An Access Control Scheme for Big Data Processing

Vincent C. Hu, Tim Grance, David F. Ferraiolo, D. Rick Kuhn
National Institute of Standards and Technology
Gaithersburg, MD, USA
vhu, grance, dferraiolo, kuhn@nist.gov

***Abstract*— Access Control (AC) systems are among the most critical of network security components. A system's privacy and security controls are more likely to be compromised due to the misconfiguration of access control policies rather than the failure of cryptographic primitives or protocols. This problem becomes increasingly severe as software systems become more and more complex, such as Big Data (BD) processing systems, which are deployed to manage a large amount of sensitive information and resources organized into a sophisticated BD processing cluster. Basically, BD access control requires the collaboration among cooperating processing domains to be protected as computing environments that consist of computing units under distributed AC management. Many BD architecture designs were proposed to address BD challenges; however, most of them were focused on the processing capabilities of the "three Vs" (Velocity, Volume, and Variety). Considerations for security in protecting BD are mostly ad hoc and patch efforts. Even with some inclusion of security in recent BD systems, a critical security component, AC (Authorization), for protecting BD processing components and their users from the insider attacks, remains elusive. This paper proposes a general purpose AC scheme for distributed BD processing clusters.**

Keywords—Access Control, Authorization, Big Data, Distributed System

I. INTRODUCTION

Data IQ News [1] estimates that the global data population will reach 44 zettabytes (1 billion terabytes) by 2020. This growth trend is influencing the way data is being mass collected and produced for high-performance computing or operations and planning analysis. Big Data (BD) refers to large data that is difficult to process by using a traditional data processing system, for example, to analyze Internet data traffic, or edit video data of hundreds of gigabytes. (Note that each case depends on the capabilities of a system.I It has been argued that for some organizations, terabytes of text, audio, and video data per day can be processed, thus, it is not BD; but for those organizations that cannot process efficiently, it is BD [2]). BD technology is gradually reshaping current data systems and practices. Government Computer News [3] estimates that the volume of data stored by federal agencies alone will increase from 1.6 to 2.6 petabytes within two years, and U.S. state and local governments are just as keen on harnessing the power of BD to boost security, prevent fraud, enhance service delivery, and improve emergency response. It is estimated that successfully leveraging technologies for BD can reduce the IT cost by an average of 48 % [4].

BD has denser and higher resolutions such as media, photos, and videos from sources such as social media, mobile applications, public records, and databases; the data is either in static batches or dynamically generated by machine and users by the advanced capacities of hardware, software, and network technologies. Examples include data from sensor networks or tracking user behavior. Rapidly increasing volumes of data and data objects add enormous pressure on existing IT infrastructures with scaling difficulties such as capabilities for data storage, advance analysis, and security. These difficulties result from BD's large and growing files, at high speed, and in various formats, as is measured by: Velocity (the data comes at high speed, e.g., scientific data such as data from weather patterns.); Volume (the data results from large files, e.g., Facebook generates 25 TB of data daily.); and Variety (the files come in various formats: audio, video, text messages, etc. [2]). Therefore, BD data processing systems must be able to deal with collecting, analyzing, and securing BD data that requires processing very large data sets that defy conventional data management, analysis, and security technologies. In simple ways, some solutions use a dedicated system for their BD processing. However, to maximize scalability and performance, most BD processing systems apply massively parallel software running on many commodity computers in distributed computing frameworks that may include columnar databases and other BD management solutions [5].

Access Control (AC) systems are among the most critical of network security components. It is more likely that privacy or security will be compromised due to the misconfiguration of access control policies than from a failure of a cryptographic primitive or protocol. This problem becomes increasingly severe as software systems become more and more complex such as BD processing systems, which are deployed to manage a large amount of sensitive information and resources organized into a sophisticated BD processing cluster. Basically, BD AC systems require collaboration among corporate processing domains as protected computing environments, which consist of computing units under distributed AC management [6].

Many architecture designs have been proposed to address BD challenges; however, most of them have been focused on the processing capabilities of the "three Vs" (Velocity, Volume, and Variety). Considerations for security in protecting BD AC are mostly ad hoc and patch efforts. Even with the inclusion of some security capabilities in recent BD systems,

practical AC (authorization) for BD processing components is not readily available.

This paper proposes a general AC scheme for distributed BD processing clusters. Section II describes current BD tools and implementations. Section III discusses BD AC requirements. Section IV introduces related work. Section V illustrates our BD AC scheme. Section VI discusses implementation considerations for the general BD model. Section VII concludes the paper.

II. General Big Data Model and Example

The fundamental model for most of the current BD architecture designs is based on the concept of distributed processing [2, 4], which contains a set of generic processing systems as shown in Figure 1:

1. **Master System** (**MS**) receives data from BD data source providers, and determines processing steps in response to a user's request. MS has the following three major functions:
   - **Task Distribution** (**TD**) function is responsible for distributing processes to the cooperated (slave) systems of the BD cluster.
   - **Data distribution** (**DD**) function is responsible for distributing data to the cooperated systems of the BD cluster.
   - **Result Collection** (**RC**) function processes, collects, and analyzes information provided by cooperating systems of the BD cluster, and generates aggregated results to users.

   Unless restricted by specific applications, the three functions are usually installed and managed in the same host machine for easy and secure management and maintenance.

2. **Cooperated System** (**CS**) (or slave system) is assigned and trusted by the MS for BD processing. The CS reports progress or problems to the TD and DD; otherwise, the CS returns the computed result to the RC of the MS.

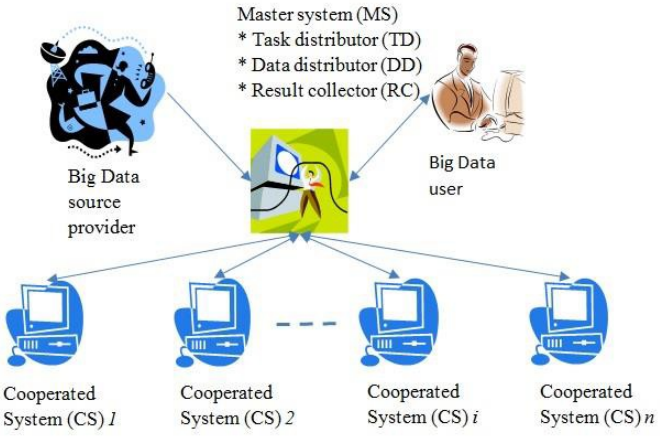

Fig. 1. General BD model

We define a **BD Cluster** as a BD distributed system that networks a MS and CSs to serve BD users' requests to process BD source data. The model in Figure 1 represents a generic distributed BD architecture such as the Apache Foundation's open source software Hadoop [7] (Figure 2), which is used throughout industry and government. Hadoop keeps data and processing resources in close proximity within the cluster. It runs on a distributed model composed of numerous low-cost computers (e.g., **Linux**-based machines with simple architecture): two main MS components: TD – **MapReduce** and DD - **File System** (HDFS-Hadoop file system), and a set of tools. The TD provides distributed data processing across the cluster, and the DD distributes large data sets across the servers in the cluster [3]. Hadoop's CSs are called **Slaves**, and each has two components: **Task Tracker** and **Data Node**. The MS contains two additional components: **Job Tracker** and **Name Node**. Job Tracker and Task Tracker are grouped as MapReduce, and Name Node and Data Node fall under HDFS.

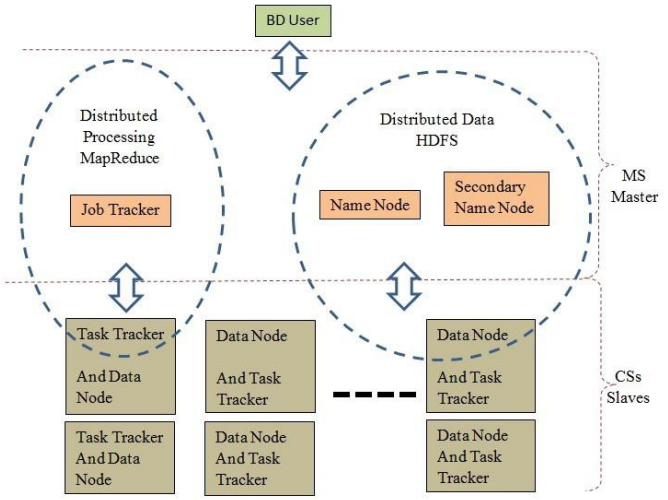

Fig. 2. Hadoop BD cluster example

Hadoop combines storage, servers, and networking to break data as well as computation down to small pieces. Each computation is assigned a small piece of data, so that instead of one big computation, numerous small computations are performed much faster, with the result aggregated and sent back to the application [2]. Thus, Hadoop provides linear scalability, using as many computers as required. Cluster communication between the MS and CSs manages what traditional data service systems would not be able to handle.

III. Big Data Access Control Challenges

Enterprises want the same security capabilities for BD as are in place for "non-BD" information systems, including user authentication and authorization (AC). According to [4], the biggest challenge in deploying BD technologies is security (50 % of those surveyed), and the biggest challenge working with and leveraging technologies for BD data is to maintain data security (47 % of those surveyed). One of the fundamental security techniques is AC policy enforcement and management [8], which allows organizations to safeguard their BD in order to meet security and privacy mandates. However, the three Vs of data are overwhelming for existing system models, which were not designed and built with AC capability in mind [9]. Thus, most of them fail to adequately manage the creation, use, and dissemination of BD data and processes. As a result, they either introduce friction into collaboration through excessively

strict rules, or risk serious data loss by sharing data too permissively [5].

Authentication is different from authorization, as distinguished in [10]; the authentication management function is not directly related to the data content. For BD, as for non-BD data systems, authentication is generally handled by MS and CSs independently. The focus of our BD scheme is on authorization (AC), which is more complex than non-BD systems, because of the need to synchronize access privileges between the MS and CSs.

BD AC must not only enforce access control policies on data leaving the MS, it must also control access to the CSs' resources. Depending on the sensitivity of the data, it needs to make certain that BD applications, the MS, and CSs have permissions to access the data that they are analyzing, and deal with the access to the distributed BD processes and data from their local users [11]. The characteristics of a BD distributed computing model, as listed below, pertain to a unique set of challenges for BD AC, which requires a different set of concepts and considerations.

Like Hadoop, support for the BD's three V features complicates a system's AC implementation, because the difficulties are in general handled by the following techniques, each with its security challenge.

• Distributed computing – BD data is processed anywhere resources are available, enabling massively parallel computation between the MS and CSs. This creates complicated environments that are highly vulnerable to attack, as opposed to the centralized repositories that are monolithic and easier to secure.

• Fragmented/redundant data - Data within BD clusters is fluid, with multiple copies moving to and from the MS and CSs to ensure redundancy and resiliency. Data can become sliced into fragments that are shared across the MS and CSs. This fragmentation adds complexity to the data integrity and confidentiality.

• Node-to-node communication – the MS and CSs usually communicate through insecure protocols such as RPC over TCP/IP [9].

IV. Related Work

Tools and techniques for BD AC should protect BD processes and data, ensuring that security policies are enforced in a cost-effective and timely manner. Currently, only a few approaches that address the unique architecture of distributed computing can meet the security requirements of BD AC. Some provide an enterprise-class security solution by generally applying traditional perimeter security solutions for a control point (gateway/perimeter such as firewalls and intrusion detection/prevention technologies) where data and commands enter the MS or CS. But traditional approaches that rely on perimeter security are unable to adequately secure a BD cluster. For example, firewalls attempt to map IP address to actual AD (Active Directory) credentials, but this is problematic in the BD cluster, because it requires a specific network design (i.e., no Network Address Translation (NAT) from internal corporate sub-nets). Even with special network configuration, a firewall can only restrict access on an IP/port basis, and knows nothing of the architecture of the BD cluster. As a result, security administrators have to segregate sensitive data on separate servers in order to control access. It would require the creation of a second BD cluster to contain sensitive data, and even then would only provide two levels of security for the data. Even without those drawbacks, perimeter security solutions represent a single layer of defense around a soft interior; for example, once a firewall is breached, the system is wide open for attack [9].

Hadoop, like many open source technologies, was not created with security in mind. It uses the MapReduce facility and a distributed file system with no built-in security. The Hadoop community realized that more robust security controls were needed, and decided to focus on security by applying technologies including Kerberos, firewalls, and basic HDFS permissions [9, 11]. The Kerberos implementation utilized the token-based framework to support a flexible authorization enforcement engine that aims to replace (but be backwards compatible with) the current AC Lists (ACLs) approaches for AC, thus to support an advanced authorization model, focusing on Attribute Based AC (ABAC) and the eXtensible Access Control Markup Language (XACML) standard. However, Kerberos is difficult to install and configure on the MS and CSs, and to integrate with Active Directory (AD) and Lightweight Directory Access Protocol, (LDAP) services. A malicious developer could easily write code to impersonate users' Hadoop services (e.g., writing a new TaskTracker and registering itself as a Hadoop service, or impersonating the HDFS or mapped users, deleting everything in HDFS, etc.). In addition, DataNodes enforce no AC; a malicious user could read arbitrary data blocks from DataNodes, bypassing AC restrictions, or write garbage data to DataNodes, undermining the integrity of the data to be analyzed. Further, anyone could submit a job to a JobTracker and it could be arbitrarily executed.

Some components of the Hadoop ecosystem have applied their own security as a layer over Hadoop; for example, Apache Accumulo [12] provides cell-level authorization, and HBase [13] provides AC at the column and family level [11]. Some of them configured Hadoop to perform AC based on user and group permissions by ACLs, but this may not be enough for every organization, because many organizations use flexible and dynamic AC policies based on security attributes of users and resources and business processes, so the ACL approach is certainly limited [11].

For decades, relational, or SQL-based databases, have been the database schema of choice to store and manage data; such databases allow data to be stored by predefined schema such as Relational database management system (RDBMS)'s (row and column in table format. SQL-based databases support AC on data queries by assigning columns or rows with security attributes so that they conform to the Attribute-Based AC (ABAC) [10] model that is central to many database security frameworks. But in the era of BD, the traditional database model has difficulty dealing with the multitude of unstructured data types, as well as the massive amounts of data that must be stored, managed, and manipulated. Many applications employ NoSQL [14] for handling unstructured, messy, and unpredictable data. NoSQL encompasses a wide variety of different database technologies

that were developed in response to a rise in the volume of data stored. They are built to allow the insertion of data without a predefined schema; NoSQL taxonomy supports key-value stores, document store, BigTable, and graph databases. However, it is useful when constraints and validation logic are not required to be implemented in a database. Thus, most BD environments only offer AC at the schema level, with no finer granularity to address attributes of users and resources. Therefore, if the AC for a BD data query is based on the structural attributes of BD data, then NoSQL needs to support the capability to create and manage AC information; such a capability may require an application layer on top of the existing NoSQL mechanisms [15].

V. General Big Data Access Control Scheme

Fundamentally, the AC requirements for a BD cluster are no different than non-BD systems. However, due to the facts that (1) BD is processed by distributing its processes and data from MS to CSs, and (2) BD data has no formal scheme for database management, BD AC needs additional AC capabilities than non-BD systems. A BD cluster is a construct of an enterprise system that requires the MS's AC mechanism to be incorporated with CSs'. In terms of AC privilege as defined in [10], when the MS passes the process/data to a CS, the MS is the **Subject**, the BD process/data and required CS local resources are the **Objects**, and the required actions are the **Actions** in the CS's AC policy. Figure 3 shows our proposed AC scheme based on the general BD model described in Section 3. The scheme includes AC components to meet the BD AC requirements as described below.

Security Agreement (SA) is a mutual agreement between a BD source provider and the MS for defining security classes of BD sources. The purpose of SA is for the BD source provider and the MS to define and agree upon security classes (ranks), so that the security classes can be referred to by the MS and CSs to decide the levels of security (or trust) that a CS is qualified for processing. For instance, a BD source may be an email log file, and in considering the confidential level, the log can be accessed only by a CS with security class say from 1 to 3.

Trust CS List (TCSL) lists the trusted CSs recognized by the MS. The TCSL categorizes CSs by the security classes according to the SAs worked with BD source providers. TCSL is managed by the MS security officer based on its knowledge of the associated CSs. For example, *CS-i* is assigned to class 1, *CS-j* and *CS-k* are assigned to class 2, and *CS-l* is assigned to class 3. In other words, TCSL allows the MS to determine how CSs are trusted for the distributions of BD processes and data.

MS AC Policy (MSP) is managed by the MS security officer. MSP specifies a set of AC rules that are imposed by the MS to enforce AC on CSs. For example, the distributed BD data can be *read* only by subjects with attribute *company employee*, or the BD process can be executed only by processes with subject attribute *system administrator* in a CS.

CS AC Policy (CSP) is managed by the CS security officer. CSP allows the CS to control the access to the distributed BD data/process by considering the processing capabilities (e.g., system load) and security requirements of the CS. For example, the distributed BD data cannot be written to disk space that is shared by other local CS users that have nothing to do with the BD, or BD data can be printed only from local printers. Additionally, the CSP needs to handle a situation in which AC rules from other CS local policies conflict with CSP rules.

Federated Attribute Definitions (FAD) list the common attributes used by the MS and CSs, so that the MSP and CSPs can be composed using the common attributes in the FAD dictionary. For example, attribute *local user* is defined as all users who can log into the CS system, *company employee* is defined as all the CS users who have company's employee identifications, and *system administrators* is defined as the CS user who has system administrator privilege on the CS system. So, the FAD serves as the federated dictionary of AC attributes that should be syntactically and semantically agreed to by the MS and CSs.

To apply the scheme, the following tasks need to be performed:

- Coordinate BD source providers and the MS for SA agreements;
- The MS collects information about CSs based on the knowledge of CSs' security capabilities, levels of assurances, or trust. The information is required for the MS to define security classes in sync with BD source providers for SA;
- Coordinate the MS and CSs to define attributes in FAD based on the common syntactic and semantic values of attributes for the BD processing needs;
- The CS prepares information about how it trusts the MS (i.e., what local resources are available for the BD processing) and considerations for performance and security capabilities of the CS's local system, as well as responsibility for disseminating the distributed BD processes and data. All the information will be translated into CSP rules;
- The CS composes meta-rules to handle conflicts between CS's local AC policies and CSP, as well as between MSP and CSP. Note that unless specifically agreed upon, CSP rules should have higher priority than MSP rules when determining access privileges. If a CS denies a BD process's access, it should notify the MS for transferring the task/data to other CSs; and
- In addition to authorization enforcement, the MS and CSs may consider including AC activity audit capability for BD access logs.

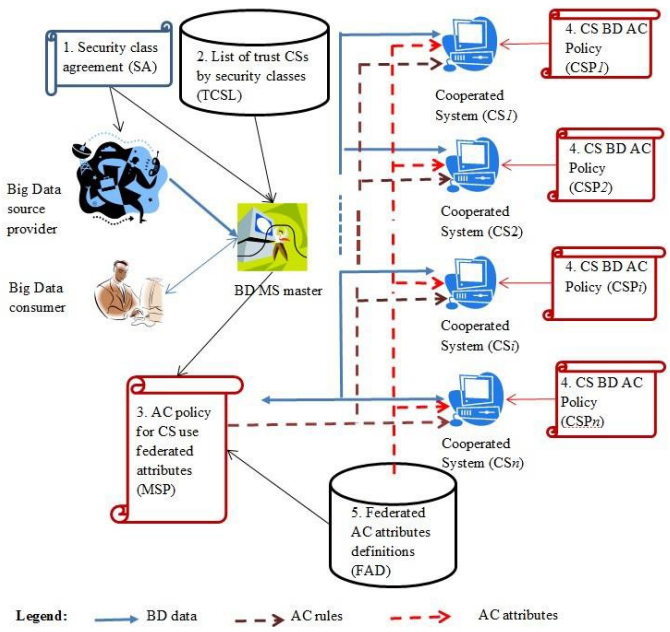

Fig. 3. A generic BD architecture

Formally, the proposed scheme can be represented by:

- A set of security classes from c_1 to c_n in the set $C = \{c_1 \ldots .. c_n\}$.
- A set of BD source providers from bd_1 to bd_n in the set $BD = \{bd_1 \ldots bd_n\}$.
- A set of CSs from cs_1 to cs_n in the set $CS = \{cs_1 \ldots . cs_n\}$.
- A set of federated attributes from at_1 to at_n in the set $FAD = \{at_1 \ldots . at_n\}$.
- A set of (bd_x, c_x) pairs in the set $SA \subset BD \times C$.
- A set of (cs_x, c_x) pairs in the set $TCSL \subset CS \times C$.
- A set of MS policy rules from mp_1 to mp_n in the set $MSP = \{mp_1 \ldots . mp_n\}$, each $mp_i = (at_x, a_x, bd_x)$ is a tuple where, $at_x \in$ FAD is an attributes, a_x is an action, and $bd_x \in BD$ is a source provider that means subject with attribute at_x is permitted to perform action a_x on object from bd_x.
- A set of AC policy rules for CS *CSi* in the set $CSPi = \{cspi_1 \ldots . cspi_n\}$, each $cspi_i = (bd_x, a_x, rs_x)$ is a tuple where $bd_x \in BD$, a_x is an action, and rs_x is a local resource from *CSi* that means subject from bd_x is permitted to perform action a_x on object rs_x.

Let the $BDU = (u, a_u, bd_u)$ represent a BD user request; where u is an authenticated BD user by the MS, a_u is a requested action, and bd_u is a BD source provider of the data/process that u is requesting to perform a_u from. The AC algorithm to accept (u, a_u, bd_u) on the CS cs_l is:

BDAC {

$C_u = \{c_1 \ldots .. c_k\}$ **such that** $(bd_u, c_i) \in SA$;

if $C_u = \varnothing$ {

request = *deny* /* for this cs_l

else

$CS_u = \{cs_1 \ldots . cs_k\}$ **such that** $(cs_i, c_i) \in TCSL$ and $c_i \in C_u$;

if $CS_u = \varnothing$ **or** $cs_l \notin CS_u$ {

request = *deny* /* for this cs_l

else

if (**there exist** $(mp_x = (at_x, a_x, bd_x) \in MSP$ **such that** $a_u = = a_x$ **and** $bd_u = = bd_x)$) **and** (**there exist** $(csp_x = (bd_x, a_x, rs_x) \in CSP_l$ **such that** $a_u = = a_x$ **and** $bd_u = = bd_x$ **and** $rs_x = =$ *resource required*)) {

request = *grant* /* for this cs_l ;

if *perform* a_u *on* $rs_x = =$ *success* {

return *result* to RC

else

return "RC resource from cs_l unavailable"

}

else

request = *deny* /* for this cs_l

}

}

}

}

The following demonstrates an example of the algorithm. The BD source provider *x* enforces security *class 2* as defined in SA. *Class 2* is a level of moderate trust by some mutual criteria between *x* and the MS. The MS sets up MSP and regulates that only *system administrator* can process *x*'s BD after the subject attribute: *system administrator* is syntactically and semantically agreed upon by the MS and participating CSs. The MS also assigns *CS-i*, *CS-j*, and *CS-k* (assuming the higher the number, the higher the security classes) to security classes greater than or equal to *class* 2 in the TCSL. Assume *CS-i*'s local AC policy does not allow processing nonlocal data, as well as *CS-i*'s CSP gives local policy higher priority than MSP. And *CS-j* does not give any *system administrator* privilege to nonlocal process; thus, only *CS-k* can process the request for *x*'s BD.

Figure 4 depicts the BD AC **control/manage domains** where the MS AC domain is enforced collectively by information in the SA, TCSL, MSP, and FAD entries, which are all configured and managed by the MS. The CS AC domain is enforced collectively by information in MSP, FAD, and CSP entries;

however, only CSP entries are configured and managed by the CS. Note that at_x in the FAD and cs_x in TCSL are used to determine if a CS is permitted to process the BD process request. The cs_x is independently decided by the MS, but at_x needs to share the responsibilities with trusted CSs for access privilege decisions. Thus, the *chain of trust* is from BD source provider to MS through SA, then from the MS to the CS through TCSL, MSP and FAD, also from the CS to the MS through FAD and CSP which shows that both MS's and CS's AC domains do not allow unauthorized access without the coordination between the MS and CSs.

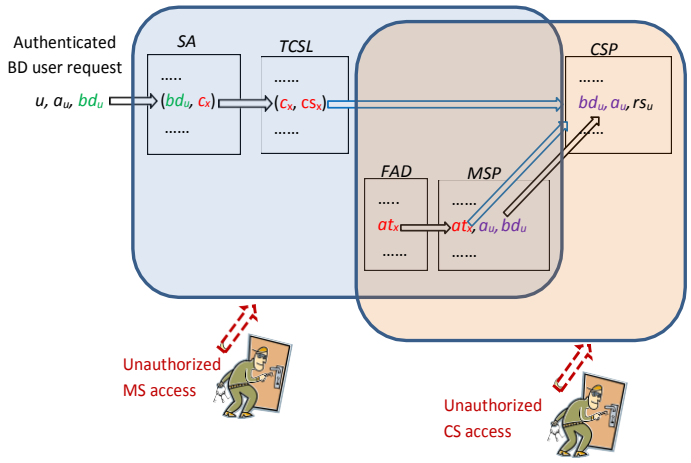

Fig. 4. BD AC control/manage domain.

Note that to be general, we provide a core concept of the scheme. For practical implementations, additional sets and rules for the algorithm that cover other system components such as backup, logs, etc., might need to be included.

VI. Implementation Considerations

In addition to the scheme, operational considerations that are common for other enterprise BD AC mechanisms also need to be addressed as below.

Trust establishment – For better performance, the MS might need to separate parallel TDs and DDs functions (e.g., Hadoop) that process immense volumes of BD. In this case, TDs and DDs should also protect the BD data from an untrusted CS, especially when their processes are delegated in a virtual environment such as that of a **Cloud**. For example, some suggest ensuring the trustworthiness of CSs by Mandatory AC (MAC) mechanisms so that a CS must be authenticated and given properties by MS, and only when they're competent can they be assigned CS tasks. After this qualification, periodic updates must be made to ensure CSs consistently meet established policies [16].

Content Attributes -- If instead of SAs, the security classes for MSP are dynamically determined by referring attributes from the BD sources contents [15], then the issue would be that as data grows in volume and complexity, it becomes harder to retrieve required attributes, for example, by traditional database queries or analytic tools. This is because established procedures for manipulating data typically rely upon imposition of a rigid structure or schema (defining elements of the data to be a location, a device type, a zip code, a user ID, etc.) and the pre-calculation of indexes to recall specific data fields (attributes), such as used in Remote Database Administration(RDBA) or NoSQL. These approaches continue to excel in managing highly structured data but are less well suited for BD [5].

AC auditing -- The nature of distributed processing in BD clusters poses a challenge for AC activity auditing, which tends to be incapable of analyzing the interconnected data between the MS and CSs, which are responsible for tracking the actual interactions with files and resources spread across their operating systems and applications. To minimize the problems, organizations may want to implement auditing functions in the infrastructure layer, simplify complexity in the application space, and adopt authentication and MAC. However, scaling these technologies to the levels necessary to accommodate BD can present its own set of unique challenges [5, 16].

Combining with Cloud computing environment -- With the emergence of Cloud platforms, many might consider processing BD in a Cloud environment to handle the growing volume and complexity of BD. However, Cloud computing has both positive and negative effects on the BD. As data becomes more accessible through the Cloud, there are three major threats to securing BD processing: malfunctioning infrastructure components, infrastructure outside attacks, and infrastructure inside attacks. To address these threats, the MS should improve trustworthiness and the usability of CSs by strengthening the MSP and fine-grained FAD.

Many AC models and mechanisms can be applied to support the proposed scheme. One of the most versatile is the Attribute-Based Access Control (ABAC) model, because of its capabilities in configuring and managing attributes and AC policies. It also supports the AC requirements of enterprise systems, where BD is usually serviced as described in [10], which provides guidance for ABAC enterprise implementation by the **initiation, acquisition and development**, **implementation and assessment**, and **operations and maintenance** phases of an enterprise. The guidance may also be applied to BD AC implementation.

VII. Summary and Conclusions

To harvest BD benefits, security challenges must be overcome. Security professionals apply most controls at the very edges of the network. However, if attackers penetrate the BD cluster, they will have full and unrestricted access to the BD [9]. Thus, many organizations are being required to enforce AC and privacy restrictions on BD to meet regulatory requirements. In this paper, we presented an AC scheme for BD data processing in a distributed processing environment.

We introduced the definition of BD, illustrated a general BD process model abstracted based on general distributed processing environment, and used the popular BD process application Hadoop as an example. We discussed BD AC issues with related work. Finally, we presented the BD AC scheme and considerations based on trust between BD source providers and the BD Master System (MS), and consequently, between the MS and Cooperating Systems (CSs). The scheme is focused on authorization (access control) to protect BD processing and data from insider attacks in the BD cluster, under the assumption that authentication is already established. In addition to AC components, we demonstrated the formal sets and algorithm, and the domain of protection for the scheme that showed that no unauthorized privileges either from the MS or CSs are possible.

Depending on the security requirements of the BD application, many AC models and mechanisms can be applied to the scheme. The proposed scheme is devised to be generally applicable for distributed system-based BD AC, and stems from the fundamental AC mechanism that uses attributes of subjects, objects, actions, and sometimes environment conditions which are the building blocks of Attribute-Based AC (ABAC) mechanisms, to determine access permission. We further discussed some operational and implementation issues for practical application. These issues are tied to the BD application, and should be handled according to the BD security requirements.

REFERENCES

[1] "Big data to turn 'mega' as capacity will hot 44 zettabytes by 2020,", DataIQ News, http://www.dataiq.co.uk/news/20140410/big-data-turn-mega-capacity-will-hit-44-zettabytes-2020, Oct. 2014.

[2] H. Mir, "Hadoop Tutorial 1What is Hadoop," ZeroToProTraining, http://ZeroTOProTraining.com http://nusmv.irst.itc.it/.

[3] J. Moore, "How big data is remaking the government data center," GCN, http://gcn.com/articles/2014/02/14/big-data-data-centers.aspx, Feb. 2014.

[4] W. Bell, "The Big Data Cure," MeriTalk, http://www.meritalk.com/bigdatacure, 2014.

[5] P. Miller, "Applying big data analytics to human-generated data," GIGAOM RESEARCH, http://research.gigaom.com/report/applying-big-data-analytics-to-human-generated-data/, Jan. 2014.

[6] V. Hu, R. Kuhn, T. Xie, and J. Hwang, "Model Checking for Verification of Mandatory Access Control Models and Properties," International Journal of Software Engineering and Knowledge Engineering (IJSEKE) regular issue IJSEKE Vol. 21, No. 1., 2011.

[7] Hadoop.apache.org

[8] V. Hu and K. Scarfone, "Guidelines for Access Control System Evaluation Metrics," NIST Interagency Report 7874, Gaithersburg, MD, USA, 2012.

[9] "The Big Data Security Gap: Protecting the Hadoop Cluster," White Paper, Zittaset, http://www.zettaset.com/wp-content/uploads/2014/04/zettaset_wp_security_0413.pdf, 2014.

[10] V. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone…., "Attribute Based Access Control Definition and Consideration," NIST Special Publication 800-162, Gaithersburg, MD, USA, 2013.

[11] K.. T. Smith, "Big Data Security: The Evolution of Hadoop's Security Model," InfoQ, http://www.infoq.com/articles/HadoopSecurityModel, Aug. 2014.

[12] "Apache Accumulo,"https://accumulo.apache.org

[13] Hbase.apache.org

[14] "NoSQL Databases Explained," mongoDB Inc., http://www.mongodb.com/nosql-explained, 2014.

[15] W. Zeng, Y, Yang, B. Luo, "Access Control for Big Data using Data Content," in Proc. 2013 IEEE International Conference on Big Data, 2013.

[16] S. Shea, "CSA top 10 big data security, privacy challenges and how to solve them," TechTarget, SearchSecurity, Nov. 2013.