engineering and Technology, 8(7), 550- 558.

[37] "A prediction scheme of mobility of cognitive femtocells LTE-A / LTE-UE under different speed scenarios",International Journal of Engineering and Technology(UAE),Volume 7, Issue 2, 2018, Pages 64-67,ISSN:2227524X

[38] "Prevention of Spoofing offensive in Wireless Sensor Networks",International Journal of Engineering and Technology(UAE),Volume 7, 2018, Pages 770-773,ISSN:2227524X

[39] Jyoti B. Kulkarni, Dr. Manna Sheela Rani Chetty, "Depth Map Generation from Stereoscopic Images Using Stereo Matching on GPGPU", Journal of Advanced Research in Dynamical and Control Systems, Volume 9, ISSN 1943- 023X, Special Issue – 02 / 2017.