CHAPTER 4
PROTECTING BIG DATA PRIVACY

Securing MapReduce computations in distributed programming frameworks are covered in Chapter 3. It covered a methodology to detect rogue worker nodes with different malicious behaviours. This chapter covers a methodology used for protecting privacy of big data. When big data is outsourced to cloud, the data owner does not have a local copy in general. This is the reason why the data owner is worried about security and privacy of data. With respect to security, cryptographic primitives are used while privacy of data is to be protected with high priority. A detail on security of big data is covered in methodologies presented in Chapter 5 and Chapter 6, while this chapter is limited to protecting privacy of big data. Disclosure of sensitive data to illegitimate users or adversaries has potential risk. Therefore non-disclosure of sensitive data is nothing but protecting privacy of data. As MapReduce computations are protected by identifying rogue nodes as presented in previous chapter, the protection of big data from privacy attacks is the main focus of this chapter. It also covers the formulation of the problem, an algorithm that works in MapReduce programming framework to protect privacy of big data, real cloud environment like Amazon Web Services (AWS) and experimental results. The empirical results show the potential utility of the proposed methodology which is based on differential privacy.

4.1 PRELIMINARIES

In the contemporary world, people witnessed technological innovations. With invent of the cloud computing, data is stored remotely using Internet based computing. Data, in other words, is outsourced to public cloud where computing resources are provided in pay as you go fashion. On demand computing resources are available round the clock in pay per use fashion. The cloud infrastructure is able to provide in-exhaustive storage and computing resources. There is growing trend of outsourcing data to cloud storage. The reason behind this is that organizations are giving importance to data and not willing to lose it. Therefore data is growing exponentially in such a way that it cannot be managed with local servers. Thus cloud
computing technology on top of virtualization came as a viable alternative. According to this solution, enterprises outsource data and computational power to cloud so as to have unlimited services and on-demand provisioning of computing resources.

Since big data is very huge and it grows continuously, storing it locally is not possible. Therefore, enterprises prefer outsourcing data to public cloud. The problem here is that cloud servers are treated as untrusted and data owners have privacy concerns. When the sensitive data is lost, it causes potential damage to an organization. Big data is used to perform business analytics and obtain business intelligence (BI). When data is outsourced, there is a possibility of misusing it thus privacy of data is lost. As enterprises may not be competent to have data analytics, they may outsource big data for data analytics too. This has potential to create privacy problems.

For data processing, map and reduce functions are written in Hadoop. The code written for the mapper and reducer might have untrusted code that may cause leakage of sensitive data. Leakage of sensitive data has potential risk that needs to be overcome. Many anonymization techniques came into existence. They are known as k-anonymity, l-diversity and t-closeness. These techniques are good in providing anonymity of data but the problem is with the utility and their shortcomings towards providing level of privacy. To overcome this problem, a methodology is proposed in this chapter to protect big data from privacy issues. The methodology works in a distributed programming framework like Hadoop as it is realized with MapReduce programming paradigm. Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3) are used for experiments.

### 4.2 BACKGROUND

The recent technology improvements in the area of storage and computing have provoked many fundamental changes in Information Technology (IT) departments of enterprises. The storage and computing strategies are changed from traditional approach to cloud based approach. Storage and computations are done in remote servers with cloud computing technology in place. It is the distributed computing phenomenon that paved way for major shift in the storage and computing
methods. The rationale behind this is the new programming model like MapReduce and new computing model such as cloud computing [128] which is on top of virtualization technology [129].

Since cloud provides a pool of computing resources that are scalable and cater to the needs of the entire world, it is a good strategy for companies to prefer remote storage and computations. There is economic feasibility as well. Due to commoditization of computing resources, it is inevitable to exploit the technologies to reap their benefits in the competitive business world. Businesses, enterprises, governments and individuals can exploit the power of cloud to grow in economy and other aspects of business. Organizations exhibit different usage patterns with respect to cloud computing. When the cloud is used to outsource data and perform mining operations for business intelligence, it is inevitable to share data to some external partner and there lies the risk. Privacy of data may be at stake. This is the cause of concern. Nevertheless, it is not possible to manage huge data without cloud support. Cloud services are characterised by parallel processing, scalability, availability, fault tolerance and high availability. In order to have these benefits, data is outsourced and such data may be used by data analysts, researchers and other companies without time and geographical restrictions. However, it is subjected to accessibility and the permissions given by data owners. Once data is permitted to have data mining on it, the third party having permissions on it may misuse it. There might be insider attacks on the privacy of data. This is the main focus of this chapter, where a solution is provided using differential privacy.

MapReduce programming paradigm has many advantages as explored in [130]. The reason is that, it can process big data and it is associated with cloud ecosystem. MapReduce framework may be vulnerable to security attacks such as replay, eaves dropping and Denial of Service (DoS). Additionally, it may be subjected to privacy attacks as well. Privacy attacks are made by leaking sensitive data to other people by writing data to an external file or by making an attack to know whether there is a presence of value in data or not. As specified in section 3.1 there are many anonymization techniques to protect data from privacy attacks. Especially when data is given to third party for data mining, it may be subjected to privacy attacks. When it
comes to cloud computing, it is important to understand that data owners do not spend time on security provisions generally. They wanted the cloud to take care of it. This is a very challenging situation where protecting privacy of big data needs to be given highest priority. Many secure computation techniques are found in [124]. However, this chapter provides a methodology to protect big data from privacy attacks based on differential privacy.

This chapter presents a methodology for protecting privacy of big data in presence of untrusted mapper and reducer. The methodology is based on differential privacy. MapReduce Privacy Protection (MPP) algorithm is proposed to exploit differential privacy technique for providing privacy to big data. MapReduce programming framework in Amazon EC2 is used to have experiments. Results revealed the utility of the proposed methodology.

4.3. PROBLEM FORMULATION

This section provides a case study that exhibits the need for protecting privacy of big data. An enterprise by name KKS is considered. It has its entire data stored in a central repository. The database contains sensitive data of customers and their transactions. Since the data became very huge and acquired quality of big data, the company was not able to manage it properly in the central repository. For this reason, the company wanted to outsource its data to public cloud like AWS. After placing their data in public cloud, the company appointed a third party known as DanTics and authorized it to use its data for data analytics. The intention of data analytics was to understand the customer behaviour and improve business further.

In this context, the data analytics are performed by the third party in a MapReduce framework in order to provide the enterprise with acquired business intelligence. The third party implemented map and reduce interfaces for data analytics. If the code is malicious either intentionally or unintentionally it may leak privacy of outsourced data. This is the problem to be addressed.
Table 4.1: Sample malicious MapReduce code

```java
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
    private Text word = new Text();
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            String keyword = tokenizer.nextToken();
            if (keyword.compareTo("Microsoft")) {
                word.set("Macrosoft");
                output.collect(word, 1500000);
            } else {
                word.set(keyword);
                output.collect(word, 1);
            }
        }
    }
}

public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            int rValue = values.next().get();
            if (rValue>=1500000){
                key.set("Sazaki");
                sum = 12345678;
                break;
            } else {
                sum += rValue;
            }
        }
        output.collect(key, new IntWritable(sum));
    }
}
```

The third party may be trustworthy to the enterprise, but its employees may launch insider attacks to steal information or to get monetary benefits by leaking sensitive data. Such potential risks needs to be eliminated. Attackers may launch
privacy attacks to know presence or absence of particular customer. This causes risk to the enterprise that outsourced big data. The attackers may write output to separate storage instead of legitimate storage area. The key/value pairs found in the generated output may disclose sensitive information.

It is understood from the Table 4.1 that there is malicious piece of code in map and reduce functions. The attacker is trying to infer the presence of a particular customer like “Microsoft”. This is made in line number 14 of the source code. In line number 15, a key variable is updated to have a big value stored in it. It is done in the map function. In the reduce function as per line number 33, there is the discovery of the fact to know whether a strange value is set in the line number 15. In case the specified customer is present in the data, the output is determined to specific value. When such value is found in the output, it does mean that there is inference attack made successfully. Thus the sensitive data is leaked. This is the potential problem with privacy of big data to be addressed. When the privacy of a customer is lost, it may cause legal issues to the enterprise that outsourced the data to public cloud. Since the problem is related to the entire community that outsources data to a public cloud, solution to this problem has a huge impact. The methodology to solve the problem is provided in section 4.4.

4.4. PROPOSED METHODOLOGY

This section provides the proposed methodology in detail. It provides a procedure to enable privacy to big data processed using MapReduce operations in a Hadoop kind of distributed programming environment. The solution needs cloud ecosystem where big data, cloud and data analytics are possible to exploit the resource risk environment. The MapReduce programming framework has an application that contains both map and reduce functions. The source code written for a map or reduce function may have a malicious piece of code that can cause leakage of sensitive data.

Therefore, it is essential to overcome this problem. In this chapter, a solution is proposed based on differential privacy. AWS cloud has its managed MapReduce framework known as Elastic MapReduce (EMR) [131]. As EMR is in a managed environment, its usage is very fast and cost effective in nature.
EMR is provided by AWS for processing huge amount of data with features like fault tolerance, high performance, high scalability and availability. The EMR runs in Amazon EC2 instances. Different use cases related to big data are supported by EMR. They include web indexing, log analysis and ETL (Extract, transform and load) to mention a few. Figure 4.1 shows the architecture of the proposed system. Architecture is determined with a computation system in the distributed environment which covers EMR, S3 and EC2.

![Architecture of the Proposed System](image)

**Figure 4.1: Architecture of the Proposed System**

The computation system is the cloud based approach in which Amazon EMR, Amazon S3 and Amazon EC2 are involved. The EC2 is the cluster instance that contains multiple nodes associated. Amazon S3 is used to store unstructured data in the form of files. The EMR is used to have MapReduce computing with map and reduce tasks. Differential privacy is used to achieve privacy of big data. Non-disclosure of big data even in the presence of malicious mapper or reducer is the aim of the proposed methodology. Here is some description about the privacy. Consider the height of a person to be sensitive data and private in nature. In this context
differential privacy has the ability to formalize and protect privacy of the data from various privacy attacks.

In order to achieve the aforementioned objective, an algorithm is proposed. According to the algorithm, two datasets are considered. They are denoted as $D$ and $D'$ where $D$ is the original dataset while $D'$ is derived by making changes to the data for privacy. In this case, consider a randomized algorithm denoted as $A$ is able to satisfy Differential Privacy for the two datasets known as neighbouring datasets $D$ and $D'$ and this is true for the output $O$ of the algorithm $A$.

$$\Pr[A(D)=O] \leq \exp(\varepsilon).\Pr[A(D')=O] \tag{4.1}$$

The degree of privacy protection is denoted as $\varepsilon$.

Since the aim of the algorithm is to deal with big data, it is very common for the data to have sensitive information. It can be applied to data of any domain such as social networks, healthcare, banking and so on. Many anonymization algorithms that provide privacy to data may cause information loss. This loss is due to the transformation of input data for providing privacy to it. Therefore, there exist tradeoffs between information loss and privacy level of data. There is a danger when the tradeoffs are not balanced as it leads to losing utility of anonymized data and defeat the main purpose of privacy protection. The reconstruction function proposed in [132] shows this fact.

$$F_{X_i}(a) = \int_{-\infty}^{\infty} \frac{1}{n} \frac{f_Y(w-1-z) f_X(z) dz}{\int_{-\infty}^{\infty} f_Y(w-1-z) f_X(z) dz} \tag{4.2}$$

Given $n$ number of random sample realizations and cumulative distribution function $F_y$, the sample realisations are estimated as $x_1+y_1$, $x_2+y_2$, ..., $x_n+y_n$ and $F_x$. As estimate in Equation (4.2), the posterior distribution is provided. Similarly the estimation of $F'_{x}$ for $x_1+y_1$, $x_2+y_2$, ..., $x_n+y_n$ is provided as follows.

$$F_{X_i}(a) = \frac{1}{n} \sum_{i=1}^{n} f_{X_i} = \frac{1}{n} \int_{-\infty}^{\infty} \frac{f_Y(w-1-z) f_X(z) dz}{\int_{-\infty}^{\infty} f_Y(w-1-z) f_X(z) dz} \tag{4.3}$$

Density function $f'_{X}$ is obtained as follows by differentiating the $F_x$. 
The randomization approaches and reconstruction problems throw an important problem that is to provide some information besides anonymization data causing leakage of sensitive data. This is the reason such approaches are not considered for the proposed solution. Only one parameter related to privacy is used in Eq. 4.1 while two parameters are studied in [133]. Therefore $(\varepsilon, \delta)$-differential Privacy is the construct that satisfied computation function that consider both $D$ and $D'$ when their difference is a single item that is associated with $D$ and not with $D'$. It is true for all outputs such as $S \subseteq \text{Range}(F)$. Eq. (4.5) is used to achieve this.

$$P_r[F(D)eS] \leq \exp(\varepsilon) \times P_r[F(D)eS]$$

As per the results of the computation, it is not possible to say with any probability that whether there is usage of any specific input value for producing the desired output. This kind of quality does not give the ability to find the presence or absence of any data item, is the main goal of the proposed differential privacy construct that is employed in MapReduce programming paradigm with Amazon EMR.

4.5. PROPOSED ALGORITHM

The proposed algorithm for protecting big data in the presence of malicious reducer and mapper in MapReduce framework is described here. The proposed algorithm is named as Noise Addition based Differential Privacy (NADP). This algorithm assumes unique values expected by the adversaries while making leakage attacks. Identity leakage is the aim of adversaries. The algorithm is employed to a big data dataset collected from EDGAR website. It is described in section 4.7 and used for evaluating the proposed algorithm. The proposed algorithm takes EDGAR dataset as input and produces another dataset after subjecting it to the proposed differential privacy. It is employed on sensitive attributes in order to have benefits of privacy protection.
Table 4.2: Proposed Noise Addition Based Differential Privacy (NADP) Algorithm

Notations Used in Algorithm

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STV</td>
<td>Values that appear only one time</td>
</tr>
<tr>
<td>MTV</td>
<td>Values that appear multiple times</td>
</tr>
<tr>
<td>V</td>
<td>Value set</td>
</tr>
<tr>
<td>V’</td>
<td>Values processed by reducer</td>
</tr>
<tr>
<td>R(x)</td>
<td>Returns a random value from -</td>
</tr>
<tr>
<td>𝜀</td>
<td>Privacy parameter set by data provider</td>
</tr>
</tbody>
</table>

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’) * (1+R(𝜀))
10 return output

In step 1 of the algorithm all values associated with key are assigned to a vector V. From line 3 to 7, there is an iterative process to obtain values to be subjected to noise. In step 8, an initial output vector is produced. The noise additive parameter is used in step 9 of the algorithm. It is read as “output = ReduceFunction(V’) * (1+R(𝜀))”. This line is responsible for adding noise to output before it is sent to storage media or output device. The epsilon value is 8.85x10⁻¹². Randomness required by the method is introduced using R parameter. When the output is subjected to noise, the privacy attack to know presence or absence of a company in terms of its IP address is defeated. Illustration of this algorithm on the noise addition is provided in section 4.8 where an IP address is discussed.
The differential privacy to big data is provided by adding noise to the output produced by reducer. In the EDGAR dataset, IP address is very sensitive attribute that may be subjected to leakage. Therefore noise related to differential privacy is provided to reducer’s output thus the generated output is protected from adversaries.

From the Table 4.2, it is evident that the algorithm is meant for applying differential privacy to big data in the context of MapReduce programming. The value that may appear single time is the target of adversaries. Adversaries often target them because they can infer the presence or absence of an entity or customer or value easily with this kind of attack. As per the EDGAR dataset, IP address is one such unique value that is the target of adversaries. In other words, the adversaries wanted to know the presence or absence of an IP address in the given data. This assumption is used by the algorithm and works accordingly. Differential privacy is applied to such target values to protect data form malicious reducer or mapper. The output of reducer is subjected to noise so as to ensure that the privacy of the data is not lost. It is important to have control over noise as too much noise can affect the utility of the output information. Hence it is important to understand what enough noise is.

After completion of map phase, the reducer is given key/value pairs for further processing. The system has lists of key/value pairs with separate key and value lists to avoid exploitation of values with manipulation. Malicious users try to obtain sensitive information such as the presence or absence of an entity that is, IP address in the EDGAR dataset. In order to achieve this, attacker may create a storage value that can be used to distinguish the entity from other values. This is actually the planned activity of adversary to know the presence or absence of a target value in the dataset. To defeat this plan, the proposed algorithm employs noise addition approach and protects privacy of big data.

4.6. EXPERIMENTAL ENVIRONMENT

Real cloud environment like Amazon AWS is chosen for experiments. EC2 is the cluster instance on which EMR and S3 are available as shown in Figure 4.1. Amazon EMR is the MapReduce programming framework that runs on top of EC2. It is similar to Hadoop which is used for experiments as described in Chapter 3. EMR is
one of the distributed programming frameworks that provide speed, high availability, scalability and fault tolerance as it is in a managed environment. EMR also has underlying MapReduce framework of Hadoop. EMR is used for experiments as it supports MapReduce programming. The experiments are meant for evaluating the proposed methodology that is to protect privacy of big data. In EC2 cluster, there are two kinds of commodity computers known as master and slave.

![Diagram of a cluster in Amazon EMR](image)

Figure 4.2: Overview of a cluster in Amazon EMR

As shown in Figure 4.2, there is one master node and multiple slave nodes. Slave nodes can communicate with master nodes and they can also communicate with other slave nodes. The entire cluster is managed by master node. This node is equipped with the software that can help in controlling slave nodes and allocate data and tasks to them. Actual data processing is the job of slave nodes. The task allocation and tracking the status of them is the job of master node. It also maintains the health of the underlying cluster. The slave nodes are divided into two categories. They are known as task nodes and core nodes. Core nodes are the slaves that can run tasks with some special software and they can also store data in HDFS. On the other hand the task nodes can run tasks only with optional presence in the cluster.
Cluster is a central theme of cloud platform. The life cycle of a cluster is shown in Figure 4.3. Every cluster has its own life cycle and EC2 is no exception. EC2 cluster starts with its life with launching the cluster. The life ends when the cluster is terminated.

![Figure 4.3: Life cycle of a cluster](image)

On starting a cluster, it is said to be in STARTING state. Once bootstrapping process is completed it enters into another state known as running state. Bootstrapping is crucial for a cluster to live and function well. If that fails, the cluster needs to be shut down. In the running state, it involves in certain processing. If the process is done without failure, the cluster is configured successfully. If the configuration of cluster is not successful then it leads to shut down. Once configuration is completed, the cluster can have instructions and execute them. Users also can terminate clusters that results in shutdown and termination.
4.7. DATASET USED

Dataset is crucial for the experiments to evaluate the proposed methodology. The dataset needs to be big data as it has to be given to MapReduce programming framework that is Amazon EMR. The dataset runs in distributed environment and it will be processed by multiple worker nodes in the EC2 cluster. The dataset used for the experiments is called EDGAR dataset. It has data related to Electronic Data Gathering, Analysis and Retrieval which is part of a system that has automated approach in collecting, indexing and accepting filings related to companies in the USA that are submitted to Securities and Exchange Commission (SEC) of the USA.

SEC filings are there in dataset and in that, IP address is considered sensitive as it can provide some inference to adversaries. Dataset is collected from [134]. The dataset is created and provided by Division of Economic Risk Analysis (DERA). It is created based on the search traffic on the internet related to SEC EDGAR filings made through a web site known as www.SEC.org. The dataset used for experiments has search data of 2016 with as many as 20160331 instances. In fact it is a log file that contains attributes like **IP**, **date**, **time**, **zone**, **CIK**, **accession**, **doc**, **code**, **size**, **idx**, **norefer**, **noagent**, **find**, **crawler**, and **browser**. Out of all the attributes the **IP** attribute is sensitive. This attribute contains IP addresses of machines that are participated in the search process and resulted in EDGAR filings. Thus it becomes sensitive and target attribute to adversaries.

The date attribute holds the date in which log entry is made. The time attribute in the same fashion holds the timestamp of the log entry. **Zone** is another attribute that contains zone information as part of log entries. **CIK** stands for Central Index Key that reflects index of associated document. It is also understood as accession number of document.

**Doc** attribute stores the name file requested in search. The status code of the request is stored under the **code** attribute. **Size** attribute contains size of file. **Idx** is an attribute that contains either 1 or 0. The former indicates that the request successfully reached index page while latter indicates the opposite. **Norefer** is an attribute that contains either 1 or 0. Zero is stored when the log entry is empty, if not, it holds 1.
Noagent is another attribute that contains zero when the field is empty. Find is another attribute that can hold any value between 1 and 10 to reflect the presence of different strings associated with the referrer field. Crawler is the attribute that stores 1 if there is any user agent otherwise it holds zero. The browser type is stored in the browser attribute. After studying the dataset, it is understood that IP address is very sensitive and needs to be protected. It can reveal the secrets of the users and their identity that made the search on EDGAR filings. Therefore, the IP attribute is treated with the proposed differential privacy. The experimental results are found in the following section.

4.8. RESULTS

The EDGAR dataset is subjected to big data processing and the IP address column is used for differential privacy to protect it from malicious attacks. The purpose of the MapReduce application is to return access count for each IP address.

<table>
<thead>
<tr>
<th>IP</th>
<th>Genuine Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>101.81.76.106</td>
<td>12455</td>
</tr>
<tr>
<td>104.40.128.107</td>
<td>9546</td>
</tr>
<tr>
<td>104.40.128.108</td>
<td>10339</td>
</tr>
<tr>
<td>104.40.128.109</td>
<td>12350</td>
</tr>
<tr>
<td>104.40.128.110</td>
<td>14452</td>
</tr>
<tr>
<td>104.40.128.111</td>
<td>7480</td>
</tr>
<tr>
<td>104.40.128.112</td>
<td>13987</td>
</tr>
<tr>
<td>104.40.128.113</td>
<td>12894</td>
</tr>
<tr>
<td>104.40.128.114</td>
<td>13654</td>
</tr>
</tbody>
</table>

The behaviour of genuine mapper and reducer is as expected. But the mapper and reducer that are under the influence of an attack can have malicious intentions. They have intention to know the presence or absence of an IP address such as 104.40.128.114. In order to protect IP address from malicious privacy attacks, the pattern of cloud function is ascertained with the help of a decompiler tool [135]. Then that knowledge is applied to the proposed algorithm. The output of the differential
privacy is as in Table 4.3. As shown in Table 4.3, it is evident that the IP addresses are in the first column and the resultant count is in the second column. The result is the genuine count as it is in the absence of attacker and the proposed differential privacy is not employed.

Table 4.4: Results of MapReduce in presence of attacker

<table>
<thead>
<tr>
<th>IP</th>
<th>Count in Presence of Attacker</th>
</tr>
</thead>
<tbody>
<tr>
<td>101.81.76.106</td>
<td>12454</td>
</tr>
<tr>
<td>104.40.128.107</td>
<td>9545</td>
</tr>
<tr>
<td>104.40.128.108</td>
<td>10340</td>
</tr>
<tr>
<td>104.40.128.109</td>
<td>12349</td>
</tr>
<tr>
<td>104.40.128.110</td>
<td>14451</td>
</tr>
<tr>
<td>104.40.128.111</td>
<td>7479</td>
</tr>
<tr>
<td>104.40.128.112</td>
<td>13986</td>
</tr>
<tr>
<td>104.40.128.113</td>
<td>12893</td>
</tr>
<tr>
<td>104.40.128.114</td>
<td>13653</td>
</tr>
</tbody>
</table>

As shown in Table 4.4, it is evident that the IP addresses are in the first column and the resultant count is in the second column. The result is the count in presence of attacker and when the proposed differential privacy is applied. The differential privacy is applied by the proposed algorithm to provide changed value when there is an attack. This will not enable adversary to know the presence or absence of an IP address targeted that is, in case of, 104.40.128.114. Similarly, the algorithm protects all IP addresses from privacy attack intended to know the presence or absence of an IP address in the dataset. Since the IP address in EDGAR dataset has some indication to adversaries such attacks are made in general. Therefore, it is understood that the proposed differential privacy algorithm is able to protect data from privacy attacks made by malicious mapper and reducer. The proposed algorithm applies noise to the actual count in order to protect the sensitive data that is IP address. As shown in Table 4.3, the count value is 13654 and corresponding IP address is 104.40.128.114. According to the algorithm the functionality of adding noise and protecting big data from privacy attack, an illustration is made as follows.
\[ \varepsilon = 8.85 \times 10^{-12} \]

Count in presence of attacker = count+ [(1+ \( \varepsilon \))+R]

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

=13653

This illustration shows the difference in the entity in order to protect the disclosure of presence or absence of the IP address 104.40.128.114.

**Figure 4.4:** Results of MapReduce application with and without privacy

As shown in Figure 4.4, it is evident that the IP address of EDGAR dataset has different sensitive values that are presented in horizontal axis while the vertical axis provides MapReduce result with respect to the count of IP in the EDGAR dataset. The results show the stacked line graph with values that are observed with genuine count and count in presence of an attacker (differential privacy is applied).
Table 4.5: Observations on performance of MapReduce with and without privacy

<table>
<thead>
<tr>
<th>Data Size (GB)</th>
<th>Time taken for execution (Sec)</th>
<th>With Privacy</th>
<th>Without Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>107</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>143</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>197</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>259</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>310</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>355</td>
<td>340</td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>417</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>440</td>
<td>420</td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>510</td>
<td>490</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>525</td>
<td>510</td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 4.5, observations are made on the MapReduce performance in terms of time taken for data processing against different data sizes with and without privacy. When privacy is applied, it adds computational complexity and also time taken is more.

Figure 4.5: Data size vs. MapReduce execution time with and without privacy
As presented in Figure 4.5, it is evident that the data size in GB is taken in horizontal axis. The values are taken from 50 GB to 500 GB incremented by 50 GB gradually. The vertical axis shows the time taken for execution. Experiments are made to see the performance with and without applying privacy.

The results showed important observations. The first one is that as data size increases, its execution time is increased gradually. The second observation is that the execution time is more when differential privacy is applied to big data.

**SUMMARY**

This chapter has covered the proposed methodology for protecting big data from privacy attacks. After addressing the problem of malicious mapper and reducer to launch malicious attacks in Chapter 3, this chapter has thrown light into the privacy of big data in MapReduce programming. It studies privacy issues in the presence of malicious mapper and reducer. The code which is used in the functions of map and reduce may have malicious code intentionally. Such code may cause leakage of sensitive data. As enterprises outsource their big data and analytics to public cloud, it is important to safeguard privacy of data in the presence of untrusted mapper and reducer. A methodology is proposed to prevent privacy attacks on big data in MapReduce computations. It is based on differential privacy that is described in this chapter. An algorithm known as Noise Addition based Differential Privacy (NADP) is proposed to achieve this. MapReduce programming framework such as Amazon EMR is used in the confines of EC2. The data is stored in S3 and output is also stored in S3 which is similar to HDFS of Hadoop. Experimental results revealed that, the adversaries cannot disclose sensitive information such as the presence or absence of IP address (sensitive attribute) in EDGAR dataset. Chapter 5 throws light into big data storage with a comprehensive methodology.