CONTENT SENSITIVITY BASED ACCESS CONTROL MODEL FOR BIG DATA

By

ASHWIN KUMAR THANDAPANI KUMARASAMY

Bachelor of Technology in Information Technology
Anna University
Chennai, Tamil Nadu, India
2010

Master of Science in Computer Science
Oklahoma State University
Stillwater, Oklahoma
2013

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2017

CONTENT SENSITIVITY BASED ACCESS CONTROL

MODEL FOR BIG DATA

Dissertation Approved:

Dr. Johnson P Thomas

Dissertation Adviser

Dr. KM George

Dr. Blayne Mayfield

Dr. Ramesh Sharda

ACKNOWLEDGEMENTS

To my mom Meera and dad Kumarasamy. I am what I am because of you and I owe it all to you. Thanks for the never-ending encouragement, which helped me to finish this degree. I am grateful to my sibling, friends and family for providing moral and emotional support. Thanks for being there for me Dakota.

I am grateful to my adviser Dr. Thomas. It was my pleasure to get to know you and work with you for the past several years. I leaned a lot from you over these years and it has made me a better person. I am also thankful to my department head and co-adviser Dr. George for his guidance and support during my PhD. I also thank all my committee members for being very supportive.

I would like to express my gratitude to Liu Hong, Xiaofei Hou and Ashwin Kannan for being my research partners. It was great to work with you. I would like to thank all the graduate students that I got a chance to work with.

Name: ASHWIN KUMAR THANDAPANI KUMARASAMY

Date of Degree: JULY, 2017

Title of Study: CONTENT SENSITIVITY BASED ACCESS CONTROL MODEL FOR BIG DATA

Major Field: Computer Science

Abstract: Big data technologies have seen tremendous growth in recent years. They are being widely used in both industry and academia. In spite of such exponential growth, these technologies lack adequate measures to protect the data from misuse or abuse. Corporations that collect data from multiple sources are at risk of liabilities due to exposure of sensitive information. In the current implementation of Hadoop, only file level access control is feasible. Providing users, the ability to access data based on attributes in a dataset or based on their role is complicated due to the sheer volume and multiple formats (structured, unstructured and semi-structured) of data. In this dissertation an access control framework, which enforces access control policies dynamically based on the sensitivity of the data is proposed. This framework enforces access control policies by harnessing the data context, usage patterns and information sensitivity. Information sensitivity changes over time with the addition and removal of datasets, which can lead to modifications in the access control decisions and the proposed framework accommodates these changes. The proposed framework is automated to a large extent and requires minimal user intervention. The experimental results show that the proposed framework is capable of enforcing access control policies on non-multimedia datasets with minimal overhead.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF PUBLICATIONS

1 Ashwin Kumar, T.K., Liu, H. and Thomas, J.P (2014). Efficient structuring of data in big data. In International Conference on Data Science & Engineering (ICDSE), 2014, pp. 1-5. IEEE.
2 Liu, H., Ashwin Kumar, T.K., and Thomas, J.P (2015). Cleaning framework for big data-object identification and linkage. In IEEE International Congress on Big Data (BigData Congress), 2015, pp. 215-221. IEEE.
3 Ashwin Kumar, T.K., Liu, H., Thomas, J.P. and Mylavarapu, G., (2015). Identifying Sensitive Data Items within Hadoop. The 1st IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2015), 2015, pp. 1308-1313.
4 Ashwin Kumar, T.K., Liu, H., Thomas, J. P and Hou, X (2016). An Information Entropy Based Approach to Identify Sensitive Information in Big Data. Accepted for publication by the 2nd IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2016).
5 Ashwin Kumar, T.K., Liu, H., Thomas, J.P and Hou, X (*). Content Sensitivity Based Access Control Framework For Hadoop, Journal of Digital and Communication Networks. [Under Review After Revision].

CHAPTER I

INTRODUCTION

Motivation

We have been generating unprecedented amounts of data over the past decade [1]. An estimate by E. Schmidt suggests that for every two days we generate roughly about five Exabytes (10^{18}) of data, which is equivalent to all the data produced until 2003 [1]. According to another survey [34], Gartner Inc. predicts that enterprise data will grow by 800 percent in five years, with 80 percent of it unstructured. This data is of several formats and it mostly comprises of social media interactions, videos, sensor data, server logs, e-mail and so on. Big data technologies can handle huge volumes and variety of data more efficiently than a traditional RDBMS. Due to this, the usage of big-data applications is increasing in both industry and academia.

There are several big data applications that extract knowledge from massive amounts of data. Sensitive information, which can reveal a person's identity for example, is invariably present in these huge volumes of data [2]. Identifying such information is tedious and complicated due to the sheer size of the data and its variety. A study conducted by L. Sweeney suggested that using only the 5-digit zip code, date of birth and gender, 87% of the population in the U.S could be identified [3]. Around half of the population can be identified from current city, date of birth and gender [3].All the sufficient information to identify a person will not necessarily occur in the same dataset, but combining several data sources could reveal identities of people. Hence there is a need for protecting this sensitive information is itself and this is a challenge.

For some time, data anonymization was viewed as a silver bullet to protect personal information [4]. But incidents such as the AOL Data Leak [5] (where search logs were used to reveal user identities) and revealing users' identities based on Netflix movie recommendations [6] have proved that anonymization is not adequate to protect sensitive information. In addition, there are several de-anonymization techniques like [6, 7, 8, 9] that render data anonymization ineffective.

A dataset might have a combination of sensitive and non-sensitive information. There is always a possibility of users who are given access to such data to misuse/abuse it. There was an incident involving a help-desk employee selling customer bank and credit card passwords to scam artists for a bounty [10, 11]. Incidents like these prove that there is a need for sophisticated mechanisms to prevent misuse of information by authorized personnel.

In the current Hadoop implementation [12, 13], all non-sensitive information that co-occur with sensitive information is often brought under restricted category to ensure the protection of sensitive data items [14]. This is because Hadoop uses the POSIX model for providing access to files and folders stored in HDFS [12, 13], where a user is given access to an entire dataset or not given access at all. This doesn't protect the data from being misused or abused by the authorized user. Discovering a misuse/abuse after it has occurred is often too little too late.

Several legislations like the Sarbanes-Oxley Act [15] (protects corporate financial information) and HIPPA Act [16] (restricts sharing of patient's health records) impose several restrictions on data sharing. Complying with these complex levels of data sharing in current Hadoop implementation is not feasible. Thereby Fine-grained access control mechanisms are the need of the hour. They will allow sharing data precisely while not compromising sensitive information [14].

Challenges

Implementing Fine-grained Access Control Mechanism for big data is challenging. Traditional methods like Access Control Lists (ACL's) and Role Based Access Control (RBAC) will not be suitable for big data applications as they grow many folds in size and will become cumbersome to manage [17]. R. Krishnan, in [17] suggests, "Attribute Based Access Control (ABAC) will be able to provide necessary flexibility in dealing with large scale of data". Attributes in a dataset can be easily identified if there is any information from data owners. With exponential growth in data, all the datasets do not necessarily come with all the required information from the data owner. Identifying attributes from a dataset manually is tedious due to volume and variety of these datasets. Research works [18, 19 and 20] makes use of data context and usage patterns to identify attributes from any non-multimedia dataset. Identified attributes contribute to the metadata of the corresponding dataset. A white paper by General Atomics [21], lists significance of the metadata. Metadata is also very helpful in enforcing attribute based access control decisions. [19] paves the way for implementing ABAC (Attribute Based Access Control) in Hadoop. Once all the attributes are identified, the next logical step is to identify which of these attributes are sensitive.

Content Sensitivity Based Approach towards Access Control

There have been many research works [22- 31], which deal with providing access control for Hadoop. All of them except [29 and 31] are solely dependent on the information provided by the data owners regarding data sharing to enforce access control decisions. In [29], the authors propose a content-based access control framework (CBAC) for Hadoop, with functions based on the data content itself.

The proposed Content Sensitivity Based Access Control framework (CSBAC) is somewhat related to [19]. The stark difference between [19] and the proposed approach in this dissertation is that the former compares the data content with a base dataset to make access control decisions whereas the latter uses the data content with no base dataset to estimate its sensitivity and make access control decisions based on the sensitivity. The advantages in the proposed approach are as follows 1) Elimination of considerable significant manual effort to construct a base dataset. It is simply impossible to construct a base dataset to cover all possible types and combinations of data; 2) Factoring in the variation in the data sensitivity when multiple datasets are aggregated and used. CBAC proposed in [19] fails to address this issue. If there is no base dataset or if a new dataset is created because of join or aggregation operations, CBAC will not work as intended which, severely limits the effectiveness and applicability of CBAC.

The scenarios given below are some empirical evidences to show that the content sensitivity changes when datasets are aggregated. The proposed approach can handle dynamic changes in data sensitivity without compromising the users' privacy.

Scenario 1: Consider a dataset maintained by the Human Resources department containing a list of all employees and a budget dataset, which contains salary forecasts and payments. These datasets are relatively less sensitive by themselves but when aggregated the resulting dataset that contains the list of all employees and their salaries will be more sensitive. [32]

Scenario 2: A single e-mail is relatively not very sensitive, but when a series of correspondences are aggregated then it will provide detailed insights of a person or a project, which is described in those e-mails. This aggregated dataset is highly sensitive. [32]

Estimating data sensitivity from the data itself and harnessing it to make access control decisions are new in the context of Big Data and this is the novelty of the proposed framework. In this dissertation, a Content Sensitivity Based Access Control (CSBAC) Framework, which utilizes data sensitivity to enforce access policies, is proposed. The CSBAC framework is an automated framework requiring minimal user intervention. The CSBAC framework is an extension of the SDD (Sensitive Data Detection) Framework proposed in [19].

Contributions

Contributions in building the CSBAC Framework are as follows.

Structuring Non-Multimedia Dataset

The framework proposed by us in [18] is capable of identifying data items in a non-multimedia dataset. The first step in protecting sensitive information is identifying the data items (attributes) present in a dataset. This process is very simple when the data owner provides information about what is in the dataset. With significant data democratization, there are many datasets without this information. When there is no data owner provided information, the data items should be identified manually. This process is tedious due to the sheer volume of data and the many types of datasets that are out there. The framework proposed in [18], makes use of data context and usage patterns of datasets to extract data items and to identify how they are related to one another.

Generating Structural and Descriptive Metadata

Metadata can be of two types, namely, structural and descriptive metadata. Structural metadata consists of information about what a dataset is comprised of, like data items, their data type, whether values of a data item is unique or not. Descriptive metadata is a textual representation of the dataset itself. The current implementation of Hadoop [12] is not capable of generating this information. However, the current Hadoop implementation stores data block level metadata [12]. This might be useful at the block level to maintain its integrity but not useful in detecting sensitive data items. The Enhanced Metadata Generator (EMG) proposed by in [19], will generate these two kinds of metadata. All the dataset stored in HDFS will have at least one of these two types of metadata. Descriptive metadata will be very helpful when the dataset is textual and cannot be coerced to fit into any relational scheme.

Tracking Usage Patterns

Data Usage patterns shed light on how a dataset is being used by a user. If two Hadoop clusters have the same datasets it cannot be guaranteed that these two Hadoop clusters will have similar usage patterns. The Data Usage Tracker (DUT) proposed in [18, 19] records data usage patterns. DUT tracks the users usage patterns, which consist of user identity, timestamp, datasets and data items accessed by a user along with the job information. All the usage patterns generated are stored in HDFS like metadata. All these usage patterns are used in identifying relationships between data items and also in estimating the sensitivity of data items.

Tracking Data Lineage

Data lineage refers to keeping track of where the data is at any given time in a process at any stage. Data lineage is important, as it can be useful in identifying where sensitive information is used in a process. The precursor to the Provenance Tracker proposed in [19] tracks data lineage. The proposed CSBAC framework modifies the Provenance Tracker by adding block-chain capabilities proposed in bitcoin [165]. By adding block-chain capabilities the lineage generated by the Provenance Tracker is made secure and cannot be edited or modified by malicious users. This data lineage can also be used for audit to discover any violations of the organization specific data access policies.

Data-Driven Approach to Estimate Data Sensitivity

The proposed CSBAC Framework uses metadata generated by EMG and usage tracked by DUT to differentiate sensitive data items from the non-sensitive ones. Two methodologies are presented to quantify the sensitivity of data items. The first methodology presented in [19] quantifies information value by assessing the security risk of a dataset. The laws regarding how to assess security risk of a dataset is given in [35]. The second methodology is to use Shannon’s entropy [36] to estimate data sensitivity. Results from these methodologies are compared and the best one of them is selected.

Enforcing Access Control Decisions Based On Data Sensitivity

Once the data sensitivity of the datasets stored in HDFS is evaluated, access control decisions are enforced based on the user role. The Access Control Enforcer (ACE) in CSBAC plays an important role in enforcing access control decisions. This process consists of a two-step approach as shown below

1. File level Access Check – ACE initially checks whether the user has file level access to the resource he/she is requesting. If so then the request enters the next step, else the request from the user is discarded.
2. Data item level access check – If the user has access control privilege to access a certain dataset, it doesn't mean that he/she can access all the data items. Decision about accessing these data items are made based on the user role and the sensitivity of the data items themselves.

Re-estimating Data Sensitivity during Data Aggregation

Whenever multiple datasets are aggregated the sensitivity of data items may change as in Scenarios 1 and 2. This dynamic change in data sensitivity is addressed by re-calculating the sensitivity of data items whenever a data aggregation operation is made.

Thesis Organization

The rest of this dissertation is organized as follows. Chapter 2 consists of a summary of related work. In Chapter 3, the framework to structure a dataset to identify data items is described in detail. In chapter 4, a data-driven approach to identify sensitive data items is described. Chapter 5 describes the Provenance Tracker in detail and chapter 6 illustrates the proposed CSBAC framework and how access control decisions are enforced. Chapter 7 concludes this dissertation and provides guidelines for future work.

CHAPTER II

REVIEW OF LITERATURE

Hadoop

"Apache Hadoop is an open-source software for reliable, scalable and distributed computing" [37]. Hadoop can handle huge volumes and different types of data. It is also capable of handling streaming data at relatively high speeds. Hadoop comprises of two key components namely HDFS (Hadoop Distributed File System) and MapReduce [38]. Two important features that make Hadoop prominent are data locality and parallelism. Parallelism is achieved through the MapReduce programming model developed by Google. In this programming model, data and tasks are distributed across several low cost commodity machines [39]. This programming model provides mass storage and parallel computation power [39].

A brief history of Hadoop

Doug Cutting created Hadoop [51]. Hadoop was initially a part of Apache Nutch [50, 52]. Apache Nutch is an open source and highly scalable web crawler [50, 51, 52]. Apache Nutch was part of a bigger text indexing and searching software called Apache Lucene [49]. The Apache Nutch project began on 2002 and started to index lots of web pages [51, 52].The disadvantage with this project was that it was not able to scale to index billions of web pages that were available. This problem was solved with the help of the Google File System (GFS) developed by

S. Ghemawat et al [47]. The Google File System was capable of providing a distributed storage to huge volumes of data produced during the indexing and crawling of web pages and in addition to that it eliminated all the book-keeping operations that were used in Apache Nutch to track the data stored in the storage nodes [51, 47].

In 2004, J. Dean and S. Ghemawat released the MapReduce framework [39]. This framework was capable of processing data that was distributed across several nodes. Apache Nutch project developers were able to make use of both GFS [47] and MapReduce [39] and port them to Nutch. The ported GFS [47] was termed as Nutch Distributed File System (NDFS) [50, 51]. As NDFS and MapReduce were beyond the scope of Apache Nutch, they were moved to a separate subproject of Apache Lucene called Hadoop [51].

The first largest Hadoop cluster was launched at Yahoo in 2008 [48, 53]. This Hadoop cluster processed roughly one trillion links between web pages and produced about 300 TB's of compressed output [53]. Due to the large user community and contribution Apache Hadoop was made a top-level project in 2008 [51]. The New York Times blog used a Hadoop cluster consisting of around 100 machines in Amazon's EC Cloud computing to process 11 million articles between 1851 and 1922 [54]. It took less than 24 hours to convert these articles of size 4TB as PDF's for viewing them in web pages.

Hadoop Distributed File System (HDFS)

The architecture of a typical Hadoop cluster is shown in Figure 1. HDFS provides the required storage capability for a Hadoop cluster. All the data in Hadoop is handled by HDFS and it distributes very large datasets across multiple commodity machines [38]. This makes HDFS a very scalable, highly distributed and reliable file system. HDFS is built on write-once read many times data access pattern [38].

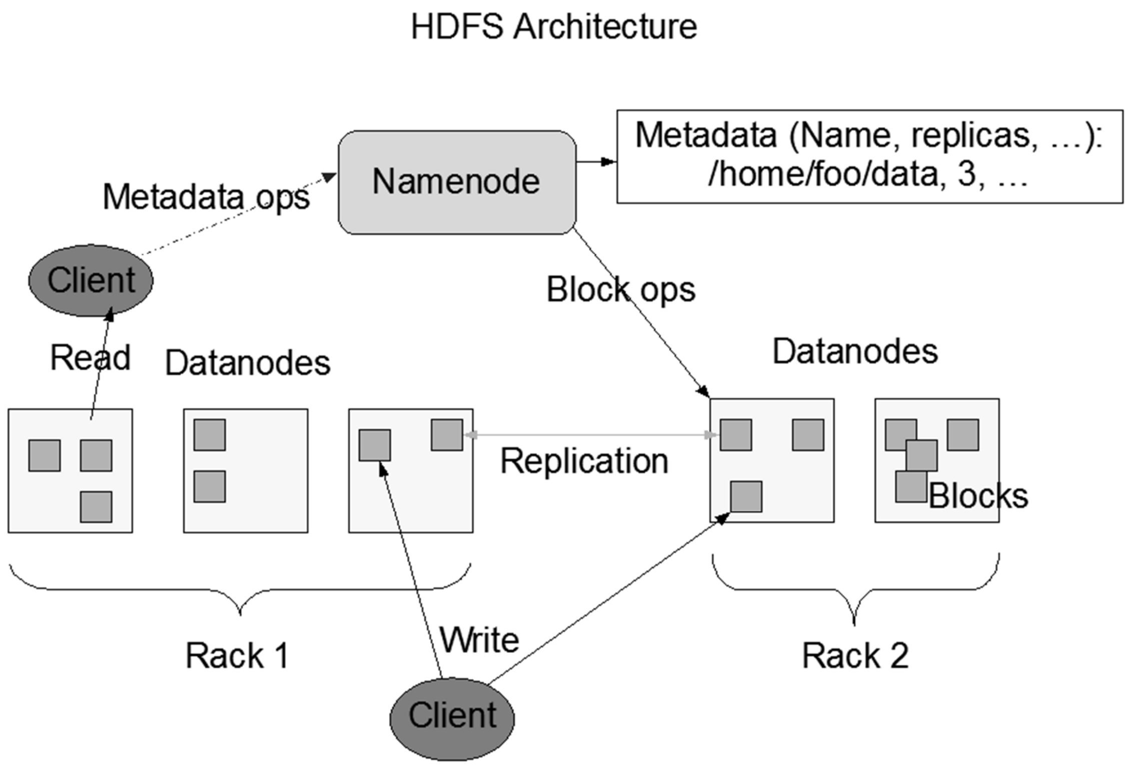


Figure 1: HDFS Architecture [38]

A large dataset is broken down into smaller components called data blocks and stored in HDFS. The user will not be able to access individual data blocks instead they will be able to access the dataset as a whole. The Hadoop administrator predetermines the size of these data blocks. The minimum block size should be at least 64 MB. 64 MB or greater is large when compared with other file systems. One of the main reasons for the large block size is to reduce the cost of disk seeks [38, 55]. By doing so the cost of data transfer for many data blocks will not be greater than the disk seek cost [55]. There are numerous advantages to storing data as blocks such as increased reliability, fault tolerance, block-level abstraction and efficient use of disk storage.

A dataset may be very well larger than the disk storage capacity of a single node, and it can still be stored in HDFS as it is not just stored in one machine but distributed over multiple machines [55]. Even though the datasets are split into several blocks, the end-users will not be aware of it and they will still view the entire dataset as a single entity. To increase data availability, reliability and the ability to handle system failures, the data blocks are replicated across the Hadoop cluster [38]. The number of replicas is determined by the replication factor (default replication factor is 3). If any data node with a certain data block fails, then the data can be accessed from another data node having this data block. Integrity of all the data blocks can be preserved with the help of CRC codes. Hadoop generates Metadata for individual data blocks and it is stored where the corresponding data blocks are stored.

A typical Hadoop cluster consists of a single name node and many data nodes. A master-slave relationship exists between the name node and data nodes. All the data in a Hadoop cluster is stored on the data nodes only. A name node monitors the HDFS namespace [39, 55]. A name space tree stored in the main memory of the name node enables this. In order to recover from any name node failure, the recent version of the HDFS namespace (fsimage) and changes done to this namespace (edit logs) are stored persistently on to a disk. If a name node fails, then the fsimage and edit log are transferred to a secondary name node. The secondary name node applies all the changes in the edit logs to fsimage to obtain an equivalent of the latest version of HDFS. Then the secondary name node functions as the master node.

All the data nodes keep sending a heartbeat message to the name node at periodic time intervals [39]. A data node is assumed to be dead by the name node when there is no heartbeat message from that node and the data blocks were in that data node are replicated to other data nodes, which have sufficient disk space.

MapReduce Programming Model

There are four major entities in the MapReduce programming model. They are the client, job tracker, task tracker and the HDFS [39]. The client is an end user who submits MapReduce jobs to the Hadoop cluster. The Job tracker is responsible for coordinating MapReduce jobs. Every MapReduce job can be divided into a set of tasks. These tasks can be categorized into map and reduce tasks and are monitored by the task tracker. All the resources related to a MapReduce job are stored in HDFS. A Hadoop cluster has only one instance of Job Tracker running whereas all the slave nodes run an instance of Task Tracker. Map tasks in MapReduce jobs can be parallelized so that they can be run on a voluminous dataset, which is stored in multiple commodity hardware. In addition to significantly reducing the running time the MapReduce framework also increases fault tolerance and scalability. One should also note that every serial program could not be parallelized using the MapReduce Framework.

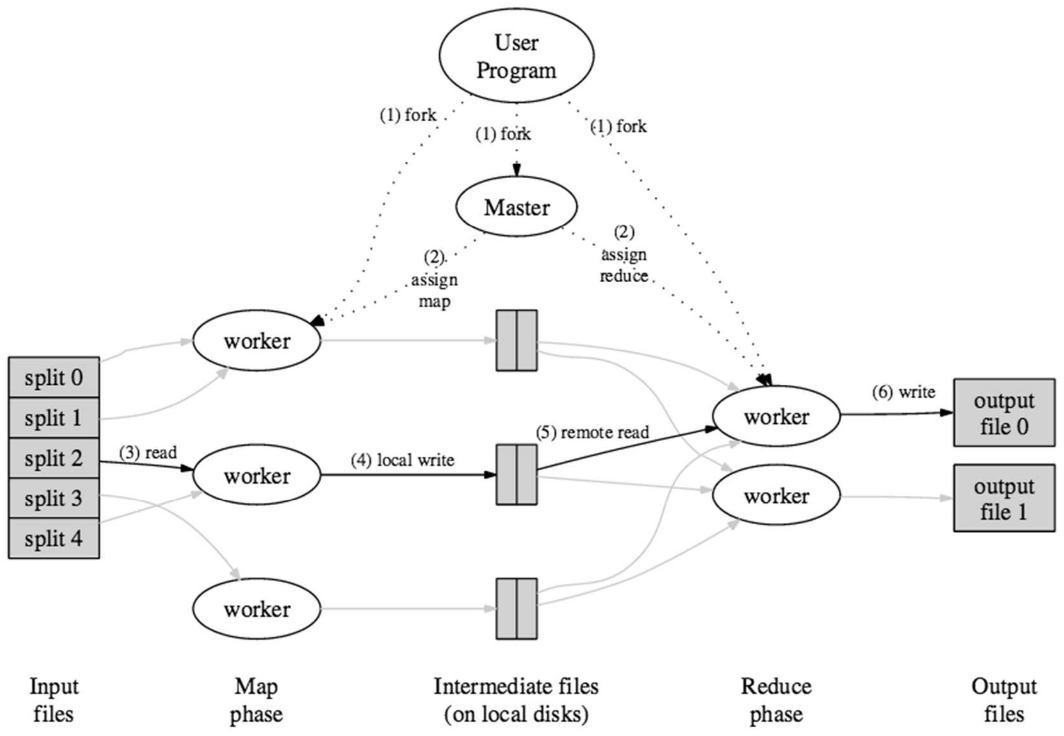


Figure 2: MapReduce Execution Overview [39]

A MapReduce job processes a set of input key/value pairs and produces another set of key/value pairs as an output [39]. The MapReduce programming model has two main functions namely map and reduce and these functions are implemented in map and reduce tasks respectively [39]. In addition to these two main functions the MapReduce model also allows users to implement partition and combiner functions. Implementation of these functions is user defined and can change from one job to another.

Overview of MapReduce Job Execution

An overview of MapReduce job execution is shown in Figure 2. In Figure 2, the User program corresponds to a MapReduce job; the master corresponds to the Job Tracker and a worker corresponds to a Task Tracker monitoring both Map and Reduce tasks. MapReduce job execution has three phases namely Job Initialization, Job Execution and cleanup [40]. The dataset is split into chunks called *"input splits"* logically by the Job Tracker for the purpose of running a MapReduce job. The size of the input splits can range from 16 to 64 MB [39]. If the input split size is not equal to the data block size, it can span across multiple data nodes.

Monitoring Job Progress and Status Updates

Running time of a MapReduce job can range anywhere between minutes and hours depending the size of the data it processes [40]. Therefore, it is necessary for a user to track the progress of these jobs. The MapReduce job and all of its tasks contains an associated status [40]. This status comprises of the state of the job (For example: running, failed, completed), map and reduce progress, job counter's values and a user-defined status message [40]. The progress of a map task is the ratio of the input data it has consumed to the size of the input data. The progress of the reduce task is also the ratio of the input consumed to the size of input data. Total progress of a reduce task is split into three equal parts between copy, sort and actual reduce task. For example, if a reduce task has used half its input then its progress is 5/6, as it completed the copy (1/3) and sort (1/3) phases and is half way through reduce phase (1/6) [40].

Fault Tolerance in MapReduce

MapReduce programming model is highly scalable and thus the parallelized programs (MapReduce jobs) can be run simultaneously on many commodity machines. Since the execution of MapReduce job involves multiple machines, it is only practical to expect these machines to fail. In case of such failures, the MapReduce framework should be able to handle it very well [39]. Data replication helps very much in case of node failures.

If a task tracker stops sending heartbeat message for a certain period of time to the Job Tracker, then that particular Task Tracker is considered to be dead. Any progress made by the tasks in these failed nodes will be reset and thus these tasks can be scheduled on other data nodes that have the same data blocks [39]. If a node, which has completed a map task, fails before or when the intermediate results are copied to the reducer task, this task will have to be scheduled and executed again. This rescheduling information will be passed along to the corresponding reducer node so that it can fetch intermediate results from the correct node [39].

If a Job Tracker fails, then it is restarted from its latest checkpoint. Job Tracker periodically creates checkpoints, which contains information about the status of jobs that are running or waiting to be run. If there is no check pointing done by the Job Tracker, then all the MapReduce jobs being executed are aborted when the Job Tracker fails.

Drawbacks of MapReduce programming model

Although there are many advantages in the MapReduce model, it also comes with significant disadvantages as well. The MapReduce model arrived with the scalability bottlenecks because the Job Tracker was responsible for both scheduling the MapReduce jobs and monitoring their progress. It became impossible for the Job Tracker to scale beyond 4000 nodes [56] due to its expensive bookkeeping operations.

Yet another Resource Navigator (YARN)

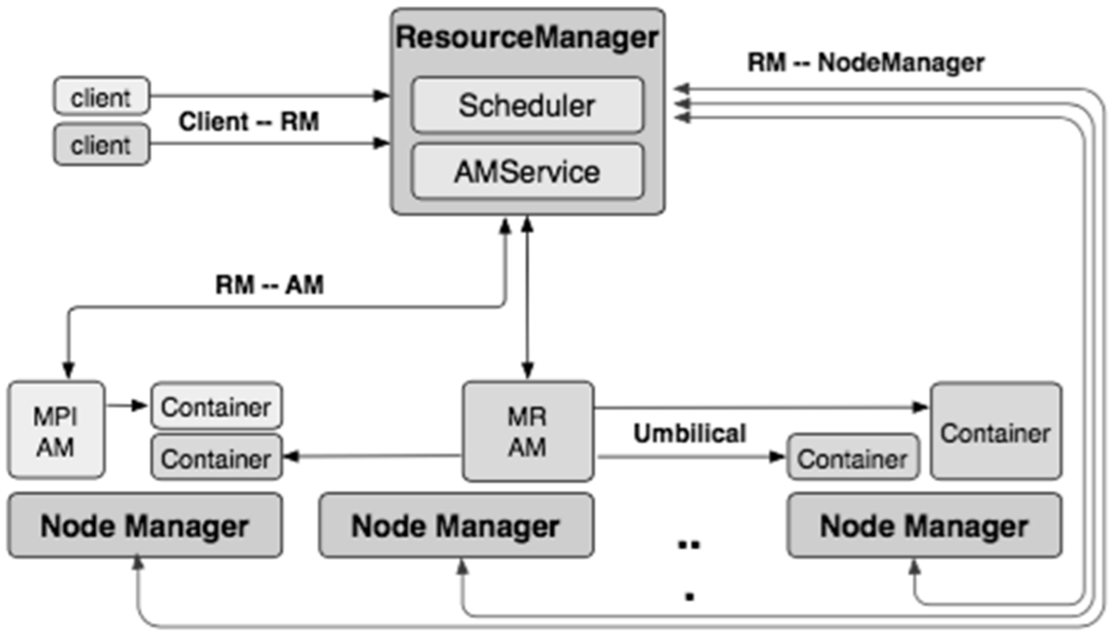


Figure 3 YARN Architecture [57]

YARN was created to address the shortcomings of the MapReduce programming model. The functions of Job Tracker are split into two separate entities called resource manager and application master in YARN model [57]. This reduces the bookkeeping workload of the Job Tracker. The Resource Manager (RM) is responsible for controlling resource usage in a Hadoop cluster, checking if a node is alive, enforcing resource allocations and resolving issues in sharing resources between users [57]. The Application Master (AM) is responsible for allocating tasks and monitoring their progress. In addition to this the AM is also responsible for coordinating with the RM for allocating required resources for a MapReduce job [57]. Architecture of YARN is shown in Figure 3.

The Resource Manager runs on a dedicated machine and manages resources in a cluster centrally [57]. The Resource Manager allocates "containers" to tasks dynamically based on demand. A container is a collection of resources, a combination of CPU and RAM belonging to a slave node

[57]. To monitor these container assignments and prevent over usage of containers, the RM communicates with the Node Manager (NM) [57]. The NM is responsible to monitor the containers and managing their life cycle at the data node level [57]. In addition to this the NM reports faults frequently to the RM and the NM communicates with the RM through a heartbeat message [57]. By putting together, the heart beat messages from several NM's running on all the data nodes, the RM will get a clear picture of resource utilization of the entire cluster.

A Job Client submits a MapReduce job to the RM through a submission protocol accessible to all the users. The jobs initially are scrutinized by an admission control phase, which checks the credentials of the user submitting the job and check if the user is authorized to access the requested resources. After the initial admission control phase, the scheduler schedules accepted jobs for execution. Based on the available resources, the jobs are executed. When a MapReduce job is executed a container for Application Master (AM) is created on one of the nodes in the Hadoop cluster [57]. Accepted MapReduce jobs (or) applications are stored in persistent storage so that they can be recovered later in an event of failure. The AM is responsible for running all the tasks in a job and managing life cycle, flow of execution, resource utilization and handling errors of all tasks involved in a MapReduce job [57]. YARN assumes most of the MapReduce programs will be written in a higher level programming language like Java, Python, etc [57]. To complete executing tasks in a MapReduce job the AM will need a set of containers on several nodes from the RM. To get hold of these containers, the AM will initially request the RM for required resources. This request will comprise of specific features about the containers requested and their locality preferences [57]. The RM will accommodate all such requests from all applications based on the current availability and scheduling policies [57]. When RM allocates a container to an AM, it creates a lease for the same and it is sent via heartbeat message to the AM. The AM passes this lease information to the Node Manager (NM) in which the container exists to prove its authenticity [57]. After the NM verifies the authenticity of the AM it grants the container to the AM. Once a container is available to the AM, it encodes a task (map/reduce) launch request with the lease information [57]. In the container either a map or a reduce task is executed. The containers communicate with the AM to inform about the job status and their health and resource utilization periodically [57].

Survey of Traditional Access Control Models

Some of the information stored on computers may be personal or sensitive in nature. If unauthorized individuals access this information, it will result in dire consequences. Examples of these consequences include but not limited to disclosing sensitive information to others, holding sensitive information for financial compensation, lawsuits for companies because they failed to protect the sensitive information safe, etc. There are a number of access control mechanisms that aid in keeping sensitive information out of reach of unauthorized individuals. A detailed description of several of these access control models is provided in this section.

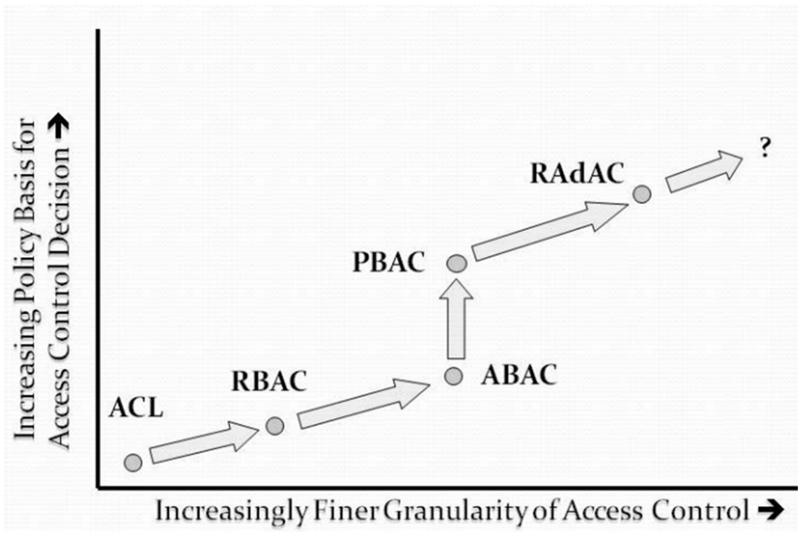

Increasing Policy Basis for Access Control Decision
ACL
RBAC
ABAC
PBAC
RAdAC
?
Increasingly Finer Granularity of Access Control

Figure 4 Various access control models [59]

A survey of several types of access control models was done in [59] by the National Institute of Standards (NIST). In [59], the authors explain the evolution of the granularity of access control models from an Access Control List (ACL) to Risk Adaptability Access Control Model (RAdAC) as shown in Figure 4.

One of the primary applications of these access control models are in RDBMS, where a user is given fine grained authorization to access data based on his/her role or privileges. RDBMS access control models fall into any of the following categories.

- Mandatory Access Control Model (MAC) [93 – 104]
- Discretionary Access Control Model (DAC) [105 – 112]
- Role-based Access Control Model (RBAC) [113-120]

Access control models shown in Figure 4 and the models listed above are discussed briefly in the following sections.

Access Control Lists (ACL)

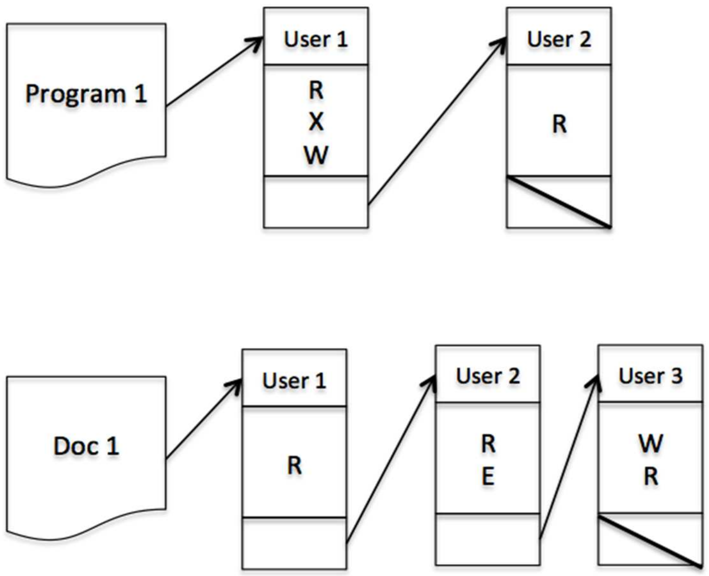


Figure 5 Example of an Access Control List (ACL)

ACLS's were initially implemented in Linux operating systems and they were increasingly used when multi user operating systems were introduced to prevent users from accessing each other's files [59]. Every resource (files, folders, etc.) in a system is referred as objects. These objects have a "*list of mappings*" [59] between users and the operations these users are allowed to perform on the objects. An example of an Access Control List with read (R), write (W) and execute (X) permissions is shown in Figure 5.

In Figure 5, the ACL has two resources (objects) Program 1 and Document 1. Both these objects have their own data structure (linked list) to contain mappings between several users and the operations, which they are allowed to perform on these resources. ACL's are used not only in Operating Systems they are used in several other applications such as network security [121], cloud security [122], and so on. In spite of its simplicity ACL's have some disadvantages as well. ACL's, which are stored in-memory, cannot scale well when the number of users and resources grow significantly. In addition to this before a user performs an operation on any resource, the corresponding access control list for that resource must be checked every time. This process can be time consuming if a number of users are given access to same resource [59]. ACL's can be difficult to maintain in an enterprise, as the users will be requiring multiple levels of access to various resources. Modifying an ACL for each and every such resource, which needs multiple levels of access, is time consuming, complicated and error-prone [59].

Discretionary Access Control (DAC)

DAC allows the owner of a resource to provide and manage access rights to other people (subjects) to use his/her resource [105]. D.D. Downs et al in [125] states that the "*basis for DAC is that an individual user or a program operating on the user's behalf is allowed to specify explicitly the types of access to other users (or programs executing on their behalf).*"In [126]

B.W. Lampson added the notions of owners and access matrix to the DAC. P.P. Griffiths and B.W. Wade in [127] extended DAC to relational databases.

DAC is based on the fact that the "owners" of resources will have complete discretion to grant/revoke access to their resources to other users [105, 128]. Ownership is attained by means of creating new resources [105, 129]. Access matrices can be used with either DAC or MAC to enforce the policies for a given user trying to access a resource. An example of such an access matrix is shown in Table 1

TABLE 1 An Example of an Access Matrix

	File 1	Folder 1	Program 1	File 2	Folder 2
User 1	Read	Read, write	Execute, read	----	----
User 2	----	----	Read	Read, write	Read
User 3	----	Read	Execute, write	-----	Write

Rows of access control matrix represent a subject or user and columns of access control matrix represent resources or files. Each cell in this access matrix contains the access rights of a user to access a specific resource. When a user requests to access a resource, the access rights of the user is verified to ensure that the requested operation can be performed on the resource by the requesting user. If this condition is satisfied, then the user's request is honored else it is denied.

The access matrices are sparse and for an organization with many users and resources, the size of these access matrices can become tremendously large. Searching through such large matrices each and every time to determine whether to accept or deny an access request will be time consuming. Due to the inefficiencies of the access matrix, the DAC splits this matrix into two lists namely access control list (ACL) and capability list [130].

A capability list (CL) focuses on users unlike the access control lists, which focus on resources. Capability lists are very useful when authorization is checked on the subject-basis [130].Even after bifurcating the access matrix into ACL's and CL's for each request these lists must be searched through to check if the user is allowed to do a certain operation on a resource. In any organization with a numerous subjects and resources the lookup time for DAC can be very significant. Updating both these lists accurately can also be challenging.

Mandatory Access Control (MAC)

The Mandatory Access Control (MAC) model is also known as the Bell-La Padula model [98, 123]. R. S. Sandhu in [96] proposed a minimalistic Bell-La Padula model called as an BLP model. The basis of the BLP model is to support the Discretionary Access Control (DAC) Model in enforcing access control policies. For a user to access a resource the BLP model proposed in [96] requires a Discretionary access matrix *D* and mandatory access control policies. The discretionary access control matrix is explained in detail in the previous section. Mandatory access control policies are labels attached to every user and resource in the system [96]. In the BLP model, labels associated with a user are called security clearances and labels associated with resources are called security classifications [96]. Examples of these labels are as follows top-secret, secret, confidential, classified, unclassified, etc. R.S. Sandhu makes an assumption known as "tranquility". According to "tranquility" only a security officer can change the labels after they have been generated [96].

Role Based Access Control (RBAC)

Role Based Access Control Models (RBAC) became increasingly relevant and useful in the beginning of the early 1970's when systems capable of accommodating several users and applications were introduced [115]. The guiding principle of RBAC is that every user in a system will be assigned at least one role, which is created as per organizational policies or based on job functions/designation of the user [115]. Each role will allow the user to perform actions relevant to that role. Users can be moved across several roles and the roles can be granted to perform certain operations or permissions revoked [115]. A study done by D.F. Ferraiolo et al [131] for the U.S. National Institute of Standards and Technology (NIST) shows that the RBAC effectively addresses privacy issues in any organization. An important feature that makes RBAC so prominent is that it ties the access control decisions based on the role of an individual [115]. In addition to this as roles evolve access control decisions can be modified in RBAC [115].

R.S. Sandhu in [115] classifies the RBAC models into four conceptual models as shown in Figure 6. These four models represent several dimensions of the RBAC model [115]. In Figure 6, $RBAC_0$ represents the base model, which satisfies all the requirements to support RBAC. Both $RBAC_1$ and $RBAC_2$ add independent features to the base model. $RBAC_1$ contains role hierarchies, which can control user role inheriting access permissions from other roles [115]. $RBAC_2$ contains constraints, which impose criteria that all components on a RBAC must satisfy [115].

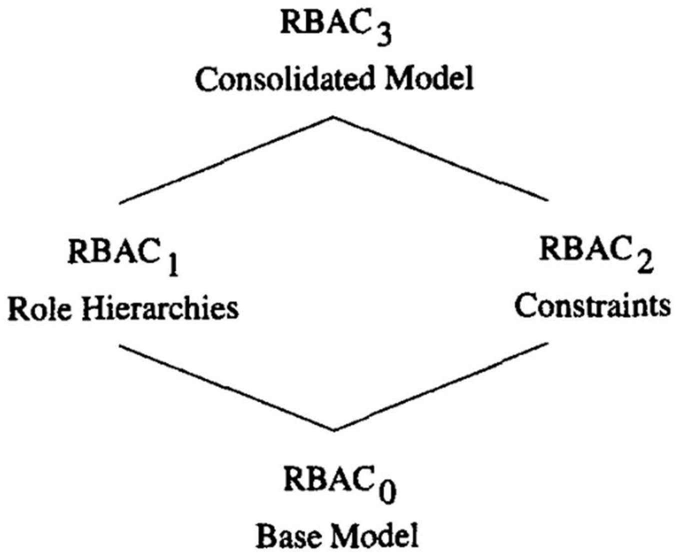


Figure 6 Relationship between RBAC96 models [115]

$RBAC_3$ is called the consolidated model as it includes both the features on $RBAC_1$ and $RBAC_2$. By transitivity one can argue that the consolidated model also includes the features of the base model $RBAC_0$ [115]. The consolidated model is often referred to as the RBAC96 family of models [115]. Relationships between the four conceptual RBAC models are shown in Figure 10 and the consolidated RBAC model is shown in Figure 7.

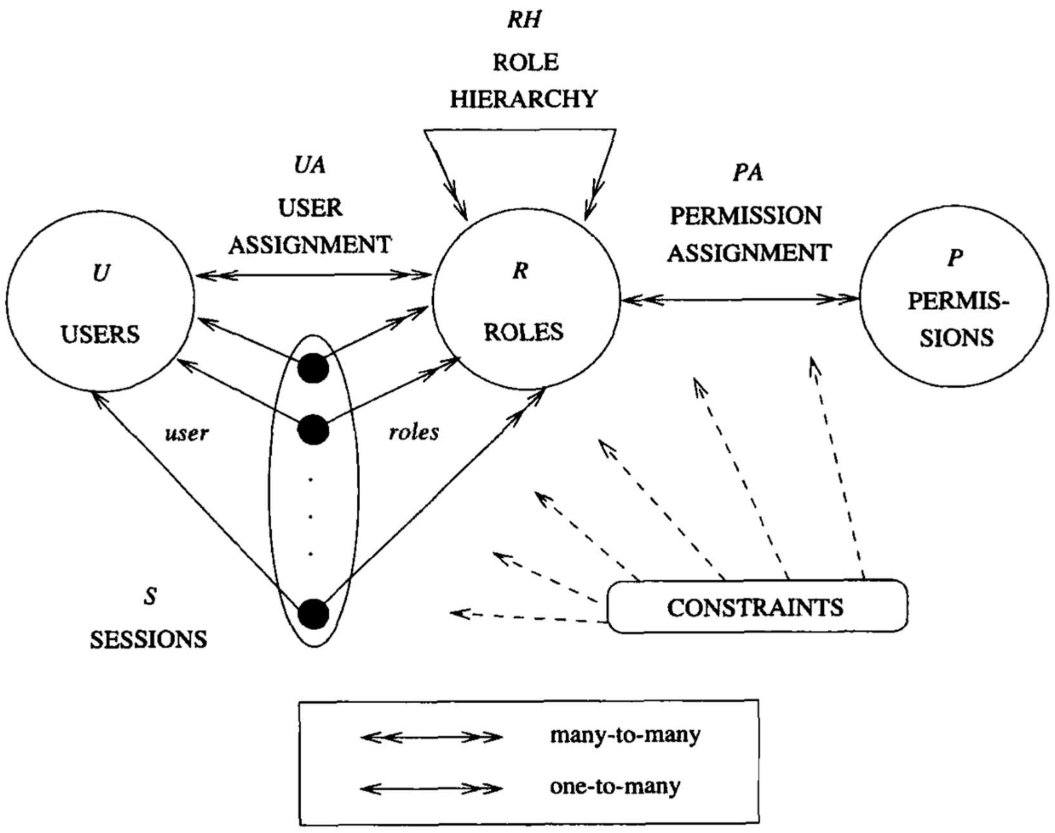


Figure 7 RBAC96 model family [115]

Base Model RBAC$_0$

RBAC$_0$ model consists of the following components shown in Figure 7 [115]. These components are described in detail in the rest of this section.

- U, R, P and S (users, roles, permissions and sessions).
- User: S → U, a mapping between each session S_i to a user U_i
- Role: S → 2^R, a mapping between each a session S_i to a set of roles satisfying the Equations 1 and 2. In Equation 1, roles can change with time as they evolve.

$$roles\ (s_i)\ \subseteq\ \{\, r\ |(user(s_i), r)\ \epsilon\ UA\} \qquad (1)$$

$$Session\ s_i\ has\ permissions\ U_{r\ \in\ roles\ (s_i)}\{\ p\ |(p, r)\ \epsilon\ PA\} \qquad (2)$$

- Permission Assignment (PA) – Many-to-many relationship between roles and permissions. PA should be a subset or equal to the cross product of the sets P and R. This is a necessary condition for PA and shown in Equation 3.

$$PA \subseteq P\,X\,R \qquad (3)$$

- User Assignment (UA) – Many-to-many relationship between user and roles. UA should be a subset or equal to the cross product of the sets U and R. This is a necessary condition for UA and shown in Equation 4.

$$UA \subseteq U\,X\,R \qquad (4)$$

A user in this model may represent a human being or any other entities that are capable of accessing data or performing a task such as computer programs, software agents [115]. Role refers to the title or a responsibility of a job/assignment in an organization [115]. Permission refers to an authorization or approval provided to a user role to access certain resources [115]. User Assignment (UA) and Permission Assignment (PA) are many-to-many relations between Users-Roles and Permissions-Roles respectively. A user can have more than one role assigned to him/her and a role can have several permissions assigned to it [115]. The prominence of RBAC relies solely on the PA and UA as they provide make the role of a user as an intermediary between the user and the permission, which he/she wants to exercise [115]. A session (S) maps a user to one or more of the roles assigned to him/her. This is shown in Figure 7 in the form of double-headed arrows originating from sessions (S) to roles (R) [115]. A session is mapped to only one user throughout its existence and this is represented in Figure 7 in form of single-headed arrows between sessions (S) and users (U) [115]. A user can have more than one active session at any given time and these sessions can have a combination of different roles that are active and assigned to the user [115]. In $RBAC_0$ the roles, which should remain active in a session, is entirely up to the decision of that user as the users control them directly [115].

RBAC models are neither a solution to all access control issues nor it is capable to enforce the security principles it satisfies [115]. The person in charge of setting up roles can set them up in violation of the principles mentioned above and the data abstraction entirely depends on the implementation [115].

Attribute Based Access Control (ABAC)

Attribute based access control model permits/denies access to resources by considering the factors such as the subject, object, requested resources, environmental attributes and rules or relationships [133]. Introduction and gaining prominence of the Service Oriented Architecture (SOA) lead to the creation of a new specification by OASIS called the eXtensible Access Control Markup Language (XACML) [134]. This specification contains the building blocks for the ABAC model. XACML introduces several reference points to accept/deny a request from a user trying to access a resource such as Policy Decision Points (PDP), Policy Enforcement Points (PEP), Policy Administration Points (PAP) and Policy Information Points (PIP) [134]. In addition to this XACML also defines protocols for communication between several entities. These entities communicate based on the request response protocols [134]. Although XACML requires the components to implement ABAC it did not provide a formal guide to implement the same [134].

ABAC Model

All the components of the ABAC model are shown in Figure 8. The ABAC Access Control Mechanism (ACM) in ABAC model receives a request from a subject to access an object. ABAC ACM then analyzes the attributes of the subject, object, any environmental conditions imposed and the access control policies and decides whether the request can be honored or not.

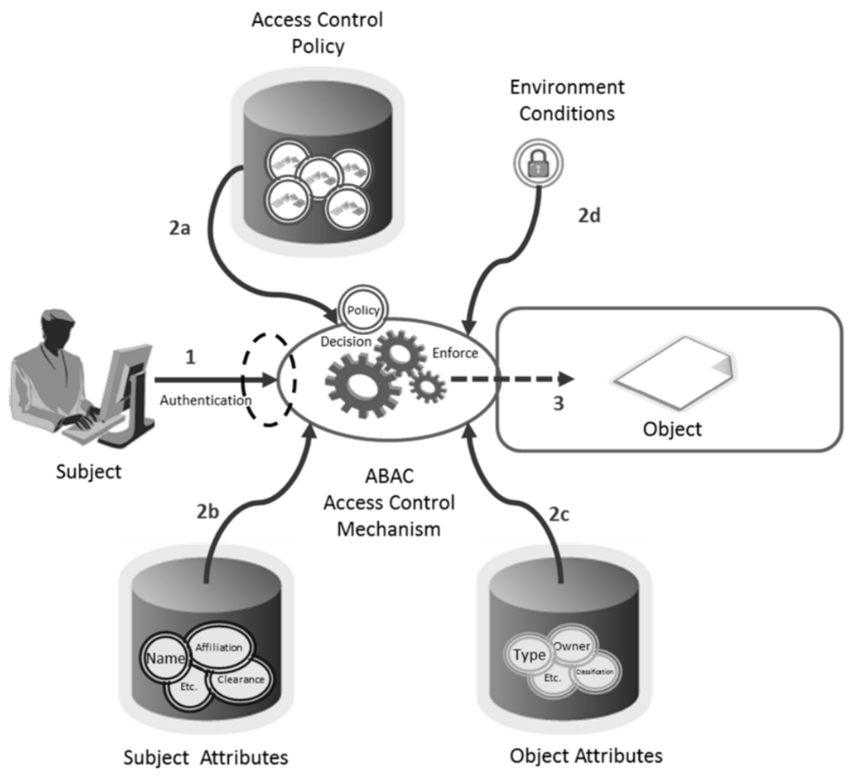


Figure 8 Attribute Based Access Control Model (ABAC) [133]

Attributes define several features of subjects (users), resources (objects), environmental factors, operations requested by the subject (user) [133]. Attributes are defined and assigned by a trusted authority. Subject is a physical or a logical entity that can request to access information [133]. Subject can be either a user or a device or a computer program, which are capable of accessing resources [133]. An object is a resource, which contains some amount of information. Examples of object include files, databases, tables, records, and logs. An operation is the process of executing a particular operation by a subject on an object [133]. Operations include but are not limited to read, write, execute, modify, delete and copy. Policy is used to represent relationships, which contains a collection of permissible operations that can be executed by a subject on an object given any environmental conditions [133].

ABAC model primarily analyzes the attributes of subject, attributes of object, environmental conditions and policies, which define all the operations that are permitted for a subject-object combination [133]. After analyzing the aforementioned features ABAC either allows or denies the operation performed by the user. ABAC ACM is both the Policy Decision Point (PDP) and Policy Enforcement Point (PEP) as it prevents sensitive information being misused or abused by users [133].

Primary disadvantage of ABAC is generating all the relevant and necessary attributes for each subject and object. In a large organization, which stores considerable amount of data and has a number of users (subjects) the effort put to generate these attributes are very significant [130]. In addition to this the ACM in the ABAC model chooses the attributes of a subject and object before making a decision. The process of choosing attributes from a number of attributes each time an operation is requested by the user is time consuming.

Policy Based Access Control (PBAC)

Policy Based Access Control model is the extension of the ABAC access control model to address the failure of the ABAC model to *"harmonize"* access control policies across several entities in an organization [130].

The first and foremost challenge is that an organization should maintain and monitor a list of attributes over an entire organization comprising many individuals, departments, objects (resources) and departments [130]. The second challenge is to convert the access control principles to enforce access control decisions [130]. However, this can be resolved by using XACML, a machine-readable access control specification language. In addition to this the PBAC clearly defines the guiding principles of a user session [135]. Functioning of PBAC is very similar to that of ABAC where both of these models can make access decisions based on several policies governing various subjects, resources and the operations [135, 119, 136].

Policy Based Access Control (PBAC) Model

In addition to all the variables in the ABAC model, the PBAC model also consists of session [135]. Access control mode of the PBAC model is shown in Figure 9.

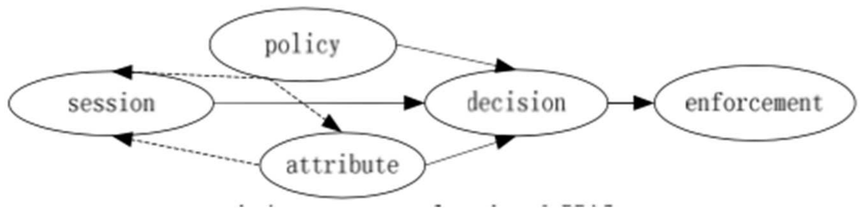


Figure 9 Access Control mode of PBAC [135]

All the variables in the PBAC except the session are explained in the preceding sections. Session contains information about which subject performed what action on which resource [135]. In other words, it comprises of subject, resource and action. Session in PBAC should be a Cartesian product between these three variables and is represented in Equation 5.

$$SESSION = SUBJECT * RESOURCE * ACTION \quad (5)$$

The major limitation of the PBAC model is the identification and maintenance process of several attributes spanning across several subjects, resources and departments [130]. This process is not only time-consuming but also very complicated and tedious.

Risk Adaptable Access Control (RAdAC)

RAdAC is intended to work on several large-scale scalable computing systems which are intended to gather, store, process, manage and allow users to access information [139, 140 and 141].The motivation for the RAdAC model was to create a *"real-time, adaptable, risk-assess access control facility for enterprises"* [130]. To facilitate real-time and adaptable access control enforcement the RAdAC model should assess each request to access information by considering the following factors the priorities of the mission, risk and cost of compromising the information and threat status of the system [139].The RAdAC model grants the users access or denies them access to resources by computing the security risk and the operational need [138].

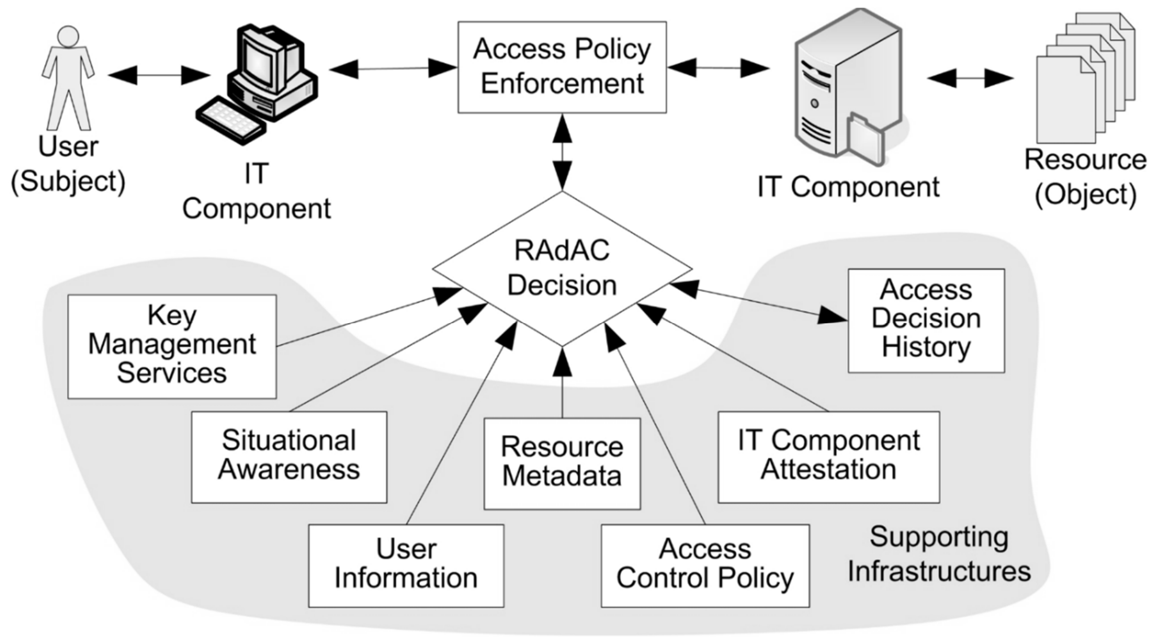


Figure 10 RAdAC Model in action [138]

In Figure 10, the functioning of the RAdAC model is described in detail [138]. Security risk involved in allowing users to access resources is entirely dependent on the type of the access granted to the resource or the nature of transaction itself [139]. For example, a user accessing banking information from a known computer doesn't pose a security risk whereas accessing the same from an unknown computer poses considerable risk. Operational need is usually referred as the need-to-know basis in the literature [139]. Examples of operational need might include membership of an individual in a community or his/her interests in a specific area [139]. Situational factors include several external or environmental variables, which are considered when making decisions [139]. Following factors are considered while determining security risk and operational needs to evaluate each access control request [138].

- Sensitivity of information, which is to be accessed

- User information (role and trust)
- Access history decisions
- How critical is the information for the operation?
- How well the information can be protected
- Uncertainty

The RAdAC model will be able to adapt the threshold in a dynamic manner when required. An example for this scenario is changing threshold *"when operational need can trump security risk"* [138]. Threshold in the RAdAC model is computed by using organization policies and relevant environmental or external variables.

The disadvantages of the RAdAC model are its implementation is very complicated and time consuming [130]. Building trust among several organizations to share mechanisms to standardize evaluation of risk is nearly impossible. Evaluating security risk will need extensive information about users, resources, external variables and other environmental factors. However, there are no standard mechanisms to gather such information [130]. There is no standard format of all the information needed to estimate risk of an access request [130]. A standard format will ensure portability of the RAdAC model across several environments.

Content Based Access Control Models

In [144] B. Gopal and U. Manber propose an access control mechanism called the Hierarchy and Content (HAC) model for traditional file systems. HAC combines the hierarchical access control model for file systems with content-based access control. Users can access resources by specifying the name and path of the resource explicitly in hierarchical file systems [144]. In [144] the authors combine HAC paradigm with the semantic file system paradigm [147] through an optional feature called Content-Based Access (CBA) option. CBA option is optional and can be enabled/disabled by the user [144].

In [146], E. Bertino et al argue that the mechanisms to protect data based on the content are the need of the hour and they will be more effective in protecting the data. In [145] E. Bertino et al propose an extension to the System R authorization model based on the content to the relational database system (RDBMS) [148]. L. Giuri and P. Iglio in [142] propose a modification to the RBAC model by adding content-based access control policies. In [142] content-based access control policies are implemented as *"parameterized privileges"* and *"role templates"* to facilitate the parameterized privileges.

In [149], N.R. Adam et al propose a content-based authorization model for digital libraries. Digital libraries are global systems, which are responsible for gathering and dissemination of information among a variety of users such as content producers, librarians and end users [150]. Contents of these digital libraries are mostly images and videos [149]. Major challenge in digital libraries is keeping information out of reach of unauthorized users. Traditional authorization mechanisms used for a RDBMS will not suffice the digital libraries due to the large volumes and variety of data [149]. The authorization model proposed in [149] makes access control decisions based on the characteristics of the user and the characteristics of a digital content or a part of the digital content.

In [143] S.K. Tzelepi et al propose an access control model based on content for database systems that store multimedia medical information. The authors in [143] extend the RBAC model to accommodate access control decisions made based on the content of medical images. The drawback of this approach is that the administrators manually enter the annotations for the medical images.

In [152] E. Bertino et al propose a hierarchical content-based access control based on semantic trees. Representing videos in form of domain dependent semantic trees will enable the model to provide fine-grained access control. In semantic tree a video is broken down to semantic clusters, scenes and shots as shown in Figure 11. An example of representing a news video in the model proposed in [152] is shown in Figure 12.

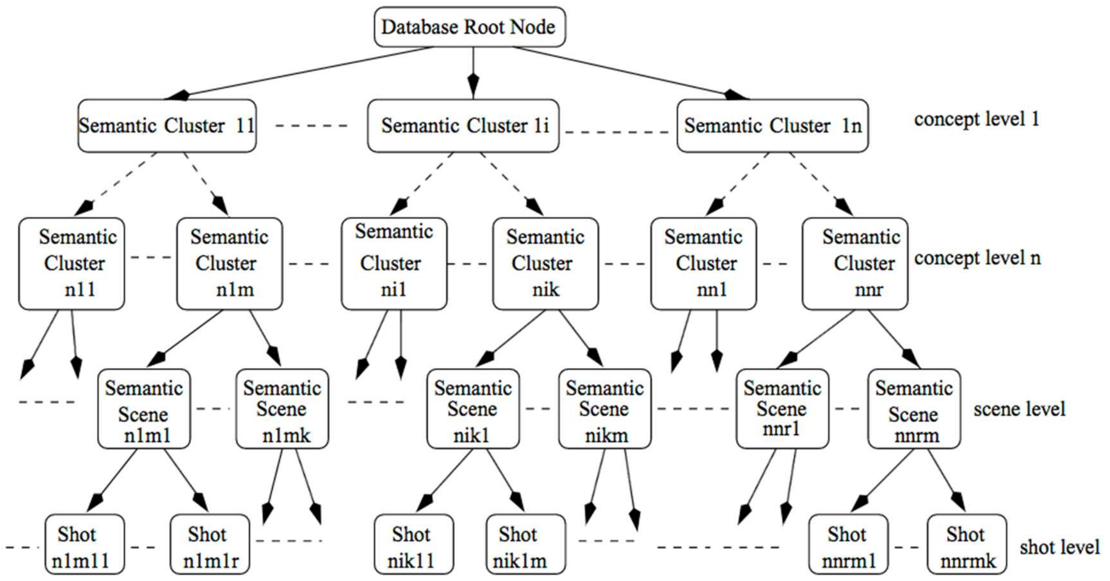


Figure 11 Hierarchical Video database model [152]

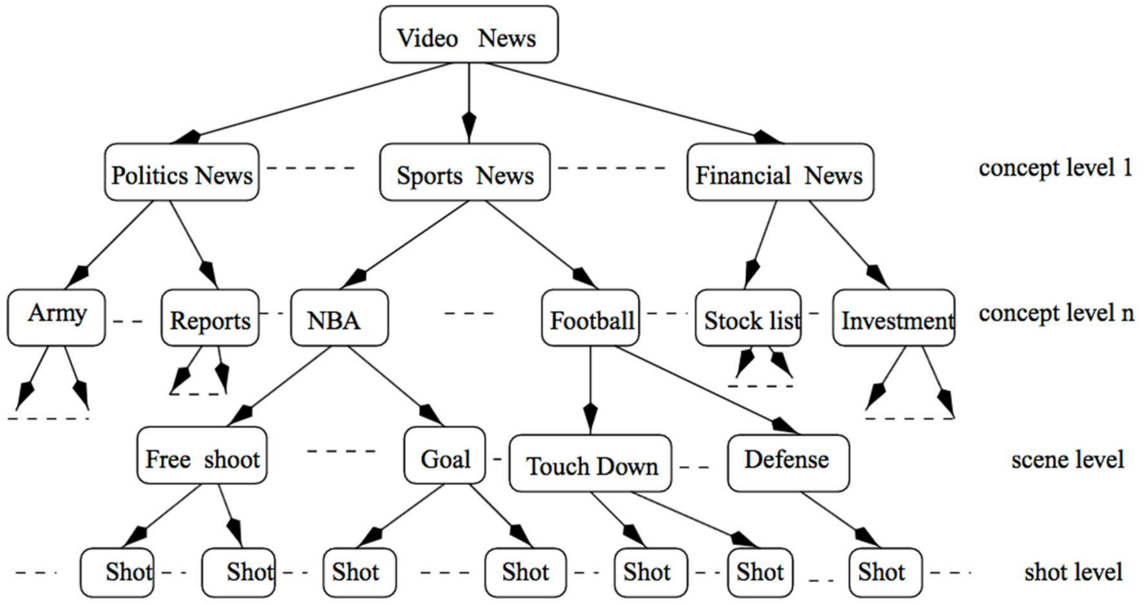


Figure 12 Example Hierarchical Semantic tree representing a News video [152]

The disadvantage of this approach is that the semantic tree cannot accommodate a large collection of videos and the level of these trees cannot be increased. In addition to this model depends on the concept hierarchy provided by a domain expert [152].

In [151] N.A.T. Tran and T.K. Dang propose a content-based access control model for video database, which extends the semantic cluster tree proposed in [152]. The extended video database proposed in [151] is shown in Figure 13.

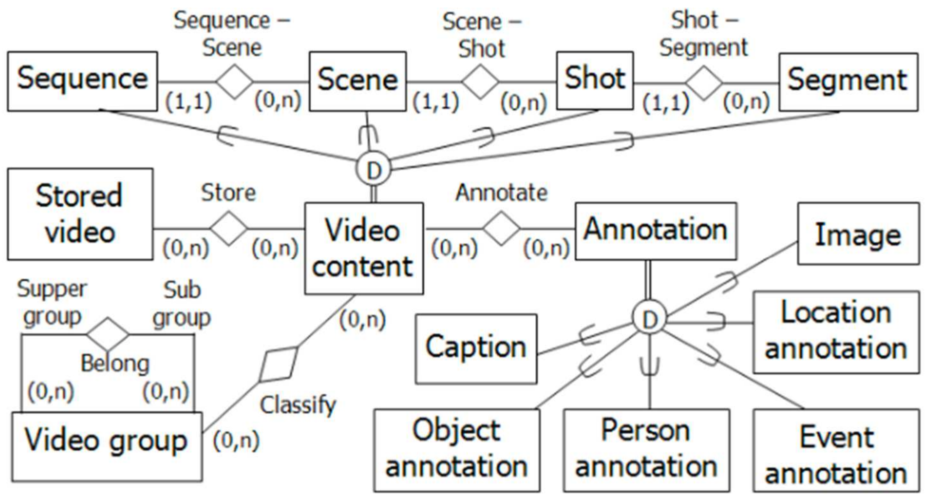


Figure 13 Extended Video database [151]

In [151] the each stored video will belong to a video group and it can be split into scenes, sequences, shots and segments. Each video will contain any number of annotations, which can range from captions, images, resources, subjects and events.

In [153], E. Bertino et al propose an access control mechanism for video database systems. This mechanism exploits both the semantics and structure of the video. In [153] the *"basic unit of authorization"* is called the video element and it can comprise of sequence of video frames or any object on the frame. Users of this authorization model sends request to either view or edit the resources (videos) [153]. Basic elements in a video are recognized explicitly by identifiers or implicitly by semantic contents in the video and users are identified by their credentials

[153].Like the model proposed in [143], the major drawback of the models in [151] and [153] are their dependence on the annotations for providing content-based access control. In addition to this in all the preceding access control models attributes regarding to the users and access control policies are defined in advance.

In [156] S. Monte proposes a Content Based Access Control (CBAC) model for web-based social networks (WSBN's). Due to the proliferation of the Internet number of users in the WSBN's have increased significantly and these users are responsible themselves to select whom they want to share their information with [156]. By allowing users to make the decision their personal information sometimes ends up in the hands of the users who are capable of misusing personal data of others. Hence the CBAC model proposed in [156] allows users to access resources based on its contents. Contents of resources in CBAC are analyzed based on computer vision and natural language processing techniques [156]. In [154, 155] M. Hart et al propose PLOG (Privacy/Policy-aware bLOGging engine), a language to enable users to use the content-based access control system. PLOG allows users to specify which part of their data they want to share with whom. In [154, 155, 156] content from WBSN's are inferred based on the tags and not the actual content within these tags, which is a major drawback.

In [158] I. Molloy et al propose a model, which make access control decisions under uncertainty. In [158] the authors use supervised learning algorithms such as SVM (Support Vector Machines) to train the model with known decisions so that the model makes access decisions when an unknown scenario is encountered. The access control decisions made by the model proposed in [158] will pose a certain degree of uncertainty and risk. Access requests with too much uncertainty or high risk should not be allowed [158]. Q. Ni et al in [159] propose an automated provisioning model, which reassign and modifies user's role assignment. Q. Ni use a variety of supervised machine learning algorithms trained using actual provisioning data. These algorithms are then evaluated in [159] for their performance when they propose modifications to provisions/role assignments.

Access control for semantic web based on concepts was proposed by L. Qin and V. Alturi in [160]. Semantic web provides a means for computers to process information in the World Wide Web (WWW) [164]. Every web page consists of annotations which define the concepts in the web pages and these annotations also contains information about a web page is related to others [160]. Several concepts are expressed together in ontology. The model proposed in [160] makes use of these annotations in the web pages and deciphers the concepts and their semantic relationship with other concepts and makes access control decisions. The access control decisions are also dependent on the security policies put in place by the organization at the conceptual level and these policies are written in OWL (W3C Web Ontology language) [160].

A.Toninelli et al in [161] argues that context also plays an important role in making access control decisions for semantic web. In [161] the authors propose a context-aware access control framework for a highly dynamic web where the availability of resources and users change more often. Context awareness proposed in [161] is however limited only to the usage of the semantic web from a user's perspective. In [162], C. Pan et al propose a Semantic Access Control Enabler (SACE), which will act as a middleware between the legacy databases and users. The SACE will make access control decisions based on the ontologies and their mapping rules. Each mapping rule will specify how a concept in ontologies can be accessed. SACE extends the RBAC access control model and uses all the components in the RBAC model in addition to the ontologies and mapping rules. In [163] B. Fabian et al propose access control model for semantic data federations. The access control model will aid in accessing information between business partners, which may be separate organizations.

The access control model proposed in this dissertation is significantly different from the others. The proposed CSBAC model enforces access control decisions based on the information sensitivity. Information sensitivity is in turn derived automatically from the metadata, data usage and information entropy.

Bitcoin

S. Nakamoto in [165] proposed a peer-to-peer electronic cash transfer in which the transfers were not required to pass through financial institutions like banks [165]. Hashing digital signatures and the timestamps to a chain is a major advantage of Bitcoin [165]. These chains are called as block chains and they are immutable.

Introduction

In E-Commerce banks generally serve as trusted third parties to process all the electronic payments [165]. The trusted third parties function on the trust model, which has its inherent weakness. Some transactions in e-commerce should be reversible to avoid disputes. The cost of mediating these disputes increases the cost of the transaction themselves [165]. There is no means to perform an online transaction without a trusted third party [165]. S. Nakamoto in [165] proposes a cryptographic model instead of trust. The transactions generated by the cryptographic model are nearly impossible to be reverse engineered. This would help in protecting consumers as well as merchants [165]. The cryptographic model will replace the trust-based model and the major advantage of this model is that solves the double spending problem using a distributed peer-to-peer timestamp server [165]. The timestamp server will generate proof of transactions occurring in chronological order. This distributed peer-to-peer system will remain secure as long as there are more honest nodes when compared to a group of synchronized malicious nodes.

Transactions

"*An electronic coin is defined as the chain of digital signatures*" [165]. Every owner in the chain hands the coin to the next by signing the hash value of previous transaction and the public key of next owner. This digital signature is added to the end of the coin. If a person who wants to pay can go through the chain and verify these signatures to check if they are real or falsified. They payee cannot verify if there was not any double spending on these transactions by any users [165]. This can be avoided by using a central authority. There is no difference between the central authority and the banks, so the drawback of using a trust based model will hold valid for the central authority as well. To avoid using the central authority the model proposed in [165] broadcasts all the transactions to all the nodes. Since multiple nodes will have the transactions stored in them the payee will have to verify whether the majority of nodes agree that there is no double spending.

Timestamp Server

The timestamp server proposed in [165] takes the hash value of a block that is to be time stamped and publicly announces it. The resulting timestamp suggests that the data must have existed for it to be included in the hash calculation [165]. Every timestamp adds the previous timestamp in the hash, thereby resulting in a chain as shown in Figure 14.

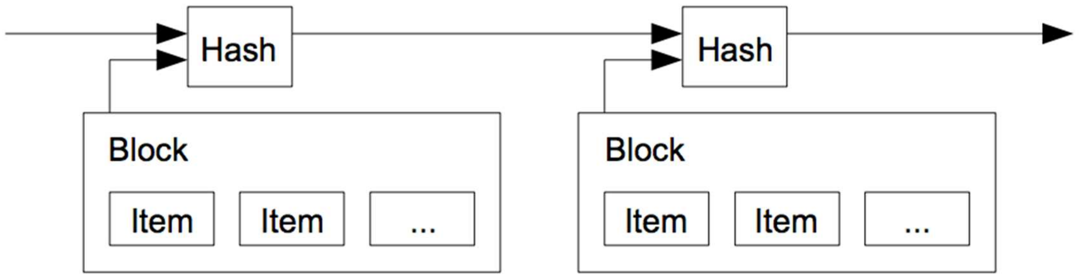


Figure 14 Implementation of a Timestamp Server in Bitcoin [165]

Proof-of-Work

Proof-of-Work system is similar to the Adam Back's Hashcash [178] and is used to implement a distributed timestamp server [165]. Proof-of-Work involves in finding for a value which when hashed results in a hash, which begins with a number of zero bits [165]. In Proof-of-Work the Nonce is incremented with every block being processed. When a Proof-of-Work is computed, the block becomes immutable and cannot be changed. The blocks Proof-of-Work can be changed only by changing the Proof-of-Works for all the blocks following it [165]. The longest chain has the most proof-of-Work's and it represents one CPU one vote rather than one IP one vote.

Network

In a Bitcoin network, all the transactions are broadcasted to all the nodes [165]. Nodes gather new transactions to a block and they try to compute the proof-of-work [165]. Once the proof-of-work is computed, it is broadcasted to all the nodes. The nodes will accept the block only if all the transactions contained within the block are valid [165]. If the node accepts the block, then they are added to the chain and the hash value of the next block is calculated by the current hash value and the previous hash value [165].

Verifying a Payment Using Block Chain

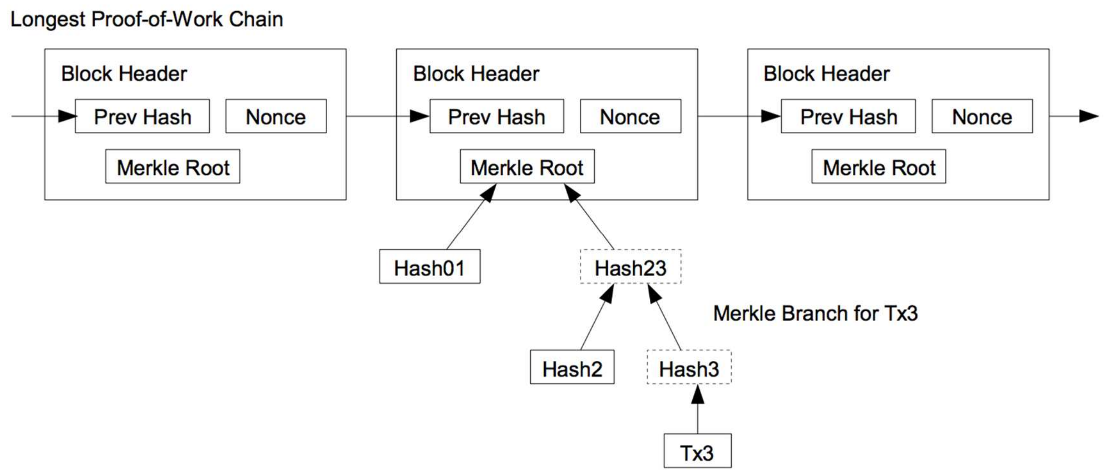


Figure 15 Verification of Payment made using Block Chain [165]

To verify the payments the user has to get access to the longest proof-of-work chain. Once the longest proof-of-work chain is obtained the user can access the Merkle [179] branch, which connects the transaction to the block [165]. The user will not be able to check for an individual transaction but the user can check if the entire block is valid or not [165]. When nodes encounter, and discover invalid blocks, these block chains will be completely replaced alerting transactions to assert their inconsistency [165] as shown in Figure 15.

Privacy

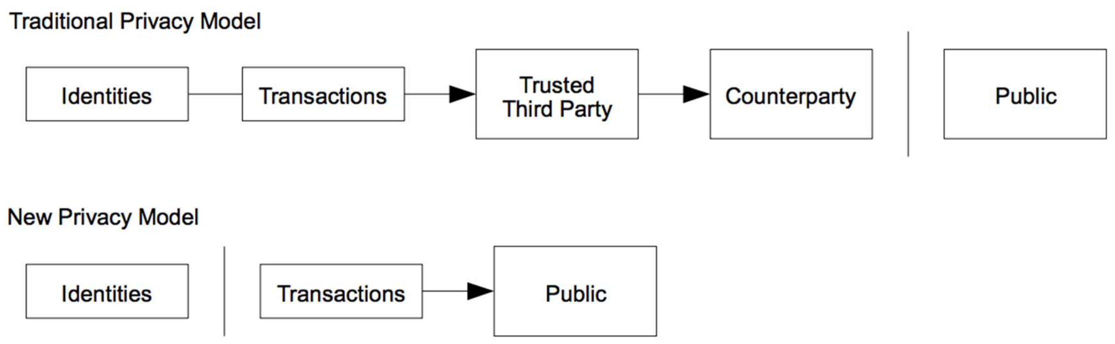


Figure 16 Traditional Privacy model vs. Bitcoin Privacy Model [165]

The traditional privacy model maintains privacy by preventing public to access the identities of users. Identities of the users are limited to the trusted third parties alone. In the privacy model in Bitcoin the transactions are made public whereas the user identities are not linked with the transactions themselves as shown in Figure 16.

CHAPTER III

STRUCTURING AND LINKING DATA

Problem Statement

The first step in protecting sensitive information is to identify all data items in a dataset and to determine similarity between data items spanning across multiple datasets. In the current Hadoop implementation, the scope of metadata generated by Hadoop is limited to the block level and not dataset level [12]. Identifying attributes (data items) in a dataset is a straightforward approach for structured datasets, which adhere to a specific format and when there is information from the data owner. Even the data owners provide this information it can be lost during transmission or get corrupted. Due to data democratization there are several datasets without any information provided by the data owner. Adding semi-structured data and unstructured datasets to this mix complicate things further. When there is no information available about datasets, these datasets are manually analyzed to know what data items (attributes) they contain. Irrespective of whether the information provided by the data owner is available or not, multiple datasets are linked often by intuition or manual analysis.

Introduction

The novelty of the proposed framework is that it makes use of only the dataset itself to generate relevant metadata. To identify related data items spanning across multiple datasets the proposed framework harnesses both context and usage patterns. Context patterns by themselves will identify the data items that may be similar in both data sets by using their names, content and other constraints such as data type. Contextual similarity can identify similar data items in multiple datasets; but fails to capture semantic relationships [46]. To augment context similarity and to identify related data items that were not identified by context similarity, the proposed framework makes use of usage patterns. This is based on the assumption that semantically related data items will be used together. Usage patterns correspond to how the data has been used over a period of time. These patterns can be used to reveal any privacy breach, user behavior and data misuse or abuse. The components in the proposed framework, which are responsible for generating metadata, tracking usage and linking data items in datasets, are Enhanced Metadata Generator (EMG), Data Usage Tracker (DUT) and Data Similarity Analyzer (DSA) respectively.

Proposed EMG generates two types of metadata namely structural and descriptive metadata. Structural metadata comprises of information about data items in a dataset whereas descriptive metadata comprises of a description or summary of a dataset. Structural metadata is generated for the datasets ranging from structured to unstructured which can be coerced into a structured dataset. Some unstructured datasets like free text datasets cannot be coerced into a structure. For these types of datasets, the EMG generates descriptive metadata.

Literature Review and Related Work

Generating Metadata and Structuring Data in Hadoop

In the current Hadoop implementation [12], metadata is limited to data blocks rather than dataset itself. Block level metadata will not be helpful in identifying data items within a given dataset. The proposed framework is the first of its kind to generate metadata using the dataset itself.

An application developed by J. Shin et al called DeepDive [60], is used to convert “dark data” into structured data. Large unstructured data constitutes “dark data” [60]. DeepDive augments database and machine learning techniques to the Knowledge Based Construction (KBC) techniques to facilitate conversion of unstructured to structured data. KBC proposed in [60] is iterative and the authors propose incremental techniques (based on sampling and variation techniques) to produce inference results for the KBC systems. Although DeepDive can generate structured data from unstructured data, it is not implemented to work on top of Hadoop i.e. KBC process is not parallelized [60]. According to the authors of DeepDive scalability of feature extraction for the KBC process is a challenge given a large amount of data [60]. The proposed framework is highly scalable because it works on top of Hadoop taking advantage of the parallelism it provides via MapReduce. The proposed framework makes use of MapReduce programming model to generate relevant metadata for the datasets.

Context Similarity Measures

Process of defining relationships or logical mappings across multiple datasets is called schema matching [62]. There are many techniques that can perform schema matching such as linguistic matching, structure-based matching or graph matching, and constraint-based matching. In the proposed framework, all the above three techniques are used to identify similarity between multiple data items across several datasets. Linguistic matching detects semantic similarities between concepts of element from different data sources. This technique evaluates similarity between element’s name and description by combining results from different processes such as stemming, tokenization, string and substrings matching and information retrieval. These processes are commonly considered as the matching conditions to evaluate the correspondences between these entities [62]. Graph matching includes two algorithms, which are fixed-point computations on similarity propagation graph [63] and probabilistic constraint satisfaction algorithms [64]. The first one needs two or more input graphs (schemas or structures) to produce an output mapping which describes the relationships among the elements of the graphs [63]. The algorithm takes a couple of ontology, and finds several practical similarity measures to implement the mappings [64]. Constraint-based matching is a technique, which consider the properties of the elements such as data types, value ranges, uniqueness, null-ability and foreign keys [62]. W. Shen et al [65] and V. Le Clément et al [66] make use of constraint-based matching techniques in entities and graphs. Constraint-based entity matching proposed in [65] is a generative model to improve matching accuracy. Constraint-based graph matching proposed in [66] is a constraint-based modeling language, which can support both Constraint Programming and Constraint-based Local Search.

Usage Pattern Similarity Measures

Usage patterns of data have been used to find semantic relations between learning resources in [46] by K. Niemann et al and in recommender systems. A recommender system provides a user with a set of items, which he or she might be interested by comparing the user behavior with other users behavior. Recommender systems can be user based, item based [68], collaborative filtering [69], content based filtering [70] or hybrid [67]. In the proposed framework item based [68] collaborative filtering approach, with a variation in measuring the similarity between different data items is used. The reason to factor in the usage pattern similarity is to identify the linked data items in addition to the aforementioned context similarity measures. If two words appear in very different contexts, then they are semantically unrelated. A word that appears in various contexts can be called polysemous. But if two words occur in very similar contexts, then they may or may not be semantically related to one another [46]. Relatedness can be specified as a metric to determine semantic similarity if two words are co hyponyms (they have a common higher-level concept, which is true for words with highly similar contexts) [46]. Thus by comparing the usage of the contexts of words their semantic similarity can be determined. By extending these to data items, comparing their usage contexts will determine their semantic similarity.

The Metadata Generator

Introduction

A precursor to the EMG is proposed in [18] to generate metadata and to link relevant data items across datasets. There are two modules created in [18] which can be interfaced to the existing Hadoop implementation. Data context analysis module is responsible for generating metadata and identifying similarity between data items based on linguistic, structure-based and constraint-based matching. Data usage pattern analysis module tracks data usage patterns and identify semantically related data items based on usage. Data usage analysis module makes use of Markov's graph clustering algorithm [71]. Results from these modules are combined together to identify related data items. The architecture of these modules is shown in Figure 17.

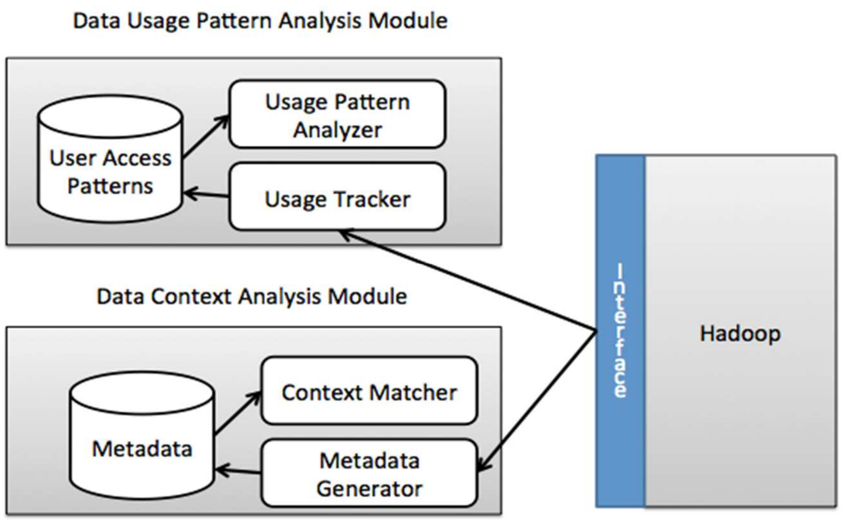


Figure 17 System Architecture proposed in [18]

Data Context Analysis Module

Data context analysis module has two major components namely metadata generator and context matcher as shown in Figure 17. A detailed description of these components is described below.

Metadata Generator

Metadata generator is a Java program that analyses blocks of data of a dataset that is stored in a Hadoop cluster. These blocks are chosen in random to be processed by the metadata generator. The data items and their corresponding data types are stored in the metadata log. When a new dataset is added to the Hadoop file system, the metadata generator generates a list of data items that the dataset contains. The metadata is removed from the HDFS when the dataset is removed from the HDFS.

Context Matcher

The context matcher makes use of techniques like linguistic matching, schema matching and constraint-based matching to estimate similarity between data items. A detailed description of these techniques is provided in this section. The similarity between datasets u and u' is measured by Equation 6.

$$S(u,u') = \alpha * S_P(u,u') + \beta * S_Q(u,u') + \gamma * S_R(u,u') \qquad (6)$$

In Equation 6 factors α, β and γ are parameters between 0 and 1, and α+ β +γ=1. S_P(u, u'), S_Q(u, u') and S_R(u, u') represent linguistic, structural and constraint matching scores respectively.

Linguistic Matching

$$S_P(u,u') = \begin{cases} \rho(u,u'), if\ \rho(u,u') \leq \varphi \\ 0, otherwise \end{cases} \qquad (7)$$

Linguistic matching scores between two datasets can be estimated using Equation 7. In Equation 7, ρ(u, u') represents similarity degree of datasets u and u', which can be searched from an auxiliary information file (A dictionary contains all relevant pairs of similar names). The factor φ is a threshold value of similarity degree.

Structure Matching

$$S_Q(u,u') = \begin{cases} 1, if\ u\ and\ u'are\ single\ elements \\ \dfrac{|f(u,d) \cap f(u'd)|}{|f(u,d) \cup f(u',d)|}, otherwise \end{cases} \tag{8}$$

Similarity scores between two datasets based on their structure can be estimated using Equation 8. In Equation 8, f(u, d) represents a dataset which contains all elements related to u in d hops, and the cardinality of f(u, d) denoted as |f(u, d)| counts the total elements in f(u, d).

Constraint Matching

$$S_R(u,u') = \begin{cases} \omega^{g(u,u')}, if\ g(u,u') \leq\ \theta \\ 0\ , otherwise \end{cases} \tag{9}$$

Similarity scores between two datasets based on user-defined constraints are estimated using Equation 9. In Equation 9, g(u, u') is the similarity degree of u and u' in the auxiliary information file which contains all relevant pairs of similar names. Factor ω is a parameter between 0 and 1; and θ is a threshold value.

Data Usage Pattern Analysis Module

Data usage pattern analysis module has two major components namely usage tracker and usage pattern analyzer as shown in Figure 17. A detailed description of these components is described below.

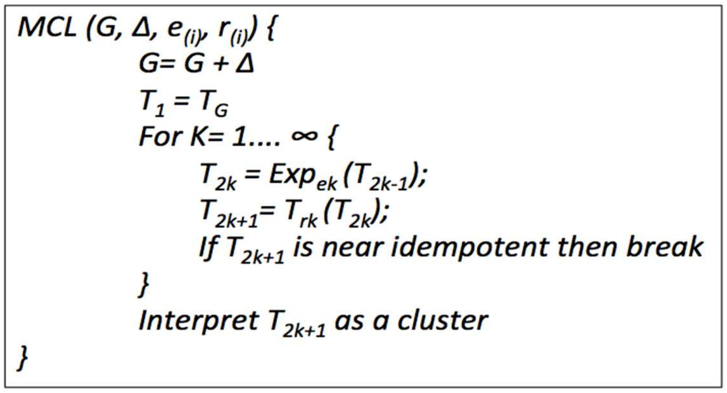
Usage Tracker

Usage Tracker tracks the users usage patterns. These usage patterns consist of user information, timestamp, datasets and data items accessed by users. Jobs submitted by users and the results of these jobs pass through the data usage tracker where it records all the necessary information as a log file. Usage tracker identifies data items with the help of the metadata generated by the metadata generator. All the usage patterns generated are stored in HDFS like metadata.

Usage Pattern Analyzer

Usage pattern analyzer identifies the semantic relationship between data items based on their usage tracked by the usage tracker. Based on the usage information, the usage pattern analyzer builds a weighted graph with data items as nodes and constructing edges between data items that were used together with their weights as their usage frequencies. In order to identify the semantic relationship, the data usage pattern analyzer implements Markov's algorithm [71] a graph-clustering algorithm. Since there is no assurance that Markov's algorithm will terminate [46], in order to prevent Markov's algorithm from running forever the data usage pattern analyzer also implements Iterative Inductance Cutting (ICC) Algorithm [46, 72]. ICC begins with a single cluster and splits it into two and proceeds further until a minimum threshold is met.

Markov's algorithm is based on the fact that a random walk will not leave a dense cluster until most of its vertices have been visited [71]. Random walk is analogous to a finite Markov Chain as the future states are not dependent on past states for a given present state [71]. Markov's algorithm implemented by the data usage pattern analyzer is shown in Figure 18.



```
MCL ($G, \Delta, e_{(i)}, r_{(i)}$) {
        $G = G + \Delta$
        $T_1 = T_G$
        For $K = 1 .... \infty$ {
            $T_{2k} = Exp_{ek}(T_{2k-1})$;
            $T_{2k+1} = T_{rk}(T_{2k})$;
            If $T_{2k+1}$ is near idempotent then break
        }
        Interpret $T_{2k+1}$ as a cluster
}
```

Figure 18 Markov's Clustering Algorithm [71]

In Figure 18 MCL represents Markov's clustering Algorithm [17]; G represents undirected, weighted graph; Δ represents Kronecker delta; e represents expansion parameter; $e_i \in N$, $e_i > 1$, $i=1 \cdots n$; r represents inflation parameter; $r_i \in R$, $r_i > 0$, $i=1 \cdots n$; T_i corresponds to intermediate matrices. Inputs to the algorithm are the adjacency matrix of the un-directed weighted graph constructed from the usage patterns, expansion parameter, inflation parameter and Kronecker delta. Self-loops are added to the adjacency matrix initially to be normalized. There are two important operations in Markov's algorithm namely expansion and inflation. The inflation operation raises each entry in Matrix M to inflation parameter r; this is followed by normalizing sum of columns to 1 [71]. The expansion operation is used to expand dense regions in the graph. Both these operations are applied alternatively in iteration beginning with the adjacency matrix. Expansion operations strengthen intra-cluster flow and minimize inter-cluster flow [71]. Initially the flow-graph is smooth and after some iteration, it becomes heightened between tightly linked nodes. Contextual similarity is not a necessary condition for data items to be in same cluster [46].

Framework to Generate Metadata, Link and Track Data Items

Introduction

In [18], a metadata generator that identifies data items and their data types for all types of non-multimedia dataset in Hadoop is proposed. But the accuracy of the proposed metadata generator in [18] was limited for unstructured data. In the Sensitive Data Detection (SDD) framework the shortcomings of the metadata generator proposed in [18] have been addressed using a trained neural network. This Enhanced Metadata Generator (EMG) aids in identifying data items, their data types and uniqueness (whether the data item has unique values) for all types of non-multimedia datasets in the framework. Some unstructured datasets, which has free text, cannot be structured. To represent these datasets, the EMG generates descriptive metadata by making use of Automatic Text Summarization (ATS) techniques proposed in [73, 74]. These ATS techniques represent the free text datasets using lexical chains. Lexical chains are chain of words, which are semantically related.

In addition to improving the metadata generator, data similarity analyzer's equations were changed so that the similarity between data items across multiple datasets is identified rather than comparing only two datasets at a time. Other than this the SDD framework includes a Provenance Tracker (PT) to track data items used by any process or user at any given time. The SDD framework is shown in Figure 19. A detailed description of the components of the components responsible for generating metadata, linking and tracking data items is described in detail in the following sections.

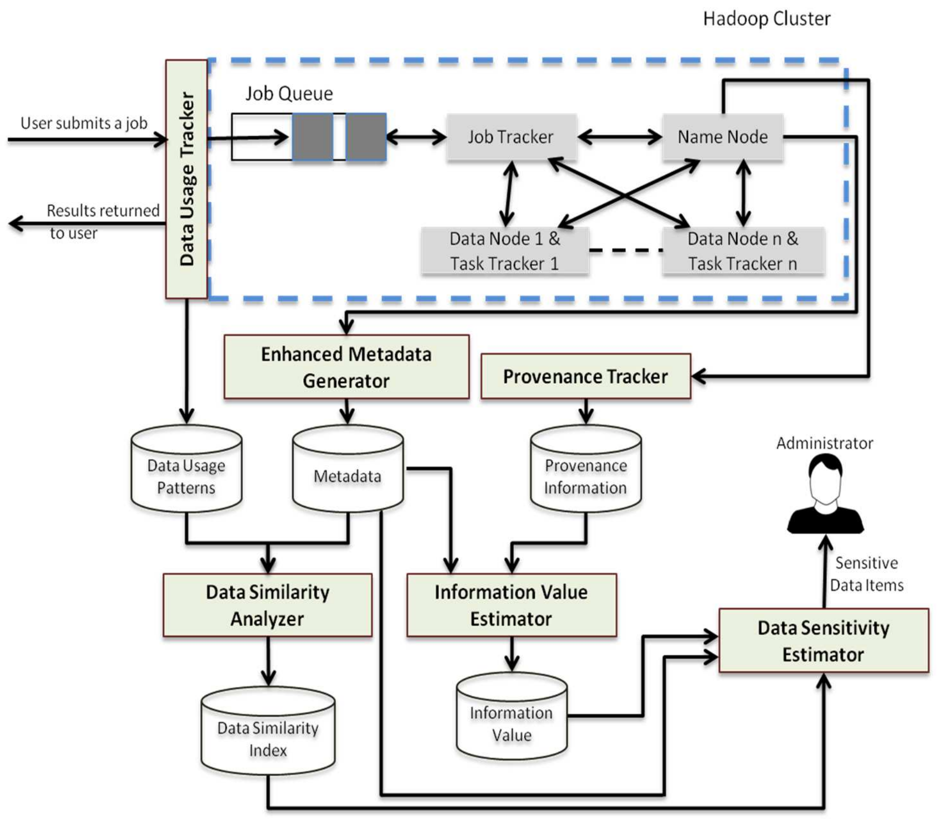


Figure 19 Sensitive Data Detection (SDD) Framework [19]

Roles and Responsibilities

Roles and responsibilities of various entities in the SDD framework are described below in detail.

Administrator

The proposed framework provides the administrator with a list of potentially sensitive data items that are present in a dataset and he or she takes appropriate actions in safeguarding these data items from misuse by authorized personnel and from accessing by authorized users.

Domain Expert

The domain expert has an important role in determining sensitivity of data items which have not been encountered before or whose information value is high. Once the domain expert determines the sensitivity of a data item, it is used to train a neural network so that it can determine the sensitivity of similar data items. This alleviates the workload of the domain expert.

Enhanced Metadata Generator (EMG)

The EMG is capable of generating both structural and descriptive metadata. Structural metadata contains information about the data items in the dataset whereas descriptive metadata provides a brief textual summary of the content in the dataset. Whenever a new dataset is stored in Hadoop, EMG generates relevant metadata and on deletion of the dataset, the corresponding metadata is also deleted. In order to make the generated metadata more reliable and available, it is stored in HDFS [12]. The EMG implements the algorithm shown in Figure 20 to generate structural metadata by identifying individual data items, its data type and uniqueness for every type of non-multimedia dataset.

```
ds: dataset;patt: frequently occurring patterns;
dtype: data set type;st: structured; unst: unstructured
Generate_Metadata(){
patt = Get_Freq_Occr_Patt(ds);
dtype = Type_of_Dataset(ds,patt); //algorithm2
If(dtype == st || dtype == unst)
   get_item_name(ds,patt); //algorithm3
  get_uniq_itype(ds,patt);
Else
  get_item_name(ds); //algorithm4
  get_uniq_itype(ds,patt);
End if
}
```

Figure 20 Pseudo code of Structural Metadata Generation Algorithm

The metadata generator algorithm as shown in Figure 20 identifies frequently occurring patterns in a non-multimedia dataset, using natural language processing. Patterns that occur more often throughout the dataset are rated based on their frequency. The pattern, which has the highest frequency, occurs more common in the dataset and is considered as the frequently occurring pattern. These patterns will determine the type of dataset. This is done using the algorithm "Type_of_Dataset" (Algorithm 2), whose pseudo code is shown in Figure 21.

```
ds: dataset;patt: frequently occurring patterns
Type_of_Dataset (ds,patt) {
c=count the occurrences of frequently occurring pattern in ds
If (c == length(ds))
   structured;
Else
   pass dataset using JSON/XML
   If no exception
      semi-structured
   Else
      unstructured
   End if
End if
}
```

Figure 21 Pseudo code of Algorithm 2

If the frequency is equal to the length of the dataset, then the dataset is structured. If the dataset can be parsed by a XML or JSON parser, then it is semi-structured else the dataset is unstructured. After determining the type of the dataset, based on the type of the dataset the structural metadata generation algorithm shown in Figure 20, calls either algorithm 3 (when dataset is either structured or unstructured) or algorithm 4 (when the dataset is semi-structured). These two algorithms are used to identify the data items present within the datasets. Pseudo code of algorithm 3, which identified data items in structured and unstructured dataset, is shown in Figure 22.

```
For a structured/unstructured dataset
If (header is available)
    Use header for data item names
Else
   For each record in dataset
        Split each row by using patt
        Pass values to the trained neural network.
        Get data items names from a trained neural network
   End for
End if
```

Figure 22 Pseudo code of Algorithm 3

If a header is available in a dataset, then algorithm 3 makes use of the header. If the header is not available in the dataset then the algorithm 3 makes use of a neural network trained with multiple training datasets [75, 76 and 77]. These training datasets will allow the neural network to identify names, cities, states and so on. If a dataset is a semi-structured dataset JSON (or) XML, then the EMG uses algorithm 4 to identify data items. Pseudo code of this algorithm is shown in Figure 23.

```
For semi-structured dataset
Identify tag names or key/value pairs
data_item_names = tag names
```

Figure 23 Pseudo code of Algorithm 4

To generate the descriptive metadata, the proposed EMG uses Automatic Text Summarization techniques proposed in [73, 74]. The proposed EMG exploits semantic relatedness between words to describe a dataset. The semantic relatedness is used to construct lexical chains by identifying and grouping related words [73]. Semantic relationships between words in English are obtained from the WordNet lexical database [78]. Words can occur in multiple senses and hence a word sense disambiguation technique is required. EMG uses word sense disambiguation technique proposed in [74]. A two-pass algorithm is proposed in [73] to compute lexical chains and to compute feature vectors from these lexical chains. The proposed uses only one pass of the two-pass algorithm proposed in [73]. Once lexical chains are computed, the lexical chain with greater strength is selected to summarize the dataset. Strength of a lexical chain is given by number of words in it.

Data Usage Tracker (DUT)

Data usage tracker remains unchanged from its precursor usage tracker in data usage pattern analysis module [18]. Its functionality remains the same.

Data Similarity Analyzer (DSA)

DSA extends the data similarity analysis module proposed in [18]. DSA estimates similarity measures between datasets, which are obtained by combining the context similarity and usage similarity measures. Both these measures are necessary because the context similarity can capture similar data items but it fails to identify semantic relatedness between them. The proposed DSA uses metadata generated by EMG and usage patterns generated by DUT and produces a data similarity index, which is in between 0 and 1. Disjoint data items have their similarity scores close to zero whereas similar data items will have a higher score. The architecture of DSA is shown in Figure 24. DSA has two major components namely data usage pattern analyzer and data context similarity analyzer.

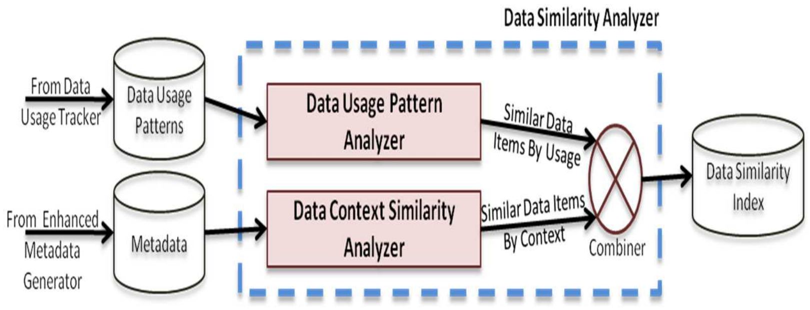

Data Similarity Analyzer
From Data Usage Tracker
Data Usage Patterns
Data Usage Pattern Analyzer
Similar Data Items By Usage
From Enhanced Metadata Generator
Metadata
Data Context Similarity Analyzer
Similar Data Items By Context
Combiner
Data Similarity Index

Figure 24 Architecture of Data Similarity Analyzer (DSA)

Data Context Similarity Analyzer

The data context similarity analyzer uses the structural metadata generated by EMG, to find relevant data items based on their names, data types and any other application specific constraints. Combining the scores of three techniques namely linguistic matching [62], structure matching [62] and constraint matching [66] identifies similar data items. All these three techniques are required because they provide an estimate of similarity based on names, structure and other constraints. Similarity between datasets U_1, U_2, U_3 … U_N is measured by Equation 10.

$$S(U_1, U_2, U_3 \ldots U_N) = \alpha * S_P(U_1, U_2, U_3 \ldots U_N) + \beta * S_Q(U_1, U_2, U_3 \ldots U_N) + \gamma * S_R(U_1, U_2, U_3 \ldots U_N) \quad (10)$$

Where factors α, β and γ are weights whose value is between 0 and 1 and $\alpha+\beta+\gamma = 1$. $S_P(U_1, U_2, U_3 \ldots U_N)$, $S_Q(U_1, U_2, U_3 \ldots U_N)$ and $S_R(U_1, U_2, U_3 \ldots U_N)$ represents similarity measures obtained from linguistic, structural and constraint matching techniques respectively.

Linguistic Matching attempts to match data items with similar linguistic features. Equation 11 estimates linguistic matching quantitatively where ρ (U_1, U_2, U_3... U_N) represents degree of similarity between datasets U_1, U_2, U_3... U_N and this can be obtained from an auxiliary information file [62]. An auxiliary information file contains all relevant ρ (U_1, U_2, U_3 ... U_N) pairs having the same name. Value of S_P (U_1, U_2, U_3 ... U_N) is between 0 and 1. When they have similar linguistic features then the similarity measure will be as high as 0.8 to 1; otherwise the similarity measure will be less than 0.8 [62]. The factor φ is a threshold value of similarity degree and its optimal value is 0.7 [62].

$$S_P(U_1, U_2, U_3, \dots, U_N) = \begin{cases} \rho(U_1, U_2, U_3, \dots, U_N), & if\ \rho(U_1, U_2, U_3, \dots, U_N) \leq \varphi; \\ 0, & otherwise. \end{cases} \tag{11}$$

Structure matching attempts to match data items with similar features such as names, data types, etc. These are obtained from the metadata generated by EMG.

$$S_Q(U_1, U_2, U_3, \dots, U_N) = \begin{cases} 1, & if\ u\ and\ u'\ are\ single\ elements; \\ \dfrac{|f(U_1, d) \cap f(U_2, d) \cap \dots\dots \cap f(U_N, d)|}{|f(U_1, d) \cup f(U_2, d) \cup \dots\dots \cup f(U_N, d)|}, & otherwise. \end{cases} \tag{12}$$

Equation 12 estimates structure similarity between datasets [62] where f (U_N, d) represents a dataset where all data items are related to u in d hops; $|f$ (U_N, d)$|$ represents the cardinality (i.e.) the number of elements in the set. Structure matching similarity values is between 0 and 1.

$$S_R(U_1, U_2, U_3, \dots, U_N) = \begin{cases} \omega^{g(U_1, U_2, U_3, \dots, U_N)}, & if\ g(U_1, U_2, U_3, \dots, U_N) \leq \theta; \\ 0, & otherwise. \end{cases} \tag{13}$$

Constraint matching attempts to estimate similarity between data items based on constraints such as similar names. Constraint matching is calculated by Equation 13 where g (U_1, U_2, U_3... U_N) is similarity degree of the datasets U_1, U_2, U_3 ... U_N, which is computed based on the relevant pairs of similar names. The factor ω is a parameter between 0 and 1; and θ is a threshold value [66].

Data Usage Pattern Analyzer

Data usage pattern analyzer remains unchanged from its precursor usage pattern analyzer in data usage pattern analysis module [18]. Its functionality remains the same. It makes use of Markov's algorithm to identify semantic relationship between data items [71].

Provenance Tracker (PT)

Business or legal liabilities may arise when the service level agreements (SLA's) are not honored. The owner of the data will have restricted access to selected parts of data, which should be enforced throughout the lifetime of the SLA. In order to keep track of any restrictions put forth by the data owner the proposed SDD framework uses a Provenance Tracker (PT). PT also aids in maintaining data lineage. Data lineage refers to the process of keeping track of data at any given time and at any stage of a process. This is very crucial because with the help of data lineage one can know where the data is in a process at any given stage.

Experimental Results

For evaluating the proposed EMG, customer churning dataset for a company, provided by Teradata is used. The dataset has customer complaints from walk-in stores, online and through call centers. Headers from the datasets were removed. To test the proposed SDD framework it is assumed that there were no restrictions from data owner for data sharing. Some sensitive information like NAI, names of customer was added to the company dataset.

Data Items	Data Type	Unique?
First Name	String	N
Last Name	String	N
Address	String	N
City	String	N

State	String	N
Zip	String	N
Phone Number	Number	Y
Customer ID	String	Y

TABLE 2Metadata Generated for Customer Dataset

A snap shot of the metadata generated for the customer dataset from the telecom company data is shown in Table 2. Similarly, metadata is generated for other datasets as well and stored in the Hadoop cluster.

Data usage pattern consists of user information, timestamp, dataset and data items being accessed. A snapshot of data usage patterns for the Telecom Company is shown in Table 3. This usage information is invaluable as it contains patterns of user behavior.

User	Time-stamp	Datasets	Data Items
User1	Sept, 10, 2014; 10:23:00 AM	Customer	First Name, Plan ID, Age, Gender
		Click-Stream	Timestamp, URLs Visited
User2	Sept, 1, 2014; 01:23:00 AM	Employee	First Name, Age, Gender
		Call-Center	Customer Phone #, Call Duration, Quality of Service

TABLE 3 Snapshot of Usage Tracked by Data Usage Tracker

Similarity between data items of based on data context is calculated based on linguistic, graph and constraint matching techniques. Table 4 shows what data items are similar between the customer dataset and other datasets.

	Cust Data	Emp Data	Walk-In Store	Call Center	Click Stream
Cust Data	------	First Name, Last Name, Phone#,	Phone #	Phone #	------

		Address, City, Zip			

TABLE 4 Results from Context Similarity Analyzer for Customer Dataset with Similarity Scores > 0.9

Usage pattern analyzer identifies similarity in usage patterns by constructing a weighted graph with data items as vertices and their usage frequency as weights of these edges. After constructing the graph, using Markov's algorithm [71], data items from multiple datasets that are semantically related are identified. Table 5 shows a snapshot of clusters of semantically related data items from Company dataset. There are three clusters shown in Table 5 (represented by the 3 columns), which widely vary based on how the data is being used by users. Even if two Hadoop clusters have same data, there is no assurance that they will have same usage patterns. Results from both data context similarity and usage patterns similarity are combined to obtain related data items.

#	Customer Dataset	Employee Dataset	Walk-In Store	Call Center	Click Stream
1	First Name, Last Name, Phone #	Employee ID, First Name, Last Name	Phone #, date, time, Service Details, Employee ID	Phone #, date, time, nature of complaint, Employee ID	
2	Online User ID				User ID, time stamp, URLs
3		Employee ID, First Name, Last Name	Date, time, Customer Satisfaction, Employee ID	Date, time, Customer Rating, Employee ID	

TABLE 5Semantically related Data Items Identified by Markov's Algorithm.

CHAPTER IV

DETECTING SENSITIVE DATA ITEMS

Problem Statement

Once all the data items are identified in a dataset, the next step is to identify which of them are sensitive. In current implementation of Hadoop implementation [12], there is no way to identify sensitive data items without prior knowledge of these data items and their sensitivity. Identifying sensitive data items without prior knowledge or without any information from data owners manually is a tedious and time consuming process due to the volume and variety of data stored in HDFS [38]. Identifying sensitive information will be very useful, as it will pave the path to protect them from misuse or abuse.

Introduction

The proposed framework is entirely data-driven to identify sensitive data items within a given dataset. To achieve this proposed framework makes use of an information sensitivity graph model. This model is implemented by the Data Sensitivity Estimator (DSE) component in the proposed framework. The Information sensitivity graph is constructed by harnessing metadata generated by the EMG, data usage tracked by the DUT and the data similarity generated by the DSA. The nodes of the information sensitivity graph are users, datasets and data items within the datasets and edges are usage patterns. After the construction of this graph, Shannon's entropy [36] and information gain [36] are used to identify sensitive data items. The proposed information sensitivity graph treats the data stored in Hadoop as a communication system. Sensitivity of data is computed by observing the effect of removing the data from the communication system. The data will be highly sensitive when the effect of removal is significant.

Literature Review and Related Work

Shannon's Information Entropy

In [36], C.E. Shannon expresses information entropy as the mean of information contained in a message that is sent through a communication system. Shannon's entropy of a discrete random variable X can be computed from Equation 14.

$$\mathbf{H}(\mathbf{x}) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \qquad (14)$$

In Equation 14, $p(x_i)$ represents probability mass function of state x_i, for a system with n different states. For every state x_i in a discreet information source there will be probabilities associated with these states $p(x_i)$ to produce several output symbols. Informally entropy can also be regarded as a measure of impurity, higher the entropy value greater the impurity.

Information Gain

Information gain is useful in identifying which attribute is important in a given feature vector. The change in entropy between the original state and the modified state can be quantitatively represented using Information Gain. Expected information gain can be calculated using Equation 15.

$$\mathrm{IG}(\mathrm{X}, \mathrm{a}) = \mathrm{H}(\mathrm{X}) - \mathrm{H}(\mathrm{X}|\mathrm{a}) \qquad (15)$$

In Equation 15, IG refers to the information gain, H(X) refers to the entropy of the system and H(X|a) refers to the entropy of a system after removal of node 'a'. In the proposed model when the value of information gain decreases, the sensitivity increases. This assumption is validated in the following sections.

Assessing security risk of a dataset

In [41], A. Harel et al., proposed a dynamic sensitivity based access control (DSBAC) framework for traditional databases. DSBAC framework is an extension of Mandatory Access Control (MAC) and it makes use of a misuseability score (m-score) to compute an access class on demand for the tuples. M-score is a quantitative measure to identify the extent of damage a user can cause when exposed to sensitive data unknowingly or by mistake [80]. In [41], misuseability score is calculated with the help of information quality, information quantity and other distinguishing factors. Although this measure is computed on demand, it is primarily applicable to traditional RDBMS. It will not be applicable to big data and m-score depends on the sensitivity function defined by a domain expert [79]. The sensitivity score function will vary based on the domain knowledge of the expert.

In [35] K. Sajko et al., state that information value obtained using a dataset can be used to assess its security risk. Assessing security risk of each dataset plays a vital role in putting security mechanisms in place for protecting the dataset. D.L. Moody and P. Walsh in [81] have proposed quantitative measures to estimate the value of information. In [81], it is stated that like an asset information also has a cost and a corresponding value but information does not follow laws of economics. R. Glazer in [82] suggests that the unique value of information must be used to compute its value.

Seven laws governing information value [81]

In [81] D.L. Moody and P. Walsh proposed seven laws, which can quantitatively estimate information value. These laws are described in detail as follows. These seven laws are used in the proposed framework to evaluate security risk of a dataset.

"Information is infinitely shareable"

Information can be shared with multiple parties without loss of its value; unlike other assets which only one party can possess and claim ownership [82]. Other assets will lose their value when shared between multiple persons (or) organizations (or) entities. This can be observed in Figure 25.

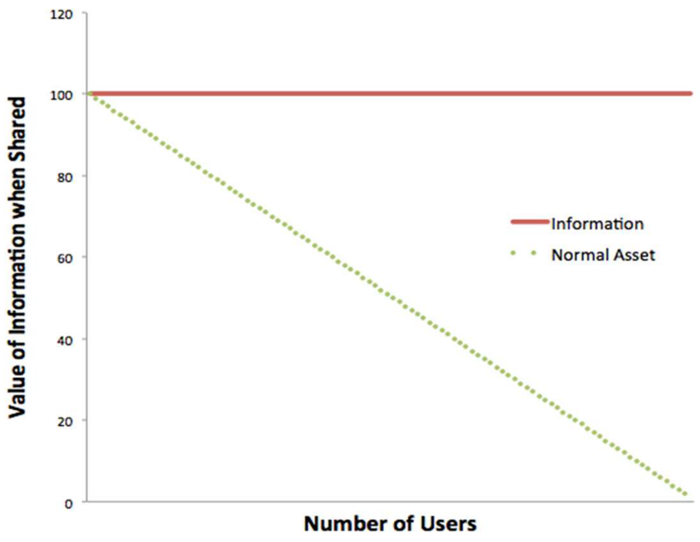


Figure 25 Value of information when shared [81]

An example of this is the World Wide Web (WWW) where information is disseminated to number of people. Sharing information is much easier and cost-effective than duplicating and maintaining several copies of the same data.

“Value of information increases with use”

Value of conventional assets will decrease over increased use. A suitable example for this will be reselling value of a car decreasing over its mileage. Value of information unlike conventional assets increases when used more. Therefore, it is right to say that it has increasing returns [81]. Cost of obtaining and maintaining information is initially high and this is paid for by the usage of information itself [81]. Unused information that sits idle in an organization is wasteful, as it is not being used to its fullest potential and ends up being a liability [81]. In Figure 26 a comparison between the value of information and value of a conventional asset over usage is shown in detail.

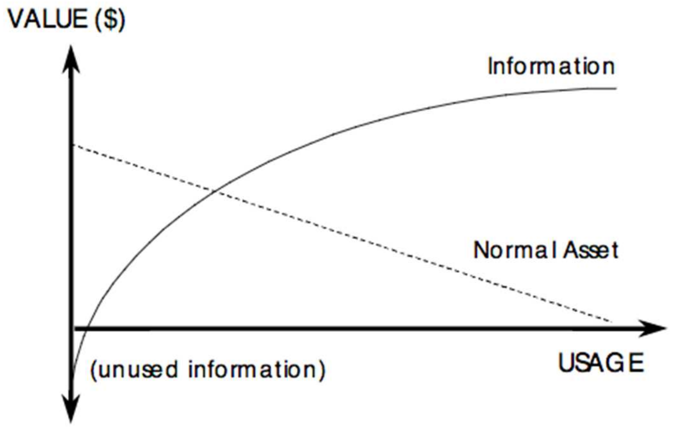


Figure 26 Value of information over usage [81]

“Information is perishable”

Information like every conventional asset loses its value over time [81]. The rate of depreciation of this value depends on the type of the information itself. In [81], the authors propose three “lives” or stages to information. These stages are “operational shelf live, decision support shelf life and statutory shelf life” respectively [81]. These three stages are depicted in Figure 27.

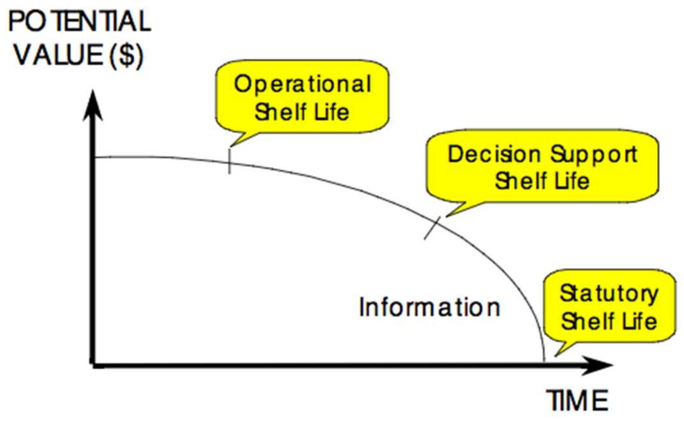


Figure 27 Value of information over time [81]

In Figure 27 “Operational shelf life” refers to the period of time up to which the corresponding information is likely to be valid. This period of time is relatively short [81]. “Decision support shelf life” refers to a period of time up to which the information can be used for identifying trends and making decisions based on these trends [81]. “Statutory shelf life” corresponds to the period of time up to which the information should be stored for legal purposes [81].

“Value of information increases with accuracy”

When the available information is accurate, it will be more valuable for decision-making and knowledge/pattern extraction [81]. Inaccurate information will be expensive for an organization, as it will result in incorrect decisions [83]. Type of information decides how accurate the information should be [81].

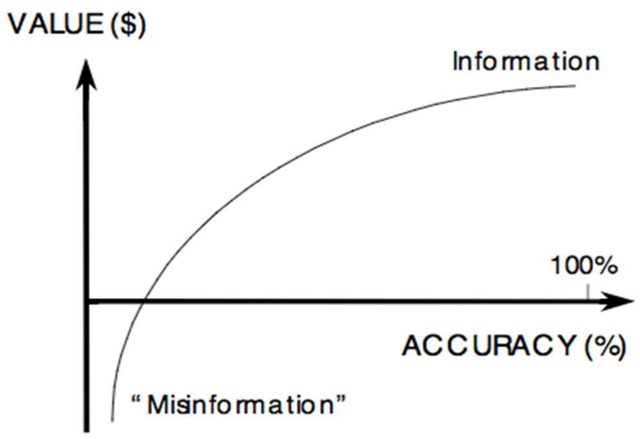


Figure 28 Value of information over accuracy [81]

In Figure 28 it can be observed that the value of information increases with increased accuracy. However, when the accuracy increases beyond a threshold increase in information value is not matched by the increase in accuracy. This can be explained by the fact that most organizations do not need 100% accurate data to make decisions [81]. If the accuracy of the data falls below a certain threshold, it is called as "*Misinformation*" [81]. Knowing the accuracy of information will help the decision makers to avoid making erroneous decisions based on incorrect facts [84].

"Value of information increases when combined with other information"

Value of a dataset increases drastically when it can be combined/aggregated with multiple other datasets. This is because the dataset by itself may be of little (or) no value for making certain decisions, but when combined with other datasets, the resulting information can be more intuitive [81].

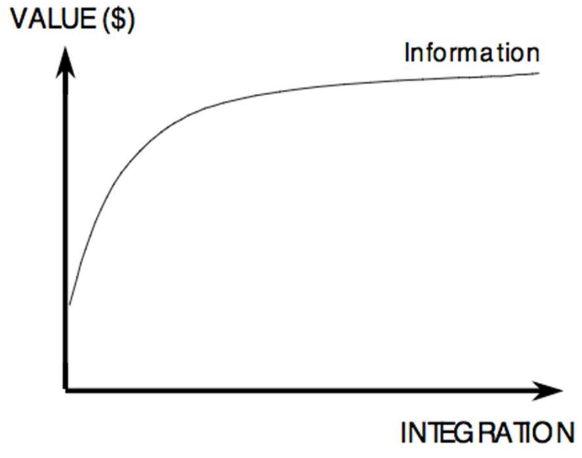


Figure 29 Value of information over aggregation (or) integration

From Figure 29, it is evident that the information value increases significantly after integrating (or) combining it with other sources. D.L. Moody and P. Walsh in [81] suggest that in an organization 80% of the benefits can be reaped by integrating just 20% of data in an organization. A study conducted by D.L. Goodhue et al., showed that integrating data beyond 20% will be counterproductive and could hamper the benefits attained through integration [85].

"More is not necessarily better"

In case of a conventional asset, the more one possesses it the better off they are [81]. With the proliferation of technology in these days, information is not scarce anymore. Mankind produces more information than ever before. The challenge facing many organizations in recent times is how to get useful insights from voluminous information. As humans, we have a limited capability to process information [86, 87]. When we are exposed to more information than we can handle it is called as information overload. As a result of information overload comprehension and decision-making capabilities of a person is affected [88, 89, 90 and 91].

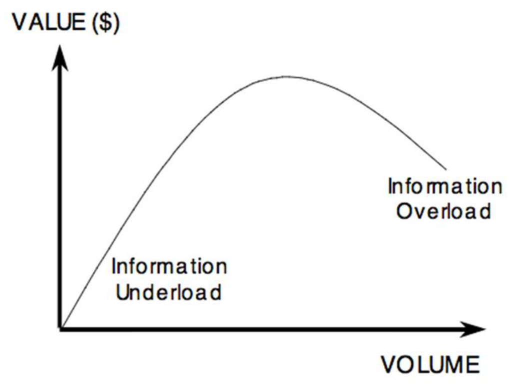


Figure 30 Information value over volume [81]

From Figure 30 one can determine that the value of information increases with volume up to a certain point and then it decreases due to information overload. Previous studies such as [89, 90 and 91] have shown that the availability of more information to people increases their confidence and satisfaction in making a decision. In spite of the above findings the aforementioned studies have also proved that if the availability of information exceeds beyond processing ability (information overload) then it certainly hampers comprehension.

“Information is not depletable”

When a conventional asset is used more it gets depleted faster. But this is not true for information. According to R. Glazer [82], information is “self-generating” and with increased use, there will be plenty of information left. This is because information unlike conventional asset can be summarized, analyzed, aggregated (or) joined with other sources, thereby generating new information. Another example can be data mining techniques as they can create new information based on existing data [81].

Information Value Model to Identify Sensitive Data Items

Introduction

A precursor to the DSE was proposed in [19] to identify sensitive data items. Identifying data items in [19] required the following components Information Value Estimator (IVE) and Data Sensitivity Estimator (DSE). Information Value Estimator (IVE) is used to identify value of information of a data item. Information value is considered a measure for assessing security risk based on the findings in [81]. Information value scores are low for data items that are highly valuable and vice-versa. Based on the similarity index produced by the DSA, metadata produced by the EMG and information value estimate produced by (IVE) the Data Sensitivity Estimator (DSE) determines the sensitivity of data items. These data items are vulnerable and can lead to compromising users' privacy.

IVE and DSE in the Sensitive Data Detection Framework are shown in Figure 19. A detailed description of these components will be provided in the following sections.

Information Value Estimator (IVE)

Information value of a dataset is predominantly used to assess security risk of a dataset [35]. Security risk assessment is essential in identification of proper security measures [35]. In the current Hadoop implementation [12], information value assessment is often done from the information from the data owners. The proposed Information Value Estimator (IVE) quantitatively estimates information value when there is no information from the data owners or from the metadata, usage patterns and similarity index. Information value is between 0 and 1. The lower value indicates higher security risk. Information value is calculated using Equation 16.

$$IV(D_i) = \alpha * Usage(D_i) + \beta * Data\ Quality(D_i) + \gamma * Relatedness\ Measure(D_i) - \delta * Lifetime(D) \qquad (16)$$

Where D_i represents an ith data item in a dataset D and factors α, β, γ and δ are weights whose value is between 0 and 1 and $\alpha + \beta + \gamma + \delta = 1$.

Equation 16 is based on the seven laws governing information value proposed in [81]. Some of these laws which were used to derive Equation 11 are 1) information value increases with the data usage; 2) information value increases with increase in the data quality; 3) information value increases when a dataset can be combined with other datasets; 4) information value decreases when the dataset is outdated [81]. Data usage is obtained from data usage patterns; relatedness is obtained from similarity index and data quality is measured based on the completeness and accuracy of dataset. Lifetime is the difference between the current time and the time at which the dataset was originally created. If there is information from the data owners regarding data sharing, then the proposed IVF uses this information solely instead of Equation 16.

Data Sensitivity Estimator (DSE)

The proposed data sensitivity estimator (DSE) identifies sensitive data items from datasets. The architecture of DSE is shown in Figure 31.

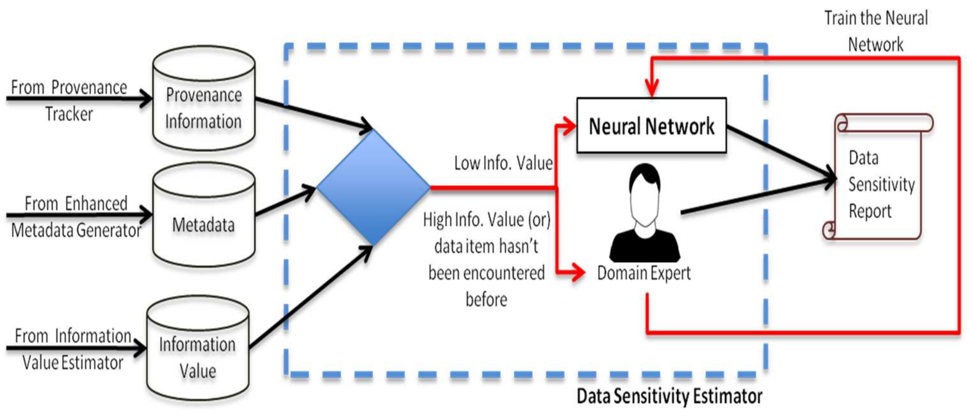


Figure 31 Architecture of Data Sensitivity Estimator (DSE) [19]

DSE uses provenance information, metadata and information value generated by PT, EMG and IVE respectively. If the information value is high or if the data item is not encountered before, then the sensitivity is determined by a domain expert. This sensitivity estimated by the domain expert is used to train a neural network. If the information value is low, then the trained neural network determines its sensitivity. Then the sensitivity report is sent to the administrator to put forth sufficient security measures to protect the identified sensitive information

Entropy Based Approach to identify sensitive data items

Introduction

Although the precursor to the entropy based approach identified sensitive items from datasets, this approach was dependent on finding optimal values for the constants α, β, γ and δ in Equation 16. The weights for these constants varied for different domains and it required a significant amount of time for deriving these weights. To avoid this intensive computation and to identify sensitive information across all domains, the entropy-based approach is proposed. A detailed description of this model is given in the following sections. The system architecture shown in Figure 32 is similar to the SDD framework in [19], without the Information Value Estimator (IVE) component.

Roles and Responsibilities

The roles and responsibilities of entities in the proposed framework shown in Figure 46are described below in detail.

Administrator

The proposed model provides the administrator with a list of potentially sensitive data items that are present in each dataset.

Domain Expert

The domain expert estimates sensitivity of data items, which are identified as potentially sensitive by the proposed model.

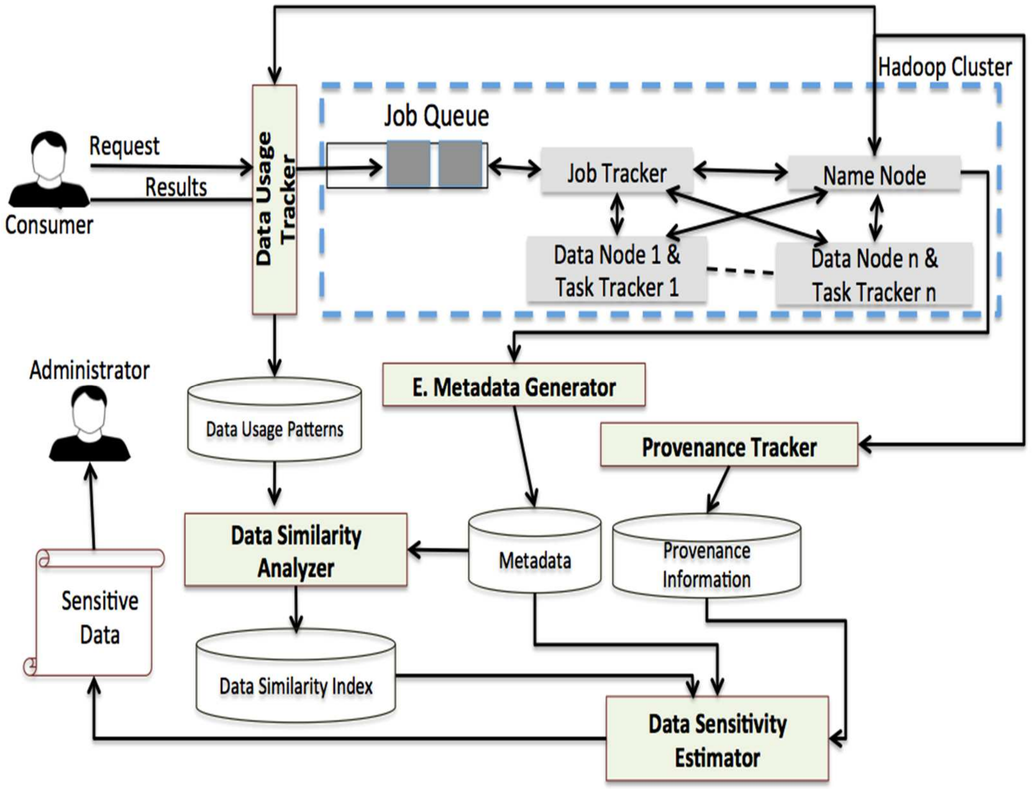


Figure 32 System Architecture – Identifying sensitive data items using Information Entropy

Data Sensitivity Estimator (DSE)

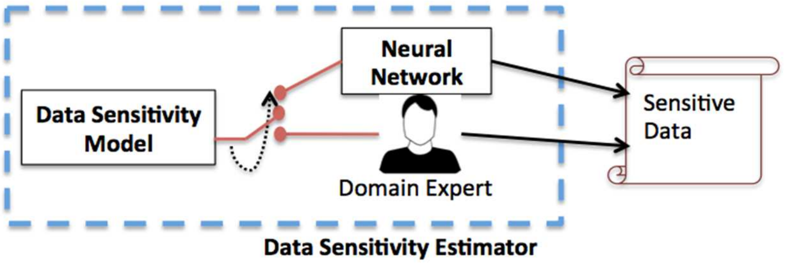


Figure 33Architecture of the Data Sensitivity Estimator (DSE)

The system architecture of the framework to identify sensitive data items using information entropy is shown in Figure 32. The proposed Data Sensitivity Estimator (DSE) shown in Figure 33, implements an entropy-based model to estimate information sensitivity. The proposed model selects data items, whose removal will have a significant effect on the system. Data items, which have no effect on the system, can also be potentially sensitive and identifying them as sensitive will increase the accuracy of the proposed model. Hence the proposed model makes uses the domain expert to estimate the sensitivity of the data items, which do not have significant effect on the system. Decisions of a domain expert are used to train a neural network. If similar data items are encountered in future, a trained neural network determines their sensitivity.

Model for Data Sensitivity

In the proposed entropy model, we consider three contributing factors to data sensitivity. These three factors were identified from the seven laws to quantitatively estimate information value proposed in [81]. Shannon's entropy [36] is adopted to estimate data sensitivity based on three factors namely data usage, data interconnectedness (similarity) and data quality. Shannon's entropy of a discrete random variable X can be computed from Equation 14. In the proposed model the changes for each node by removing that node from the network and recording the entropy change caused after removal of that corresponding node is calculated. This will give a measure of sensitivity of the node removed. If the removal of a node disconnects the network during entropy calculation, then the largest connected sub-graph is used to calculate entropy. Information Gain can represent the change in entropy between the original state and the modified state (after removal of the node). Expected information gain can be calculated using Equation 15. The factors contributing to the information sensitivity are as listed below.

Data Usage

The more a data item is used, the more sensitive it is because the probability of the data item being abused or misused increases with its usage.

Data Similarity

When there are many data items across multiple datasets that are similar to a particular data item, the probability of these items being sensitive increases because as the number of similar data items increases, the number of datasets, which can be, aggregated with the dataset increases.

Data Quality

The higher the data quality of a dataset in terms of missing data, corrupted or erroneous data, higher its sensitivity.

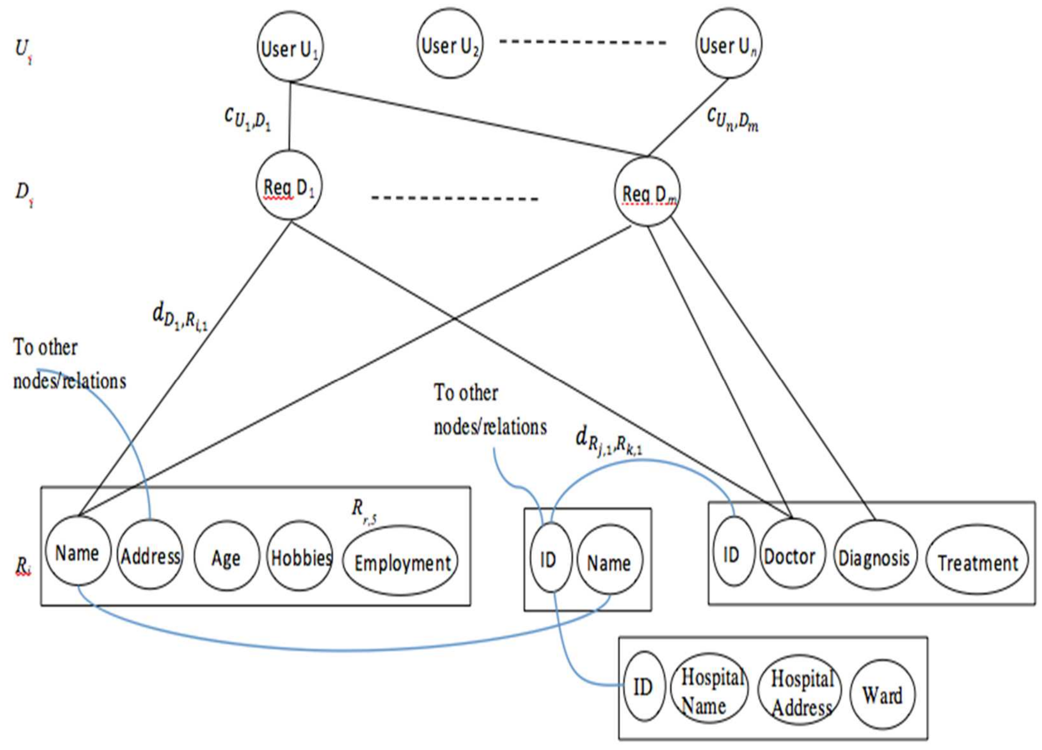


Figure 34 An Example of Information Sensitivity Graph

Information Sensitivity Graph

A sample data sensitivity graph is shown in Figure 34. Based on the information from EMG, DUT, and DSA we model the datasets stored in HDFS as a graph as shown in Figure 34.

In Figure 34 nodes represent the following:

- U_i represent a user *i*
- D_i represents a request *i* sent by user
- R_i represents a dataset *i* (In Figure 34 Patient, Doctor are examples of datasets)
- $R_{i,j}$ is a specific data item *j* in a dataset R_i.

In Figure 34 edges represent the following:

- c_{U_i,D_j} is the number of times a user u_i has made the request D_j.
- $d_{D_i,R_{j,k}}$ is an edge from a request *i* to a specific data item *k* in a dataset R_j.
- $d_{R_{i,j},R_{m,n}}$ is an edge from a specific data item *j* in a dataset *i* to a specific data item *n* in a dataset *m*.

Proposition 1: In the data sensitivity graph, whenever datasets are aggregated and used there exists a path, which starts at the node representing the dataset D_i and ends in the node representing the dataset D_j.

Proof: For every request involving aggregation and usage of multiple datasets ($D_1,\ldots D_N$), there exist edges $d_{D_1,R_{i,j}}$ …. $d_{D_N,R_{m,n}}$ that connects the datasets being accessed (D_i) to the data items being accessed in that dataset. In addition to these there will be edges $d_{R_{i,j},R_{m,n}}$ that connects similar data items across multiple datasets based on the similarity index. Thus a cyclic path can be traced from D_i back to itself when multiple datasets are aggregated.

Proposition 1 indicates the data items or attributes in the path accessible by a user. For example, in Figure 34, a patient's name and ID are now available to the user because the nodes are in the path. The goal of our work is to restrict the arcs that connect the nodes and thus maintain privacy.

Calculating Information Gain

Change in entropy (sensitivity) of a dataset based on the usage, data similarity, data quality and data vulnerability are calculated as shown below.

Probability mass function for data usage c_{U_i,D_j} in the data sensitivity graph is calculated using Equation 17.

$$p(R_i) = \frac{\sum_{i=1}^{m} c_{U_i,D_j}}{\sum_{\boldsymbol{j=1}}^{\boldsymbol{k}} \sum_{i=1}^{m} c_{U_i,D_j}} \quad (17)$$

In Equation 17, $\sum_{i=1}^{m} c_{U_i,D_j}$represents the number of times a request D_j has been made by all users; $\sum_{j=1}^{k} \sum_{i=1}^{m} c_{U_i,D_j}$represents the total number of requests made by all users. The usage entropy is computed for all the requests in the graph. Data items in a dataset can be accessed by several requests.

Probability mass function for data connectivity is calculated using Equation 18.

$$p(R_i) = \sum_{j,m,n=1,1,1}^{j,m,n=a,b,c} d_{R_{i,j},R_{m,n}} \Big/ \sum_{i,j,m,n=1,1,1,1}^{i,j,m,n=d,a,b,c} d_{R_{i,j},R_{m,n}} \quad (18)$$

A dataset may serve as a connection point between other datasets. The dataset in Figure 34 (with attributes ID and Name) contains data items, which are similar in a number of other datasets. Such nodes are very sensitive data as they connect different datasets. Data items, which are similar to data items in other datasets, are sensitive as they provide access to data items, which are potentially sensitive in other datasets. In equation 18, $\sum_{j,m,n=1,1,1}^{j,m,n=a,b,c} d_{R_{i,j},R_{m,n}}$represents the number of arcs or paths incident to dataset R$_i$, and $\sum_{i,j,m,n=1,1,1,1}^{i,j,m,n=d,a,b,c} d_{R_{i,j},R_{m,n}}$is the sum of all arcs or paths incident to all datasets.

$$p(R_i) = \frac{\sum_{s_r} co(R_j)}{s_j} \Big/ \sum_{i=1}^{n} \frac{\sum_{s_i} co(R_i)}{s_i} \quad (19)$$

The probability mass function based on data quality can be computed using Equation 19. In the proposed model, we represent data quality in terms of missing data and erroneous/corrupted data. The higher the data quality higher the data sensitivity. In equation 19, *co(R$_i$)* represents the number of correct entries for all data items in *R$_i$, s$_i$* represents total number of entries for all data items in *R$_i$*; $\frac{\sum_{s_r} co(R_j)}{s_r}$ represents the proportion of correct data in a single dataset; $\sum_{i=1}^{n} \frac{\sum_{s_i} co(R_i)}{s_i}$represents the total number of correct entries in all datasets.

The combined entropy measure is the product of all the three entropy measures computed using Equations 17, 18 and 19. Combined entropy measure is represented in the following equation

$$\mathrm{H}(x_i) = \mathrm{H}(c_{u,d_i}).\mathrm{H}(d_{r_i}).\mathrm{H}(N_{r_i}) \quad (20)$$

In Equation 20, $\mathrm{H}(x_i)$denotes the combined entropy measure for a dataset D$_i$, $H(c_{u,d_i})$denotes the entropy measure calculated by data usage and is calculated as shown in Equation 21, *H(d$_r$)* denotes the entropy measure calculated by data similarity and is calculated as shown in Equation 22 and *H(N$_r$)* denotes the entropy measure calculated by data quality and is calculated as shown in Equation 23.

$$\mathrm{H}(c_{u,d_i}) = -p(R_i)\log p(R_i) \ (21)$$
$$H(d_{r_i}) = -p(R_i)\ log\ p(R_i)(22)$$

$$H\left(N_{r_i}\right) = -p(R_i)\, log\, p(R_i)(23)$$

Sensitivity score of a dataset is determined by the effect of removal of the dataset in the data sensitivity graph. It is the difference of the sum of the combined entropy measure of all datasets and the entropy of the dataset.

$$\mathrm{C}(x_i) = \sum_{k=1}^{n} H(x_k) - H(x_i) \quad (24)$$

In Equation 24, $\mathrm{C}(x_i)$denotes the sensitivity of a dataset i, $H(x_i)$denotes the entropy score of a dataset i, $\sum_{i=1}^{n} H(x_i)$ denotes the sum of entropies of *n* datasets. The adjusted sensitivity measure is calculated as shown in Equation 25.

$$C_{adj}(x) = \alpha * \mathrm{C}\,(\mathrm{x}) \quad (25)$$

In Equation 25, α represents a score assigned by the domain expert. This is a weight that is given to the dataset indicating the sensitivity. A dataset with a high *C (x)* score but which is not deemed to be very sensitive by the domain expert will receive a low α value. This score ranges between 0 and 1. Characteristics of the dataset, which made the domain expert to assign the score, are used to train a neural network, which in turn will determine the score when a similar dataset is encountered in the future.

It is important to note that the proposed framework does not have to wait until the Usage Tracker has a lot of usage information. The framework will work even when there is no usage information, as the sensitivity of the data will be determined initially based on its quality and data connectivity. As usage information becomes available the sensitivity is dynamically updated by including usage data as well. When there is no usage information available $\mathrm{H}\left(c_{u,d_i}\right)$ in Equation 20 will be replaced by a constant 1 (one). So there is no overhead in the system because it does not have to wait for the usage data from the usage tracker. The same process is repeated whenever one of these three data sensitivity measures is not present.

CHAPTER V

TRACKING DATA LINEAGE

Problem Statement

Big data platform tools like Hadoop allows users to store huge volumes of different varieties of data. Data stored in Hadoop can go through several transformations after being consumed by different MapReduce jobs executed by several users. Tracking provenance information can be helpful in detecting data misuse. Data provenance deals with tracking the data lineage (i.e.) information about the data origin, ownership and the transformations it goes through. In the native Hadoop implementation there are no means or mechanisms to track the transformations a dataset goes through. In the current Hadoop implementation [12], the data lineage can be only tracked manually. Tracking data lineage manually is very time consuming and a tedious process.

Introduction

Provenance information consists of data about several entities, processes and users that are associated in producing a piece of information [251]. Provenance data is useful to assess the quality and reliability of data [251]. Provenance data is also useful to know about the origin of data and the transformations it went through. With the help of provenance information, one can identify which users transformed what datasets. In [252], the authors propose an Open Provenance Model (OPM), which specifies how provenance data can be gathered and exchanged in a system consisting of multiple layers. In [253] the authors stat that reasons for why one should use provenance data and they also suggest the advantages of using the provenance information.

Data provenance has been a well-studied field and there are some researches that use data provenance for scientific workflows [197, 198, 199, 200, 201, 214, 215, 235], data driven workflows [234], e-science [202], bioinformatics [203], storage systems [204], sharing structured data [236], cloud [196], data warehouses [205, 207], databases [208, 212, 213, 227, 241], access control models [209], web browsers [210], preventing forgeries [211], Resource Description Framework (RDF) Stores [242] and reproducing computational results [226]. While data provenance has been used to identify data quality the authors in [245] identify certain dimensions and measures to assess the quality of the provenance data itself. These authors argue that the quality of provenance data itself is essential as it can affect the outcome [245].

Provenance data is usually represented as graphs and are traditionally stored in relational databases. In [246] the authors use the Earth System Science Server (ES3), to capture provenance information automatically by tracking the interaction of the tasks with the execution environment. Provenance data stored in ES3 is assembled as provenance graphs later and ES3 provides a report on what actually happened during execution rather than what was requested during execution [246]. In [247] the authors suggest that compressing provenance graphs and using dictionary encoding can be used to store and query provenance data effectively. In [248], the authors suggest decreasing provenance data size by using factorization and inheritance. In [250] the authors identify challenges in automatic collection of provenance data on operating system level. These authors identify granularity, versioning and cycles in provenance data as issues in automatic collection of provenance data [250].

In [190] B. Galvic coins a term “Big Provenance” which refers to the provenance information obtained from big data. B. Galvic introduces two types of provenance for big data namely the transformation provenance and data provenance [190]. Provenance information can be used for debugging data, security and other purposes [190]. In [254], R. Agarwal suggests that provenance in big data can be used for "*validating, debugging, auditing, evaluating quality of data and determining reliability of data*".

J. Wang et al, in [188] identifies the following as challenges in big data when collecting provenance to be the four major challenges

- Size of provenance data collected
- Overhead of provenance collection
- Storing and aggregating provenance data
- Reproducing executions from provenance

Provenance Tracker proposed in the SDD framework discussed in Chapter III has some drawbacks such as the tracked provenance information can be modified by a malicious user so that the damages caused by that user will be unknown. This drawback is addressed in the provenance tracker based on the block chain used in Bitcoin [165]. In block chain once a block is created it becomes immutable and even if the data in the block is changed it will contradict with the hash value in the next block. This principle is applied in the block chain based model to track computation and data provenance in Hadoop [12]. This model poses no additional overhead to capture provenance data as it is fed the data from the Data Usage Tracker and the provenance information is stored in HDFS to increase availability and fault tolerance. In addition to this the users will be able to query the provenance information, which they are authorized to access using Apache Hive interface [243]. To impose additional security the block chain based model encrypts all the raw data stored in the block and only the administrator has the key to decrypt and view the data. In the proposed block chain based model it is assumed that the administrator is a trusted entity to keep the encryption key safe. Detailed information on how the block chains are constructed, maintained and monitored is explained in detail in rest of this chapter.

Summary of Existing Lineage Tracking Frameworks for Big Data

In [175] W. Zhou et al, propose a distributed network provenance tracking system called the Extensible Provenance Aware Networked Systems (ExSPAN). ExSPAN Framework is developed based on RapidNet a declarative networking engine [176, 177]. Network provenance data is stored as relational tables in ExSPAN [175]. The provenance data in ExSPAN is distributed based on two approaches namely value-based and reference-based [175]. In the value-based approach the data is piggybacked onto the general network communication whereas in the reference-based approach references to resources are created, which can be traced back to the source when querying for that information [175].

In [174] W. Zhou et al, propose a platform called NetTrails, which is used for querying and maintaining the provenance data in a distributed network. NetTrails provenance engine comprises of the features from RapidNet declarative networking engine [176, 177] and the ExSPAN network provenance engine [175]. The network provenance engine proposed in [174], stores provenance graphs as tables that are distributed over the network. Network Provenance in [174] is modeled as an acyclic graph whose vertices represents either a base tuple or result of an operation and edges represent the rules applied on the input data (tuples) to get the desired output. The other major advantage on the framework proposed in [174] is that it supports queries to access the distributed provenance data.

In [166], R. Agarwal et al propose a layer-based architecture to capture provenance data in big data. The authors use MongoDB, a NoSQL database [167] to track and store provenance information. The three layers in this architecture are the storage, application and access layers [166]. The authors state the major advantage of using this layer-based framework is that the changes in one layer will not affect the others. The provenance information is stored in BSON format (a binary format of JSON) in MongoDB. In [166], GridFS is used to store data in MongoDb as it has provisions to store both the data and the metadata. Large data objects are split into chunks and are stored in chunk collection whereas the metadata is stored in the file collections [166]. The authors in [166] limit the users to access each others provenance information. Although MongoDB can store considerable amount of data it cannot store as much information as Hadoop.

In [168], D. Ghoshal proposes a mechanism to extract provenance information from huge amounts of log files. Provenance information is captured by adding hooks to applications, which are a part of a workflow system [168, 169 and 170]. These hooks are known as "*program instrumentation*" and require all the source code to use these hooks. Capturing different types of logs and extracting provenance information from them is complicated as the logs can be structured, semi-structured and unstructured. In [168] the authors propose a rule-based framework to identify provenance information from log files. The framework consists of two phases namely the event capture and provenance derivation [168]. The authors use XML-based rule language for extracting provenance information [168]. Although the proposed framework identifies, links and remaps provenance information from log files the provenance information cannot be identified when there are no log files and the rules keep changing when the structure of log files keep changing.

In [171] D. Crawl et al, the provenance information of the MapReduce workflow is tracked using the Kepler framework [169, 172]. The framework proposed in [171] consists of a data model, which can capture provenance information from individual MapReduce jobs or entire workflows. Provenance information is stored in MySQL database and users can query this information [171]. The Task Tracker running on the data nodes in the Hadoop cluster adds data to the MySQL database whenever a MapReduce job executes. The Name Node (master nodes) in the Hadoop cluster runs the MySQL manager that monitors all the MySQL servers running on the data nodes. However the framework stores provenance information and tracks the same it requires an instance of MySQL server running on all the slave nodes (data nodes) in a Hadoop cluster [171].

In [173] R. Ikeda et al, propose a Reduce and Map Provenance (RAMP) framework, which is an extension of the Generalized Map And Reduce Workflows (GMRWs). The RAMP framework provides a wrapper function for the default Mapper and Reducer functions in the native Hadoop implementation [12]. In addition to the Mappers and Reducers the RAMP framework implements wrapper functions around the *RecordReader* and *RecordWriter* Classes in the native Hadoop implementation [12]. *RecordReader* class fetches record from the input split and sends it to the mapper for consumption, whereas the *RecordWriter* class writes the output from the Reducer to the HDFS. RAMP framework is used for both *forward tracing* and *backward tracking*. Forward tracing is used to identify which elements contributed the output and the backward tracing allows users to identify the elements, which were responsible for the creation of the output element. Although the RAMP framework implements both forward and backward tracing they have not been implemented efficiently [173].

In [185], Y.W. Cheah et al propose a lightweight framework for tracking provenance in big data called “Milieu”. Like the framework proposed in [166], “Milieu” also stores the provenance information in MongoDB [167]. “Milieu” facilitates in collection of semi-structured provenance information by collecting provenance data for storage and analysis separately. Provenance data collected in “Milieu” comprises of three levels. Level 1 data consists of basic information such as job submission details, user information and the results whereas level 2 data consists of more detailed information by including the resources required for computation and level 3 data is even more detailed as it comprises of detailed I/O information [185].Appropriate users access these different levels of data. Provenance data is stored based on the Job Id’s or the location Id’s [185]. Job Id’s are used to track MapReduce jobs whereas location Id’s are used to track command line access to the datasets stored in HDFS.

In [186], C. Olston and A.D. Sharma propose a provenance-tracking framework for Big Data called “Ibis”. The provenance information is stored in a relational database called SQLite. “Ibis” also defines an Ibis Query Language (IQL), which is similar to the Structured Query Language (SQL) [186]. IQL is used to query provenance data stored in SQLite database. Major advantage in "Ibis" is that it allows users to specify granularity sets called as "gsets". These granularity sets allow users to determine the granularity of the provenance data captured [186]. Granularity sets can be applied for both the data and the MapReduce jobs as well.

HadoopProv a provenance tracking system is built by modifying the Hadoop implementation by S. Akoush et al in [187]. This framework imposes less than 10% overhead on the running time of the MapReduce jobs. At the end of map task in a MapReduce job HadoopProv maps the intermediate keys to the input splits and when the combine task finishes its execution HadoopProv aggregates all the positions at the input split for records with same keys [187]. At the end of the reduce task HadoopProv stores the provenance data generated at the end of the map task and the locations of the records in the result of the MapReduce job [187]. HadoopProv stores "record level" provenance information and thereby it takes more space because the size of the provenance data depends on the number of key-value pairs in the entire dataset [187].

In [189] Y. Amsterdamer et al propose a provenance framework that aggregates both database and workflow style of provenance information. Database style provenance information has fine-grained dependencies whereas the workflow-style provenance information has coarse-grained dependencies [188, 189]. In [189] the provenance information is represented as compact graphs and is used by the workflow analysis queries [188, 189].

V. Korolev and A. Joshi in [191] propose a tool called "PROB" to track provenance and to aid in reproducing job execution for big data. "PROB" tool uses the following software to function

- Git
- Git2Prov [192]
- Git-Annex [193].

Git refers to a version control repository that tracks changes. Git-Annex [193] extends Git by allowing the software to track very large files without adding them to the repository. Git-Annex only stores the hash of the datasets in the repository and stores the datasets in a different path [191, 193]. Git2Prov [192] is used to represent the information in a Git repository as provenance data by converting it to the PROV W3C standard [191]. PROB tool is developed only to work with Apache Pig [194], an open-source data munging and manipulating tool that work on Hadoop [12].

In [195] the authors propose a Distributed Time-aware Provenance (DTaP) model, which is used to track provenance data about the state changes. The authors in [195] provides an implementation of DTaP model called as "DistTape", which is responsible for distributed storage of provenance data and this implementation allows users to query time-aware provenance data [195]. The authors also show how DistTape can be used in the context of MapReduce and discuss about the overhead and computational cost.

In [228], the authors collect data on how the tasks run on a high performance environment using provenance management system and store this data as structured data. The authors in [228] state that data on how the tasks behave when they are executed are helpful in identifying resource usage patterns and how the output data is obtained. In [228], the authors improve and use the Swift parallel scripting system [229, 230 and 231] to track data provenance. Although Swift parallel scripting system is capable of tracking provenance information there is a mismatch between its data structures and the data stored in relational model [228].

In [232], the authors demonstrate how the traditional provenance tracking systems will not scale in a completely distributed environment where the provenance information generated by individual nodes has to be verified for its authenticity and when some nodes in these distributed systems fail. In order to resolve these issues the authors in [232], propose a lineage authentication scheme called “Bonsai”. This scheme is a completely decentralized, introduces failure tolerance and reduces latency [232]. The provenance data collected by “Bonsai” is authentic, complete, only the operations and data required to reconstruct the data lineage are recorded [232]. Rather than transmitting metadata along with the resulting data in “Bonsai” only the data is transferred to the node which requests it and the metadata is stored on the node which performs the computations [232]. The cryptographic information required to access and verify the metadata is sent to the node requesting to access the data so the node requesting the data can verify the authenticity of the data if required [232]. This approach drastically reduces the number of times the metadata is transmitted and it can save network bandwidth when the metadata is large [232].

In [233], the authors propose a framework called “PReServ” to track lineage for services. “PReServ” records all the services, which were responsible for transforming the data, and it is helpful in reproducing experimental results. The provenance data tracked in “PReServ” is stored in a database [233].

In [237], W. Zhou et al propose a Time Aware Provenance (TAP) framework for distributed systems in order identify any discrepancies in a systems behavior, identify any intrusions, detect performance bottlenecks and diagnose issues related to configuration of protocols. This will aid the system administrators to identify any issues and rectify them [237]. TAP also allows users to query the provenance data securely.

In [238], the authors propose a general-purpose framework to track data provenance by using dynamic instrumentation. Users can avoid modification in their code by using dynamic instrumentation and the framework identifies the points in the user’s code where the dynamic instrumentation has to be added [238]. The authors build the framework proposed in [238] on DTrace [239] and test it on HTTP requests from browsers, transactions on databases and operations on file-systems.

In [240], T. Malik et al propose a framework to track provenance sketches on distributed applications and this decentralized framework is able to handle queries on provenance data from users effectively. The dependencies in workflows are tracked both within a host and across multiple hosts in [240]. The provenance data tracked in [240] is used to construct a provenance graph with resources (e.g. files) and operations (e.g. sum, difference, etc.) as vertices and direction of data flow as edges. Gathered provenance data will be stored in a RDBMS in [240].

The advantages of the proposed Provenance Tracker (PT) that runs on the Hadoop implementation [12] over these frameworks are as follows

- There is no additional overhead to capture provenance data as it uses the data collected by the Data Usage Tracker (DUT).
- Provenance data is fault-tolerant as it is stored on the HDFS itself.
- Provenance data is also secure since all the data except the hash values are encrypted and only the administrator can decrypt and view the raw provenance data.
- Provenance data can be queried using Apache Hive [243] and the queries should be in compliance with HiveQL [244] specifications.
- Provenance data can be checked periodically to identify violations in users accessing data.

Block Chain Approach to Track Data Lineage

The block chain based model to track data lineage creates a block whenever a new dataset is created or stored in HDFS [12].The block in the block chain, which is created whenever a new dataset is created or stored in HDFS, is called as the Genesis block. Whenever a dataset is accessed via MapReduce jobs (MR) or Command Line Interface (CLI) a new block is added to a block chain corresponding to the dataset. When multiple datasets are aggregated and accessed via MapReduce jobs then a block is created from the block chains corresponding to the datasets, which are aggregated. Every block in the block chain consists of a header and data associated with the block. In the proposed a block chain based approach contents of a block in a block chain are depicted in Figure 35.

Header Data
Version Number *Dataset name* *Data items being accessed* *Type of access (MR/CLI)* *User Information* *Hash value of the root of Merkle Tree* *Hash value of previous block in chain*

Figure 35Contents of a block in block chain

To enhance security and to prevent users tampering with the provenance information, data in every block such as the dataset name, data items being accessed, type of access and user information are encrypted using an asymmetric encryption algorithm like RSA [124] before being stored in HDFS.

Header data in a block consists of the hash value of root of Merkle tree, hash value of previous block in block chain and the little endian representation of the following after they have been converted to hexadecimal representation

- Version number
- Timestamp
- Size of data block in bytes

In the little endian representation the Most Significant Bits (MSB) are swapped with the Least Significant Bits (LSB). Merkle tree is constructed using the name of the dataset, data items accessed, type of access and user information as shown in Figure 36.

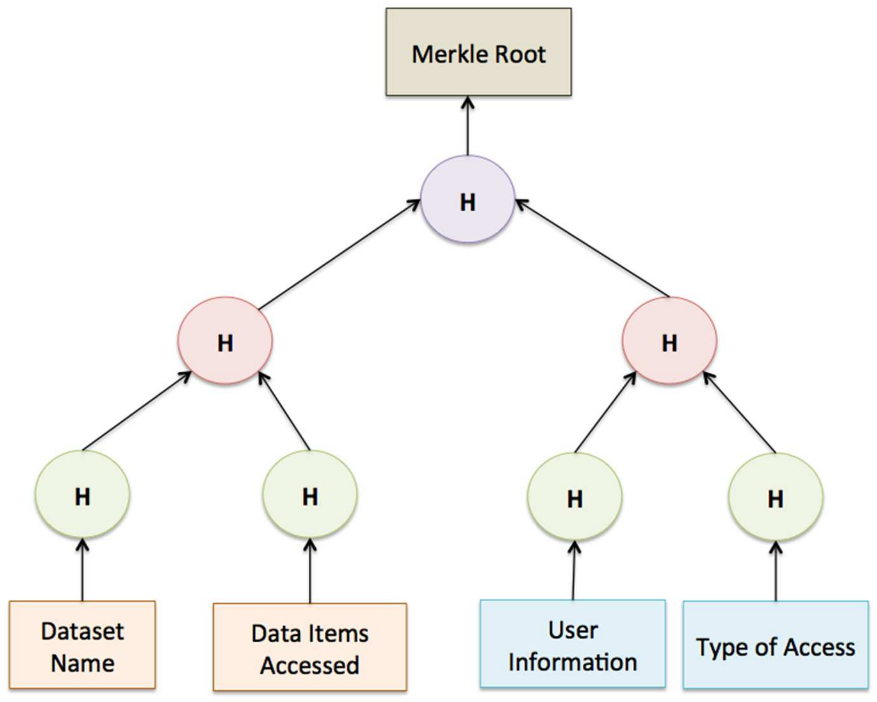


Figure 36 Merkle Tree Calculations

H in Figure 36 denotes a hashing algorithm such as SHA-256. To calculate the root of the Merkle tree, hash values for dataset name and data items accessed are calculated and then they are hashed with the hash value obtained by hashing user information and type of access. Instead of storing the entire Merkle tree the proposed model stores only the hash value of the root thereby saving space in HDFS.

Creating Genesis Block

Genesis block is created only when a new dataset is stored or being written to HDFS. Genesis block lacks information such as the user information, type of access, hash value of previous block and data items being accessed. Thus they are created using the algorithm shown in Figure 37.

```
Input: Dataset Name
Output: Genesis Block
Initialize the following constants
Version_Num ← 0
Nonce ← rand()
Timestamp ← now()
Header ← Hash (Version_Num + Timestamp + Nonce + Dataset Name)
Block_Data ← Encrypted (Dataset Name), Version_Num, Timestamp
Return Header,Block_Data
```

Figure 37Algorithm to Create Genesis Block

During the creation of the genesis block due to the lack of previous hash value a random number (nonce) is used. Header of the genesis block consists of the hash value of the concatenation of the version number, timestamp, nonce and name of the dataset. To create genesis block source code of HDFS shell commands [132], which are used to create a new dataset (touchz), transfer files to HDFS (put, copyFromLocal, moveFromLocal, getmerge) and modify files (appendToFile) is modified to create genesis block.

Adding a Block in a Block Chain with no Data Aggregation

```
Input: Dataset name, data items, user information, type of access

Output: None

Prev_hash ← Header of the last block in block chain

Root_Merkle_Tree ← Compute Merkle Tree using dataset name, data items, user information and
type of access and return root of tree

Version_Num ← Little Endian(Hex(Version_Num of last block + 1))

timestamp ← Little Endian(Hex(timestamp))

data_size ← Little Endian(Hex(data size in bytes))

Block_Header ← Hash (Version_Num + timestamp + data_size + Prev_hash + Root_Merkle_Tree)

Block_Data ← Encrypted (Dataset name, data items, user information, type of access),
Version_Num, timestamp, data_size, Root_Merkle_Tree

Add Block with Block_Header as Header and Block_Data as Data for new block
```

Figure 38 Algorithm to add a Block to the Block Chain

Whenever a dataset is being accessed by itself by either a MapReduce job or via Command Line Interface (CLI) Algorithm shown in Figure 38 is invoked and an example of the same is shown in Figure 39. When inserting a block to a block chain the first task is to identify which block chain to insert the data into and after identifying the block chain the algorithm fetches the header of the last block in the block chain. After this the Merkle tree is calculated by using the example shown in Figure 36. After calculating the Merkle tree the root of this tree is returned and then the header of the block to be inserted is computed by calculating the hash value of concatenation of version number, timestamp, size of data, previous header, and root of Merkle tree. Data of the block to be inserted is computed by encrypting the user information, dataset name, data items accessed and type of access of dataset. Once the block header and data is calculated the block is added to the block chain.

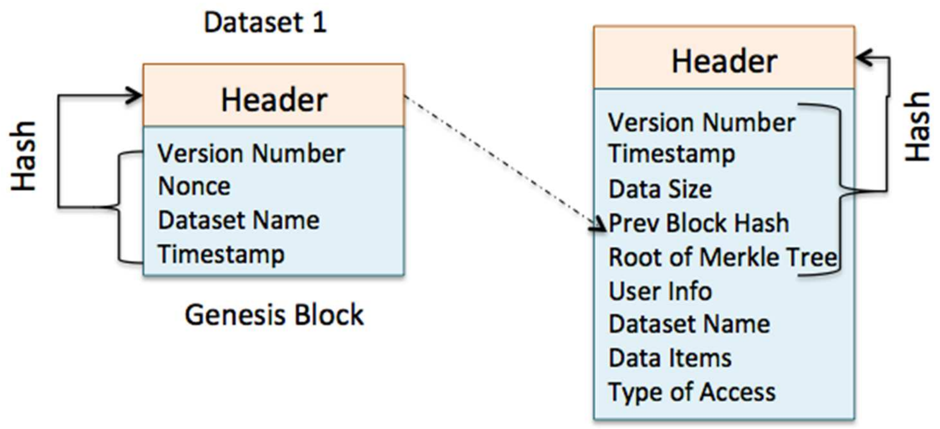


Figure 39 Depiction of adding a Block to a Block Chain

Adding a Block in a Block Chain with Data Aggregation

```
Input: Dataset names (1 …. n), data items, user information, type of access

Output: None

If aggregation is performed already then

        Prev_hash ← Fetch the header of last block in block chain corresponding to the agg.

Else

        Prev_hash ← Compute hash of headers of the genesis blocks in block chain for all datasets (1 … n)

End If

Root_Merkle_Tree ← Compute Merkle Tree using dataset names (1 … n), data items, user information and
type of access and return root of tree

Version_Num ← Little Endian(Hex(Version_Num of last block + 1))

timestamp ← Little Endian(Hex(timestamp))

data_size ← Little Endian(Hex(data size in bytes))

Block_Header ← Hash (Version_Num + timestamp + data_size + Prev_hash + Root_Merkle_Tree)

Block_Data ← Encrypted (Dataset name, data items, user information, type of access), Version_Num,
timestamp, data_size, Root_Merkle_Tree

Add Block with Block_Header as Header and Block_Data as Data for new block
```

Figure 40Algorithm to add a Block to the Block Chain when Datasets are aggregated

When the user submits a MapReduce job by aggregating *n* datasets the algorithm shown in Figure 40 is invoked. Whenever multiple datasets are aggregated one has to check if the aggregation has been made before or is the aggregation is being made for the first time. If the aggregation has been made for the first time then the previous hash value is the hash of all the genesis block headers of the datasets that are being aggregated. If the aggregation has been performed before then the previous hash value is the header of the last block in the block chain representing that aggregation. Once the previous hash value is estimated the root of Merkle tree is computed as shown in Figure 41.

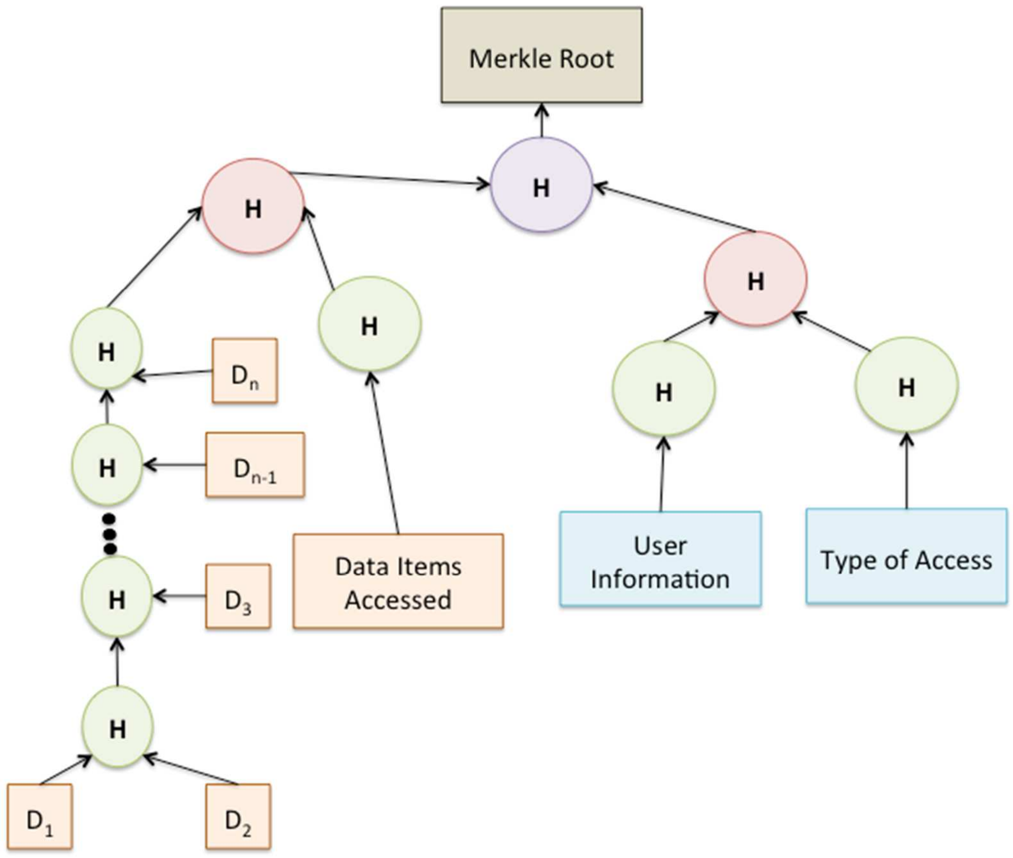


Figure 41 Calculating Merkle root when multiple datasets are aggregated

Computing Merkle tree when datasets are aggregated is close to how Merkle tree is computed when the datasets are accessed without aggregation as shown in Figure 36. The major difference is that during the aggregation of datasets the dataset names are hashed until there is one hash value representing the dataset name. Calculating this for every aggregation can be tedious. It can be optimized my making blocks in block chain to store this hash value of all dataset names as a part of data in the blocks so that when the blocks are aggregated this information can be fetched from the previous block in the block chain rather than calculating it again.

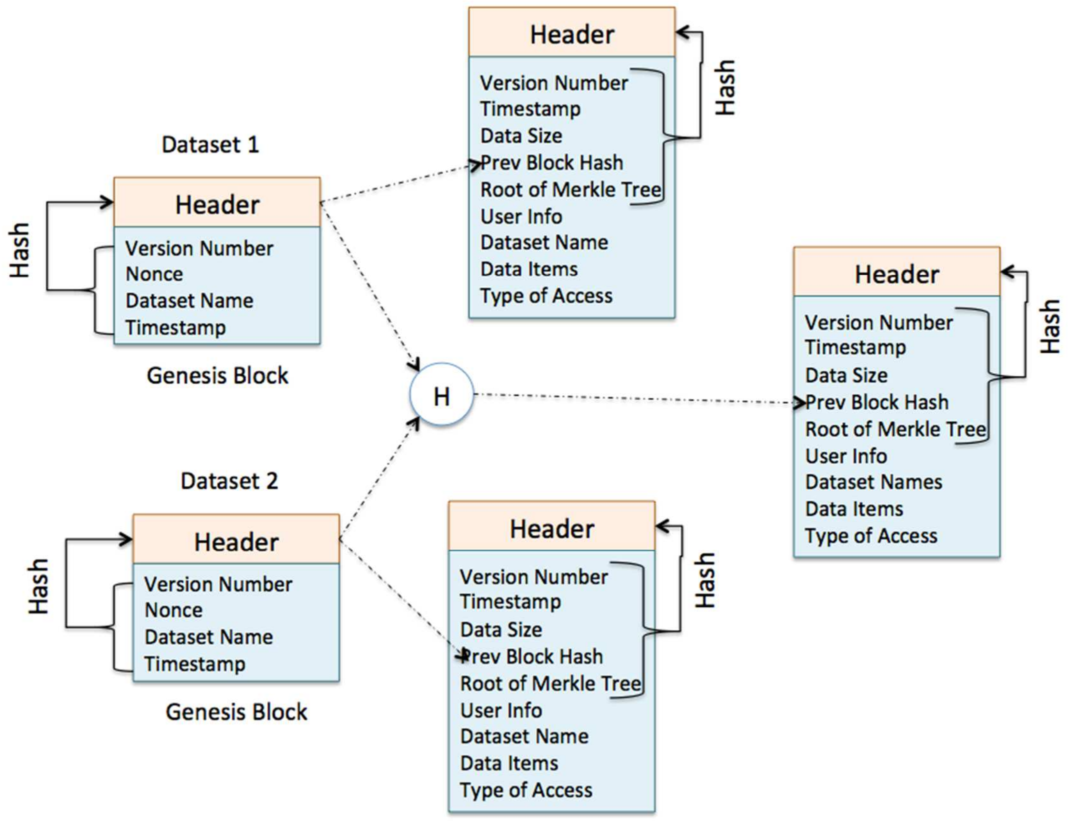


Figure 42 Scenario 1 - Adding a Block to a Block Chain when multiple datasets are aggregated

Figure 42 depicts an example of how a new block is inserted to the block chain when an aggregation of two datasets is performed. Whenever two datasets are aggregated and the aggregation has not been done before the previous hash value is computed by hashing all the block headers of the genesis blocks representing the datasets that are being aggregated. Then the similar procedure is followed to calculate the header of new block when the data is not aggregated.

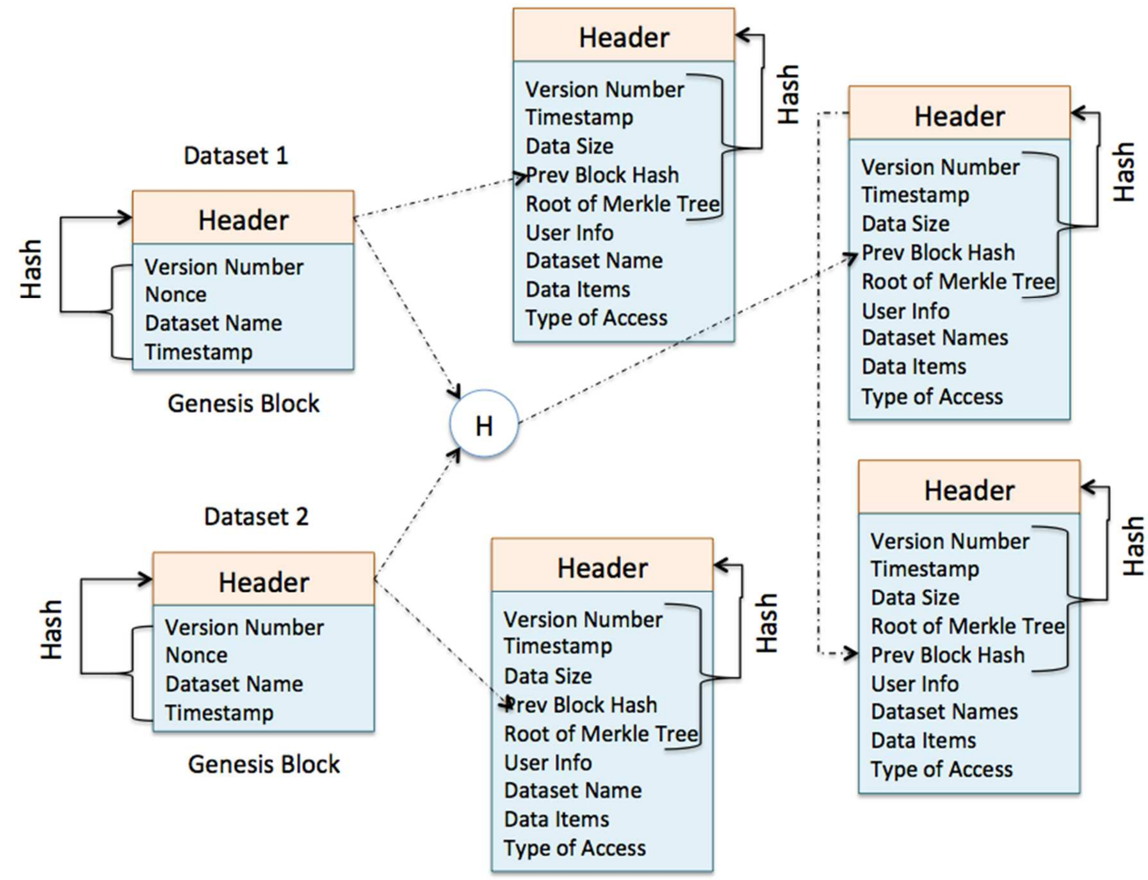


Figure 43 Scenario 2 - Adding a Block to a Block Chain when multiple datasets are aggregated

Figure 43 depicts an example of how a new block is inserted to the block chain when an aggregation of two datasets is performed and this aggregation has been performed before. Whenever two datasets are aggregated and the aggregation has been done before the previous hash value is the header of the last block in the block chain representing the data aggregation. Then the similar procedure is followed to calculate the header of new block when the data is not aggregated.

Handling Deletion of Datasets

In HDFS datasets can be added and removed. When the datasets are removed from HDFS the changes should be reflected in the block chain as well. Algorithm shown in Figure 44 is invoked whenever a dataset is deleted in HDFS. To handle deletion the algorithm shown in Figure 44 checks if the dataset is an actual dataset or if it is obtained as a result of aggregating multiple datasets.

```
Input: Dataset name
Output: None
If the dataset is not produced as a result of aggregation of multiple datasets then
    Remove the entire block chain representing the dataset
    If the dataset is involved in any aggregations then
        Recalculate header for all the blocks in the block chain representing the
        aggregation without including the dataset deleted
    End If
Else
    Remove the block in chain corresponding to the aggregated dataset being removed
    Recalculate header for all the blocks in the block chain (start from block after the deleted
    block)
    Prev_Hash of the block after the block being deleted is initialized to the header of the block
    before the block being deleted
End If
```

Figure 44 Algorithm to handle Deletion of data in HDFS

When the dataset is an actual dataset and not formed by aggregation the algorithm checks if the dataset is involved in any aggregation operations. If the dataset is not involved in any aggregation operations then all the blocks in the block chain starting from the genesis block that represents the dataset are removed. If the dataset is involved in any aggregation then previous hash value of the block representing the data aggregation is recomputed by hashing all the genesis block headers of remaining datasets. Once the previous hash value is set then the headers of rest of the block chain is computed one block at a time. If the dataset represents an aggregated dataset then the block representing the dataset being deleted is identified and is removed from the block chain. Let the block deleted be k. Previous hash value of the k+1th block is initialized to the header value of k-

1th block. To understand the functioning of the algorithm when the datasets are being deleted from HDFS three scenarios are explained as shown in Figures 45 through 49.

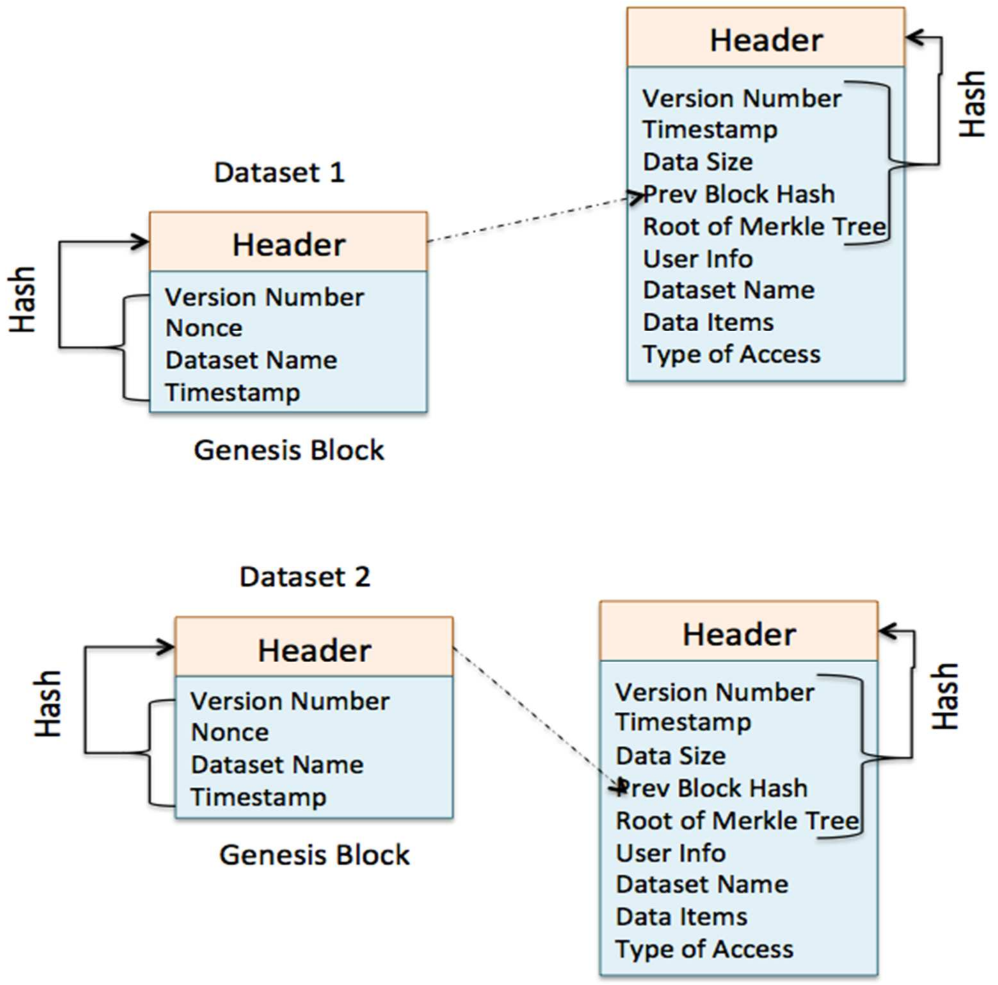


Figure 45 Scenario 1 – Deleting a non-aggregated dataset – Before Deletion

Scenario 1

Consider two datasets in HDFS namely dataset 1 and dataset 2 and let us also assume in this scenario these datasets are not aggregated. Figure 45 depicts the block chains of these two datasets 1 and 2, which are used one time each. Now let us assume that dataset 2 is removed from HDFS. In that case all the blocks in the block chain representing dataset 2 will be removed since it is not involved in any aggregation. This type of removal is easy, as it does not require any further changes in the block chain. Resulting block chain after the removal of dataset 2 is shown in Figure 46.

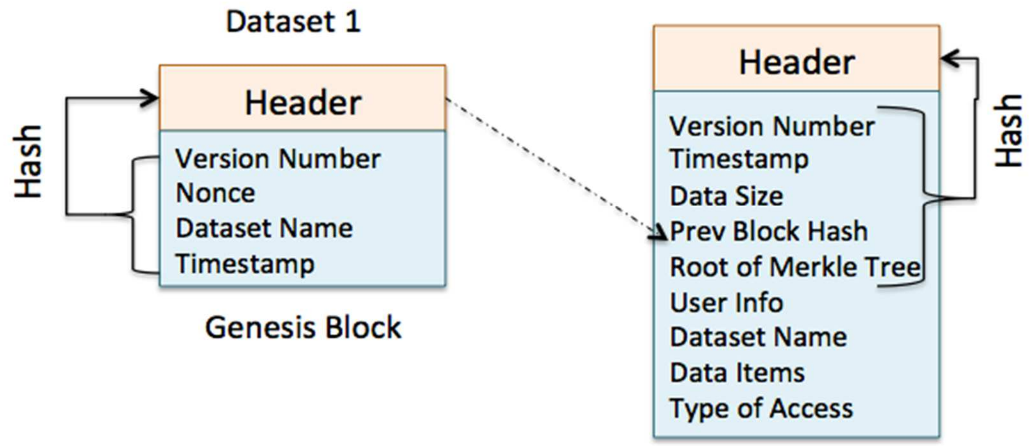


Figure 46 Scenario 1 – Deleting a non-aggregated dataset – After Deleting Dataset 2

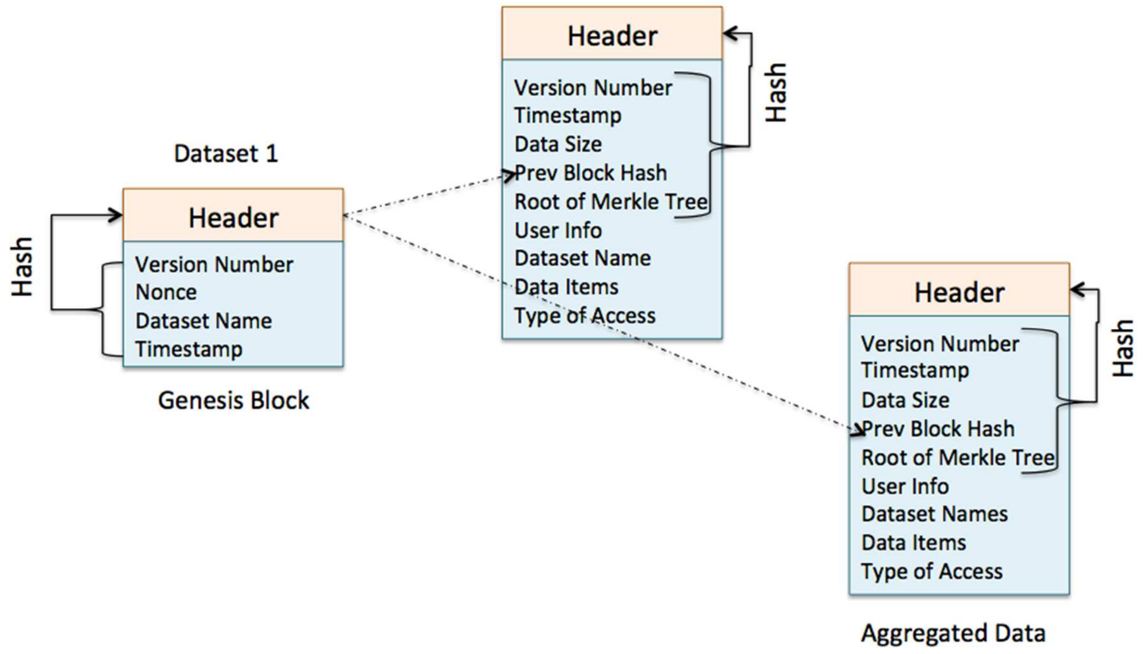


Figure 47 Scenario 2 – Deleting a non-aggregated dataset – After Deleting Dataset 2

Scenario 2

Consider two datasets in HDFS namely dataset 1 and dataset 2. Figure 47 depicts the block chains of these two datasets 1 and 2, which are used one time each and these datasets are aggregated once as well. Now let us assume that dataset 2 is removed from HDFS. In that case all the blocks in the block chain representing dataset 2 will be removed. In this scenario dataset 2 is involved in aggregation. Let k be the block created as a result of aggregation of these two datasets. After removal of dataset 2, previous hash value of the block k must be a hash of all the genesis blocks of the datasets that exist in HDFS. In this case there is only one other dataset and hence the previous hash value of the block k is set to the header of genesis block of dataset 1 as shown in Figure 48.

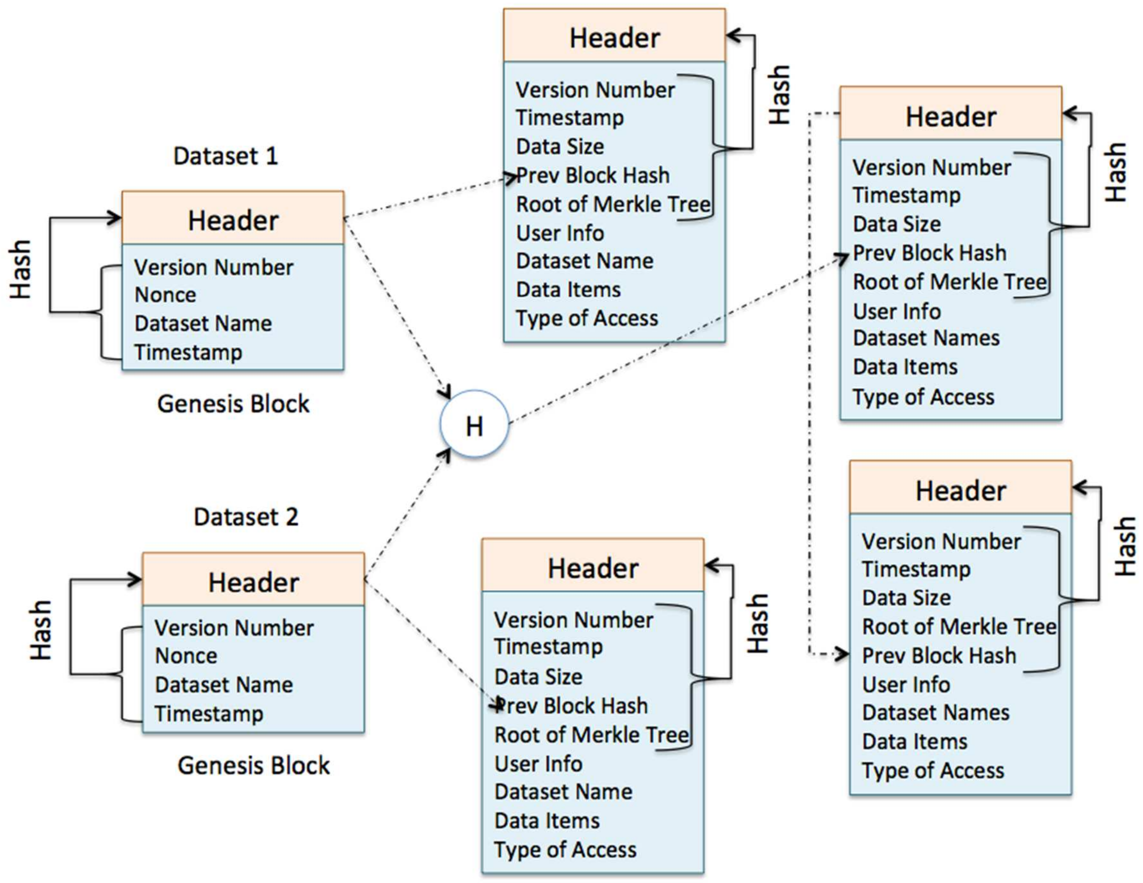


Figure 48 Scenario 3 – Deleting an aggregated dataset – Before deletion

Scenario 3

Consider two datasets in HDFS namely dataset 1 and dataset 2. Figure 48 depicts the block chains of these two datasets 1 and 2, which are used one time each and these datasets are aggregated twice. These aggregations are called aggregations 1 and 2. Now let us assume that aggregation 1 is removed from HDFS. Let k be the block in the block chain, which is after the block that is supposed to be deleted. Previous hash value of the block k should be set to the previous hash value of block k-1. The block chain after removing aggregation 1 is shown in Figure 49.

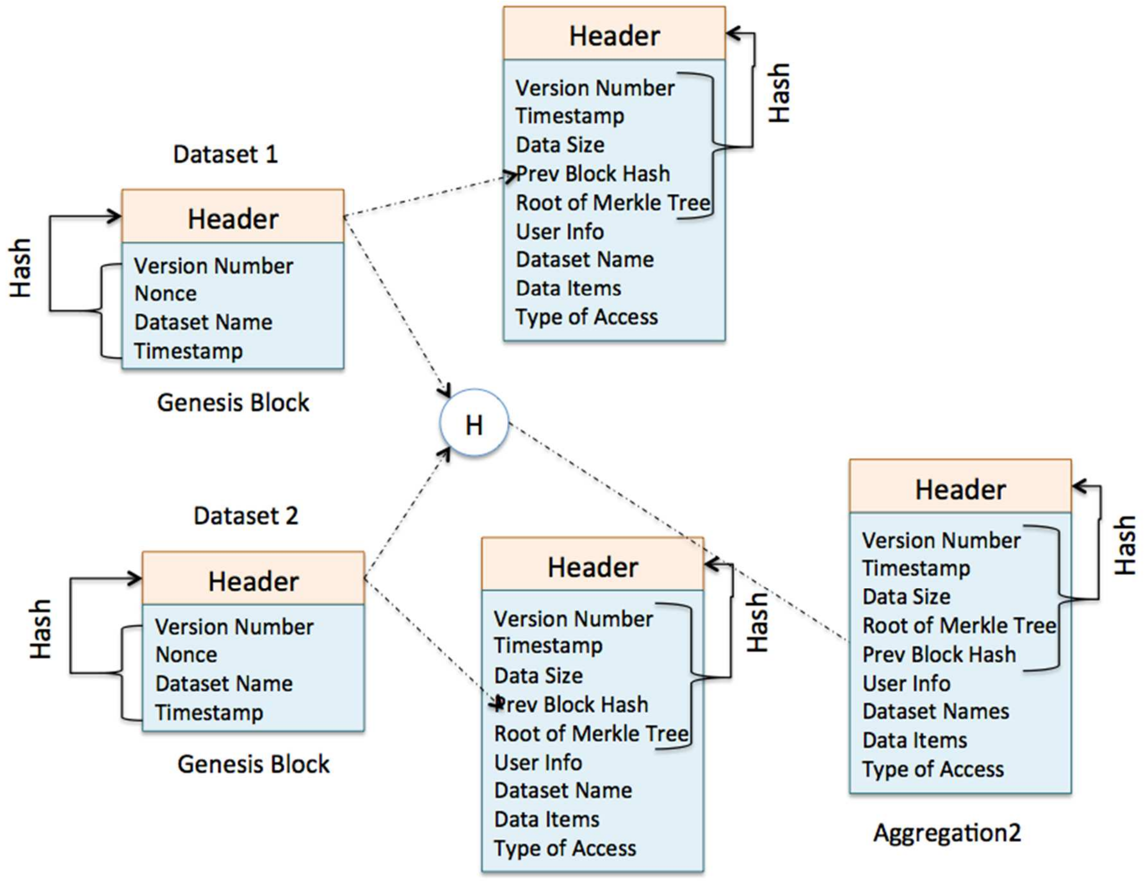


Figure 49 Scenario 3 – Deleting an aggregated dataset – After deleting Aggregation 1

Validating Provenance Data

The provenance data captured and stored in the HDFS must be guaranteed to be correct at all time. Assuming that a malicious user is capable of modifying contents of a block in the provenance block chain then there must exist a mechanism to identify any violations. These violations must be reported to the administrator and the actions will be taken by the administrator as they deem fit. Violations in the block chain can be calculated by comparing the comparing the block header with the hash value of the previous block header, version number, timestamp, data size and root of Merkle tree. Violations detection process can be scheduled to run at specific time intervals. The algorithm shown in Figure 50 will be invoked whenever the violation detection process is executed.

```
For every block chain in the provenance data
    For every block in the block chain
        If the block is not a genesis block then
            If Hash (Prev_hash + Root_Merkle_Tree + data_size + timestamp + Version_Num)
            == Block_Header then
                Move to next block
            Else
                Alert the administrator
            End If
        End If
    End Loop
End Loop
```

Figure 50Algorithm to validate Block Chain

The validation process is scheduled to run on specific intervals rather than running constantly in order to avoid this process taking much of the resources of the Hadoop cluster so that the users will be able to submit jobs and use the cluster productively.

CHAPTER VI

PROPOSED CSBAC FRAMEWORK

Problem Statement

Once the data items are identified in the datasets and their sensitivities are estimated, the next step is to provide a fine-grained access control based on users privileges. The challenge in providing such a fine-grained access control is making dynamic access control decision to decide whether to allow a user to access a dataset or a part of the dataset. If these decisions were user based, then the ACL (Access Control List) tends to get larger. In a very large organization maintaining a very large ACL becomes cumbersome. Since the ACL's are larger the decisions cannot be made with less overhead. In addition to this one must also consider changes in information sensitivity when multiple datasets are aggregated. Example scenarios provided in [32] prove that some information may be not that sensitive by themselves but when combined with multiple datasets it may become sensitive. A good access control mechanism should address these changes in data sensitivity when datasets are aggregated.

Introduction

Proposed CSBAC framework will be an extension of the frameworks proposed in Chapters III and IV. In CSBAC, access control decisions are made based on the users roles. There can be a large number of users in an organization but these users can be separated into different handful roles.

Note that the users can have multiple roles in such case the proposed CSBAC framework will consider only the role of the user, which can access the sensitive data items. In CSBAC framework sensitivity of data items is re-estimated whenever multiple datasets are combined and used together. For correct functioning the CSBAC framework will need the organization specific Access Policy Document (APD), which specifies the role of user and the corresponding sensitivity score of the data items that the role can access. The Administrator can amend APD if needed. The Sensitivity score of data items are expressed as a numerical value.

Like Vigiles [24], CSBAC filters out sensitive information in the results of the MapReduce job in Hadoop. It also prevents the MapReduce program from using unauthorized data items. A user depending on his or her access rights may not be authorized to access the sensitive data.

The roles and responsibilities of an administrator are as follows: The administrator provides the Access Policy Document (APD) to the framework. This document is important and mandatory for functioning of the proposed framework.

Related Work

In [2], A. Cavoukian et al., describe how privacy issues are big concerns that needs in big-data and they also investigate the role of Attribute-based access control (ABAC) technology in protecting sensitive information from inadvertent/deliberate misuse or abuse of data. A. Cavoukian et al., argue that in ABAC instead of just comparing the role of the user many other attributes can be used to grant access to a particular resource. Thus providing an effective restriction against misuse or abuse of resources. They specify how ABAC technology can benefit several sectors like healthcare, insurance, airlines and telecommunication. Their work provided necessary motivation and direction to use attribute-based approach for the proposed CSBAC. In the proposed CSBAC framework user role and attributes are considered to enforce access control decisions.

Access control for sensitive data in HDFS is proposed by Y.B. Reddy in [22]. This work highlights the fact that the user should be granted permission to access sensitive information only for a limited period of time. In [22] Y.B Reddy proposes an access control model and compares it with the security schemes in federated systems. Access control model proposed in [22], in solely dependent on the guidelines created by the data owner regarding data sharing.

"Airavat" proposed in [23], is a novel MapReduce based system that combines Mandatory Access Control and Differential Privacy. Like [22], this system adheres to the security policy specified by the data providers. In addition to this "Airavat" will not be able to guarantee privacy for MapReduce jobs with malicious mappers generating keys.

H. Ulusoy et al., proposed a first system to enforce fine grained access control in Hadoop in [24]. The system proposed in [24] is known as "Vigiles" and it doesn't impose any modifications to the underlying MapReduce system's source code. "Vigiles" acts as a middleware layer between the users and the Hadoop Cluster. Reference Monitors in cloud will filter the output before sending to the users to prevent unauthorized access of data. Reference Monitors will have to adhere to the policies specified for a dataset. These policies will have to be enforced by the predicates specified by the administrator. A user can access data only if corresponding predicate is satisfied.

"Efficient access control mechanism with dynamic policy updating for big data in the cloud" is proposed in [25]. K. Yang et al., propose an efficient access control scheme that will allow the data owners to change the data access policies without the need for re-encrypting and re-transmitting it. This prevents heavy communication and computation burden.

Apache Accumulo [42] is a distributed and parallel processing database that supports structured to unstructured data. Apache Accumulo [42] also offers fine-grained access control and user authentication. Administrator can restrict access up to cell level on Apache Accumulo. Apache

Accumulo [42] is a part of big data ecosystem and works on top of Hadoop. The administrator usually places access control restrictions in Apache Accumulo [42].

V. C. Hu et al., propose an access control scheme for big data processing in [26]. They propose an authorization component for big data to avoid misconfiguration of access control policies. The proposed Authorization component protects data from insider attacks. Data provider should provide a security class agreement for functioning of this Authorization component.

H. Chen et al., propose a scalable access control for big data based on multi-labels in [27]. They use multi-labels to protect PHR's (Patient Health Record). Parts of PHR's are sensitive and there are strict restrictions put forth by laws such as HIPPA in sharing PHR's. In [27], the data owner will have to decide on the labels initially and the administrator can modify it later on basis of necessity.

In [28], Q. Yuan et al., propose a fine grained access control mechanism for big data in cloud based on Ciphertext Policy Attribute Based Encryption (CP-ABE). The authors claim that this scheme is able to provide fine-grained access control and implement changes made by data owner to the data sharing policies effectively. This approach eliminated the need for trusted third parties and the responsibility of authorizing consumers to access data falls solely on the data owners.

In [30], R. Nasim and S. Buchegger proposes an eXtensible Access Control Markup Language (XACML), based access control model for data from Online Social Networks (OSN). Data from OSN is one type of big data, which has information about the users and their behavior. The model proposed in [30], makes use of XACML along with Security Assertion Markup Language (SAML) along with secret key authentication. A requester's request will have to be authorized by the owner every time such a request arises. By doing so privacy of users and their data from OSN's is preserved.

C. Rong et al., [43] combined the idea of virtual pieces from BitTorrent [44] and secure sharing over cloud [45] to implement an access control scheme for data stored in Hadoop. The scheme proposed in [43], encrypts block level metadata generated by Hadoop and distributes it as torrent file. Any consumer wanting to access the data should download the torrent file and decrypt the metadata using shared key and access these data blocks.

All these research works implement some form of access control mechanism to protect sensitive information. Nonetheless, in these methodologies except [43], the administrator or data owner should explicitly provide specifications on how to share the data. In other words, the administrator and/or the data provider determine the data sensitivity. But the proposed CSBAC framework uses the data itself to determine its sensitivity.

Content Based Access Control (CBAC), was proposed by W. Zeng et al., in [29]. The CBAC framework uses the data content to make access control decisions. However, it requires a base set to compare with for sensitivity. The base set consists of a number of datasets and a number of users are given access to these datasets. When a new dataset (D2) similar to dataset (D1) in the base set is stored in HDFS, all the users who are able to access D1 will be able to access D2. Data similarity is determined by a top-k similarity measure. CBAC exploits semantic relatedness by exploiting content similarity. However semantic relationships cannot be completely identified by just using content similarity [46]. Some semantic relationships can be identified using data usage as well. The proposed CSBAC framework addresses this and makes use of both data content and usage to identify similar content.

In [31], the authors who proposed CBAC Framework use it on top of existing Role-Base Access Control (RBAC) scheme Multi-level Security (MLS) System. They explain how CBAC can work in tandem on top of these existing security schemes to provide access control decisions based on data content. Similarity between structured and semi-structured datasets is computed by equation 26 in [19, 31].

$$Sim_d(d_i, d_j) = \sum_{x=1}^{N} w_x * Sim_{a_x}(d_{i,x}, d_{j,x}) (26)\ [31]$$

In Equation 26, Sim_{a_x}represents a normalized similarity function, which is defined on a specific domain a_x, w_x is the weight of the attribute weight. For unstructured data the authors in [19, 31] compute the similarity measure using equation 27 [31].

$$Sim_d(d_i, d_j) = \frac{d_i \, . \, d_j}{|d_i|\ X\ |d_j|} (27) \quad [31]$$

In equation 27, d_i represents a record in a dataset, which can be expressed as an array of TF-IDF weight of terms as shown in Equation 28 [31].

$$d_i = [w_{1,i}, w_{2,i}, w_{3,i}, \ldots \ldots, w_{N,i}] (28)\ [31]$$

In Equation 28, $w_{t,i}$ represent the TF-IDF weight of a term t in record i. $w_{t,i}$ can be computed using Equation 29.

$$w_{t,i} = tf_{t,i} * idf_1 = tf_{t,i} * log \frac{N}{df_t} (29)[31]$$

In Equation 29, tf_i represents the term frequency (number of times a term occurs in a record) of a term t in record i, df_t is the number of records in the dataset that contain the term t [31]. The work in [29] and [31] has two major drawbacks. Firstly, it depends on a base dataset. Steps to identify a good base dataset are not discussed. The scheme fails if there is no base dataset. Defining and implementing base datasets to cover all possible types and combinations of data is simply impossible. Secondly these works do not factor in the change of data sensitivity when multiple datasets are combined. CBAC Framework fails to address problems that are similar to scenarios 1 and 2 discussed in Section I. To be universally applicable, access controls must be determined based on the data itself, rather than by comparing to a base dataset (if it exists). The proposed framework is the first one to use the data itself to estimate its sensitivity and make access control decisions based on it. This framework is therefore applicable to any dataset, aggregated or not and there is no need to define a base dataset. The goal of the proposed framework is to keep the sensitive information out of reach of unauthorized users.

Content Sensitivity Based Access Control (CSBAC) Framework

The CSBAC Framework is an extension of the SDD (Sensitive Data items Detection) framework proposed in [21]. In [21], the SDD framework identifies whether individual data items in a dataset are sensitive or not. The sensitive data items are reported to the administrator constantly. The onus is on the administrator to ensure the protection of sensitive data. In addition to this, SDD doesn't account for variations in data sensitivity when multiple datasets are combined. The CSBAC framework uses some of the components proposed in the SDD framework along with the information gain model described in Section IV to identify sensitive items. The sensitivity of data items is re-estimated whenever multiple datasets are combined and used together. For correct functioning the CSBAC framework will need the organization specific Access Policy Document (APD), which specifies the role of a user and the corresponding sensitivity score of the data items that the user can access. The Administrator can amend the APD if needed. The Sensitivity score of data items are expressed as a numerical value. Like Vigiles [14], CSBAC filters out sensitive information in the results of the MapReduce stage in Hadoop. A user depending on his or her access rights may not be authorized to access the sensitive data. The roles and responsibilities of an administrator are as follows: The administrator provides the Access Policy Document (APD) to the framework and the default values for various data types. This document is important and mandatory for functioning of the proposed framework. Architecture of the proposed framework is shown in Figure 51.

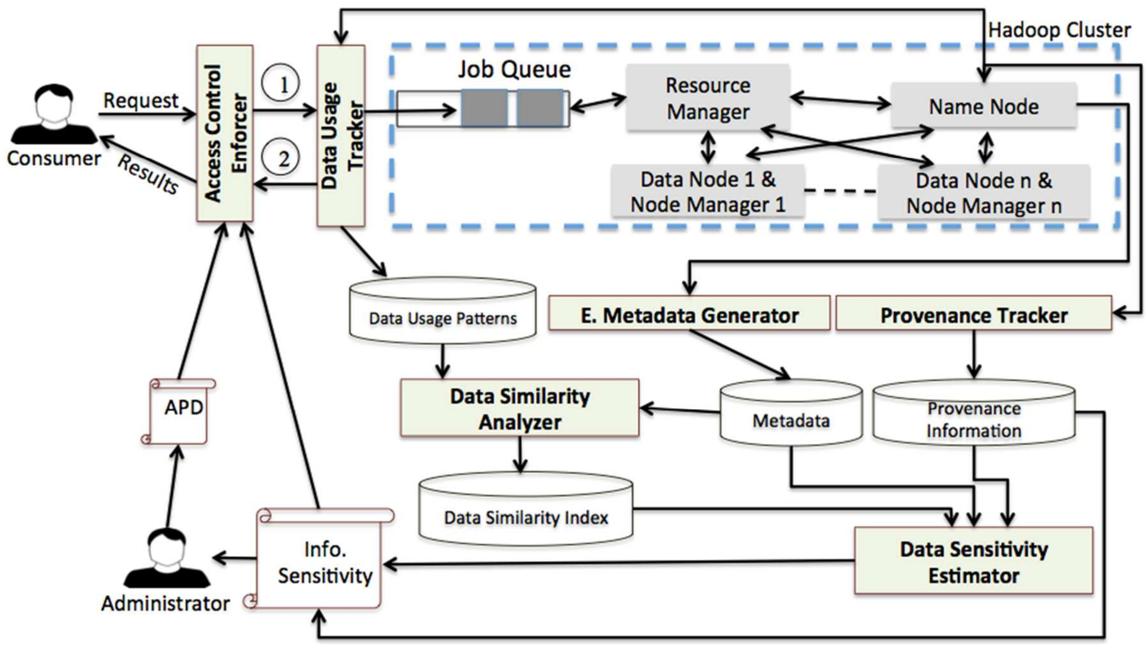


Figure 51 Architecture of CSBAC Framework

Access Control Enforcer (ACE)

The Access control enforcer (ACE) is the central component of the proposed CSBAC Framework. ACE receives the user requests and returns the results to the users. The results contain only the data, which the user is authorized to view or access. ACE can handle user requests, which can be either a MapReduce, job being submitted for execution or a shell command for accessing a dataset.

Access Control Rule denotes whether a user is entitled to access a piece of information or not. A simple Access Control Rule [19, 31] is denoted as shown in Equation 30.

$$ACR = \{user, data, action, decision\} \qquad (30)$$

In Equation 30, action denotes the operation that the user wants to perform on the data and decision, represents a Boolean value. Decision is true if the user is allowed to perform the action on the data and vice-versa. Equation 30, presented in [19 and 31] is used to represent the access control rules in CSBAC. Since native Hadoop Implementation [10] allows only users to create, delete, read and append datasets the parameter ‘action’ in Equation 30 is limited to read, append and delete. Only owner of a dataset or administrator will be allowed to delete the dataset they created in CSBAC.

A user *u* can access all the data items with sensitivity score greater than or equal to the sensitivity score S_r corresponding to the role of the user *u* as described in the APD. Access control decisions (ACD) are made dynamically based on data sensitivity as shown in Equation 31.

$$ACD(U_r, S_i) = \begin{cases} true, & S_i \geq S_r \\ false, & S_i < S_r \end{cases} \quad (31)$$

In Equation 31, U_r represents a user U of role *r*; S_i represents sensitivity of a data item *i*; S_r represents the sensitivity score of the data items, which the user is authorized to access. A user is allowed to access a dataset only if the ACD for the action is true. Enforcement of the ACD’s by the proposed ACE occurs at two levels namely the MapReduce level and the command line level to access to the HDFS.

Enforcement at Command Line Level

The algorithm shown in Figure 52 shows the sequence of events whenever a dataset, which is stored in HDFS, is read via the command line (FS Shell).When a user tries to read a dataset using FSShell, an InputStream object is created for the input file paths. InputStream object cannot be created when the file path does not exist or when the user does not have permission to access the file. If verify checksum flag is set a checksum is computed for the file path and compared against the checksum stored in HDFS. If the checksums match then the file is not altered and the contents are printed on a standard output device. To prevent unauthorized access of data items or datasets via the command line the proposed ACE implements algorithms are shown in Figures 53, 54 and

55. Algorithm 3 shown in Figure 55 is a modification of the algorithm in Figure 52. Algorithm 55 allows users to view subset of a dataset whereas in a traditional Hadoop implementation the user gets to view either entire dataset or nothing at all [10].

```
Input: Input File Path, verifyChecksum
Output: Contents of a dataset
try:
  InputStream ←InputStream (InputFilePath)
  If verifyChecksum == true then
    Check if the checksum of the file in input path is valid
    If checksum is invalid then
      Throw an exception & return
    Else
      Print(InputStream, std.out)
    End If
  Else
    Print(InputStream, std.out)
  End If
catch:
  Print(Exception, std.err)
```

Figure 52 Algorithm for reading a dataset via command line in Hadoop [10]

Algorithm 1 shown in Figure 53 accepts minimum sensitivity of a user role given in APD and the sensitivity of a dataset computed by the DSE and returns either true or false. If the minimum sensitivity S_u of the user role is less than the sensitivity of the dataset S_d then the algorithm returns true, which implies that the user can access the required dataset. In other words a user's role with sensitivity S_u is allowed to access all datasets whose sensitivity is greater than or equal to S_u. If the condition mentioned above is not satisfied then the user would not be able to access the dataset.

```
Input: User Sensitivity ($S_r$), Dataset sensitivity ($S_d$)
Output: True/False
If $S_u < S_d$ then
  return true
Else
  return false
End If
```

Figure 53 Algorithm 1

```
Input: User Sensitivity ($S_r$), FSDataInputStream (f), Dataset (d)
Output: Sanitized Records
sanitizedRecords ← ""
for record in f:
  sanitizedRecord ← ""
  for attribute in record:
    $S_{attr}$ ← getDataItemSensitivity(attribute, d)
    if $S_r < S_{attr}$ then
      sanitizedRecord += attribute
    else
      sanitizedRecord += defaultValue(attribute_data_type)
    end if
  end for
  sanitizedRecords += sanitizedRecord
end for
return sanitizedRecords
```

Figure 54 Algorithm 2

Algorithm 2 in Figure 54 is responsible for displaying only the portion of dataset, which a user is authorized to view. This algorithm accepts the *FSDataInputStream*, minimum sensitivity of a user role given in the APD and the dataset name and returns set of sanitized records. Sanitized records contains only the data attributes (data items/columns) whose sensitivity S_{attr} is less than the sensitivity of user role S_u and the data attributes whose sensitivity S_{attr} is greater than or equal to the sensitivity of the user role S_u are replaced with a default value for that data type specified by an administrator. Data is read from the original dataset using the *FSDataInputStream* object and before being displayed to the user, the attributes whose sensitivity value estimated by the DSE is less than the minimum sensitivity of the user role are replaced with the default values as specified by the administrator. Data items, which do not satisfy the aforementioned condition are returned, "as-is".

Algorithm 3 in Figure 55 is called whenever a user wants to read a dataset via the command line. This algorithm determines the minimum sensitivity for the user role using the APD and it also determines the sensitivity of the dataset based on the results of the DSE. Algorithm 3 invokes algorithms 1 and 2 to identify if the user is allowed to access the dataset and to retrieve sanitized data if the user is allowed to access the dataset respectively. The Data usage tracker is notified with all the relevant information about the user and dataset after the data is displayed to the user.

```
Input: Input File Path, verifyChecksum
Output: Contents of a dataset
try:
  InputStream ← InputStream (InputFilePath)
$s_r$ ← getUserRoleSensitivity(determineUser(), APD)
  $s_d$ ← getDatasetSensitivity(InputFilePath)
  If verifyChecksum == true then
    Check if the checksum of the file in input path is valid
    If checksum is invalid then
      Throw an exception & return
    Else
If Algorithm_1($s_r$, $s_d$) then
        Print(Algorithm_2($s_r$, InputStream, InputFilePath), std.out)
      End If
    End If
  Else
    If Algorithm_1($s_r$, $s_d$) then
      Print(Algorithm_2($s_r$, InputStream, InputFilePath), std.out)
    End If
  End If
catch:
  Print(Exception, std.err)
```

Figure 55 Algorithm 3 - Modified Algorithm for reading a dataset via command line in ACE

Enforcement at MapReduce Level

When a MapReduce job is submitted to the Resource Manager (RM), the RM checks if the user has sufficient permissions to access the input dataset(s). After determining that the user is authorized to access the dataset, the input dataset is split into a number of splits based on the split size. Each of these input splits is then transformed into key/value pairs before being passed to the map function by the RecordReader class as shown in Figure 56. Initialize() is called to initialize a RecordReader object once per input split. The method nextKeyValue() is called to read each record in the input split to create a key-value pair. This method returns false when there are no records left in the input split to process. getCurrentKey() and getCurrentValue() are called to get the key/value pairs for the record being processed and these key/value pairs are sent to the mapper using sendToMap() method. This process is depicted in Figure 56.

```
Input: Data Split
Output: <Key, Value> pairs for mapper
initialize()
while nextKeyValue() == true
  key ← getCurrentKey()
  value ← getCurrentValue()
  sendToMap(key,value)
end while
```

Figure 56 Input preprocessing in Hadoop [10, 14]

In the proposed ACE checking whether the user has sufficient permissions to access a dataset before running a MapReduce job is verified using algorithm 1 in Figure 53. Once the ACE establishes the user has sufficient permissions to access the dataset, the pre-processing of data is implemented based on the algorithm shown in Figure 57. During the preprocessing of input data, the proposed ACE determines the minimum sensitivity for the user role based on the APD and the data set from the input file path (obtained from the JobConf). For every record processed by the RecordReader class before sending the key, value pair to the mapper, their sensitivity (S_{Key} and S_{Value}) is checked against the sensitivity of the user role (S_r). If sensitivity of the key S_{Key} is less than the sensitivity of user role S_r then the key is sent to the mapper "as-is". If the above condition is not satisfied then the key is substituted with the default value specified by the administrator for the corresponding data type and sent to the mapper by sendToMap() method. The same process is repeated for the value before sending it to the mapper.

```
Input: Data Split, configuration
Output: <Key, Value> pairs for mapper
initialize()
$s_r$ ← getUserRoleSensitivity(determineUser(), APD)
d ← getDatasetInfo(InputFilePath)
while nextKeyValue() == true
  key ← getCurrentKey()
  value ← getCurrentValue()
  $s_{key}$ ← getDataItemSensitivity(key, d)
  $s_{value}$ ← getDataItemSensitivity(value, d)
  if $s_r >= s_{key}$ then
    key ← defaultValue(key_data_type)
  end if
  if $s_r >= s_{value}$ then
    value ← defaultValue(value_data_type)
  end if
  sendToMap(key,value)
end while
```

Figure 57 Input preprocessing in ACE [10, 14]

In the proposed ACE checking whether the user has sufficient permissions to access a dataset before running a MapReduce job is verified using algorithm 1 in Figure 53. Once the ACE establishes the user has sufficient permissions to access the dataset, the pre-processing of data is implemented based on the algorithm shown in Figure 57. During the preprocessing of input data, the proposed ACE determines the minimum sensitivity for the user role based on the APD and the data set from the input file path (obtained from the JobConf). For every record processed by the *RecordReader* class before sending the key, value pair to the mapper, their sensitivity (S_{Key} and S_{Value}) is checked against the sensitivity of the user role (S_r). If sensitivity of the key S_{Key} is less than the sensitivity of user role S_r then the key is sent to the mapper "as-is". If the above condition is not satisfied then the key is substituted with the default value specified by the administrator for the corresponding data type and sent to the mapper by sendToMap() method. The same process is repeated for the value before sending it to the mapper.

```
Input: Datasets $D_1$ ... $D_n$
D ← NULL
for each dataset $D_i$ from the input
  D ←{D U Data Items in $D_i$}
end for
If D has occurred before
    Use the sensitivity estimated before for each data item
Else
    Estimate the effect of the set 'D' on the system
    Store the sensitivity estimated by DSE for future use
```

Figure 58 Re-estimation of Sensitivity

In case of Scenarios like 1 and 2 presented in Section I, sensitivities of data items will vary when datasets are joined. Multiple datasets will be joined when they are passed as input paths to a MapReduce job. In this case the ACE implements the algorithm shown in Figure 58. ACE can handle multiple requests from several users and makes use of Fair Scheduler [37].

Experimental Results

Data Used

To evaluate the correctness and overhead of the proposed framework three types of datasets (relational/structured, semi-structured and unstructured) are used in the experiments. Synthetic patient data generator called "*Synthea*" proposed in [180] is used to generate structured data. *Synthea* generates high quality realistic patient data modeled on the top 10 leading causes of years of lost life from the United States Institute for Health Metrics and Evaluation (IHME) [181]. Dataset generated by *Synthea* is relational (structured). Medical history is associated with each patient based on the relational data and this dataset is unstructured. Data is collected from Twitter using the real-time streaming API [182]. Twitter data obtained is in JSON format and it constitutes semi-structured data. All these three datasets are divided into 20GB, 40GB, 60GB, 80GB, 100GB, 120GB, 140GB, 160GB, 180GB, 200GB and 220GB.

MapReduce Jobs Used

The datasets are accessed via the command line and also by three MapReduce jobs. The first MapReduce job filters all the data corresponding to a patient (structured and unstructured datasets) or a twitter handle (semi-structured dataset). The second MapReduce job counts the number of records for each patient (structured and unstructured datasets) or a twitter handle (semi-structured dataset). The third MapReduce job calculates the average elapsed time between hospital visits for every patient (structured and unstructured datasets) or the average elapsed time between tweets for every user (semi-structured dataset). Mappers and Reducers for these jobs were implemented in python and executed in Hadoop using the Streaming API [183]. Mappers and reducers were written in python version 2.7.

Results

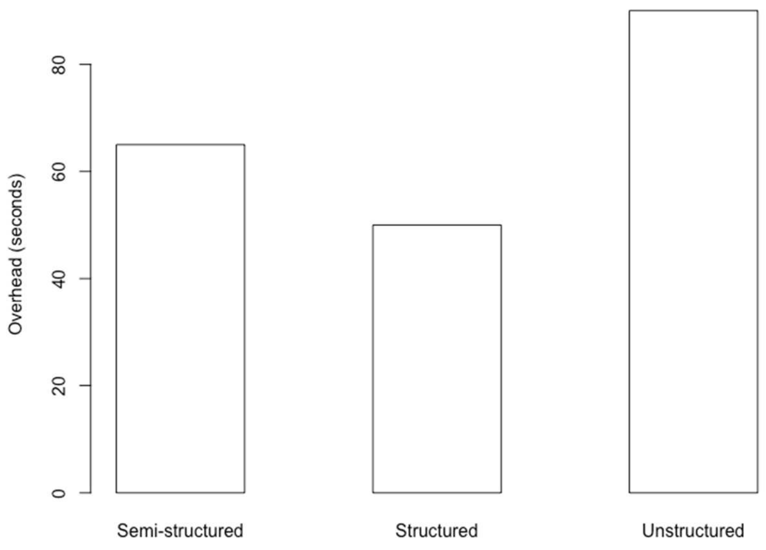


Figure 59 Overhead posed by the CSBAC Framework when accessing Data via CLI

The overhead posed by the proposed CSBAC Framework on accessing data via the command line interface (CLI), is shown in Figure 59. The overhead shown in Figure 59 is the time taken by the proposed framework to display 100 records/lines on the standard output device.

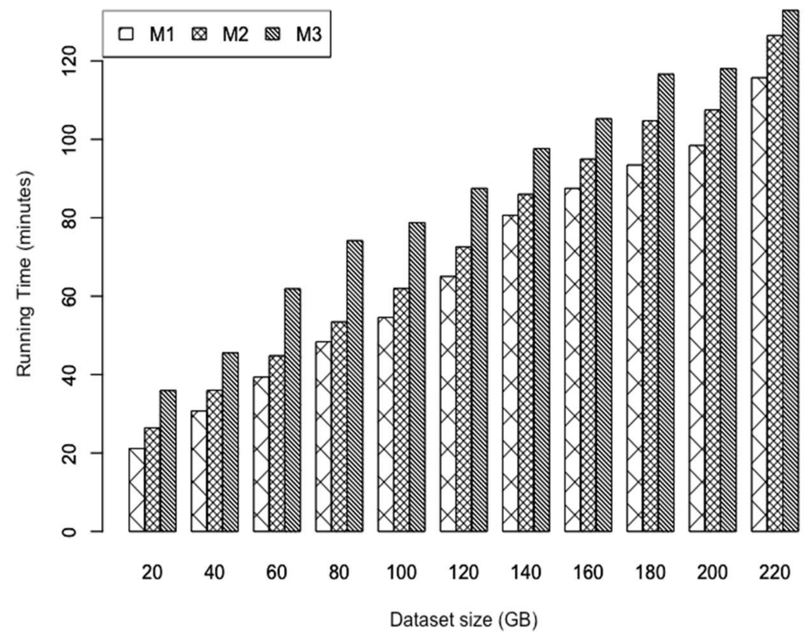


Figure 60 Running time of MR jobs on the structured data

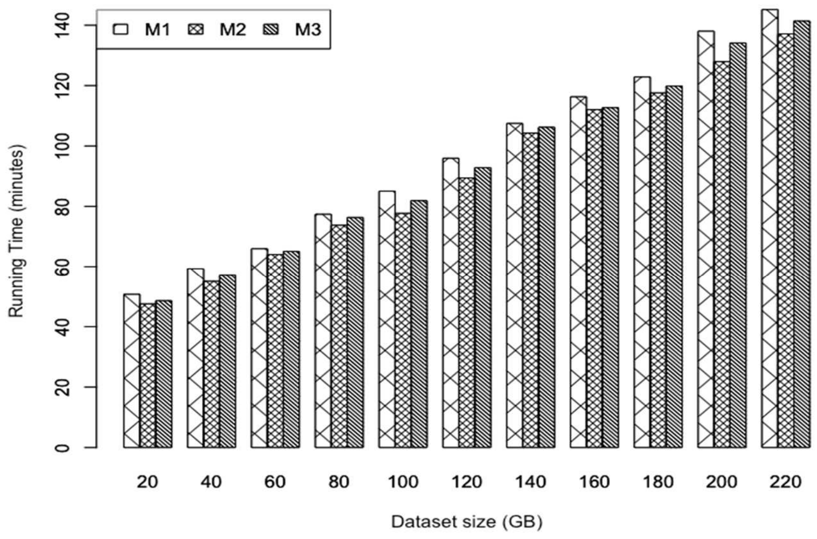


Figure 61Running time of MR jobs on the unstructured data

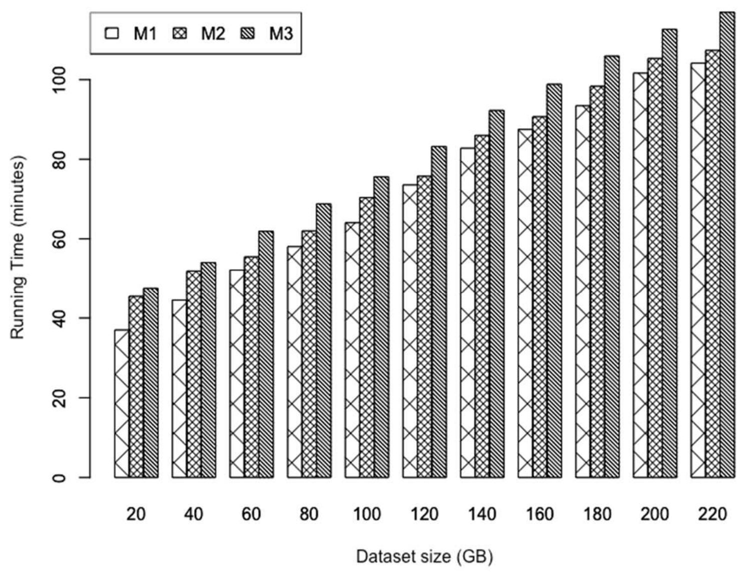


Figure 62 Running time of MR jobs on the semi-structured data

In Figure 60, 61 and 62, keys M1, M2 and M3 stand for the three MapReduce jobs discussed above. Figure 59 depicts the overhead imposed by the framework to access datasets via the command line. From the figures 60, 61 and 62 it can be observed that the overhead for accessing structured data is the least and the overhead for accessing unstructured data is the highest. Since structured dataset follows a specific format it is sufficient to identify the data items (attributes) and fetch their sensitivity to estimate which data items the user is authorized to access once. After the data items that the user is authorized to view has been identified for a single row it is unchanged for the rest of the dataset. In an unstructured dataset the structure is non-uniform and varies from record to record. Hence the data items have to be identified first for each record and then their sensitivity must be compared with the user's sensitivity to check if the user is authorized to access the data item or not. Since this process is repeated for every record the overhead is high. Overhead to access semi-structured datasets via command line is in between the overhead for the structured and unstructured datasets. For every record in a semi-structured dataset the tags (in XML files) or keys (in JSON files) are identified and their sensitivity is checked against the user's sensitivity. Even though the data items are identified for every record in the semi-structured dataset like the unstructured dataset, the identification process is itself simpler because the tags/keys corresponds to the data item names in the semi-structured datasets. Thus the overhead for the semi-structured dataset is lower than the unstructured dataset. In Figure 61 the running time for the jobs on semi-structured data increases at a slower rate with increase in data size when compared to structured data. The processing time increases at a faster rate for structured data when compared to the semi-structured data, as the entire record in structured data has to be processed to get the desired results whereas in the semi-structured data the required information can be selected by just performing a tag searching. Thus the jobs running on a large structured dataset takes longer time to complete. Mappers used in our MapReduce jobs use the "*json*" library that is shipped with python to parse JSON data. A detailed analysis of benchmarking of various python libraries to parse JSON data is discussed in [184]. In [184], A. Krylysov identifies that this library takes a longer time to parse a smaller JSON data. This is the reason for higher running times for jobs processing smaller sized semi-structured dataset. From Figure 60, it can be seen that it takes more time to run job 3 than job 1 on the structured dataset, as MapReduce job 3 has to compute the average whereas the MapReduce job 2 just prints the records matching a patient name. Running time of MapReduce job 1 on the unstructured dataset is higher because it emits more data to the reducer compared to the other two jobs. This can be seen in Figure 61. Running time of the MapReduce jobs on the semi-structured dataset is shown in Figure 62 and it follows the same pattern as the structured dataset.

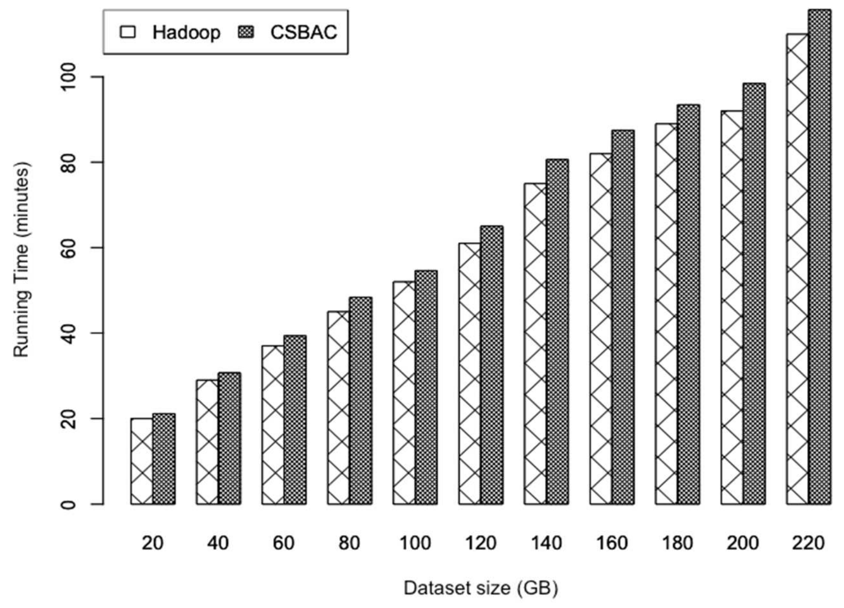


Figure 63 Overhead posed by CSBAC for Job 1 on Structured data

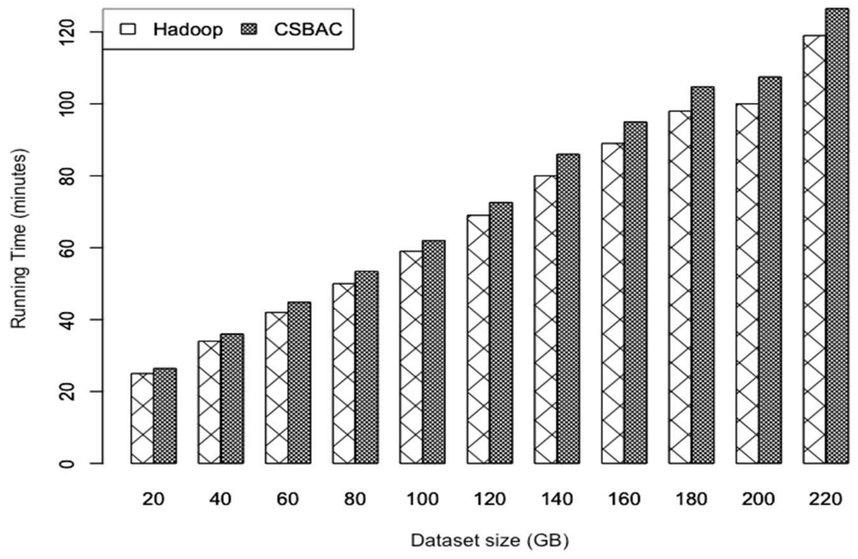


Figure 64 Overhead posed by CSBAC for Job 2 on Structured data

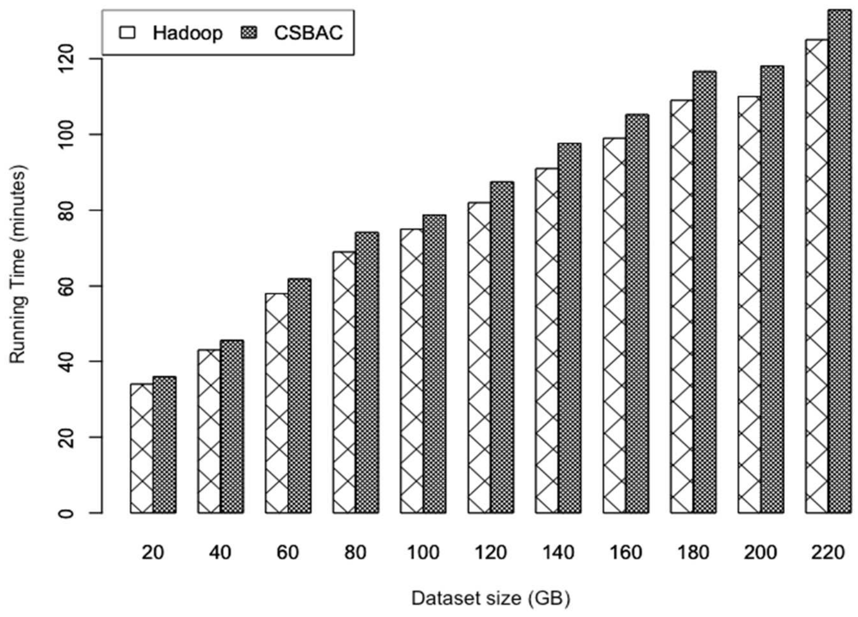


Figure 65 Overhead posed by CSBAC for Job 3 on Structured data

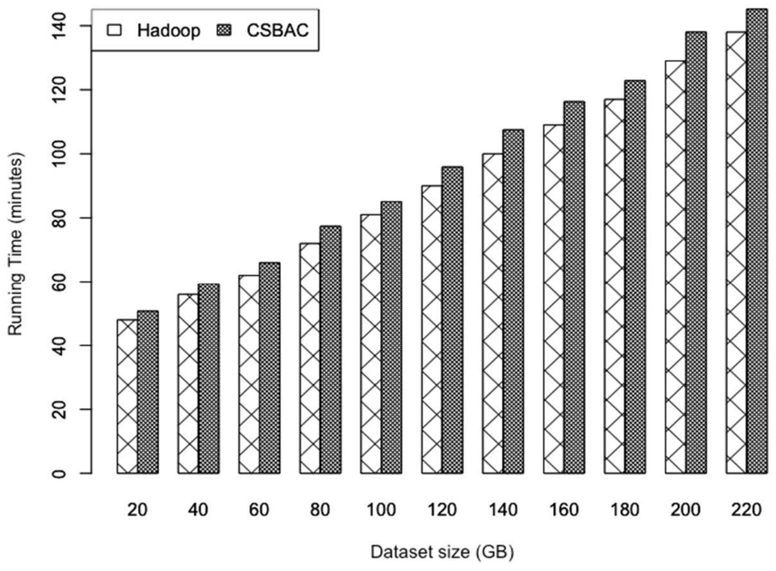


Figure 66 Overhead posed by CSBAC for Job 1 on unstructured data

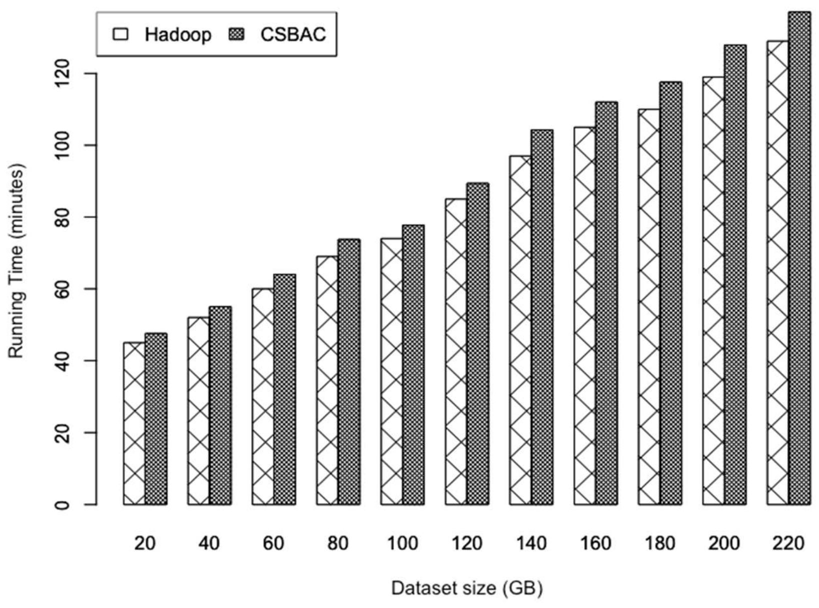


Figure 67 Overhead posed by CSBAC for Job 2 on unstructured data

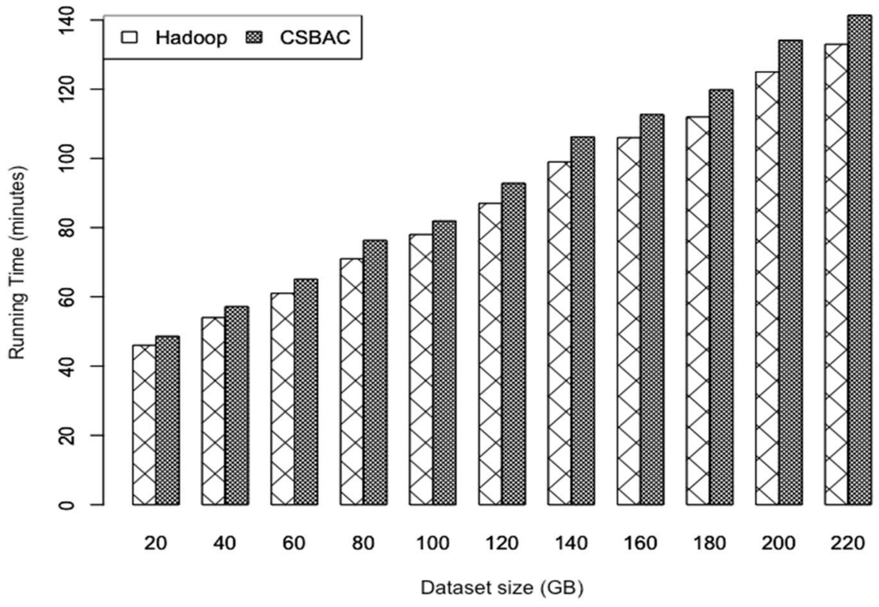


Figure 68 Overhead posed by CSBAC for Job 3 on unstructured data

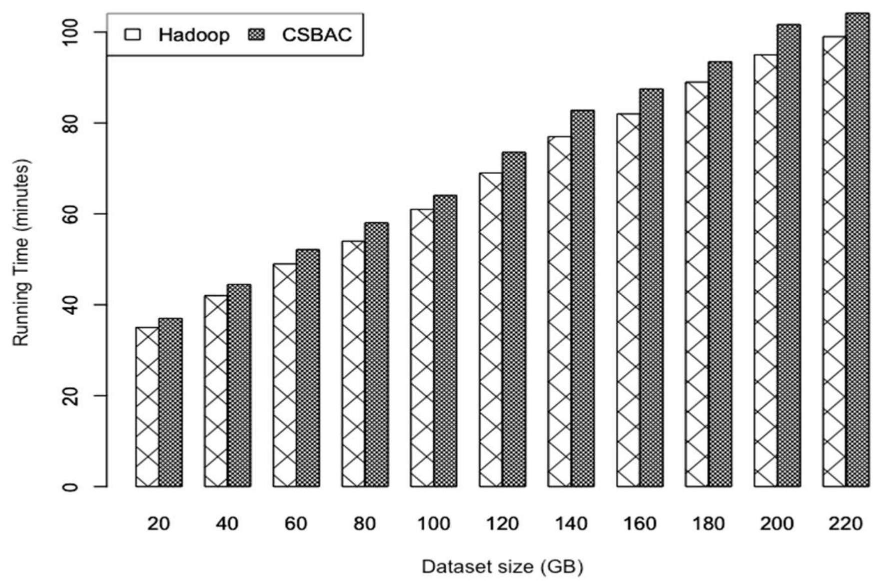


Figure 69 Overhead posed by CSBAC for Job 1 on semi-structured data

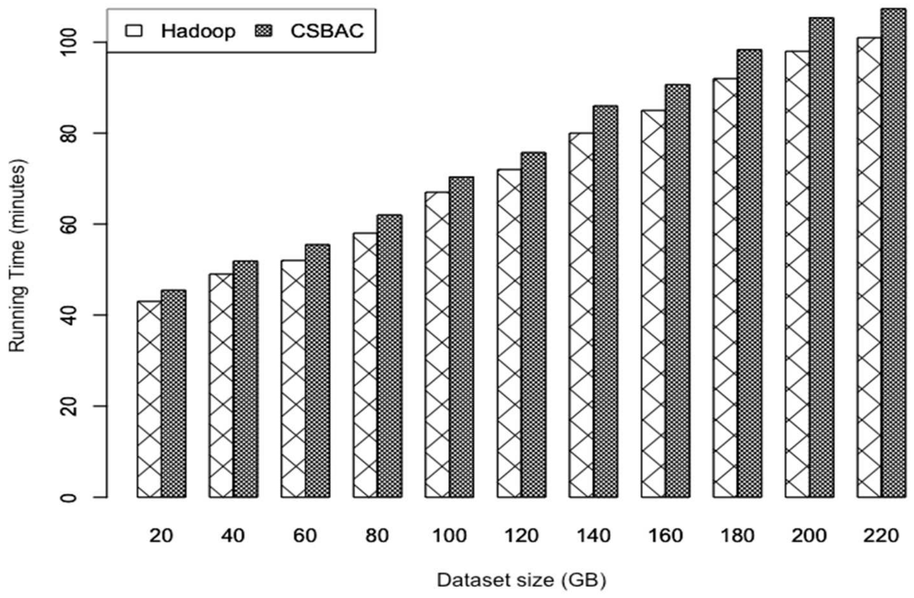


Figure 70 Overhead posed by CSBAC for Job 2 on semi-structured data

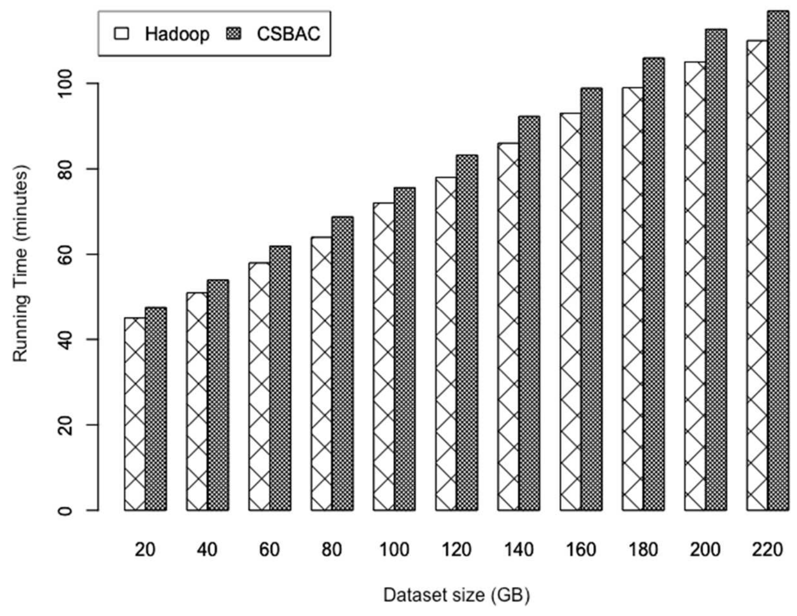


Figure 71 Overhead posed by CSBAC for Job 3 on semi-structured data

Figures 63 through 71 compares the running times of MapReduce jobs on naïve Hadoop implementation and the proposed CSBAC framework on different types of textual datasets. On average about 6.5% overhead is imposed by the proposed framework. This is a tradeoff to protect vulnerable data items from misuse or abuse. Figures 63, 64 and 65 compares the running times of MapReduce jobs 1, 2 and 3 on the structured dataset. Figures 66, 67 and 68 compares the running times of MapReduce jobs 1, 2 and 3 on the unstructured dataset. Figures 69, 70 and 71 compares the running times of MapReduce jobs 1, 2 and 3 on the semi-structured dataset.

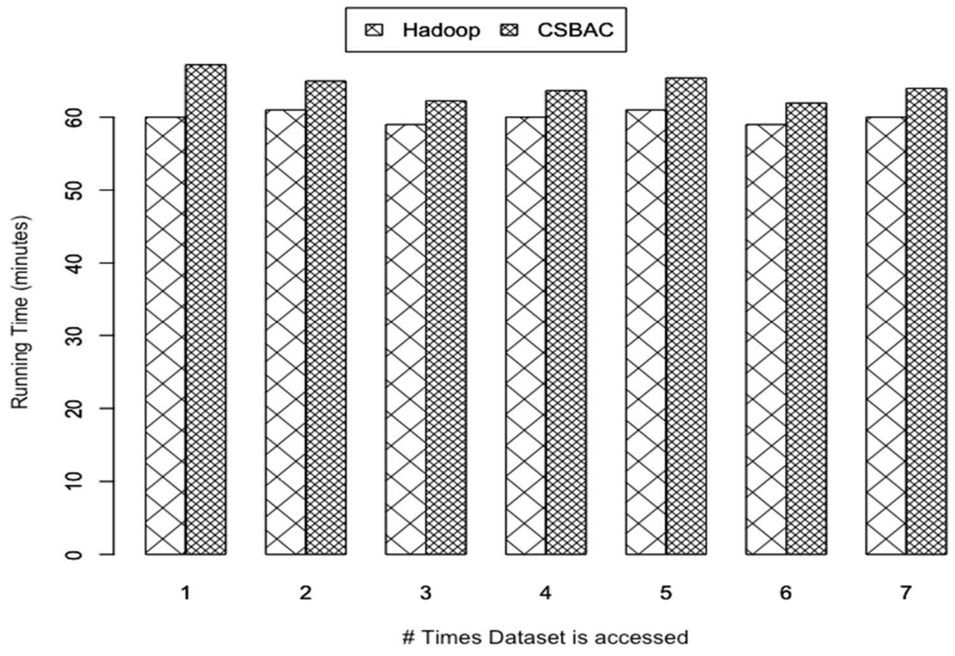


Figure 72 Overhead Analysis when multiple datasets are joined

Figure 72, analyzes the additional overhead imposed by the proposed CSBAC framework when multiple datasets are combined. To illustrate this scenario we run a MapReduce job that counts the number of entries in the diagnosis history for each patient. This is achieved by joining the relational dataset and the unstructured dataset. From our experiments and results we identify that the CSBAC framework imposes an additional 5-6% overhead when datasets are joined. The combined sensitivity estimate is stored in the DSE and re-used whenever these datasets are joined again in MapReduce jobs.

Comparison with CBAC

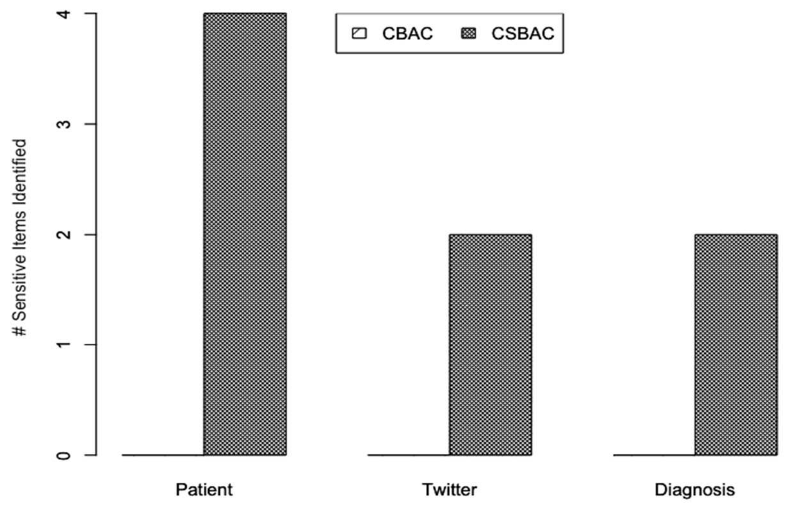


Figure 73Comparison between CBAC and CSBAC with no base set

Performance of the proposed framework is compared with the CBAC Framework in identifying sensitive data items from a dataset. The CBAC framework proposed in [19, 31] uses top-k similarity algorithm and requires a base set to function. To compare the proposed CSBAC framework and the CBAC framework proposed in [19, 31], two scenarios are considered namely 1) with no base set, and 2) with a partial base set. The probability of constructing a complete base set covering all scenarios is impossible, this scenario is not considered. When there is no base set, the CBAC performs poorly and doesn't identify any data items that are potentially sensitive because there is no dataset for the scheme to compare it with. This is evident in Figure 73 where CBAC identified no sensitive data items. The proposed CSBAC framework can function well by identifying sensitive data items even though there is no base set. CSBAC framework identifies about 67%, 40% and 67% of sensitive data items in the patient, diagnosis and twitter datasets respectively. Comparing the results of the proposed framework and CBAC with a manual identification process derives these percentages.

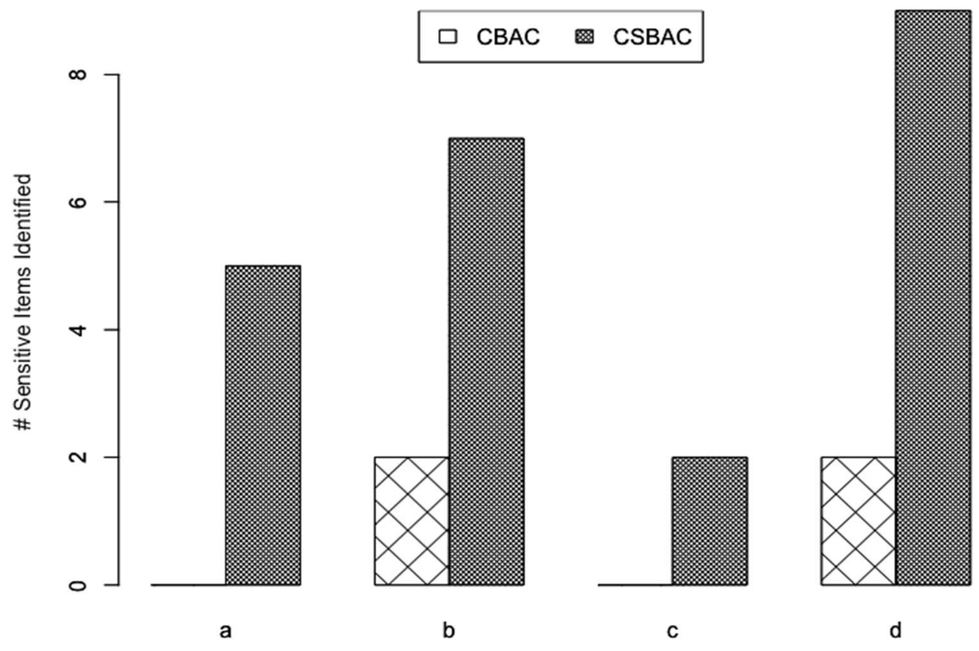


Figure 74 Comparison between CBAC and CSBAC with a partial base set.

When there is a partial base set, the CBAC performs relatively well but not as well as the proposed framework and does not identify all the data items that are potentially sensitive. In Figure 74, x-axis labels (a) through (d) refer to four simulation scenarios as listed below. Label a denotes the number of sensitive data items identified when the partial base set contains patient dataset and when the sensitivity is estimated for a join operation for patient and twitter datasets. Since both the datasets do not contain any similar items the top-k similarity algorithm proposed in [19 and 31] (CBAC) does not identify any sensitive data items but the proposed CSBAC algorithm identifies 5 sensitive data items. Label b denotes the number of sensitive data items identified when the partial base set contains patient dataset and when the sensitivity is estimated for a join operation for patient and diagnosis datasets. Since both the datasets contains patient name the top-k similarity algorithm proposed in [19 and 31] (CBAC) identifies 2 sensitive data items but the proposed CSBAC algorithm identifies 7 sensitive data items. Label c denotes the number of sensitive data items identified when the partial base set contains patient dataset and when the sensitivity is estimated for a join operation for patient and twitter datasets. Since both the datasets do not contain any similar data items the top-k similarity algorithm proposed in [19 and 31] (CBAC) identifies no sensitive data items but the proposed CSBAC algorithm identifies 2 sensitive data items. Label d denotes the number of sensitive data items identified when the partial base set contains patient dataset and when the sensitivity is estimated for a join operation for patient, twitter and diagnosis datasets. Since the patient and diagnosis datasets contains patient name the top-k similarity algorithm proposed in [19 and 31] (CBAC) identifies 2 sensitive data items but the proposed CSBAC algorithm identifies 9 sensitive data items. For scenarios a and c, the proposed CSBAC framework identifies 55% and 22% of sensitive data items whereas the CBAC framework does not identify any sensitive data item at all. For scenarios b and d the proposed CSBAC framework identifies 63% and 64% of the sensitive data items whereas the CBAC framework identifies 18% and 14% of the sensitive data items respectively. Comparing the results of the framework with a manual identification process derives percentages. One should note that the number of data items identified by the proposed CSBAC framework took into account the variation in the data sensitivity when multiple datasets are joined in Figure 87. The number of sensitive data items identified by the proposed framework will change if themselves access the datasets. The CBAC framework proposed in [19, 31] did not consider variation in the data sensitivity into account. Furthermore, effort is expended in creating the base data set for CBAC. There is also no way to measure the '*goodness*' or '*effectiveness*' of the base dataset. This is not the case for the proposed scheme, as no base set is needed.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this dissertation a Content Sensitivity Based Access Control (CSBAC) Framework for Hadoop is proposed. This framework makes use of the data itself to make access decisions. To the very best of our knowledge using the data to estimate its sensitivity and to use it to enforce access control policies are novel for Hadoop. In addition to it the CSBAC Framework captures changes in sensitivity when multiple datasets are joined. CSBAC is automated framework requiring minimal user intervention. It saves lots of effort put forth by the data owner and administrator of the Hadoop cluster. CSBAC Framework adheres to all the seven principles of privacy by design (PbD) [32]. CSBAC is a hybrid access control framework as it uses both attributes (to estimate sensitivity) and user role to make and enforce access control decisions. There is an overhead imposed by the proposed framework, but it is a tradeoff for keeping sensitive information out of reach from the unauthorized users. By doing so data misuse or abuse can be prevented even before it happens. In future, different methods will be used to estimate data sensitivity such as Bayesian networks and cognitive computing in place of information gain and the results of these various methods will be analyzed.

REFERENCES

[1] Schmidt, E., (2014). Tech-crunch article on data production, http://techcrunch.com/2010/08/04/schmidt-data/, [Accessed 09-22-2014].

[2] Cavoukian, A., Chibba, M., Williamson, G., Ferguson, A., (2015). The Importance of ABAC: Attribute-Based Access Control to Big Data: Privacy and Context, http://www.ryerson.ca/content/dam/pbdi/Resources/The%20Importance%20of%20ABAC%20to%20Big%20Data%2005-2015.pdf [Accessed 08-25-2015]

[3] Sweeney, L., (2000). Simple demographics often identify people uniquely." Carnegie Mellon University, Data Privacy Working Paper 3, Pittsburgh (2000), pp: 1-34.

[4] Tene, O., Polonetsky, J., (2012). Big data for all: Privacy and user control in the age of analytics. 11 Nw. J. Tech. & Intell. Prop. Vol: 11, No: 5 (2012), pp: 239-273

[5] Nakashima, E., (2006).AOL Takes Down Site With Users' Search Data, Washington Post, http://www.washingtonpost.com/wp-dyn/content/article/2006/08/07/AR2006080701150.html, [Accessed 09-23-2014].

[6] Narayanan, A., Vitaly, S., (2008). Robust de-anonymization of large sparse datasets. In IEEE Symposium on Security and Privacy, pp. 111-125. IEEE, 2008.

[7] Aggarwal, C., C., (2005). On k-anonymity and the curse of dimensionality. In Proceedings of the 31st international conference on Very large databases, August 2005, pp: 901-909. VLDB Endowment, 2005.

[8] Chawla, S., Dwork, C., McSherry, F., Smith, A., and Wee, H., (2005). Toward privacy in public databases. In Theory of Cryptography, pp. 363-385. Springer Berlin Heidelberg, 2005.

[9] Sweeney, L., (2002). Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol: 10, No. 05 (2002), pp: 571-588.

[10] Simons, B., Spafford, E., H., (2003). Letter from ACM to Senate Committee on Armed Services, The Electronic Frontier Foundation, https://w2.eff.org/Privacy/TIA/acm-letter.php, [Accessed 09-06-2015].

[11] Heffner, K., Popkin, R., Reem, A., Kannan, A., Report on Data Aggregation, The Electronic Frontier Foundation, http://www.eecs.harvard.edu/cs199r/bd-aggregation/kheffner_etal.pdf, [Accessed 09-06-2015]

[12] Hadoop Implementation, http://Hadoop.apache.org/docs/ [Accessed 09-01-2015].

[13] HDFS Permissions Guide, http://Hadoop.apache.org/docs/stable/Hadoop-project-dist/Hadoop-hdfs/HdfsPermissionsGuide.html [09-01-2015].

[14] Cloud security Alliance, (2013). Top Ten Big Data Security and Privacy Challenges, https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf [Accessed 09-06-2015]

[15] The Sarbanes-Oxley Act, http://www.soxlaw.com/ [Accessed 09-06-2015]

[16] The HIPPA Act, http://www.hhs.gov/ocr/privacy/ [Accessed 09-06-2015]

[17] Krishnan, R., Access Control and Privacy Policy Challenges in big data, Position Paper, NSF Workshop on Big Data Security and Privacy, http://csi.utdallas.edu/events/NSF/papers/paper10.pdf [Accessed 09-06-2015]

[18] T, K, Ashwin Kumar., Liu, H.,Thomas, J, P., (2014). Efficient structuring of data in big data. In International Conference on Data Science & Engineering 2014, pp. 1-5.

[19] T, K, Ashwin Kumar., Liu, H.,Thomas, J, P., Mylavarapu, G., (2015). Identifying Sensitive Data Items within Hadoop. In 1st IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2015), 2015, pp. 1308-1313.

[20] Liu, H.,T, K, Ashwin Kumar., Thomas, J, P., (2015). Cleaning Framework for Big Data-Object Identification and Linkage. In IEEE International Congress onBig Data (BigData Congress), 2015, pp. 215-221.

[21] General Atomics, (2015). Tackling the Big Data Deluge in Science With Metadata, White Paper, http://whitepapers.insidehpc.com/?option=com_categoryreport&task=viewabstract&pathway=no&title=44242 [Accessed 09-06-2015]

[22] Reddy, Y. B., (2013). Access control for sensitive data in hadoop distributed file systems. In Third International Conference on Advanced Communications and Computation, 2013, pp. 17-22.

[23] Roy, I., Setty, S. T., Kilzer, A., Shmatikov, V., Witchel, E. (2010). Airavat: Security and Privacy for MapReduce. In NSDI vol. 10, 2010, pp. 297-312.

[24] Ulusoy, H., Kantarcioglu, M., Pattuk, E., Hamlen, K. (2014). Vigiles: Fine-grained access control for mapreduce systems. InIEEE International Congress onBig Data (BigData Congress), 2014, pp. 40-47.

[25] Yang, K., Jia, X., Ren, K., Xie, R., & Huang, L. (2014). Enabling efficient access control with dynamic policy updating for big data in the cloud. In Proceedings of INFOCOM, 2014, pp. 2013-2021. IEEE.

[26] Hu, V. C., Grance, T., Ferraiolo, D. F., Kuhn, D. R. (2014). An Access Control scheme for Big Data processing. In International Conference onCollaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014, pp. 1-7. IEEE.

[27] Chen, H., Bhargava, B., & Zhongchuan, F. (2014). Multilabels-Based Scalable Access Control for Big Data Applications. Cloud Computing, IEEE, Vol: 1, no. 3, 2014, pp. 65-71.

[28] Yuan, Q., Ma, C., & Lin, J. (2015). Fine-Grained Access Control for Big Data Based on CP-ABE in Cloud Computing. In Intelligent Computation in Big Data Era, 2015, pp. 344-352. Springer Berlin Heidelberg.

[29] Zeng, W., Yang, Y., Luo, B. (2013). Access control for Big Data using data content. In International Conference onBig Data, 2013, pp. 45-47. IEEE.

[30] Nasim, R., Buchegger, S. (2014). XACML-Based Access Control for Decentralized Online Social Networks. In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014, pp. 671-676, 2014.IEEE Computer Society.

[31] Zeng, W., Yang, Y., Luo, B. (2014). Content-Based Access Control: Use data content to assist access control for large-scale content-centric databases. In IEEE International Conference on Big Data (Big Data), 2014, pp. 701-710. IEEE.

[32] Cobb, M., (2011). How to create a data aggregation risk mitigation plan, ComputerWeekly.com, January 2011, http://www.computerweekly.com/tip/How-to-create-a-data-aggregation-risk-mitigation-plan, [Accessed 09-06-2015]

[33] Cavoukian, A., (2009). Privacy by design: The 7 foundational principles. Information and Privacy Commissioner of Ontario, Canada (2009), https://www.iab.org/wp-content/IAB-uploads/2011/03/fred_carter.pdf, [Accessed 09-06-2015]

[34] Gartner, Inc. http://www.gartner.com/[Accessed 09-06-2015]

[35] Sajko, M., Rabuzin, K., & Bača, M. (2006). How to calculate information value for effective security risk assessment. Journal of Information and Organizational Sciences, Vol. 30, No. 2, 2006, pp. 263-278.

[36] Shannon, C., E., (2001). A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications, Review 5, No. 1, 2001, pp 3-55.

[37] Apache Software Foundation. http://hadoop.apache.org/[Accessed 09-06-2015]

[38] Borthakur, D., (2013). HDFS design Guide. Apache Software Foundation. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html[Accessed 09-06-2015]

[39] Dean, J., Ghemawat, S., (2004). Mapreduce : Simplified Data Processing on Large clusters, In the proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2004, pp. 137-150

[40] White, T., (2012). How MapReduce works, In "Hadoop: The definitive guide", O'Reilly Media, Inc., 2012. pp. 187-195.

[41] Harel, A., Shabtai, A., Rokach, L., Elovici, Y., (2011). Dynamic Sensitivity-Based Access Control. In ISI, 2011, pp. 201-203.

[42] Apache Accumulo, https://accumulo.apache.org/index.html [Accessed 09-06-2015]

[43] Rong, C., Quan, Z., Chakravorty, A. (2013). On Access Control Schemes for Hadoop Data Storage. In 2013 International Conference onCloud Computing and Big Data (CloudCom-Asia), 2013, pp. 641-645. IEEE.

[44] Somani, M., Swamiraj, M., Rengarajan, S., Shankar, H. (2012). BitTorrent for large package distribution in the enterprise environment. In 2012 International Conference on Recent Advances in Computing and Software Systems (RACSS), 2012, pp. 281-286. IEEE.

[45] Zhao, G., Rong, C., Li, J., Zhang, F., Tang, Y. (2010). Trusted data sharing over untrusted cloud storage providers. In 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 2010, pp. 97-103. IEEE.

[46] Niemann, K., Schmitz, H. C., Scheffel, M., Wolpers, M. (2011). Usage contexts for object similarity: exploratory investigations. In Proceedings of the 1st International Conference on Learning Analytics and Knowledge, 2011, pp. 81-85. ACM.

[47] Ghemawat, S., Gobioff, H., Leung, S, T., (2003). The Google File System, In ACM SIGOPS operating systems review, vol. 37, no. 5, 2003, pp. 29-43. ACM.

[48] O'Malley, O., (2008). Hadoop at Yahoo!, http://www.cs.umd.edu/class/spring2014/cmsc433-0201/Lectures/hadoopyahoo.pdf, [Accessed 09-06-2015]

[49] Apache Lucene, Apache Software Foundation, https://lucene.apache.org, [Accessed 09-06-2015]

[50] Apache Nutch, Apache Software Foundation, http://nutch.apache.org/, [Accessed 09-06-2015]

[51] White, T., (2012). Meet Hadoop, In Hadoop: The definitive guide, O'Reilly Media, Inc., 2012. pp. 9-12.

[52] Cafarella, M., Cutting, D., (2004). Open Source Search, ACM Queue, 2004,pp. 54-61.

[53] Baldeschwieler, E., (2008). Yahoo! Launches World's Largest Hadoop Production Application, Yahoo Developer Network,

https://developer.yahoo.com/blogs/hadoop/yahoo-launches-world-largest-hadoop-production-application-398.html, [Accessed 09-06-2015]

[54] Gottfrid, D., (2007). Self-Service, Prorated Supercomputing Fun! , The New York Times Blog, http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/, [Accessed 09-06-2015]

[55] White, T., (2012). The Hadoop File System, In Hadoop: The definitive guide, O'Reilly Media, Inc., 2012. pp. 45-54.

[56] Murthy, A. C., (2011). The Next Generation of Apache Hadoop MapReduce, Yahoo Developer Network, Hadoop Blog. https://developer.yahoo.com/blogs/hadoop/next-generation-apache-hadoop-mapreduce-3061.html [Accessed 01-22-2016].

[57] Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S. and Saha, B., (2013). Apache hadoop yarn: Yet another resource negotiator. In the Proceedings of 4th ACM Annual Symposium on Cloud Computing, 2013, Article-5.

[58] Sahafizadeh, E. and Parsa, S., (2010). Survey on access control models. In 2010 2nd International Conference on Future Computer and Communication (ICFCC), 2010, pp. 1-3, IEEE.

[59] NIST (2009). A Survey of Access Control Models, A Working Draft, http://csrc.nist.gov/news_events/privilege-management-workshop/PvM-Model-Survey-Aug26-2009.pdf [Accessed 03-28-2016]

[60] Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C. and Ré, C., (2015). Incremental knowledge base construction using deepdive. In the Proceedings of the VLDB Endowment, 8(11), 2015, pp.1310-1321.

[62] Bernstein, P.A., Madhavan, J. and Rahm, E., (2011). Generic schema matching, ten years later. In the proceedings of the VLDB Endowment, 4(11), 2011, pp.695-701.

[63] Melnik, S., Garcia-Molina, H. and Rahm, E., (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In the proceedings of the 18th International Conference on Data Engineering, 2002, pp. 117-128. IEEE.

[64] Doan, A., Madhavan, J., Domingos, P. and Halevy, A., (2002). Learning to map between ontologies on the semantic web. In Proceedings of the 11th international conference on World Wide Web, 2002, pp. 662-673. ACM.

[65] Shen, W., Li, X. and Doan, A., (2005). Constraint-based entity matching. In AAAI, 2005, pp. 862-867.

[66] Le Clément, V., Deville, Y. and Solnon, C., (2009). Constraint-based graph matching. In Principles and Practice of Constraint Programming-CP, 2009, pp. 274-288. Springer Berlin Heidelberg.

[67] Candillier, L., Jack, K., Fessant, F. and Meyer, F., (2009). Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling, Chapter State-of-the-Art Recommender Systems.

[68] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, 2001, pp. 285-295. ACM.

[69] Schafer, J.B., Frankowski, D., Herlocker, J. and Sen, S., (2007). Collaborative filtering recommender systems. In the adaptive web, 2007, pp. 291-324. Springer Berlin Heidelberg.

[70] Pazzani, M.J. and Billsus, D., (2007). Content-based recommendation systems. In the adaptive web, 2007, pp. 325-341. Springer Berlin Heidelberg.

[71] Dongen, S.V., (2000). Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht, 2000.

[72] Kannan, R., Vempala, S. and Vetta, A., (2004). On clusterings: Good, bad and spectral. Journal of the ACM (JACM), 51(3), 2004, pp.497-515.

[73] Jayarajan, D., Deodhare, D. and Ravindran, B., (2008). Lexical chains as document features.Proceedings of the Third International Joint Conference on Natural Language Processing, Volume-I, http://aclweb.org/anthology/I08-1015 [01-15-2015].

[74] Berker, M., (2011). Using genetic algorithms with lexical chains for automatic text summarization, Doctoral dissertation, Bogaziçi University, 2011.

[75] List of First Names - Social security dataset, http://www.ssa.gov/oact/babynames/limits.html [09-22-2014].

[76] Genealogy Data - U.S. Census Bureau, https://www.census.gov/genealogy/www/data/1990surnames/names_files.html [09-22-2014].

[77] List of cities, states & counties - U.S. Small Business Administration, http://www.sba.gov/ [09-22-2014].

[78] Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K.J., (1990). Introduction to wordnet: An on-line lexical database. International journal of lexicography, 3(4), 1990, pp.235-244.

[79] Harel, A., Shabtai, A., Rokach, L. and Elovici, Y., (2011). Eliciting domain expert misuseability conceptions. In Proceedings of the sixth international conference on Knowledge capture, pp. 193-194, 2011. ACM.

[80] Harel, A., Shabtai, A., Rokach, L. and Elovici, Y., (2010). M-score: estimating the potential damage of data leakage incident by assigning misuseability weight. In Proceedings of the 2010 ACM workshop on Insider threats, pp. 13-20, 2010. ACM.

[81] Moody, D.L. and Walsh, P., (1999). Measuring the Value Of Information-An Asset Valuation Approach. In Seventh European Conference on Information Systems (ECIS), pp. 496-512, 1999.

[82] Glazer, R., (1993). Measuring the value of information: The information-intensive organization. IBM Systems Journal, 32(1), pp.99-110, 1993.

[83] Wang, R.Y. and Strong, D.M., (1996). Beyond accuracy: What data quality means to data consumers. Journal of management information systems, 12(4), pp.5-33, 1996.

[84] Haebich, W., (1997). A Quantitative Model to Support Data Quality Improvement. Proceedings of the Conference onInformation Quality MIT, Boston, pp. 194-208, 1997.

[85] Goodhue, D.L., Wybo, M.D. and Kirsch, L.J., (1992). The impact of data integration on the costs and benefits of information systems. MIS Quarterly, pp.293-311, 1992.

[86] Miller, G.A., (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological review, 63(2), p.81, 1956.

[87] Newell, A. and Simon, H.A., (1972). Human problem solving, Vol. 104, No. 9. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[88] Lipowski, Z.J., (1975). Sensory and information inputs overload: Behavioral effects. Comprehensive Psychiatry, 16(3), pp.199-221, 1975.

[89] O'Reilly, C.A., (1980). Individuals and information overload in organizations: is more necessarily better? Academy of management journal, 23(4), pp.684-696, 1980.

[90] Driver, M.J. and Mock, T.J., (1975). Human information processing, decision style theory, and accounting information systems. The Accounting Review, 50(3), pp.490-508, 1975.

[91] Jacoby, J., Speller, D.E. and Kohn, C.A., (1974). Brand choice behavior as a function of information load. Journal of Marketing Research, pp.63-69.

[92] Ashwin Kumar, T.K., George, K.M., and Thomas, J.P., (2015). An Empirical Approach to Detection of Topic Bubbles in Tweets. In 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC), 2015, pp. 31-40. IEEE

[93] Sandhu, R. and Chen, F., (1998). The multilevel relational (MLR) data model. ACM Transactions on Information and System Security (TISSEC), 1(1), 1998, pp.93-132.

[94] Jajodia, S. and Sandhu, R., (1991). Toward a multilevel secure relational data model. ACM SIGMOD Record, 20(2), 1191, pp.50-59.

[95] Denning, D.E. and Lunt, T.F., (1987). A multilevel relational data model. In IEEE Symposium on Security and Privacy, 1987, pp. 220-220. IEEE.

[96] Sandhu, R.S., (1993). Lattice-based access control models. Computer, 26(11), 1993, pp.9-19.

[97] Benantar, M., (2006). Mandatory-Access-Control Model. Access Control Systems: Security, Identity Management and Trust Models, pp.129-146.

[98] Osborn, S., (1997). Mandatory access control and role-based access control revisited. In Proceedings of the second ACM workshop on Role-based access control, 1997, pp. 31-40. ACM.

[99] Winslett, M., Smith, K. and Qian, X., (1994). Formal query languages for secure relational databases. ACM Transactions on Database Systems (TODS), 19(4), 1994, pp.626-662.

[100] Li, L., He, Y.Z. and Feng, D.G., (2004). A fine-grained mandatory access control model for xml documents. Journal of software, 15(10), 2004, pp.1528-1537.

[101] Lindqvist, H., (2006). Mandatory access control. Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901, 87.

[102] Thuraisingham, B., (2009). Mandatory access control. In Encyclopedia of Database Systems, 2009, pp. 1684-1685. Springer US.

[103] Upadhyaya, S., (2011). Mandatory access control. In Encyclopedia of Cryptography and Security, 2011, pp. 756-758. Springer US.

[104] McCune, J.M., Jaeger, T., Berger, S., Caceres, R. and Sailer, R., (2006). Shamon: A system for distributed mandatory access control. In Computer Security Applications Conference, 2006, pp. 23-32. IEEE.

[105] Ahn, G.J., (2009). Discretionary Access Control. In Encyclopedia of Database Systems 2009, pp. 864-866. Springer US.

[106] Li, N., (2011). Discretionary access control. In Encyclopedia of Cryptography and Security, 2011, pp. 353-356. Springer US.

[107] Dittrich, K.R., Härtig, M. and Pfefferle, H., (1988). Discretionary Access Control in Structurally Object-Oriented Database Systems. In DBSec, 1988, pp. 105-121.

[108] Moffett, J., Sloman, M. and Twidle, K., (1990). Specifying discretionary access control policy for distributed systems. Computer Communications, 13(9), 1990, pp.571-580.

[109] Bertino, E., Samarati, P. and Jajodia, S., (1993). High assurance discretionary access control for object bases. In Proceedings of the 1st ACM conference on Computer and communications security, 1993, pp. 140-150. ACM.

[110] Thomas, R.K. and Sandhu, R.S., (1993). Discretionary access control in object-oriented databases: Issues and research directions. In Proceedings of the 16th National Computer Security Conference, 1993, pp. 63-74.

[111] Bugliesi, M., Colazzo, D. and Crafa, S., (2004). Type based discretionary access control. In CONCUR 2004-Concurrency Theory, 2004, pp. 225-239. Springer Berlin Heidelberg.

[112] McCollum, C.J., Messing, J.R. and Notargiacomo, L., (1990). Beyond the pale of MAC and DAC-defining new forms of access control. In Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy, 1990, pp. 190-200. IEEE.

[113] Ferraiolo, D., Kuhn, D.R. and Chandramouli, R., (2003). Role-based access control. Artech House.

[114] Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E., (1996). Role-based access control models. Computer, (2), 1996, pp.38-47.

[115] Sandhu, R.S., (1998). Role-based access control. Advances in computers, 46, 1998, pp.237-286.

[116] Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R. and Chandramouli, R., (2001). Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC), 4(3), 2001, pp.224-274.

[117] Ferraiolo, D., Cugini, J. and Kuhn, D.R., (1995). Role-based access control (RBAC): Features and motivations. In Proceedings of 11th annual computer security application conference, 1995, pp. 241-48.

[118] Ferraiolo, D.F. and Kuhn, D.R., (2009). Role-based access controls. arXiv preprint arXiv:0903.2171. 2009.

[119] Moyer, M.J. and Abamad, M., 2001, April. Generalized role-based access control. In 21st International Conference on Distributed Computing Systems, 2001, pp. 391-398. IEEE.

[120] Ni, Q., Bertino, E., Lobo, J., Brodie, C., Karat, C.M., Karat, J. and Trombeta, A., (2010). Privacy-aware role-based access control. ACM Transactions on Information and System Security (TISSEC), 13(3), 2010, p.24.

[121] Qian, J., Hinrichs, S. and Nahrstedt, K., (2001). ACLA: A framework for access control list (ACL) analysis and optimization. In Communications and Multimedia Security Issues of the New Century, 2001, pp. 197-211. Springer US.

[122] Yu, S., Wang, C., Ren, K. and Lou, W., (2010). Achieving secure, scalable, and fine-grained data access control in cloud computing. In proceedings of IEEE Infocom, 2010, pp. 1-9. IEEE.

[123] Bell, D.E. and LaPadula, L.J., (1973). Secure computer systems: Mathematical foundations (No. MTR-2547-VOL-1). MITRE CORP BEDFORD MA.

[124] Zhou, X. and Tang, X., (2011). Research and implementation of RSA algorithm for encryption and decryption. In 6th International Forum on Strategic Technology (IFOST), 2011, Vol. 2, pp. 1118-1121. IEEE.

[125] Downs, D.D., Rub, J.R., Kung, K.C. and Jordan, C.S., (1985). Issues in discretionary access control. In IEEE Symposium on Security and Privacy, 1985, pp. 208-208. IEEE.

[125] Graham, G.S. and Denning, P.J., (1972). Protection: principles and practice. In Proceedings of the spring joint computer conference, 1972, pp. 417-429. ACM.

[126] Lampson, B.W., (1974). Protection. ACM SIGOPS Operating Systems Review, 8(1), 1974, pp.18-24.

[127] Griffiths, P.P. and Wade, B.W., (1976). An authorization mechanism for a relational database system. ACM Transactions on Database Systems (TODS), 1(3), 1976, pp.242-255.

[128] Bishop, M., (2012). Computer security: art and science (Vol. 200). Addison-Wesley.

[129] Castano, S., Fugini, M.G., Martella, G. and Samarati, P., (1995). Database security. ACM Press Books, Wokingham, England: Addison-Wesley, 1995.

[130] Zeng, W., (2015). Content-Based Access Control, Doctoral dissertation, University of Kansas, 2015.

[131] Ferraiolo, D. F., Gilbert, D. M. and Lynch, N., (1993). An examination of federal and commercial access control policy needs. In the proceedings of National Information Systems Security and National Computer Security Conference (NIST-NCSC), 1993, pp. 107-116. DIANE Publishing.

[132] HDFS Shell Commands, https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/FileSystemShell.html, [Accessed 06-28-2017].

[133] Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R. and Scarfone, K., (2013). Guide to attribute based access control (ABAC) definition and considerations (draft). NIST Special Publication, 800-162, 2013.

[134] OASIS (2013). eXtensible Access Control Markup Language (XACML) Version 3.0, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf , [10-04-2016].

[135] Zhi, L., Jing, W., Xiao-su, C. and Lian-xing, J., (2009). Research on policy-based access control model. In International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009. Vol. 2, pp. 164-167. IEEE.

[136] Sousa, M.S. and Melo, A.C., (2006). Packageblast: an adaptive multi-policy grid service for biological sequence comparison. In Proceedings of the 2006 ACM symposium on Applied computing, 2006, pp. 156-160. ACM.

[137] IETF Policy Working Group. Policy Framework, https://datatracker.ietf.org/wg/policy/charter/, [Accessed 10-16-2016]

[138] McGraw, R., (2009). Risk-adaptable access control (RAdAC). In Privilege (Access) Management Workshop. NIST–National Institute of Standards and Technology–Information Technology Laboratory, 2009, http://csrc.nist.gov/news_events/privilege-management-workshop/presentations/Bob_McGraw.pdf, [Accessed 10-16-2016]

[139] Kandala, S., Sandhu, R. and Bhamidipati, V., (2011). An attribute based framework for risk-adaptive access control models. In Sixth International Conference on Availability, Reliability and Security (ARES), 2011, pp. 236-241. IEEE.

[140] Department of Defense.Directive 8000.01, (2009). http://www.dtic.mil/whs/directives/corres/pdf/800001p.pdf, [Accessed 10-16-2016]

[141] Department of Defense. Global information grid architectural vision.Vision for a Net-Centric, Service-Oriented DoD Enterprise. (2009). http://cio-nii.defense.gov/docs/GIGArchVision.pdf, [Accessed 10-16-2016]

[142] Giuri, L. and Iglio, P., (1997). Role templates for content-based access control. In Proceedings of the second ACM workshop on Role-based access control, 1997, pp. 153-159. ACM.

[143] Tzelepi, S.K., Koukopoulos, D.K. and Pangalos, G., (2001). A flexible content and context-based access control model for multimedia medical image database systems. In Proceedings of the 2001 workshop on Multimedia and security: new challenges, 2011, pp. 52-55. ACM.

[144] Gopal, B. and Manber, U., (1999). Integrating content-based access mechanisms with hierarchical file systems. In Operating Systems Design and Implementation (OSDI), Vol. 99, 1999, pp. 265-278.

[145] Bertino, E., Samarati, P. and Jajodia, S., (1997). An extended authorization model for relational databases. IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 1, 1997, pp.85-101.

[146] Bertino, E., Ghinita, G. and Kamra, A., (2011). Access control for databases: concepts and systems. Foundations and Trends® in Databases: Vol. 3, No. 1–2, 2011, pp. 1-148. http://dx.doi.org/10.1561/1900000014

[147] Gifford, D.K., Jouvelot, P. and Sheldon, M.A., (1991). Semantic file systems. In ACM SIGOPS operating systems review, Vol. 25, No. 5, 1991, pp. 16-25. ACM.

[148] Astrahan, M.M., Blasgen, M.W., Chamberlin, D.D., Eswaran, K.P., Gray, J.N., Griffiths, P.P., King, W.F., Lorie, R.A., McJones, P.R., Mehl, J.W. and Putzolu, G.R., (1976). System R: relational approach to database management. ACM Transactions on Database Systems (TODS), Vol. 1, No. 2, 1976, pp.97-137.

[149] Adam, N.R., Atluri, V., Bertino, E. and Ferrari, E., (2002). A content-based authorization model for digital libraries. IEEE Transactions on knowledge and data engineering, Vol. 14, No. 2, 2002, pp.296-315.

[150] Adam, N. and Yesha, Y., (1996). Strategic directions in electronic commerce and digital libraries: towards a digital agora. ACM Computing Surveys (CSUR), Vol. 28, No. 4, 1996, pp.818-835.

[151] Tran, N.A.T. and Dang, T.K., (2007). A novel approach to fine-grained content-based access control for video databases. In 18th International Workshop on Database and Expert Systems Applications, 2007, pp. 334-338. IEEE.

[152] Bertino, E., Fan, J., Ferrari, E., Hacid, M.S., Elmagarmid, A.K. and Zhu, X., (2003). A hierarchical access control model for video database systems. ACM Transactions on Information Systems (TOIS), Vol. 21, No. 2, 2003, pp.155-191.

[153] Bertino, E., Hammad, M.A., Aref, W.G. and Elmagarmid, A.K., (2000). An access control model for video database systems. In Proceedings of the ninth international conference on Information and knowledge management, 2000, pp. 336-343. ACM.

[154] Hart, M., Johnson, R. and Stent, A., (2007). More content-less control: Access control in the web 2.0. IEEE Web, 2.

[155] Hart, M., A. (2012). Content-based access control. PhD Dissertation, Stony Brook University

[156] Monte, S. (2010). Access control based on content, TKK Technical Reports in Computer Science and Engineering, B. TKK-CSE-B10. http://www. cse.tkk.fi/en/publications/B/10/papers/Monte final.pdf, [Accessed 10-16-2016]

[157] Carminati, B., Ferrari, E. and Perego, A., (2009). Enforcing access control in web-based social networks. ACM Transactions on Information and System Security (TISSEC), Vol. 13, No. 1, 2009, p.6.

[158] Molloy, I., Dickens, L., Morisset, C., Cheng, P.C., Lobo, J. and Russo, A., (2012). Risk-based security decisions under uncertainty. In Proceedings of the second ACM conference on Data and Application Security and Privacy, 2012, pp. 157-168. ACM.

[159] Ni, Q., Lobo, J., Calo, S., Rohatgi, P. and Bertino, E., (2009). Automating role-based provisioning by learning from examples. In Proceedings of the 14th ACM symposium on Access control models and technologies, 2009, pp. 75-84. ACM.

[160] Qin, L. and Atluri, V., (2003). Concept-level access control for the semantic web. In Proceedings of the 2003 ACM workshop on XML security, 2003, pp. 94-103. ACM.

[161] Toninelli, A., Montanari, R., Kagal, L. and Lassila, O., (2006). A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In International semantic web conference, 2006, pp. 473-486. Springer Berlin Heidelberg.

[162] Pan, C.C., Mitra, P. and Liu, P., (2006). Semantic access control for information interoperation. In Proceedings of the eleventh ACM symposium on Access control models and technologies, 2006, pp. 237-246. ACM.

[163] Fabian, B., Kunz, S., Konnegen, M., Müller, S. and Günther, O., (2012). Access control for semantic data federations in industrial product-lifecycle management. Computers in Industry, Vol. 63, No. 9, 2012, pp.930-940.

[164] Berners-Lee, T., Hendler, J. and Lassila, O., (2001). The semantic web. Scientific american, Vol. 284, No. 5, 2001, pp.28-37.

[165] Nakamoto, S., (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, [Accessed 06-26-2017]

[166] Agrawal, R., Imran, A., Seay, C. and Walker, J., (2014). A layer based architecture for provenance in big data. In IEEE International Conference on Big Data (Big Data), 2014, pp. 1-7. IEEE.

[167] MongoDB, https://www.mongodb.com/, [Accessed 06-28-2017]

[168] Ghoshal, D. and Plale, B., (2013). Provenance from log files: a BigData problem. In Proceedings of the Joint Workshops on EDBT/ICDT, 2013, pp. 290-297. ACM.

[169] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J. and Zhao, Y., (2006). Scientific workflow management and the kepler system. Concurrency and Computation: Practice and Experience, 2006, 18(10), pp.1039-1065.

[170] Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D. and Li, P., (2006). Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience, 2006, 18(10), pp.1067-1100.

[171] Crawl, D., Wang, J. and Altintas, I., (2011). Provenance for mapreduce-based data-intensive workflows. In Proceedings of the 6th workshop on Workflows in support of large-scale science, 2011, pp. 21-30. ACM.

[172] Altintas, I., Barney, O. and Jaeger-Frank, E., (2006). Provenance collection support in the kepler scientific workflow system. In International Provenance and Annotation Workshop, 2006, pp. 118-132. Springer, Berlin, Heidelberg.

[173] Ikeda, R., Park, H. and Widom, J., (2011). Provenance for generalized map and reduce workflows. In proceedings of 5th Biennial conference on Innovative Data Systems Research (CIDR' 11), 2011.

[174] Zhou, W., Fei, Q., Sun, S., Tao, T., Haeberlen, A., Ives, Z., Loo, B.T. and Sherr, M., (2011). NetTrails: a declarative platform for maintaining and querying provenance in distributed systems. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, 2011, pp. 1323-1326. ACM.

[175] Zhou, W., Sherr, M., Tao, T., Li, X., Loo, B.T. and Mao, Y., (2010). Efficient querying and maintenance of network provenance at internet-scale. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 2010, pp. 615-626, ACM.

[176] Muthukumar, S.C., Li, X., Liu, C., Kopena, J.B., Oprea, M. and Loo, B.T., 2009. Declarative toolkit for rapid network protocol simulation and experimentation. SIGCOMM, Association for Computing Machinery's Special Interest Group (demo).

[177] RapidNet, http://netdb.cis.upenn.edu/rapidnet/, [Accessed 06-28-2017]

[178] Back, A. (2002). Hashcash: A Denial of Service Counter-Measure, http://www.hashcash.org/papers/hashcash.pdf, [Accessed 06-28-2017]

[179] Merkle, R.C., (1980). Protocols for public key cryptosystems. In IEEE Symposium on Security and Privacy, 1980, pp. 122-122. IEEE.

[180] Synthetic patient dataset generator, https://github.com/synthetichealth/synthea, [Accessed 10-06-2016]

[181] Institute for Health Metrics and Evaluation, http://www.healthdata.org/united-states, [Accessed 10-06-2016]

[182] Twitter Streaming API, https://dev.twitter.com/overview/api, [Accessed 10-06-2016]

[183] Hadoop streaming API, https://hadoop.apache.org/docs/r1.2.1/streaming.html, [Accessed 10-06- 2016]

[184] A. Krylysov, 2015, Benchmark of Python JSON libraries, http://artem.krylysov.com/blog/2015/09/29/benchmark-python-json-libraries/, [Accessed 10-06-2016]

[185] Cheah, Y.W., Canon, R., Plale, B. and Ramakrishnan, L., (2013). Milieu: Lightweight and configurable big data provenance for science. In IEEE International Congress on Big Data (BigData Congress), 2013, pp. 46-53. IEEE.

[186] Olston, C. and Sarma, A.D., (2011). Ibis: A Provenance Manager for Multi-Layer Systems. In CIDR, 2011, pp. 152-159.

[187] Akoush, S., Sohan, R. and Hopper, A., (2013). HadoopProv: Towards Provenance as a First Class Citizen in MapReduce. In: Proceedings of the fifth USENIX conference on theory and practice of provenance (TaPP′13), 2013, pp. 1–4.

[188] Wang, J., Crawl, D., Purawat, S., Nguyen, M. and Altintas, I., (2015). Big data provenance: Challenges, state of the art and opportunities. In IEEE International Conference on Big Data (Big Data), 2015, pp. 2509-2516. IEEE.

[189] Amsterdamer, Y., Davidson, S.B., Deutch, D., Milo, T., Stoyanovich, J. and Tannen, V., (2011). Putting lipstick on pig: Enabling database-style workflow provenance. Proceedings of the VLDB Endowment, 2011, 5(4), pp.346-357.

[190] Glavic, B., (2014). Big data provenance: Challenges and implications for benchmarking. In specifying big data benchmarks, 2014, pp. 72-80. Springer, Berlin, Heidelberg.

[191] Korolev, V., Joshi, A. and Grasso, M.A., (2014). PROB: A tool for tracking provenance and reproducibility of big data experiments. In Proceeding of Workshop on Reproducible Research Methodologies (REPRODUCE'14), 2014, vol. 11, pp. 264-286.

[192] De Nies, T., Magliacane, S., Verborgh, R., Coppens, S., Groth, P., Mannens, E. and Van de Walle, R., (2013). Git2PROV: exposing version control system content as W3C PROV. In Proceedings of the 2013th International Conference on Posters & Demonstrations, 2013, Track-Volume 1035 (pp. 125-128). CEUR-WS. org.

[193] Git-Annex, http://git-annex.branchable.com/, [Accessed 06-28-2017]

[194] Apache Pig, https://pig.apache.org/, [Accessed 06-28-2017]

[195] Zhou, W., Mapara, S., Ren, Y., Li, Y., Haeberlen, A., Ives, Z., Loo, B.T. and Sherr, M., (2012). Distributed time-aware provenance. In Proceedings of the VLDB Endowment, 2012, Vol. 6, No. 2, pp. 49-60. VLDB Endowment.

[196] Muniswamy-Reddy, K.K., Macko, P. and Seltzer, M.I., (2010). Provenance for the Cloud. In Proceedings of the 8th USENIX conference on File and storage technologies (FAST), 2010, Vol. 10, pp. 15-14.

[197] Davidson, S.B. and Freire, J., (2008). Provenance and scientific workflows: challenges and opportunities. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 1345-1350. ACM.

[198] Simmhan, Y.L., Plale, B. and Gannon, D., (2006). A framework for collecting provenance in data-centric scientific workflows. In International Conference on Web Services. (ICWS'06), 2006, pp. 427-436. IEEE.

[199] Heinis, T. and Alonso, G., (2008). Efficient lineage tracking for scientific workflows. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 1007-1018. ACM.

[200] Crawl, D. and Altintas, I., (2008). A provenance-based fault tolerance mechanism for scientific workflows. Provenance and Annotation of Data and Processes, 2008, pp.152-159.

[201] Bose, R. and Frew, J., (2005). Lineage retrieval for scientific data processing: a survey. ACM Computing Surveys (CSUR), 2005, 37(1), pp.1-28.

[202] Simmhan, Y.L., Plale, B. and Gannon, D., (2005). A survey of data provenance in e-science. ACM Sigmod Record, 2005, 34(3), pp.31-36.

[203] Goble, C., (2002). Position statement: Musings on provenance, workflow and semantic web annotations for bioinformatics. In Workshop on Data Derivation and Provenance, Chicago, 2002, Vol. 3.

[204] Muniswamy-Reddy, K.K., Holland, D.A., Braun, U. and Seltzer, M.I., (2006). Provenance-aware storage systems. In USENIX Annual Technical Conference, General Track, 2006, pp. 43-56.

[205] Cui, Y. and Widom, J., (2000). Practical lineage tracing in data warehouses. In proceedings of 16th International Conference on Data Engineering, 2000, pp. 367-378. IEEE.

[207] Cui, Y. and Widom, J., (2003). Lineage tracing for general data warehouse transformations. The VLDB Journal—The International Journal on Very Large Data Bases, 2003, 12(1), pp.41-58.

[208] Ré, C. and Suciu, D., (2008). Approximate lineage for probabilistic databases. Proceedings of the VLDB Endowment, 2008, 1(1), pp.797-808.

[209] Park, J., Nguyen, D. and Sandhu, R., (2012). A provenance-based access control model. In Tenth Annual International Conference on Privacy, Security and Trust (PST), 2012, pp. 137-144. IEEE.

[210] Seltzer, M.I. and Margo, D.W., (2009). The Case for Browser Provenance, First workshop on Theory and practice of provenance, 2009, pp.1-5.

[211] Hasan, R., Sion, R. and Winslett, M., (2009). The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance. In FAST, 2009, Vol. 9, pp. 1-14.

[212] Tan, W.C., (2007). Provenance in databases: past, current, and future. IEEE Data Eng. Bull, 2007, 30(4), pp.3-12.

[213] Buneman, P., Chapman, A. and Cheney, J., (2006). Provenance management in curated databases. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data, 2006, pp. 539-550. ACM.

[214] Kim, J., Deelman, E., Gil, Y., Mehta, G. and Ratnakar, V., (2008). Provenance trails in the wings/pegasus system. Concurrency and Computation: Practice and Experience, 2008, 20(5), pp.587-597.

[215] Biton, O., Cohen-Boulakia, S., Davidson, S.B. and Hara, C.S., (2008). Querying and managing provenance through user views in scientific workflows. In IEEE 24th International Conference on Data Engineering (ICDE 2008), 2008, pp. 1072-1081. IEEE.

[226] Chirigati, F.S., Shasha, D.E. and Freire, J., (2013). ReproZip: Using Provenance to Support Computational Reproducibility. Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, 2013.

[227] Benjelloun, O., Sarma, A.D., Halevy, A. and Widom, J., (2006). ULDBs: Databases with uncertainty and lineage. In Proceedings of the 32nd international conference on Very large data bases, 2006, pp. 953-964. VLDB Endowment.

[228] Gadelha Junior, L.M.R., Wilde, M., Mattoso, M. and Foster, I., (2011). Exploring provenance in high performance scientific computing. In Proceedings of the first annual workshop on High performance computing meets databases, 2011, pp. 17-20. ACM.

[229] Wilde, M., Foster, I., Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B. and Raicu, I., (2009). Parallel scripting for applications at the petascale and beyond. Computer, 2009, 42(11).

[230] Wilde, M., Hategan, M., Wozniak, J.M., Clifford, B., Katz, D.S. and Foster, I., (2011). Swift: A language for distributed parallel scripting. Parallel Computing, 2011, 37(9), pp.633-652.

[231] Zhao, Y., Hategan, M., Clifford, B., Foster, I., Von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T. and Wilde, M., (2007). Swift: Fast, reliable, loosely coupled parallel computation. In IEEE Congress on Services, 2007, pp. 199-206. IEEE.

[232] Gehani, A. and Lindqvist, U., (2007). Bonsai: Balanced lineage authentication. In Twenty-Third Annual Conference on Computer Security Applications (ACSAC), 2007, pp. 363-373. IEEE.

[233] Groth, P., Miles, S. and Moreau, L., (2005). Preserv: Provenance recording for services. In Proceedings of the UK OST e-Science Second All Hands Meeting 2005 (AHM'05), http://www.ecs.soton.ac.uk/~lavm/papers/Groth-AHM05.pdf, [Accessed 06-28-2017]

[234] Simmhan, Y.L., Plale, B. and Gannon, D., (2008). Karma2: Provenance management for data-driven workflows. Int'l J. Web Services Research, vol. 5, no. 1, 2008.

[235] Anand, M.K., Bowers, S., McPhillips, T. and Ludäscher, B., (2009). Efficient provenance storage over nested data collections. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009, pp. 958-969. ACM.

[236] Green, T.J., Karvounarakis, G., Ives, Z.G., Tannen, V., (2010). Provenance in orchestra. IEEE Data Eng. Bull. 2010, 33(3), pp. 9–16. IEEE.

[237] Zhou, W., Ding, L., Haeberlen, A., Ives, Z.G. and Loo, B.T., (2011). TAP: Time-aware Provenance for Distributed Systems. Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance, 2011.

[238] Gessiou, E., Pappas, V., Athanasopoulos, E., Keromytis, A. and Ioannidis, S., (2012). Towards a universal data provenance framework using dynamic instrumentation. Information Security and Privacy Research, 2012, pp.103-114.

[239] Coekaerts, W., (2011). Trying out DTrace, Oracle blog, https://blogs.oracle.com/wim/trying-out-dtrace,[Accessed 06-28-2017]

[240] Malik, T., Nistor, L. and Gehani, A., (2010). Tracking and sketching distributed data provenance. InIEEE Sixth International Conference on e-Science (e-Science), 2010, pp. 190-197. IEEE.

[241] Widom, J., 2004. Trio: A system for integrated management of data, accuracy, and lineage. In Proceedings of CIDR, 2004.

[242] Wylot, M., Cudre-Mauroux, P. and Groth, P., (2014). Tripleprov: Efficient processing of lineage queries in a native rdf store. In Proceedings of the 23rd international conference on World Wide Web, 2014, pp. 455-466. ACM.

[243] Apache Hive, https://hive.apache.org/,[Accessed 06-28-2017]

[244] Language Manual – Apache Hive, (2017), https://cwiki.apache.org/confluence/display/Hive/LanguageManual, [Accessed 06-28-2017]

[245] Cheah, Y.W. and Plale, B., (2012). Provenance analysis: Towards quality provenance. In IEEE 8th International Conference on E-Science (e-Science), 2012, pp. 1-8. IEEE.

[246] Frew, J., Metzger, D. and Slaughter, P., (2008). Automatic capture and reconstruction of computational provenance. Concurrency and Computation: Practice and Experience, 2008, 20(5), pp.485-496.

[247] Xie, Y., Feng, D., Tan, Z., Chen, L., Muniswamy-Reddy, K.K., Li, Y. and Long, D.D., (2012). A hybrid approach for efficient provenance storage. In Proceedings of the 21st ACM international conference on Information and knowledge management, 2012, pp. 1752-1756. ACM.

[248] Chapman, A.P., Jagadish, H.V. and Ramanan, P., (2008). Efficient provenance storage. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 993-1006. ACM.

[249] Braun, U., Garfinkel, S., Holland, D.A., Muniswamy-Reddy, K.K. and Seltzer, M.I., (2006). Issues in automatic provenance collection. International Provenance and Annotation Workshop (IPAW), 2006, 6, pp.171-183.

[250] Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J. and Silva, C., (2008). Tackling the provenance challenge one layer at a time. Concurrency and Computation: Practice and Experience, 2005, 20(5), pp.473-483.

[251] Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J. and Miles, S., (2013). Prov-dm: The prov data model, http://www.w3.org/TR/prov-dm/, [Accessed 06-28-2017]

[252] Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J. and Plale, B., (2011). The open provenance model core specification (v1. 1). Future generation computer systems, 2011, 27(6), pp.743-756.

[253] Buneman, P., Khanna, S. and Tan, W.C., (2001). Why and where: A characterization of data provenance. In ICDT, 2001, Vol. 1, pp. 316-330.

[254] Agarwal, R., (2014). Provenance in Big Data, NIST Big Data Working Group, https://bigdatawg.nist.gov/2014_IEEE/03_04_NIST_BDWS_ProvenancePresentation.pdf, [Accessed 06-28-2017].

VITA

Ashwin Kumar Thandapani Kumarasamy

Candidate for the Degree of

Doctor of Philosophy

Thesis: CONTENT SENSITIVITY BASED ACCESS CONTROL MODEL FOR BIG DATA

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma in 2017.

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in 2013.

Completed the requirements for the Bachelor of Technology in Information Technology at Anna University, Chennai, India in 2010.

Experience:

Graduate Teaching Associate/Assistant (Aug 2014 – May 2017)
Oklahoma State University, Stillwater, OK

Graduate Research Assistant (Jan 2012 – May 2014)
Oklahoma State University, Stillwater, OK

Professional Memberships:

IEEE Student Member
ACM Student Member