PROTECTING PRIVACY OF BIG DATA IN PRESENCE OF UNTRUSTED MAPPER AND REDUCER

N. Madhusudhana Reddy
Assoc. Prof., Dept. of CSE
RGM College of Engineering and Technology(Autonomous)
Nandyal, AP, India.
Madhusudhan.nooka@gmail.com

Dr. C. Naga Raju
Assoc. Prof., Dept. of CSE
YSR Engineering College of Yogi Vemana University
Proaddatur, YSR district, AP, India.
cnrcse@yahoo.com

Dr. A. Ananda Rao
Prof in Dept. of CSE
Director of Academics and Planning,
JNTUA, Anantapuramu, AP, India.
akepogu@gmail.com

Abstract

The world is in the era where the Internet based data storage, retrieval and processing are prevailing. There is increasing demand for knowledge discovery from big data. In order to have comprehensive business intelligence from big data, MapReduce is the programming paradigm used. This kind of computation occurs in clusters of a large number of commodity computers in an economical fashion. However, source code written for mapper and reducer is untrusted and can lead to leakage of sensitive data. This is the potential and challenging privacy problem to be addressed. Anonymization techniques such as k-anonymity, t-closeness and l-diversity are used to anonymize sensitive information. Unfortunately, anonymization cannot provide the level of privacy needed. In this paper a methodology is proposed. It is to ensure that privacy is guaranteed in distributed computing frameworks such as MapReduce. The methodology is realized using the MapReduce framework of Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Empirical study revealed that our methodology is useful in privacy preserving big data mining.

Index Terms – Big data, big data mining, MapReduce framework, untrusted mapper and reducer, privacy of big data

1. INTRODUCTION

Technological innovations of late caused significant changes in computing. The phenomenon of computations done in remote servers is increasingly evidenced. This is due to the emergence of a novel computing model known as cloud computing [34] and its underlying technology known as virtualization [35]. As the cloud is a big pool of shared computing resources over Internet, a huge amount of data known as big data with characteristics such as volume, variety and velocity [6] is being outsourced to cloud in pay per use fashion. This realization of commoditization of computing resources is the reason for the world to have significant or unprecedented changes in the way of storing, sharing, and processing of data. Enterprises in the real world, governments and individuals are using cloud with different usage patterns. The services rendered by cloud are characterized by high performance, parallel processing and scalable storage. Developers, data analysts, and researchers across the globe can use cloud services without time and geographical restrictions.

Apart from plethora of advantages, computations using MapReduce paradigm brings many problems as well. Hadoop [32] is a distributed programming framework that supports MapReduce. The MapReduce framework [33] faces many security attacks such as Denial of Service (DoS), replay and eavesdropping. In addition to these attacks, MapReduce also faces privacy issue as malicious code MapReduce may leak sensitive data to adversaries by writing to a file or outputting specific value to indicate the presence or absence of sensitive data. Many techniques came into existence for anonymzing sensitive data before exposing data to the public or third

party for data mining. Very popular anonymization algorithms are k-Anonymity [18], l-diversity [16] and t-closeness [20]. The problem with cloud computing is that cloud users or service consumers like to spend as less time and effort as possible on secure computations in cloud computing. This is the big challenge for security and privacy in cloud computing. In [40] emerging technologies are reviewed on secure computations in the cloud.

Our focus in this paper is to preserve privacy of big data by protecting MapReduce application in the presence of untrusted mapper and reducer. A methodology is proposed which ensures privacy of big data being processed using MapReduce phenomenon. Our contributions in this paper are as follows.

- A methodology is proposed for privacy preserving big data processing to protect big data from malicious mapper or reducer. It is based on differential privacy.
- An algorithm known as MapReduce Privacy Protection (MPP) is proposed to exploit differential privacy in order to protect privacy of big data.
- The proposed methodology is realized using the MapReduce framework of Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Our empirical study revealed that our methodology is useful in privacy preserving big data mining.

The remainder of the paper is structured as follows. Section 2 provides review of literature on privacy preserving approaches, big data, cloud computing and differential privacy. Section 3 provides the problem statement. Section 4 presents the proposed methodology. Section 5 presents proposed algorithm. Section 6 provides experimental environment. Section 7 provides datasets used. Section 8 presents the results and empirical study while Section 9 concludes the paper besides providing directions for future work.

2. RELATED WORKS

This section provides review of related works. A review of security issues in big data processing is found in [7] and [9]. Xiao *et al*. [1] explored the differential privacy using wavelet transforms. They proposed a data publishing technique using ε-Differential Privacy. Li *et al*. [2], on the other hand used differential privacy for optimizing linear counting queries. Dwork and Naor [3] studied differential privacy and proposed a variant of it. It is known as differential privacy under continual observation for accurate information of system state. Li and Miklau [4] used differential privacy for accurate query answering. Their algorithm optimizes strategy with the help of differential privacy. Rosen *et al*. [5] proposed Iterative MapReduce for large scale processing of data using machine learning methods. Moca *et al*. [8] presented a distributed result checker in the MapReduce programming for privacy. Similar kind of work was carried out in [10] for verification and validation of MapReduce application in the Hadoop environment. In the same fashion, Huang *et al*. [14] also studied result verification schemes for MapReduce.

Grolinger *et al*. [11] focused on MapReduce challenges in big data. The challenges identified by them include data storage, analytics, online processing, privacy and security. Xie *et al*. [12] focused on optimizing performance of MapReduce while Qi *et al*. [13] provided information on implementations in MapReduce programming. Chatzikokolakis *et al*. [15] enhanced the functionality of differential privacy by using metrics. Holtz *et al*. [17] studied intrusion detection mechanisms in the MapReduce framework in distributed environment. Zhifeng Xiao and Yang Xiao [19] worked out on accountable MapReduce which takes care of machines to ensure that they are responsible for the given task.

Many researchers contributed towards privacy preserving data publishing (PPDP), privacy preserving data mining (PPDM) and privacy preserving distributed data mining (PPDDM). PPDP takes care of publishing of data with privacy given at the data level while PPDM and PPDDM focus on the privacy at process level. PPDM techniques are explored in [21], [22], and [26]. PPDP techniques are explored in [27], [28], [29] and [31]. Dwork *et al*. [23] and Dwork *et al*. [24] focused on statistical validity for privacy in case of adaptive analysis of big data. Clifton *et al*. [25] studied tools for PPDM including secure sum, secure set union, and secure size of set intersection. Wright *et al*. [30] focused on protocols for privacy preserving distributed data mining protocols such as Bayesian Network (BN) and anonymization techniques.

As found in the literature many techniques came into existence for privacy preserving data mining. However, protecting privacy of big data from malicious mapper and reducer functions provided by adversaries is the area which needed further research. Protecting mapper and reducer in the real distributed programming environment is an essential research activity it is an open problem to be addressed. Towards this end, in this paper, an algorithm is proposed based on differential privacy for protecting privacy of big data from malicious mapper and reducer.

3. PROBLEM FORMULATION

Consider an enterprise named KKS that has central repository of its sales data which contains information about customers and their transactions. The data assumed characteristics of big data. KKS could not maintain it in

local server. Therefore, the enterprise moved its data to cloud. KKS authorized DanTics Company to perform mining its data for obtaining trends in customer behaviour. The programmers at DanTics typically perform MapReduce computations in order to obtain business intelligence required by KKS. Therefore they implement interfaces such as mapper and reducer for the map and reduce methods of the framework. The source code written for mapper and reducer can have potential risk for KKS as the code may be intentionally or unintentionally malicious. Such code in mapper and reducer can steal sensitive data maintained by KKS. This can cause an embarrassing situation to KKS as it loses customer loyalty. Attackers can find the presence or absence of specific customers in the transactions and leak sensitive data in the form of output values, keys, set of key/value pairs, even relationships among different keys in the output.

```
1 public static class Map extends MapReduceBase implements
2       Mapper<LongWritable, Text, Text, IntWritable> {
3
4   private Text word = new Text();
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("Microsoft")){
15                          word.set("Macrosoft");
16                          output.collect(word, 1500000);
17                  } else {
18                           word.set(keyword);
19                          output.collect(word, 1);
20                          }
21                      }
22                  }
23 }
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>=1500000)){
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 }
```

List 1: Sample malicious MapReduce code

From List 1, the codes reveal the presence of malicious piece of code in both map and reduce functions. In line number 14, the attacker tries to find the presence of a specific customer “Microsoft”. Key variable is modified in line 15 and a big value is associated. In the reducer section line 33 finds whether there is the strange value set in line 15. If there is presence of a specific customer, output is set to a specific value in line 35 to indicate sensitive data thus violating the privacy of a customer. This is the problem to be addressed. The solution to this problem can have a huge impact on the big data infrastructure and its computations in secure and privacy preserving fashion. Section 3 provides methodology to address this problem.

4. PROPOSED METHODOLOGY

The purpose of this methodology is to have well-organized procedure that can help in enabling privacy of data in the map and reduce source codes of MapReduce application. It is believed that this is essential to protect data from malicious mapper or reducer. This is already discussed in section 3. In order to ensure privacy of data in MapReduce paradigm differential privacy is employed. A managed Hadoop framework of an Amazon Web Services (AWS) cloud is used for empirical study. The framework in AWS is a part of Amazon Elastic MapReduce (EMR) [36]. Since it is a management framework it is ready, fast and cost effective. Moreover, it can process a huge amount of data (big data) with high scalability and availability on Amazon EC2 instances. Moreover EMR can handle different use cases of big data such as ETL, log analysis and web indexing. Architecture of the proposed computation system is shown in Figure 1.

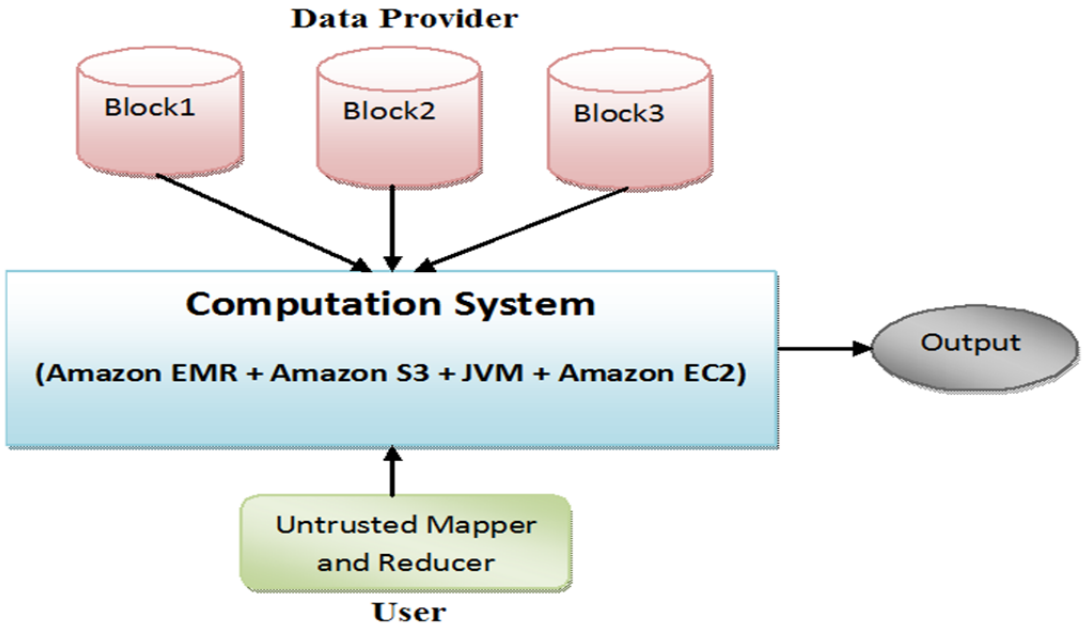

Figure 1: Architecture of the Proposed System

Differential privacy is used to ensure privacy of big data to protect it from malicious mapper or reducer. Privacy is nothing but non-disclosure of sensitive data. For instance, height of an individual may be private and sensitive. But the average height of individuals in India is definitely not private. Differential privacy is the framework that formalizes privacy and protects data from de-anonymization techniques. An algorithm is defined to achieve this. Considering two neighbouring datasets named D and D' where D' is obtained by changing a single tuple in D. A randomized algorithm A satisfies $\boldsymbol{\varepsilon}$-Differential Privacy for any two neighbouring datasets D and D' and for any output O of A.

$$\Pr[A(D)=0] <= \exp(\boldsymbol{\varepsilon}).\Pr[A(D')=0] \quad (1)$$

Here $\boldsymbol{\varepsilon}$ determines the degree of privacy protection.

In the context of large datasets or big data, sensitive personal information is increasingly common. This is true with all domains such as healthcare, social networks and even search engines. Most of the anonymization methods cause information loss to some extent when input data is transformed for preserving privacy. There are natural tradeoffs between privacy and information loss. With respect to output privacy, data perturbation when carried out indiscriminately, it leads to unpredictable impact. There are some reconstruction solutions. For instance Aggrawal and Srikanth [37] formulated reconstruction function as follows.

$$F x_i(a) = \frac{\int_{-\infty}^{a} fy\,(w1-z)\,fx(z)\,dz}{\int_{-\infty}^{\infty} fy\,(w1-z)\,fx(z)\,dz} \quad (2)$$

Givne a cumulative distribution function F_y and n random sample realizations such as X_1+Y_1, X_2+Y_2..., X_n+Y_n, F_x is estimated. The posterior distribution for given point is estimated as (2). In the same fashion, in order to estimate F'_x for given x_1+y_1, x_2+y_2, ..., x_n+y_n the distribution functions for all X_i is averaged as follows.

$$F x_i(a) = \frac{1}{n}\sum_{i=1}^{n} \dot{f} x_i = \frac{1}{n}\frac{\int_{-\infty}^{a} fy\,(w1-z)\,fx(z)\,dz}{\int_{-\infty}^{\infty} fy\,(w1-z)\,fx(z)\,dz} \quad (3)$$

By differentiating F_x, the corresponding density function f'$_x$ is obtained as follows.

$$F\dot{x}_i(a)=\frac{1}{n}\sum_{i=1}^{n}\dot{f}x_i=\frac{y\,(w1-a)fx(a)}{\int_{-\infty}^{\infty}fy\,(w1-z)fx(z)dz} \quad (4)$$

An important concern about the randomization approaches and the reconstruction problems is that they support reconstruction besides giving us information about original values, causing sensitive data leakage. This is the rationale behind the decision to use differential privacy for our solution. In (1) only one privacy parameter is used. For effective privacy, two parameters are considered as explored in [38]. A computation function satisfies $(\in, a)$-differential Privacy if, for all datasets D and D' whose only difference is a single item that is present in D but not in D' and for all outputs S⊆ Range(F). This is as achieved follows.

$$P_r[F(D)\epsilon S] \leq exp(\epsilon) \times P_r\left[F(\dot{D})\epsilon S\right] \quad (5)$$

According to the output of this computation, no one can tell whether any specific input value is used as the probability of producing this output is same even if that item does not exist. This kind of not being able to tell the presence or absence of an item is the goal of this privacy preserving big data processing which protects data from malicious mapper or reducer.

5. PROPOSED ALGORITHM

This section provides the proposed algorithm known as Noise Addition Based Differential Privacy (NADP). Assuming unique values are the target of adversaries to leak identity information, the algorithm is applied to values that appear only one time. EDGAR dataset, described in section 7, is used to evaluate this algorithm. It takes EDGAR dataset's summarized data and performs differential privacy on sensitive attributes. IP is the sensitive attribute considered in the dataset. Differential privacy is achieved by adding noise to reducer output so as to protect output from privacy attacks.

Notation	Meaning
STV	Values that appear only one time
MTV	Values that appear multiple times
V	Value set
V'	Values processed by reducer
R(x)	Returns a random value from -|x| to |x|
$\boldsymbol{\varepsilon}$	Privacy parameter set by data provider

Table 1: Notations Used in Algorithm

Algorithm 1: Noise Addition Based Differential Privacy (NADP) Algorithm

```
1   AddNoiseToData(key, <v1, v2, ..., vn>
2     V = <v1, v2, ..., vn>
3     for i in 1 to n
4       if vi in STV then
5         vi = Get(V)
6       end if
7     end for
8     V'= <v1, v2, ..., vn>
9     output = ReduceFunction(V') x (1+R(ε))
10    return output
```

List 2: proposed algorithm

As shown in List 2, the proposed algorithm is to apply differential privacy to data subjected to MapReduce programming. The data is verified to know values appear single time as adversaries target them to infer the presence of intended IP. Based on this assumption, the algorithm considers values that appear one time. For

such values differentia privacy is applied to protect data from malicious mapper and reducer. By adding noise to the output of reducer it is ensured that the privacy of output is protected. As too much noise can affect the accuracy of information, it is better to add enough noise.

A reducer gets values associated with keys for processing. The key and value lists are separated in the system. This avoids malicious users to exploit a key to manipulate values. Malicious user who wants to get sensitive information tries to find the presence an entity in the dataset. Towards this end, malicious user is likely to create a strange value that will be able to distinguish an entity from other entities. This is the actual plan of an adversary that needs to be addressed. The proposed algorithm employs differential privacy in the form of noise addition so as to overcome privacy issues. The utility of the algorithm in terms of privacy protection is shown in the section 8 of the paper.

6. EXPERIMENTAL ENVIRONMENT

Elastic MapReduce (EMR) of Amazon EC2 is used for empirical study. Amazon EMR is a managed cluster platform that helps in running distributed programming frameworks like Hadoop. Hadoop's MapReduce is the underlying MapReduce framework with EMR. This is used to study the proposed privacy mechanism used to preserve privacy of big data from malicious mapper and reducer. An Amazon Simple Storage Service (S3) is used to store input and output files. The concept of a cluster is central to Amazon EMR. A cluster is a collection of Amazon EC2 instances. Each instance of EC2 in a cluster is called a node. The nodes are of two types slave node and master node. EMR supports installing different software in nodes.

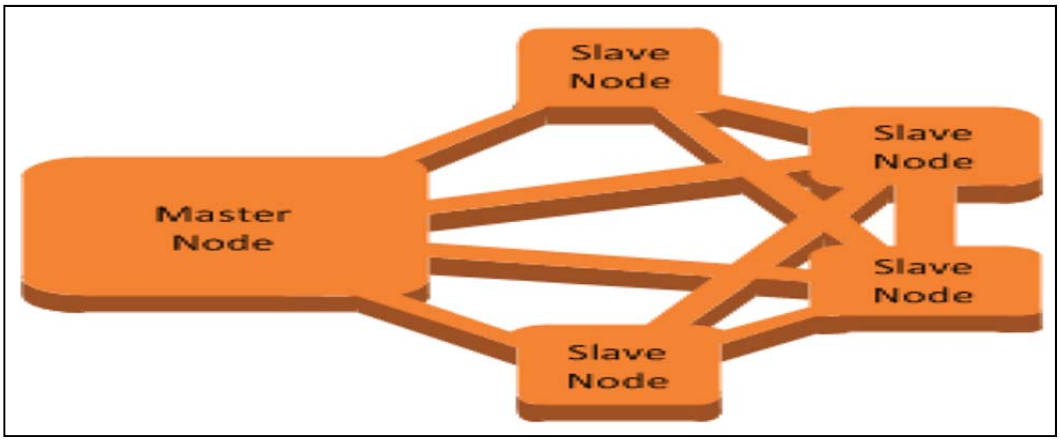

Figure 1: Overview of a cluster in Amazon EMR

From Figure 1, it is evident that there is one master node and many slave nodes. A node that manages a cluster is known as a master node. This node has software that helps in the distribution of data and tasks to slave nodes. Actual processing of data is done by slave nodes. The master node is responsible to keep track of the status of tasks and the health of a cluster. The slave nodes are of two types. They are core nodes and task nodes. Core node is a slave node that has software to run tasks and store data in HDFS while task nodes can only run tasks. Task nodes are optional.

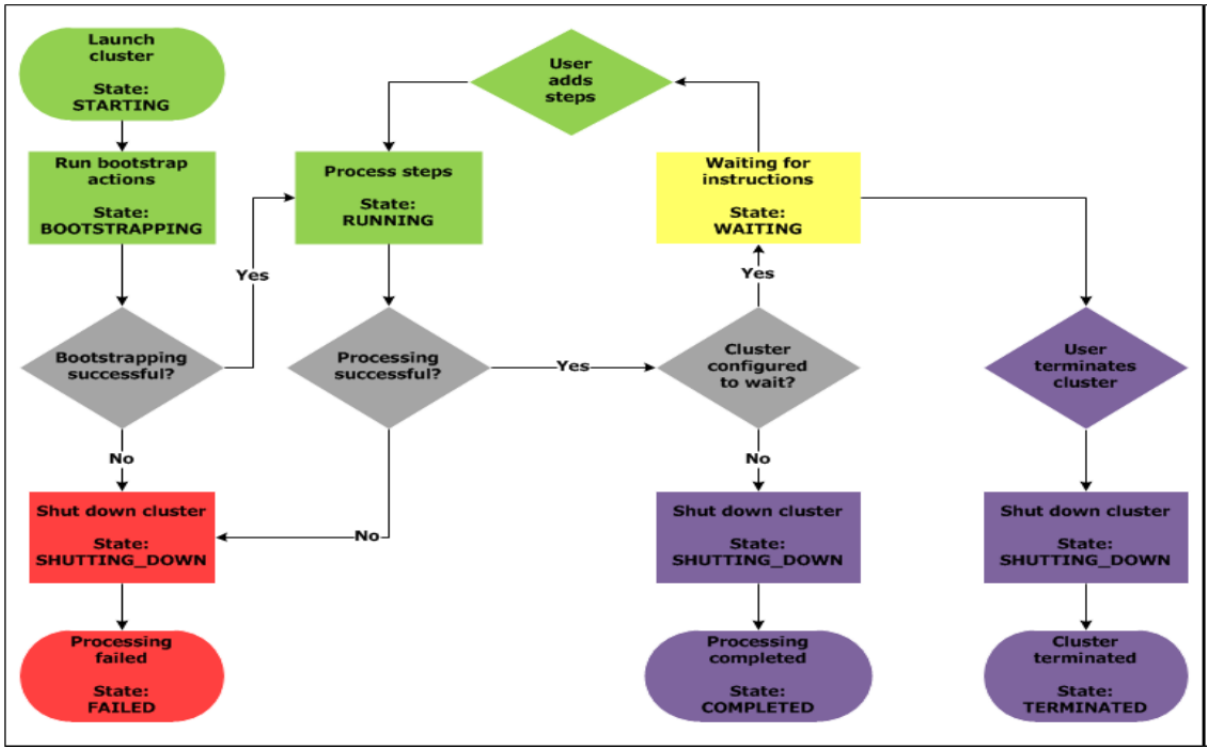

Figure 2: The Life cycle of a cluster

From Figure 2, it is evident that the life cycle starts with launching a cluster and ends with termination of cluster. When a cluster is launched, it is into STARTIGN state. If bootstrapping of a cluster is successful, it moves into running state. If the bootstrapping is not successful, then the cluster gets shutdown. In the running state if the processing is successful, cluster is configured successfully. If the cluster is not successfully configured, it may lead to shut down. After configuration, the cluster can take instructions and execute the same. Then user may terminate the cluster which leads to shut down and termination.

7. DATASET USED

Experiments in this paper need big data as the data is processed in a distributed programming environment. The dataset used is known as EDGAR dataset. EDGAR stands for Electronic Data Gathering, Analysis, and Retrieval. EDGAR is a system that automatically collects, indexes, accepts and forwards filings pertaining to various companies submitted to the U.S Securities and Exchange Commission (SEC). It other words, the dataset is related to SEC filings of public companies. Dataset is collected from [39] which is related to filings of a company named EDGAR. The dataset is assembled by Division of Economic and Risk and Analysis (DERA) based on Internet search traffic on EDGAR filings through the SEC web site named www.SEC.org. The dataset contains search information for 2016. It has 20160331 instances. The dataset is a log file which has attributes such as ip, date, time, zone, cik, accession, doc, code, size, idx, norefer, noagent, find, crawler, and browser.

The IP attribute contains an IP address of the machine from which search is made for EDGAR filings. Date attributes holds apache log file date. Time attribute holds apache log file time. Zone attribute holds zone information of apache log file. CIK is an attribute and it stands for Central Index Key (CIK). It is the index key associated with a document which is requested. In other words, it is the document accession number. Doc attributes provides the file name of a requested document. Code attribute provides a status code of the request. Filesize attributes holds the size of document file. Idx attribute holds 1 if the requester was able to reach an index page of documents else it holds zero. Norefer attribute holds 1 if the referrer log field is empty else it holds zero. Noagent attribute holds one if the user agent field is empty else it holds zero. Find attribute holds the value between 0 and 10 to indicate the presence of different character strings in the referrer field. Crawler attribute holds one if an user agent itself is a crawler else it holds zero. Browser attribute holds browser type such as chr to denote chrome. Out of these attributes, IP is considered as sensitive attribute as it can reveal the secrets of the users who made search on EDGAR filings. Therefore, it is subjected to differential privacy.

8. RESULTS

The MapReduce program used for the experiment takes the dataset and provides access count for each IP. The genuine mapper and reducer codes behave as expected. However, an adversary or attacker might have intention

to know the presence or absence of web access from specific IP address such as 104.40.128.114. To avoid this, the count function pattern is understood using a decompiler tool [41] and applied the proposed algorithm. The results are presented in the Table 1.

IP	Crawler	Genuine Count	Count in Presence of Attacker
101.81.76.106	0	12455	12454
104.40.128.107	0	9546	9545
104.40.128.108	0	10339	10340
104.40.128.109	0	12350	12349
104.40.128.110	0	14452	14451
104.40.128.111	0	7480	7479
104.40.128.112	0	13987	13986
104.40.128.113	0	12894	12893
104.40.128.114	0	13654	13653

As the algorithm applied its differential privacy, it is able to provide different value when there is attacker presence or in the presence of malicious code. The adversary thus cannot identify the presence or absence of the target IP, 104.40.128.114 availability in the data. Thus the differential privacy algorithm can protect data from malicious map and reducer codes. The differential privacy related noise is applied to actual count. The application for the value 13654 for an IP address 104.40.128.114 is as follows.

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

=count+ [(1+ $\boldsymbol{\varepsilon}$)+R]

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

=13653

9. CONCLUSION AND FUTURE WORK

Privacy issues of big data are studied in the presence of untrusted mapper and reducer. The malicious code used for mapper and reducer can lead to sensitive data leakage and misuse of information. As cloud computing became reality, enterprises are moving their data to cloud where storage and processing takes place. With this phenomenal change in computing, cloud also brings about privacy challenges. Specifically, MapReduce paradigm in distributed programming frameworks like Hadoop can cause the disclosure of sensitive information when mapper or reducer is under an influence of attack. In this paper a methodology is proposed for secure and privacy preserving computations in MapReduce framework. The methodology is based on our differential privacy algorithm. The methodology is realized in the MapReduce framework of Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Our empirical study revealed that our methodology is useful in privacy preserving big data mining. This research can be extended to have further optimization of security and privacy to MapReduce programming in the presence of untrusted mapper and reducer.

REFERENCES

[1] Xiaokui Xiao, Guozhang Wang and Johannes Gehrke. (2009). Differential Privacy via Wavelet Transforms. IEEE, p1-15.

[2] Chao Liy, Michael Hayy, Vibhor Rastogiz, Gerome Miklauy, Andrew McGregor. (2010). Optimizing Linear Counting Queries Under Differential Privacy. ACM, p1-22.

[3] Cynthia Dwork,Moni Naor,Toniann Pitassi and Guy N. Rothblum. (2010). Differential Privacy Under Continual Observation. ACM, p1-10.

[4] Chao Li and Gerome Miklau. (2012). An Adaptive Mechanism for Accurate Query Answering under Differential Privacy. ACM, p1-13.

[5] Joshua Rosen, Neoklis Polyzotis,Vinayak Borkar, Yingyi Bu, Michael J. Carey,Markus Weimer, Tyson Condie and Raghu Ramakrishnan. (2013). Iterative MapReduce for Large Scale Machine earning. ACM, p1-9.

[6] Avita Katal,Mohammad Wazid and R H Goudar. (2013). Big Data: Issues, Challenges, Tools and Good Practices. IEEE, p1-6.

[7] Priya P. Sharma and Chandrakant P. Navdeti. (2014). Securing Big Data Hadoop: A Review of Security Issues, Threats and Solution. International Journal of Computer Science and Information Technologies. 5 (2), p1-6.

[8] Mircea Moca, Gheorghe Cosmin Silaghi and Gilles Fedak. (2011). Distributed Results Checking for MapReduce in Volunteer Computing. IEEE International Parallel & Distributed Processing Symposium, p1-8.

[9] Diogo A. B. Fernandes • Liliana F. B. Soares • João V. Gomes , Mário M. Freire and Pedro R. M. Inácio. (2014). Security issues in cloud environments: a survey. Springer , p1-58.

[10] Amresh Kumar,Kiran M , Saikat Mukherjee and Ravi Prakash G.. (2013). Verification and Validation of MapReduce Program model for Parallel K-Means algorithm on Hadoop Cluster. International Journal of Computer Applications. 72 (8), p1-8.

[11] Katarina Grolinger,Michael Hayes,Wilson A. Higashino and David S. Allison. (2014). Challenges for MapReduce in Big Data. Electrical and Computer Engineering , p1-10.

[12] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin. (2010). Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters. ACM, p1-10.

[13] HAI JIN, SHADI IBRAHIM, LI QI, HAIJUN CAO, SONG WU and XUANHUA SHI. (2011). THE MAPREDUCE PROGRAMMING MODEL AND IMPLEMENTATIONS. Springer , p1-16.
[14] Chu Huang, Sencun Zhu and Dinghao Wu. (2010). Towards Trusted Services: Result Verification Schemes for MapReduce. Springer , p1-8.
[15] Konstantinos Chatzikokolakis, Miguel Andres, Nicolas Bordenabe, Catuscia Palamidessi. (2013). Broadening the Scope of Diferential Privacy Using Metrics. ACM, p1-34.
[16] Ashwin Machanavajjhala,Johannes Gehrke , Daniel Kifer Muthuramakrishnan and Venkitasubramaniam. (2012). ℓ-Diversity: Privacy Beyond k-Anonymity. ACM, p1-12.
[17] Marcelo D. Holtz, Bernardo M. David and Rafael Tim´oteo de Sousa J´unior. (2011). Building Scalable Distributed Intrusion Detection Systems Based on the MapReduce Framework. REVISTA TELECOMUNICAÇÕES. 13 (2), p1-11.
[18] Charu C. Aggarwal. (2011). On k-Anonymity and the Curse of Dimensionality. ACM, p1-9.
[19] Zhifeng Xiao and Yang Xiao. (2014). Achieving Accountable MapReduce in cloud computing. Future Generation Computer Systems. 30 , p1-13.
[20] Ninghui Li, Tiancheng Li and Suresh Venkatasubramanian. (2011). t-Closeness: Privacy Beyond k-Anonymity and -Diversity. ACM, p1-10.
[21] Chris Clifton. (2001). Privacy Preserving Distributed Data Mining. Computer Sciences p1-10.
[22] S.Selva Rathna, Dr. T. Karthikeyan. (2015). Survey on Recent Algorithms for Privacy Preserving Data mining. computer scince. 6 (2), p1-6.
[23] Cynthia Dwork, Vitaly Feldman, Moritz Hardt ,Toniann Pitassi, Omer Reingold and Aaron Roth. (2015). Preserving Statistical Validity in Adaptive Data Analysis. Computer Sciences p1-29.
[24] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann PitassiOmer Reingold and Aaron Roth. (2015). The reusable holdout: Preserving validity in adaptive data analysis. Computer Sciences. 349 ,p1-4.
[25] Chris Clifton, Murat Kantarcioglu, Xiaodong Lin, Michael and Y. Zhu . (2002). Tools for Privacy Preserving Distributed Data Mining. IEEE 4 (2) p1-7.
[26] V. Baby and N. Subhash Chandra. (2016). Privacy-Preserving Distributed A Survey Data Mining Techniques:. International Journal of Computer Applications. 143(10), p1-5.
[27] Pawel Jurczyk and Li Xiong. (2008). Privacy-Preserving Data Publishing for Horizontally Partitioned Databases. IEEE p1-2.
[28] BENJAMIN C. M. FUNG,KE WANG,RUI CHEN and and PHILIP S. YU. (2010). Privacy-Preserving Data Publishing: A Survey of Recent Developments. ACM. 42 (4), p1-53.
[29] V. V. Nagendra kumar and C. Lavanya. (2014). Privacy-Preserving For Collaborative Data Publishing. IJCSIT. 5 (3), p1-4.
[30] Rebecca N. Wright , Zhiqiang Yang and Sheng Zhong. (2006). Distributed Data Mining Protocols for Privacy: A Review of Some Recent Results. IEEE, p1-13.
[31] A N K Zaman and Charlie Obimbo. (2014). Privacy Preserving Data Publishing: A Classification Perspective. IJACSA. 5 (9), p1-6.
[32] The Apache Software Foundation. (2016). Welcome to Apache™ Hadoop. Available: http://hadoop.apache.org/. Last accessed 01 December 2016.
[33] Apache Software Foundation. (2016). MapReduce Tutorial. Available: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. Last accessed 01 December 2016.
[34] NIST (2016). Final Version of NIST Cloud Computing Definition. Available online at: https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published. Accessed on 01 December 2016.
[35] Malhotra, L., Agrawal, D. and Jaiswal, A. (2014). Virtualization in Cloud Computing. Information Technology & Software Engineering, 4 (2), p1-3.
[36] Amazon Web Services. (2016). Amazon EMR. Available: https://aws.amazon.com/emr/. Last accessed 01 December 2016.
[37] 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.
[38] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor.Our data, ourselves: Privacy via distributed noise generation. In EUROCRYPT, 2006.
[39] Securities and Exchange Commission. (2016). EDGAR Log File Data Set. Available: https://www.sec.gov/data/edgar-log-file-data-set. Last accessed 10th November 2016.
[40] Madhusudhan Reddy, N., and Nagaraju, C. (2015). Survey on Emerging Technologies for Secure Computations of Big Data. I-Manager's Journal of Cloud Computing, 2 (1), p1-6.
[41] Decompilers Onlne (2016). Available at: http://www.javadecompilers.com/jad [accssed on 10 November 2016]