Access Control for Distributed Processing Systems: Use Cases and General Considerations

Ang Li†, Qinghua Li†, Vincent C. Hu§
†Department of Computer Science and Computer Engineering, University of Arkansas
§Computer Security Division, National Institute of Standards and Technology
†{angli, qinghual}@uark.edu, §vincent.hu@nist.com

***Abstract*—Access control (AC) is critical for preventing sensitive information from unauthorized access. Various AC systems have been proposed and enforced in different types of information systems (e.g., bank and military). However, existing AC systems cannot thoroughly address the challenges in emerging distributed processing systems (DPS), such as Big Data (BD) and Cloud, due to their dynamic and complex architecture. Generally, AC for DPS needs to consider the protection for collaboration among distributed processing domains. Even though some DPS architectures were proposed to address DPS challenges, most of them only focus on processing capabilities without consideration of AC. Even with some inclusion of security in recent DPS, they are mostly ad hoc and patch efforts. In this paper, we analyze the general features and use cases of BD and Cloud, which are two of most widely applied DPS applications, and propose a set of general and comprehensive considerations for AC in DPS, which can provide a guideline for designing AC systems for DPS.**

***Index Terms*—Access Control, Distributed Processing System, Design Considerations**

I. Introduction

Distributed Processing Systems (DPSs) are computing systems where multiple computers or processors connected through networks cooperatively work together to provide computing services. In this paper, DPSs refer to modern distributed processing systems (in comparison with traditional distributed model) whose most distinguished features are dynamic resource allocation and relocation. A typical DPS contains a Master System (MS; i.e., application provider) and many Slave Systems (SS; i.e., computing nodes or computing service providers) that collaboratively fulfill computing tasks (see Figure 1). Master System and Slave Systems may be in the same administration domain, but they may also be in different administration domains. For example, the manager of an online supermarket (e.g., Amazon) may want to query which product makes the largest contribution to revenue from billions of transaction records for the supermarket in last year. The records were separately stored in different SSs. The manager sends a request to MS, which distributes the computing tasks to different SSs that serve as distributed processors. SSs carry out computations to sum the revenue generated by each product based on local transaction records, and return the results to the MS. Finally, the MS collects and analyzes the results from SSs, and returns the aggregate result to the user. DPS can be implemented in many systems, such as Big Data (BD), Cloud, Mobile Crowdsourcing, Internet of Things (IoT) [1]–[3] and Grid Computing [4] that benefit many sectors such as business, finance, and healthcare.

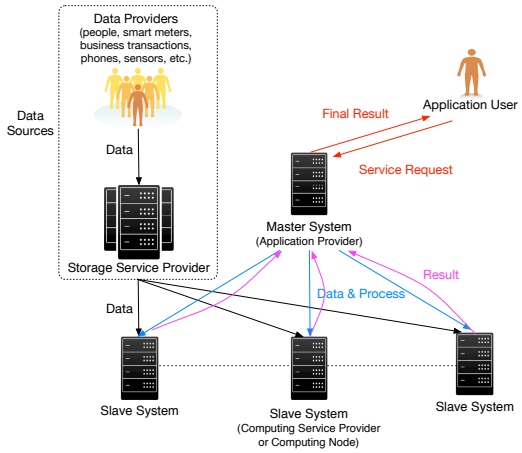

Fig. 1. A General DPS Model

Although distributed computing has been around for decades, it has gained many new features as well as technological breakthroughs in recent years driven by the emergence and proliferation of applications such as Cloud and BD. Traditional distributed systems usually use static job distribution, while modern DPSs are dynamic in nature to be adaptive to the ever-changing workload and available resources. The data input to DPSs can be directly from data providers such as mobile devices and sensors, or from Cloud-based storage service providers. An example of DPS is the Apache Hadoop [5] which uses MapReduce and Hadoop Distributed File System (HDFS) for task and data distribution.

In spite of DPS's processing features, data providers have concerns about data misuse and want to govern the private-and public-sector use of their data. Master and slave systems also want to secure the usage of their resources; thus, security and privacy protection are crucial for DPS. Access Control (AC) systems controls which principals (i.e., processes and users) have access privileges to which resources based on the AC policy. AC has been widely used and well-studied in traditional computing systems; however, DPS poses many challenges unprecedented before [6], [7] to AC. For instance, SSs may reside in different protection domains with their own AC schema and policy over local and shared resources. Thus, how

to coordinate their computing activities that achieve global AC goals without violating local AC policies at each SS is a critical challenge. Also, in DPS, access privileges are by nature not as straightforward or definite actions as commonly seen in traditional computing systems (e.g., "read", "write", and "execute"). Instead, they are sometime vaguer due to possible hidden information flow between data. In the aforementioned example of supermarket, to process the manager's query, SSs need to sum the revenue generated from each product based on their local transaction records, but the returned results may leak information about its local data. Another challenge is that DPS's AC system requires high efficiency, short response time, flexibility, and reconfigurability to adapt to Volume, Velocity, and Variety (for example, for BD) of DPS.

Most of the frameworks for DPS such as Hadoop are not developed with AC in mind [8]. In recent years, some work have been done in the literature about AC in Big Data and Cloud systems [6], [9]–[25]; however, they are mostly ad hoc and patch efforts, which do not provide comprehensive and systematic approaches for DPS AC. A comprehensive solution for AC in DPS is still elusive. Thus, there is a necessity to systematically rethink the design of AC for DPS from the fundamental properties of subjects, objects, and actions, to the design of AC policy and policy enforcement mechanism.

In this paper, we develop a comprehensive set of general considerations for DPS AC to facilitate design and deployment of AC in DPS. The rest of paper is organized as follows. Section II analyzes the characteristics of DPS that should be considered in the design of AC policies/models. Section III presents some real-world BD and Cloud AC cases. Section IV analyzes the new features of AC subjects, objects, and actions within the context of DPS and their impacts on the design of AC policy. Section V reviews related work. Section VI concludes this paper.

II. Characteristics of DPS's AC

Before proposing general AC considerations for DPS, we need to analyze the characteristics of DPS, which affect the DPS's AC design and deployment. Some key features of DPS [26], [27] with respect to their impact on AC are summarized as follows:

- **Volume** describes how much data is coming in. For example, in the context of BD, this typically ranges from gigabytes to exabytes and more [28]. Consequently, such huge volume of data is usually disseminated to distributed nodes (e.g., Slave Systems) for processing, and then Master System aggregates the result from each Slave System and delivers it to the user. Note that computing nodes may belong to different protection domains with their own AC schema and policy over local and shared resources. How to coordinate their computing operations complying to global AC policies without violating their local AC restriction should be carefully planned.
- **Velocity** measures the speed at which data is generated and processed. Usually, data comes in batches or continuous streams. To efficiently handle such rapid arrival of data, the processing capability of DPS is continuously improved, such as BD systems (e.g., MapReduce). Thus, high efficiency, short response time, flexibility and reconfigurability of AC functions need to be considered.
- **Variety** indicates the format of the data: structured, semi-structured, or unstructured. In the era of DPS, one significant shift is that unstructured data is rapidly increasing, such as video, photo and social relationship. Unstructured data types are common in DPS which are not anticipated and hard to protect in traditional systems such as Relational database. To handle the variety of data in DPS, metadata is usually applied for data management. However, metadata usually causes security and privacy leakage (e.g., it may contain when and where the data is generated). Therefore, both data and metadata should be protected.
- **Virtualization** is a key feature of DPS, which is capable of abstracting computing resources as virtual machines (VM) to associated storage and network resources. In a virtual environment, one computing host that previously ran only a single operating system (OS) can now run multiple OSs as virtual machines, which are created rapidly to facilitate resource sharing and rapid elasticity. With virtualization, computing resources (e.g., storage, CPU and bandwidth) are pooled to services. Resources are assigned and reassigned dynamically based on users' demand such that users have little interaction with the underlying infrastructure. Note that a virtualized environment of DPS introduces the problem of covert channel [29], which can leak information from DPS users. For instance, two VMs must be ensured that no application from each VM will leak information to the other virtual machine. Isolation between VMs should be warranted to keep each VM running independently from each other, and resource usage should be protected so that no VM can exhaust computation resources, especially in a distributed environment. If a malicious application consumes most computation resources of some computing nodes, then other applications cannot obtain sufficient resources and their performance will be seriously affected. Another consideration is that a user might pause the execution of a process before it is finished. Then the state and data of this VM will be saved as a guest OS image. When the process is continued later, the VM may have been migrated from the previous computing node to another node, which means the guest OS image has been moved as well. Thus, guest OS images and related storage must be protected from unauthorized access and tampering.
- **Rapid Elasticity** is the ability to add or remove capacity (e.g., computing resources) from DPS when needed so that computation capabilities can be rapidly scaled up and down to the effect that computing resources seem infinite for VM users. This can be done in two ways: nodes and servers can be added to or removed; resources (e.g., memory and storage) can be added to or removed from existing nodes or servers. How to rapidly verify the

security of new computing nodes and determine whether the newly added nodes are qualified to execute a specific task is a challenge.

- **Distributed Collaboration**: DPS offers services to users by integrating multiple services either from the same DPS or from other DPSs for efficient collaboration. Both resource sharing within the same DPS and resource sharing among different DPSs should be protected by the AC policy of each DPS. For instance, in the same Cloud, suppose a user's processing request is sent to two computing nodes, and one of them denies the request due to some context constraints (e.g., no enough memory) while the other node can take over the processing. How can the system know which candidate computing node is secure enough to inherit the processing? The problem becomes more complicated for multiple DPSs environment. For instance, what available resources of a particular DPS can be exposed to others? This leads to the question of how trust between DPSs should be analyzed,and how AC policies should be correctly coordinated.

.

III. Use Cases

We illustrate some typical applications of DPS to demonstrate the considerations in various application areas including **commercial**, **government** and **healthcare**. Most use cases utilize both BD and Cloud which represent typical applications. For each application area, one use case is described to provide a high-level view of AC requirements.

A. Commercial

Use Case 1: Netflix Movie Service Netflix allows streaming of user-selected movies to satisfy multiple objectives (for different stakeholders)—but with the focus on retaining subscribers [30]. The company needs to find the best possible ordering of a set of videos for each user (e.g., household) within a given context in real time, with the objective of maximizing movie consumption. Recommendation systems and streaming video delivery are core Netflix technologies. Recommendation systems are always personalized and use logistic/linear regression, elastic nets, matrix factorization, clustering, LDA, association rules, gradient-boosted decision trees, and other tools. Digital movies are stored in the cloud with metadata, along with individual user profiles and rankings for small fraction of movies. The current system uses multiple criteria: a content-based recommendation system, a user-based recommendation system, and diversity. Algorithms are continuously refined with A/B testing (i.e., two-variable randomized experiments used in online marketing). Therefore, data should be protected by AC system ranging from video to user rankings and profiles, which are continually updated. The AC system prevents invalid subscribers from watching protected videos (e.g., paid video). Also, user's profile and rankings are also protected by the AC system against privacy leakage.

B. Government

Use Case 2: NIST IT Service Management National Institute of Standards and Technology (NIST) is interested in moving its service ticketing system to the Cloud as part of a larger move to an IT Service Management model for providing services to end users [31]. One of the main drivers for moving the trouble ticket system to the Cloud is to allow IT to focus its resources on applications that directly implement functional aspects of the NIST mission. Moving non-core applications to the Cloud eliminates the need to patch and update software and servers. In this way, a single service request can be routed to appropriate service providers within NIST in a seamless way. The use of a Cloud application would provide flexibility in the timing of deployments and the availability of resources for testing and training. To meet the above presented capability and performance requirements, AC system guarantees the secure virtualization, rapid elasticity of Cloud. Furthermore, in order to fulfill functional demands, AC system protects cooperation of resources between different service groups, namely federated Clouds between groups. In addition, inside the same service group, AC system allows users to access data at different granularities based on their roles and responsibility.

C. Healthcare

Use Case 3: Electronic Medical Records Large national initiatives around health data are emerging. These include developing a digital learning healthcare system to support increasingly evidence-based clinical decisions with timely, accurate, and up-to-date patient-centered clinical information; using electronic observational clinical data to translate scientific discoveries into effective clinical treatments; and electronically sharing integrated health data to improve healthcare process efficiency and outcomes. With these methods in place, feature selection, information retrieval, and enhanced machine learning decision-models can be used to define and extract clinical phenotypes from non-standard, discrete, and free-text clinical data. Clinical phenotype data must be leveraged to support cohort selection, clinical outcomes research, and clinical decision support.

The Indiana Network for Patient Care (INPC) [30], the nation's largest and longest-running health information exchange, houses clinical data from more than 1,100 discrete logical operational healthcare sources. More than 20 TB of raw data, these data describe over 12 million patients and over 4 billion discrete clinical observations. Between 500,000 and 1.5 million new real-time clinical transactions are added every day.

Running on an Indiana University supercomputer, Teradata, PostgreSQL, and MongoDB will support information retrieval methods to identify relevant clinical features (e.g., term frequency-inverse document frequency [tf-idf], latent semantic analysis, and mutual information). NLP techniques will extract relevant clinical features. Validated features will be used to parameterize clinical phenotype decision models based on maximum likelihood estimators and Bayesian networks.

Decision models will be used to identify a variety of clinical phenotypes such as diabetes, congestive heart failure, and pancreatic cancer.

In this case, the AC system prevents such medical records generated in real time from unauthorized access. Such huge volume of medical records as well as the metadata associated with them (such as patient's name, record creation date and type of record) are distributedly stored and protected by the AC system. Additionally, the AC system allows users and clinics to access medical records at various granularities based on the metadata of medical records.

IV. Access Control Considerations of DPS

To comprehensively ensure the AC security of DPS, we need to explore the new features brought by DPS to the AC elements that affect how the AC policy is designed and deployed. In consequence, AC policy design, enforcement and management also need to be analyzed.

A. Considerations of AC Elements

The three basic elements of DPS's AC – **subject**, **object**, and **action** have many new features that are not presented or not prominent in other systems. These features affect how AC should be designed and implemented.

1) Subjects: In DPS, access to object may span across multiple security and administration domains. Therefore, processes activated by an access request may be handled across domains. Thus, subjects are unpredictable to non-local domains due to the dynamic elasticity nature of DPS systems. These features are not generally studied in other systems, which mainly focus on single/static domains with static AC frameworks.

Multiple security/administration domains: DPS users include:

- Data provider which is the creator or source of data, can be individual users (e.g., considering user's ratings for movies in Use Case 2), government (e.g., data in Use Case 1), businesses, mobile devices (e.g., smart phones and smart cars that generate traffic data), smart electricity meters, and so on. Storage service provider, application provider, and computing service provider are now commonly implemented through Cloud-based services, and they comprise a collection of fine-grained subject classes such as system administrator, data analyst, and processes.
- Application provider (i.e., MS) which provides generic computing services such as data analytics.
- Computing service providers (i.e., SS) which provide computing resources for processing data.
- Application users which receive service from the application provider.

Cross-domain processes: Due to the high volume of data in DPS, instead of moving data to processing nodes, software programs may be moved to the data to perform processing there, and then return the results back to the MS. Such moving processes are unprecedented in other AC systems. For instance, as to the trillions of medical records as described in Use Case 3, it is impossible to store and process such high volume of data locally, so those data need to be distributed and processed in DPS. Therefore, a question arises that whether the processes moved to distributed nodes should be considered as local or remote subjects. The approach of cross-domain cooperation proposed in [6], [25] might be applied to deal with this challenge.

Data owner's control: A data provider's data is usually hosted in another domain of DPS environment such as Cloud. How the data provider manages access to its data hosted by different companies is a new AC challenge. Questions to be addressed include how long the data provider should own the data and if the hosting company has the privilege to determine access right to the data that it hosts. As presented in Use Case 3, the medical records and genomic information are stored and processed in another domain of BD. It is very challenging to allow those data owners to make AC policy in such cases, since subjects of access requests are unknown to data owners. Even if the data provider has the capability to determine the access right, how to ensure the up-to-date AC policy is always enforced is a challenge. Policy making can be very challenging for the data provider in such cases, since the subjects for access requests are unknown to it. In general, the subjects of access requests are better known to the hosting company; a contract and policy synchronization mechanism may be needed. Attribute-based encryption schemes [6], [19], [22], [32], [33] can be considered to provide data owner's control.

2) Objects: Data objects in DPS can be complex and vary featuring unstructured formats, which are fundamentally different from traditional structured, relational data. Moreover, the ways that object is generated, stored, accessed, used, and managed are also different from traditional structured data. These new properties affect the design, deployment, and management of AC system when accesses to databases are needed.

Data and metadata: Besides raw data (e.g., text, images and videos), metadata also plays an important role in data analytics especially in AC for DPS. Metadata describes information about data, e.g., the relationships between data elements and the intended usage of data. An example metadata is provenance metadata, which describes where the data is generated, when the data is created, how the data is generated, and so on. In Use Case 3, both medical records and the metadata of them should be protected by AC. Such description is crucial for reliable data analytics. As data itself and its metadata are both needed for different applications, they may cause security and privacy leakage in different ways. Thus, data and metadata should be protected by different measures (e.g., policies). The access patterns to data and metadata as well as their relation to privacy leakage should be considered. Schemes proposed in [13]–[15] might be considered to deal with the challenge.

Granularity of data for AC: Non-DPS systems usually categorize data at a fixed level of granularity, but DPS may allow data access at different granularities. Different categories of data should be accessed at different granularities. Based

on the least privilege principle for security, it is desirable to ensure that applications only access the data no more than what is necessary for the assigned processing. Thus, the AC system should support different granularity requirements of access. Granularity considerations can be based on a single data, a group/part of data, or combinations of data objects. The mechanism of access granularity for AC policy design, representation, and enforcement needs to be considered. The schemes proposed in [13], [15], [18] might facilitate achieving fine-grained AC.

Stream of data: Usually, AC systems are designed for the protection of data at rest, such as records in a database and files in a computer system. However, DPS frequently involves streamed data sources. For instance, as described in Use Case 3, between 500,000 and 1.5 million new real-time clinical transactions are generated every day. These streamed data must be protected by the AC system in the same pace as the data is generated. This requires fast and frequent updates to the AC policy and may even to the policy enforcement mechanism. The reason is that unlike AC policy management for a single type of data objects which is straightforward and needs no update when a new object is added to the system, a data stream may contain various types of objects and different levels of AC protection may be necessary. For example, in a data stream of location information, some locations are sensitive (e.g., military base) while others are not as sensitive (e.g., school). Thus, they require different levels of AC protection. These challenges with AC design should be addressed. Solutions proposed in [19], [32], [34] might be considered to handle the stream of data.

Variety of data: Variety is a key feature of DPS, which can be structured (e.g., medical record in Use Case 3), semi structured (e.g., NoSQL database of Netflix in Use Case 2), and unstructured. It is different from data structured into tables, records and fields for management such as relational databases. An example of unstructured data is a text file with keywords. Semi-structured and unstructured data have access patterns and protection needs that are quite different from traditional structured data. However, existing DPS systems are not designed with AC in mind, and they usually relegate to middleware for security.

Variety can also be characterized by various types of data such as text, image, audio, video, network graphs, and so on. Some data that are deemed to be too big for analysis by traditional computing systems need to be handled by DPS such as video imaging and geospatial data. Variety is also reflected by the sources of data; data may be collected from various providers including mobile devices, individual users, organizations, companies, and governments. Furthermore, complex combinations of data sources and data types are common in DPS which is not anticipated and hard to protect in traditional computing systems. Note that the variety of data implies that there need multiple ways to access data. For instance, text data can be downloaded, edited, queried, and mined with machine learning algorithms, and images can be tagged with location information. AC for DPS should be enforced upon all types of data.

Multiple data sources: When multiple data sources that are not intended to be accessed in the same processing domain are accessed in the same domain, privacy leakage and security breach may occur [35]. For example, if one dataset contains records of {Date of Birth, Zip Code, Disease} and another dataset contains records of {Name, Date of Birth, Zip Code}. Then when they are put to the same processing node, this node might be able to link two records from the same user using Date of Birth and Zip Code and know what disease this user has. Traditional AC techniques that usually consider data sets from a single AC domain become inadequate in BD. New techniques for this concern should be developed.

3) Actions: In DPS, the types of action for data accessing are not only read, write, and execute as in general for non-DPS AC access requests. Due to the large volume of data in DPS, a process ranging from simple queries to complex machine learning algorithms is moved and executed at where the data is stored, instead of moving data to a processing node. The results are then sent back. Since there are many types of processes, the results vary regarding the levels of security. For example, on a dataset of salaries of university employees, one analytical process asks for the average salary of employees which is not sensitive; but another analytical process asking for the salary of a specific employee is very sensitive from a privacy perspective. Whether the AC system should allow such analytical process to access data depends on what information is retrieved from the data and if this information violates the security goal of the data provider. Note that for two analytical processes from the same user, if each renders different information, different permissions should be given. This is usually not considered in other systems, which control access based on users or subjects. As fine-grained AC over processes is needed in DPS, one possible strategy is to decompose the access to a process into a set of basic actions and protect these basic actions in a fine-grained manner. Another strategy is to introduce new action types to enrich the traditional set of actions.

B. Considerations of DPS's AC Policy Design

The design of AC policy for DPS must accommodate the additional properties of DPS's subjects, objects, and actions as described in Section IV-A1, Section IV-A2 and Section IV-A3.

1) Syntactic and Semantic Support for Specifying AC Rules: Data processed by DPS can be accessed at different granularity levels, the processing operation on data therefore more complicated than traditional systems such as relational database. To support fine-grained access control, syntactic and semantic support may need to be considered when specifying AC rules, specifically, privileges and constraints may need to be specified in complex expressions with Boolean logic relations such as AND, OR, $<$, $=$, $>$, and $*$ (wildcards) between AC elements. Logic operators provide AC policy authors with efficient ways to specify granularities of objects for the intended policy rules.

Beyond logic operators, AC policies for each collaborative processing unit must be carefully designed to ensure that they

are semantically consistent even though they may be specified in different environments. For instance, data provider's policy could be abstract for ease of management by human but a storage service provider's policy should be actionable for computers, thus requiring different interpretations for these two AC policy contexts. This means that data provider's AC policy might need to be described in informal language, while the storage/computing service provider's policies enforceable by computers might need to be specified in machine readable format (such as XACML which is a standard specification language). Current AC languages are designed with a specific application or architecture in mind, so they are not universally applicable for all AC models or mechanisms. Thus, AC policies may need to be specified with a programming language, which provides syntactic as well as semantic support for specifying AC policies.

2) Template of Standard AC Models: Many existing AC policies/models are applicable to DPS, including Role-Based Access Control (RBAC) [36], Attribute-Based Access Control (ABAC) [37], Chinese Wall [38], Multi-Level Security (MLS) [39], etc. In a traditional single host system, usually subjects are known and objects are static, so one of existing AC models may be able to achieve security goals. However, in DPS, each individual AC policy model may not be able to fulfill all the demands, but applying multiple policies/models together with other enhancements may satisfy the requirements. Such AC models can significantly facilitate users in AC policy design. Hence, we need to take the support from standard AC models into account.

3) Conflict Resolution: In DPS, due to its dynamic and complicated architecture, the AC policy may contain AC rules that are not practical to be generated by a single system. Instead, AC rules can be composed through the cooperation between multiple systems involved, or automatically generated by a specially designed method. So, there may exist conflicts between those separately developed AC rules (especially if them are made by different administrators), where the specifications of two or more access rules result in conflicting decisions about whether a subject's access request should be permitted. Policy rule conflicts can also be a result of deadlock in access rule specifications. Deadlock [40] can be defined as: a *rule r* has a dependency on other *rule(s)*, which eventually depend back on *r* itself such that the subject's request will never reach a decision because of the cyclic referencing. In addition to policy rules, when multiple policies are evoked for granting permission, conflicts of policy may occur between policy *X* and policy *Y*. To support conflict resolution, an DPS's AC system may need to provide automatic conflict identification with suggested corrections.

C. Considerations of DPS's AC Policy Enforcement

AC policy enforcement includes support for granularity and safety constraints of AC.

1) Granularity of AC: In DPS, every processing node relies on a local enforcement mechanism for AC policy. Such mechanism is traditionally implemented as a reference monitor, which checks every access request against the Access Control List (ACL) or access matrix to determine if the request is permitted or denied. In DPS, since data and metadata need to be protected interdependently in different ways such that two correlated ACLs or access matrices may be required. The AC enforcement mechanism should also be able to handle different granularities of data access, such as at both file/dataset-level and block-level. Considerations should also include virtual machines, which are commonly used in Cloud environment and serve as a type of protection.

2) Safety Constraints: Safety is an important feature of an AC system, and is needed to ensure that the AC configuration (e.g., AC model) will not result in leakage of permissions to an unauthorized principal. Thus, a configuration is said to be safe if no privilege can be escalated to an unauthorized or unintended principal. Safety is achieved either using restricted AC models that can be proven in general for that model, or via expressions called constraints that describe the safety requirements of any configuration. Compared with traditional single host system, the safety constraints in DPS are more complicated due to its dynamic architecture. For instance, the processing task will be dispatched to some specific SSs; however, they cannot be predicted before the task is executed. Hence, safety constraints based on the context of each SS are challenging to specify beforehand. To enforce safety, the AC implementation should include a mechanism for preventing leakage of privileges through either constraints or confinement.

D. Considerations of DPS's AC Policy Management

AC policy management includes activities such as policy update, dissemination, translation and other additional functions required by the application environment, which need to be effectively and efficiently managed.

1) Policy Update: To support security goals and system environment changes, policy maker may need to update the AC policy to provide adaptive protection. For example, it needs to reassign a user to a different role, or reclaim access permissions from a user. When making a change to policy, it is crucial to ensure that the security goals are still met after the change, and that the change will not introduce faults to policy, such as permission conflict, cyclic inheritance, privilege escalation, and Separation of Duty faults [41]. Compared with traditional single host system, policies may be stored in distributed repositories in DPS instead of a central repository. It is very difficult to efficiently detect and update AC policies that are affected. Hence, such assurance becomes very challenging.

2) Policy Dissemination: In DPS, data is dynamically disseminated to distributed computing nodes during processing. Correspondingly, the AC policy and the supportive information for policy enforcement should also be disseminated to relevant computing nodes in a timely manner. A secure, reliable, and efficient dissemination protocol is needed. In some scenarios, a possible alternative is to send the access request to the maker of AC policy for checking and authorization.

3) Policy Translation: In DPS, one system domain may enforce AC policies from other domains. For example, the data

provider's policy must be enforced by data hosting companies. However, different domains' policy may be syntactically or semantically different (e.g., data provider's policy is abstract at high-level but the data hosting company's policy is at low level and readable to machine). Therefore, policy translation between multiple domains is needed.

E. Other Considerations

In addition to the above proposed considerations, there are general considerations for DPS's AC design as discussed below. More general considerations can be found in [42].

1) Performance: In some conditions, DPS may be accessed by many users simultaneously. Thus performance is one of the critical factors that needs to be considered when implementing the AC system, which should be able to process access request within a time that is consistent with the operational needs. The response time of AC system depends on the number of operations required for granting a subject's access request, the processing time of the underlying hardware/network as well as retrieving AC policies from different repositories.

Policies can be stored in and retrieved from local, global, federated, or subscribed (e.g., Cloud) repositories. Furthermore, some AC policies might require connecting to multiple repositories simultaneously. It is important to balance the cost of hardware and software with efficiency based on the organization's requirements. AC policy might be distributed to designated hosting domains including a local policy repository, a global policy repository, or across federated policy repository. An AC system may support distribution by manually or automatically pushing policies to the endpoints. Then the consideration is what mechanisms for endpoint policy distribution and retrieval the AC system should support: Security Content Automation Protocol (SCAP), Active Directory, Lightweight Directory Access Protocol (LDAP) or proprietary.

2) Policy Import and Export: Designing an AC system for DPS is a costly task, but the cost will significantly be reduced if policy import and export capabilities are provided, e.g., translating from system commands to AC rules (e.g., a Simple Network Management Protocol (SNMP) command to a XACML rule), or if the AC mechanism is supported by an AC language, translating a policy from one structured language into another structured language (e.g., from OWL to XACML). An AC system may provide a selection of different forms for policy export or policy translating back to the original source from which it was derived (e.g., an XACML rule back to the original English language rule from which it was translated).

V. Related Work

In recent years, some work has been done on DPS AC in both industry and academia. For example, distributed Big Data systems apply traditional perimeter security solutions to enforce AC. However, those approaches are not efficient and adequate because the breach of perimeter leads the system to be wide open for attack [9]. The Hadoop community applies technologies including Kerberos, firewall and basic Hadoop Distributed File System (HDFS) permission to deploy AC [9], [10]. However, it is difficult to deploy and configure Kerberos on the MS and SSs in the DPS. Other distributed Big Data systems deployed their own security as a layer over the core Big Data processing system. For example, HBase [11] provides column and family level authorization; Apache Accumulo [12] improves the BigTable design in the form of cell-level mandatory and attribute-based access control.

Zeng et al. [13] propose a content-based access control model for Big Data, which is suitable for content-centric information sharing. In their model, the content similarity between records are assessed and utilized to make AC decisions. Rizvi and Mitchell [43] present a novel generic access control scheme that is capable of working with available access control policies using a global resource management system to effectively handle both local and remote authentication requests. Hu et al. [6] present a general AC schema for distributed Big Data systems, which is based on the trust between data providers and the Big Data Master System, and between the Master System and Cooperating Systems. The schema is focused on authorization under the assumption that authentication is already established. Almutairi et al. [25] design a distributed security architecture for cloud computing, where AC policy interoperation among multi-cloud environments is considered, such that AC mechanism and SLA are implemented at each layer (SaaS, PaaS and IaaS) of every cloud service provider. But the mediation for conflict resolution of heterogeneous policies and how to enforce local AC policies of each individual MS and SS in the infrastructure layer are not discussed. Both the work from Hu and Almutairi present a generic AC framework for DPS. However, the former one designs the framework from the perspective of physical architecture in a DPS, while the later one proposes the architecture from the point of service models.

Additionally, in order to protect the confidentiality of stored data in DPS, Sahai and Waters propose an Attribute-Based Encryption (ABE) scheme [32] using a user's identity as attributes, and a set of attributes are used to encrypt and decrypt data, which require data owner to use every authorized user's public key to encrypt data. Nail et al. [33] propose a threshold attribute-based encryption method which can be used to prevent collusion attacks. Goyal et al. [34] propose a key-policy attribute-based encryption (KP-ABE) scheme, which embeds access control policies into the user's private key and the encrypted data is described with user's attributes. It is more flexible to use KP-ABE to achieve fine-grained access control than the ABE scheme. However, data owner cannot choose who has the right to decrypt the data, since the access policy is built into a user's private key. Yu et al. [19] propose a scheme to achieve fine-grained access control in Cloud computing, which defines and enforces AC policies based on data attributes and allows data providers to delegate most of computation tasks involved in fine-grained access control to untrusted Cloud servers without disclosing the underlying data content. Kan et al. propose a scheme enabling AC with dynamic policy updating for BD in the Cloud [20], which is focused on policy updating for ABE systems. Zhu et al. [22]

present a temporal AC encryption scheme for Cloud services with the help of cryptographic integer comparisons and a proxy-based re-encryption mechanism on the current time. Li et al. [23] present a way to implement scalable fine-grained access control systems based on ABE. It defines and enforces AC policies based on data attributes and implements user accountability by traitor tracing. Jensen et al. [24] propose an approach for data anonymization to prevent the Cloud provider from misusing user's data, which provides anonymous yet reliable AC and accountability based on ring and group signatures. These approaches pay more attention to data storage than the data processing, which is an attractive feature of current DPS.

However, the previous works on DPS AC are mostly ad hoc and patch efforts. Comprehensive and systematic understandings on the issues are still lacking.

VI. Conclusion and Future Directions

DPS allows users to efficiently store, manage and analyze current huge volume of unstructured data. However, existing DPS is not designed with comprehensive considerations of authorization and access control. Hence, before enjoying the benefit from DPS, security challenges must be addressed. In this work, we introduced the general architecture of DPS. We first analyzed the characteristics and use cases of BD and Cloud. Then we proposed a set of general considerations for AC policy design, enforcement and management based on the aforementioned features and use case.

This work only makes an initial step towards holistic AC design for DPS. Many issues still remain, and new challenges might arise with the evolution of DPS. Future directions include designing generic AC frameworks to address all the considerations in this paper, implementation, prototyping, and evaluations.

Acknowledgment

This work is supported in part by NIST under Award Number 60NANB16D018.

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.

[2] H. Mir, "Hadoop tutorial 1what is hadoop," ZeroToProTraining, http://ZeroTOProTraining.com.

[3] W. Bell, "The big data cure," MeriTalk, http://www.meritalk.com/bigdatacure.

[4] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *2008 Grid Computing Environments Workshop*. Ieee, 2008, pp. 1–10.

[5] Hadoop.apache.org.

[6] V. C. Hu, T. Grance, D. F. Ferraiolo, and D. R. Kuhn, "An access control scheme for big data processing," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*. IEEE, 2014, pp. 1–7.

[7] "Nist big data public working group - reference architecture subgroup," DRAFT NIST Big Data Interoperability Framework: Volume 6, Reference Architecture. Retrieved in February 2015.

[8] "Top 10 challenges in big data security and privacy," 2012, cloud Security Alliance Big Data Working Group.

[9] "The big data security gap: Protecting the hadoop cluster," White Paper, Zittaset, 2014, http://www.zettaset.com/wp-content/uploads/2014/04/zettaset_wp_security_0413.pdf.

[10] K. T. Smith, "Big data security: The evolution of hadoop's security model," InfoQ, Aug 2014, http://www.infoq.com/articles/HadoopSecurityModel.

[11] Apache Hbase, Hbase.apache.org.

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

[13] W. Zeng, Y. Yang, and B. Luo, "Access control for big data using data content," in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 45–47.

[14] C. Mohan, "History repeats itself: sensible and nonsensql aspects of the nosql hoopla," in *Proceedings of the 16th International Conference on Extending Database Technology*. ACM, 2013, pp. 11–16.

[15] H. Ulusoy, M. Kantarcioglu, E. Pattuk, and K. Hamlen, "Vigiles: Fine-grained access control for mapreduce systems," in *2014 IEEE International Congress on Big Data*. IEEE, 2014, pp. 40–47.

[16] E. Pattuk, M. Kantarcioglu, V. Khadilkar, H. Ulusoy, and S. Mehrotra, "Bigsecret: A secure data management framework for key-value stores." in *IEEE CLOUD*, 2013, pp. 147–154.

[17] W. Wei, J. Du, T. Yu, and X. Gu, "Securemr: A service integrity assurance framework for mapreduce," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 73–82.

[18] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and privacy for mapreduce." in *NSDI*, vol. 10, 2010, pp. 297–312.

[19] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Infocom, 2010 proceedings IEEE*. Ieee, 2010, pp. 1–9.

[20] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2013–2021.

[21] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[22] Y. Zhu, H. Hu, G.-J. Ahn, D. Huang, and S. Wang, "Towards temporal access control in cloud computing," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2576–2580.

[23] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, L. Tang, and Y. Tang, "Fine-grained data access control systems with user accountability in cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 89–96.

[24] M. Jensen, S. Schäge, and J. Schwenk, "Towards an anonymous access control and accountability scheme for cloud computing," in *Proceedings-2010 Ieee 3rd International Conference on Cloud Computing, Cloud 2010*, 2010.

[25] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref, and A. Ghafoor, "A distributed access control architecture for cloud computing," *IEEE software*, vol. 29, no. 2, p. 36, 2012.

[26] G. I. Glossary, "Big data (definition)," http://www.gartner.com/it-glossary/big-data.

[27] I. B. San Murugesan, Ed., *Encyclopedia of Cloud Computing*, 1st ed. Wiley-IEEE Press, July 2016.

[28] S. NIST, "1500-1 nist big data interoperability framework (nbdif): Volume 1: Definitions, september 2015 [online] http://nvlpubs. nist. gov/nistpubs/specialpublications/nist."

[29] J. Wu, L. Ding, Y. Wu, N. Min-Allah, S. U. Khan, and Y. Wang, "C2detector: a covert channel detection framework in cloud computing," *Security and Communication Networks*, vol. 7, no. 3, pp. 544–557, 2014.

[30] S. NIST, "1500-3 nist big data interoperability framework (nbdif): Volume 3, use cases and general requirements [online] http://nvlpubs. nist. gov/nistpubs/specialpublications/nist."

[31] L. Badger, R. Bohn, S. Chu, M. Hogan, F. Liu, V. Kaufmann, J. Mao, J. Messina, K. Mills, A. Sokol *et al.*, "Us government cloud computing technology roadmap," *NIST Special Publication*, vol. 11, pp. 500–293, 2011.

[32] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2005, pp. 457–473.

[33] D. Nali, C. M. Adams, and A. Miri, "Using threshold attribute-based encryption for practical biometric-based access control." *IJ Network Security*, vol. 1, no. 3, pp. 173–182, 2005.

[34] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings*

of the 13th ACM conference on Computer and communications security. ACM, 2006, pp. 89–98.

[35] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[36] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[37] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo, "Attribute-based access control." *IEEE Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[38] D. F. Brewer and M. J. Nash, "The chinese wall security policy," in *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*. IEEE, 1989, pp. 206–214.

[39] P. Samarati and S. C. de Vimercati, "Access control: Policies, models, and mechanisms," in *International School on Foundations of Security Analysis and Design*. Springer, 2000, pp. 137–196.

[40] A. Li, Q. Li, V. C. Hu, and J. Di, "Evaluating the capability and performance of access control policy verification tools," in *Military Communications Conference, MILCOM 2015-2015 IEEE*. IEEE, 2015, pp. 366–371.

[41] A. Gouglidis, I. Mavridis, and V. C. Hu, "Verification of secure interoperation properties in multi-domain rbac systems," in *Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on*. IEEE, 2013, pp. 35–44.

[42] V. C. Hu and K. A. Kent, *Guidelines for access control system evaluation metrics*. US Department of Commerce, National Institute of Standards and Technology, 2012.

[43] S. Rizvi and J. Mitchell, "A new access control scheme for protecting distributed cloud services and resources."