DOI: 10.3969/j.issn.1673-5188.2018.02.009
http://kns.cnki.net/kcms/detail/34.1294.TN.20180608.1702.002.html, published online June 8, 2018

Open Source Initiatives for Big Data Governance and Security: A Survey

HU Baiqing, WANG Wenjie, and Chi Harold Liu
(Beijing Institute of Technology, Beijing 100081, China)

Abstract

With the rapid development of Internet technology, the volume of data has increased exponentially. As the large amounts of data are no longer easy to be managed and secured by the owners, big data security and privacy has become a hot issue. One of the most popular research fields for solving the data security and data privacy is within the scope of big data governance and security. In this paper, we introduce the basic concepts of data governance and security. Then, all the state-of-the-art open source frameworks for data governance and security, including Apache Falcon, Apache Atlas, Apache Ranger, Apache Sentry and Kerberos, are detailed and discussed with descriptions of their implementation principles and possible applications.

Keywords

big data; security; governance; open source initiatives

1 Introduction

In recent years, the value of big data has been much recognized by both research community and governmental agencies. However, the rapid development of big data technology has brought more unsolved problems [1]. Data have different values in different spatial-temporal domains as well as in different businesses. In order to maximize its value, using the Internet to share data is inevitable. However, as various enterprises are independent from each other, their data systems and data storage structures are also different. It is thus quite challenging to share the data between them, resulting in a common phenomenon of information islands. Meanwhile, it is challenging to guarantee the data security and privacy when sharing data between different data systems back and forth.

As enterprises start to collect, store, process and exchange large volume of data in the course of addressing these opportunities, they face increasing challenges in the areas of data security, maintaining data privacy, and meeting related compliance obligations. Traditional IT security approaches mainly focus on protecting the organizations' IT infrastructure, by securing the network edge and end points and protecting the data that are stored and moved through the infrastructure. However, the focus of this paper is on how to provide efficient services for managing the entire data lifecycle while protecting its security, which relatively speaking has sparse research exposure from the software development perspective.

In order to solve these problems, big data governance and security has become one of the hottest research areas. Big data governance aims to establish a unified standardized platform, which obtains data from different data sources and can satisfy various data operational requirements as well as conducting lifecycle data management (such as data audit, selection, and migration), to maximize the data value. Moreover, this unified standardized platform can enforce permission settings for different metadata, securing the data for different users on the basis of time points and IP addresses.

Towards this end, this paper aims to introduce how to achieve the governance and security of big data from open source component design and implementation perspectives. Specifically, our contribution is threefold.

1) We extensively review the related studies for big data governance and security, and compare their advantages and disadvantages.
2) We describe five open source initiatives, namely, Apache Atlas, Falcon, Ranger, Sentry, and Kerboros, from both conceptual and architectural perspectives, and discuss their usages in different scenarios.
3) We introduce four future research directions for big data governance and security.

2 Concepts of Big Data Governance and Security

The term "big data" refers to very large or complex data sets that cannot be managed by traditional data processing software. Big data can be converted into very useful knowledge for enterprises to make efficient decisions. It is generally accepted that big data has three "v"s: velocity, variety, and volume [2], and IDC believes big data should also have value besides the three "v"s. Moreover, IBM thinks big data is certain to be of veracity [3].

With the continuous development and popularization of the Internet technology, the data quantity is growing exponentially. Thus the concept of big data is introduced. With the deep inte-

gration of big data and cloud computing technology, the data is no longer easy to be managed by data owners using the traditional technologies. Therefore, big data security and privacy has attracted much attention [4]. At the same time, how to govern the data is also a conundrum.

This paper summarizes all the state-of-the-art technologies for governing big data, from three aspects: principle, scope, and implementation and assessment. Big data governance principle refers to the primary and basic instructive principle that big data follows, which is useful for big data management. In order to efficiently collect, effectively integrate and sufficiently utilize data, big data management principle can be subdivided into the principle of effectiveness, the principle of value, the principle of unity, the principle of openness, and the principle of security. The governance of big data mainly involves five key fields: the lifecycle of big data, the frame of big data, the safety and privacy of big data, the quality of big data and the service innovation of big data. The implementation and evaluation of big data provide an instructive project for enterprises from three aspects: the implementation environment, implementation procedure and assessment of implementation results.

Big data provides a new opportunity for all application areas as well as a challenge for information security. It is of great research value and importance to governments, enterprises, and individuals, thus data security, the precondition of big data development, has become a hot research issue in academic and industrial circles [5]. Big data not only refers to the massive amount of information but also its complexity and sensitivity, which will attract potential aggressors. Furthermore, collected big data includes lots of enterprise operational data, customer data, and individual privacy and detailed records of all kinds of behaviors. The concentrated storage of these data increases the risk of privacy leakage. Meanwhile, the lack of certain definition of data ownership and usage right also increases the risk [6]. Despite the multifaceted advantages of cloud computing, concerns about data leakage or abuse impede its application for security-sensitive tasks. Recent investigations have revealed that the risk of unauthorized data access is one of the biggest concerns for users of big data [7].

Big data imposes challenges and opportunities for auditing. Big data audit is conducted by third-party auditors who are independent of auditing targets; the auditors make comprehensive examination and evaluation of the procedure of big data governance and conduct a series of activities such as putting forward questions and suggestions to the supreme leader of the auditing targets. Big data audit aims to understand the overall situation of an organization's big data activities, to review and evaluate the organization's goal of achieving big data governance, to fully identify and assess the risks associated with the evaluation, to make comments and suggestions for improvement, and to achieve the goal of big data governance. The process of big data audit generally includes setting audit objectives, determining the risk areas of big data audit, setting an audit plan, building the environment of big data audit, carrying out the plan and issuing audit results and governance recommendations [8].

In order to ensure the quality of data governance, its audit mainly focuses on the data supervision and evaluation, being of four aspects: content, architecture, security, and lifecycle. The audit follows certain standards. At present, big data audit methods are mainly divided into traditional audit methods, IT internal audit methods, and big data audit methods. Furthermore, the audit of big data also needs a certain technical means to avoid any blind review and evaluation. The current audit models of provable data possession (PDP) and proof of retrievability (POR) for cloud storage can only be applied to static data audits but fail to support auditing of dynamic data. In order to solve this problem, the third party auditor (TPA) model is proposed, which can efficiently audit the data and also completes the pubic audit for protecting user privacy [9].

3 Related Work

3.1 Big Data Governance

Businesses and enterprises have recognized that increasing expenses on data management solutions are becoming unbearable. They need to use effective data governance methods to solve big data problems. Data governance involves the adoption of data models, data quality standards, data security and lifecycle management methods, as well as the processing procedures the application defines. However, data governance has not been well applied, due to the void of a particular enterprise repository, lacking structures and requiring broader support of organizations. Therefore, despite its importance, data governance is still under investigation.

Al-Ruithe et al. proposed six key dimensions that must be taken into consideration for cloud data governance, such as data governance structure, organizational factors, and technical/environmental factors [10]. A new technology requires a good data governance strategy for its successful implementation. Furthermore, as increasingly large amount of personal and confidential data are transferred to the cloud, related stakeholders' accountabilities have emerged as a critical issue that is related to data protection in cloud ecosystems. From this angle, Felici et al. introduced a conceptual model, consisting of attributes, practices and mechanisms [11], to form the basis for characterizing accountability relationship between cloud actors, and chains of accountability in cloud ecosystems as well. However, these two research efforts did not give specific solutions to the above mentioned problems, no matter from software development or implementation perspective.

3.1.1 Technologies for Big Data Governance

A. Corradi et al. pointed out that the discovery, aggregation and manipulation of distributed and diversified data sets play

an important role in supporting core business processes [12]. It has been agreed that the semantic method can effectively deduce the relation and dependency from the heterogeneous information set, but when the real joint data navigation is performed, the current de facto standard query language is not sufficient and there is no accurate knowledge of data distribution. Therefore, the authors set up a model to propose a lightweight federation ontology for crossing the organization mapping information source to add the current SPARQL limit based on a priori network knowledge. Then, a single query was conducted both on academia and on municipality endpoints, facilitating the development efforts of the overall solution. However, it has certain limitations in terms of endpoint time-outs and unavailability. This process is obviously inefficient and poorly extensible, because web services should be extended anytime while new data sources are added, to query new endpoints and combine results with old ones.

T. Priebe et al. presented a methodology to gather and structure data requirements to improve data-intensive projects and enable data governance [13]. The methodology facilitates data harmonization by introducing a semantic business information model as a central point of reference on top of physical and logical data models. In addition, M. Al-Ruithe et al. postulate that as the "smart" continuum continues to innovate and grow, so will the velocity and volume of data [14]. Their efforts add to body of research and frameworks are proposed to identify a roadmap to address data governance and security challenges in Internet-of-Things (IoT) cloud converged environments. Also, they propose a data governance and security layer, which describes roles, responsibilities and policies as key pillars. However, as more IoT cloud converged domains continue to evolve, their roles, responsibilities and policies will remain central to governance and security processes and procedures, that brings certain questions of whether this framework can continue to function or not. Furthermore, software development and implementation details are still missing.

When data moving across multiple systems, it may cause more mistakes or bad changes of related processes and systems. The lack of awareness of corporate data landscape impacts the ability to govern data, which in turn impacts overall data quality within organizations. R. J. DeStefano et al. propose tools and techniques for companies to better gain awareness of the landscape of their data, processes, and organizational attributes through the use of linked data, via the Resource Description Framework (RDF) and ontology [15]. The outcome of adopting such techniques is an increased level of data awareness within the organization, resulting in improved ability to govern corporate data assets, and in turn increased data quality. However, the application of such techniques into real-life big data systems still needs time.

3.1.2 Applications of Big Data Governance

It is all agreed that data has become a major need in nearly all businesses. However they must be accurate and valid. Organizations usually face data problems, like data duplication, inaccurate data, incomplete data, invalid data, and unavailable data. Yulfitri et al. [16] analyzed these problems that occurred in a governmental agency in Jakarta, Indonesia, and concluded an approach that was used in sequential stages, that is studying best practices regarding operational model of data governance, analyzing current conditions, reviewing organizational structure, analyzing business processes and human resources, and mapping out data governance activity. Another example is about operational model for clinical data governance. Thiel et al. [17] proposed a method to identify the legal and ethical challenges in Europe for clinical data governance in health informatics and to classify the various legal bases for sharing a dataset.

3.2 Big Data Security

Data security is one of the major challenges in the era of big data, including the protection against security breaches and data leakage, penetrability in public databases, and third party data sharing, etc. From research only perspective, there exists certain literature reviews reporting the recent progresses along this direction. However, from software development and implementation perspectives, open source initiatives are missing. For example, how to protect sensitive information from the security threats brought by data mining has become a hot topic in recent years. To solve the above threats, Xu et al. surveyed the privacy issues related to data mining by using a user-role based methodology [18]. They differentiate four different user roles that are commonly involved in data mining applications, i.e., data provider, data collector, data miner and decision maker. Furthermore, Ye et al. proposed three categories about security issues in big data infrastructure security, data privacy, and data management [19]. Finally, Tan et al. presented a survey of recent security advances in smart grid, centered around the security vulnerabilities and solutions within the entire lifecycle of smart grid data [20].

4 Open Source Initiatives for Big Data Governance

In the first two sections, we presented a technical overview of the governance and security of big data, including big data lifecycle governance, security protection, and data auditing. In the following sections, we will detail the realization of governance and security of big data. This section focuses on Apache Falcon and Apache Atlas, which play an important role in big data governance. Apache Falcon can perform data lifecycle management, including data collection, data processing, data backup and data cleansing, for big data platforms, as well as for fine scheduling of components of big data platforms. Apache Atlas can perform tasks including metadata management, data lifecycle auditing and visualization, lineage search,

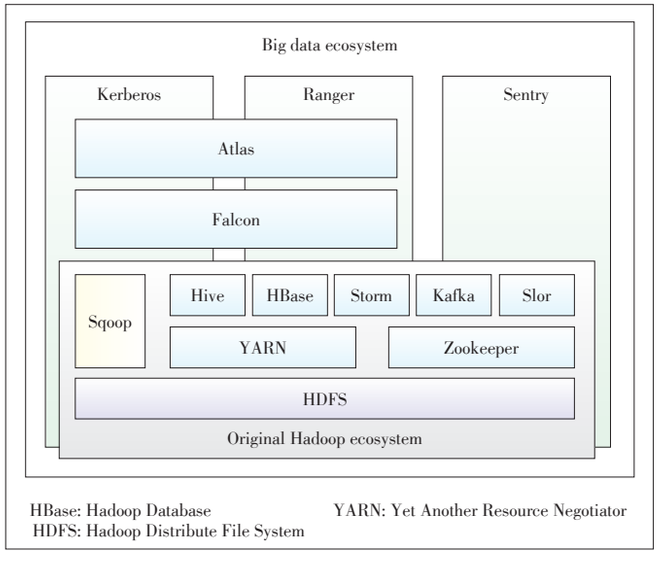
and data security and privacy, for big data platforms.

Fig. 1 shows the overall big data ecosystem that supports the data governance and security functionalities, based on the existing Hadoop ecosystem. Apache Falcon, Atlas, Kerberos, Ranger and Sentry are used. Falcon and Atlas components can interact with each other, while Atlas can be used as metadata sources for Atlas. Meanwhile, Hive, Sqoop, Falcon, and Storm can also be used as metadata sources. Then, Apache Ranger is used as a centralized security management solution for Hadoop that enables administrators to secure authentication mechanisms and configurations with Hadoop components such as Hadoop Distributed File System (HDFS), Hive, Hadoop Database (HBase), and Kafka. The components Kerberos, Ranger, and Sentry provide security protection to all Hadoop components. Furthermore, Kerberos and Ranger can interact with Falcon and Atlas, to provide data governance and security solutions simultaneously.

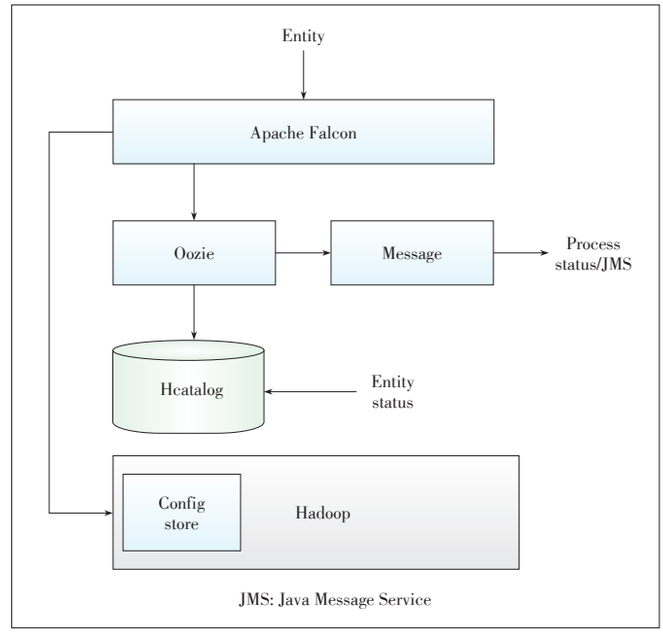
4.1 Apache Falcon

Apache Falcon solves the problems of Hadoop data replication, business continuity and lineage tracking by declaring data management and processing solutions. Falcon centrally manages the data lifecycle, facilitates quick data replication for business continuity and disaster recovery and provides a foundation for audit and compliance by tracking entity lineage and collection of audit logs. It also helps user set data management and the way of process and submit it for scheduling on Hadoop Cluster.

Apache Falcon is a management platform built on Hadoop data set and **Fig. 2** shows its processing flows. A user can submit an entity to the Apache Falcon through the Falcon client or the Rest API. Falcon generates the workflow entity based on the declaration information and stores it in the config store of



▲**Figure 1. Big data ecosystem that supports data governance and security.**



▲**Figure 2. Falcon architecture diagram [21].**

the Hadoop. As processing the workflow, Apache Falcon performs task scheduling mainly through Oozie and stores the entity processing status in HCatalog. During scheduled tasks, Oozie will return status information during execution as well as execute command messages and send them back to the Java Message Service (JMS) message announcement and return the results to Apache Falcon. Falcon essentially translates the user's data set and its process configuration into a series of repetitive activities through a standard workflow engine, without doing anything cumbersome. All functions and workflow state management requirements are entrusted to the workflow scheduler for scheduling. Because it does not do extra work on the workflow itself, the only thing Falcon has to do is to keep the dependencies and links between the data flow entities. This allows the developer to completely feel the Oozie scheduler and other underlying components when creating a workflow using Flacon so that they can focus on the data and processing itself without any unnecessary operation.

Although Falcon distributes the workflow to the scheduler (the default scheduler is Oozie; due to Oozie's limitations, Flacon also performs the scheduler functionality), Falcon also maintains communication (for example, JMS messages) with the scheduler to generate message traces for each workflow in the execution path, to ensure the progress of the current workflow task and the specific situation.

Falcon simplifies the development and management of data processing pipelines with a higher layer of abstraction, taking the complex coding out of data processing applications by providing out-of-the-box data management services. This simplifies the configuration and orchestration of data motion, disaster recovery and data retention workflows.

Falcon enables this simplified management by providing a framework to define, deploy, and manage data pipelines. As an open source project of data lifecycle management, Apache Falcon can provide the following services:

- Establishing relationship between various data and processing elements on a Hadoop environment
- Feeding management services such as feed retention, replications across clusters, and archival
- Onboarding new workflows/pipelines easily, with support for late data handling and retry policies
- Integrating with metastore/catalog such as Hive/HCatalog
- Providing notification to end customer based on availability of feed groups
- Enabling use cases for local processing in colo and global aggregations
- Getting lineage for feeding and processing.

In general, Apache Falcon meets enterprise data governance needs in three areas, as shown in **Table 1**.

In the workflow implementation, Apache Oozie is mainly responsible for task scheduling, and the entity execution status is stored in HCatalog. During the scheduled execution of the task, Oozie returns the status information during execution, executes the command message and returns the result to the Apache Falcon by sending it to JMS message announcement.

The default scheduler for Apache Falcon is Oozie. Since Falcon relies on Oozie for scheduling and workflow execution, which limits the natural return of feed. In order to achieve better scheduling capabilities, the current Apache Falcon project has also started with native scheduler development.

The scheduler functions include:

1) Submitting and scheduling Falcon to run the process regularly (no data dependencies are required). The program can be a PIG script, an Oozie workflow, or a Hive.

2) Monitor/query/modify scheduled processes: All used entity APIs and instance APIs remain in their original state. Falcon provides a data management function in the form of a life dataset that allows a user to submit a dataset location as a time-based partitioned directory in an HDFS included file.

Although the actual responsibility of the workflow is with the scheduler (such as Oozie), Apache Falcon still remains the execution path of the workflow by subscribing to messages that may be generated by each workflow. When Apache Falcon generates a workflow in Oozie, after that, it uses additional steps, including JMS messaging, to detect workflow execution. The Apache Falcon system itself subscribes to these control messages and, if necessary, performs operations such as retrying and handling the latest input data.

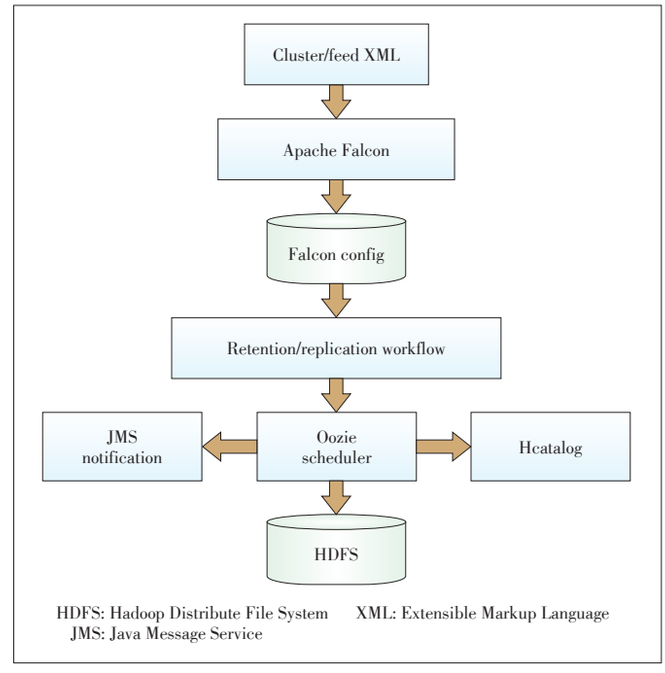
As shown in **Fig. 3**, the user submits and declares the cluster configuration information and data set information to the Apache Falcon through the Cluster XML cluster declaration file and feeds XML dataset declaration file. Falcon generates the cluster entity and feed data based on two files, sets the entities, and then stores them according to the Falcon configuration store, and finally generates the relational graphs. When a reservation or backup is required to operate related data set, Falcon reads the execution entity information in the configuration store and generates the corresponding workflow that will be dispatched by the Oozie scheduler. Oozie outputs the scheduling results to HDFS or Hive's Catalog Service and generates JMS message announcements for each action.

Therefore, Apache Falcon plays a role like Oozie's more advanced abstraction layer. It is the hub of the drives, such as Hive, Sqoop, Map Reduce and other series of Hadoop component tasks, which is not directly responsible for data processing in the actual data processing.

▼**Table 1. Apache Falcon requirements and features [22]**

Need	Feature
Unified management of data lifecycle	• Centralized definition and management of pipelines for data ingest, process and export
	• Ensuring disaster preparedness and business continuity
	• Out-of-the-box policies for data replication and retention
	• End-to-end monitoring of data pipes
Compliance and audit	• Visualization of data pipeline lineage
	• Tracking the data pipeline audit log
	• Taging data with business metadata
Database replication and archival	• Replication across on-premise and cloud-based storages targets: Microsoft Azure and Amazon S3
	• Data lineage with supporting documentation and examples
	• HDFS in heterogeneous tiered storage
	• Definition of data hot/cold storage layer within a cluster

HDFS: Hadoop Distribute File System



▲**Figure 3. The data set workflow.**

At present, Apache Falcon has been successful in a number of areas, including advertisement, healthcare, mobile communications applications, etc. For example, InMobi is one of the largest users of Falcon; it services the advertisement industry that has more than 200 complex big data pipelines and different data sources, and the data is still growing. InMobi can quickly deal with massive amount of data with the help of Apache Falcon, and keep up with the market ever-changing rhythm.

In addition, Expedia.com also carries out data management with Falcon. Expedia introduces the Falcon platform to integrate various data and rules for data processing in the Hadoop environment, sets the relationship of data and rules, perfectly solving the problems caused by the fast development of business. Falcon also provides a great deal of help for Expedia in terms of security. It provides security at the transport level to ensure data confidentiality and integrity.

4.2 Apache Atlas

Apache Atlas is also an important component for the management of big data. It supports metadata management, data lifecycle audit and visual display, data lineage collection, data security and privacy, and other content for big data. Apache Atlas play a very important role in big data governance.

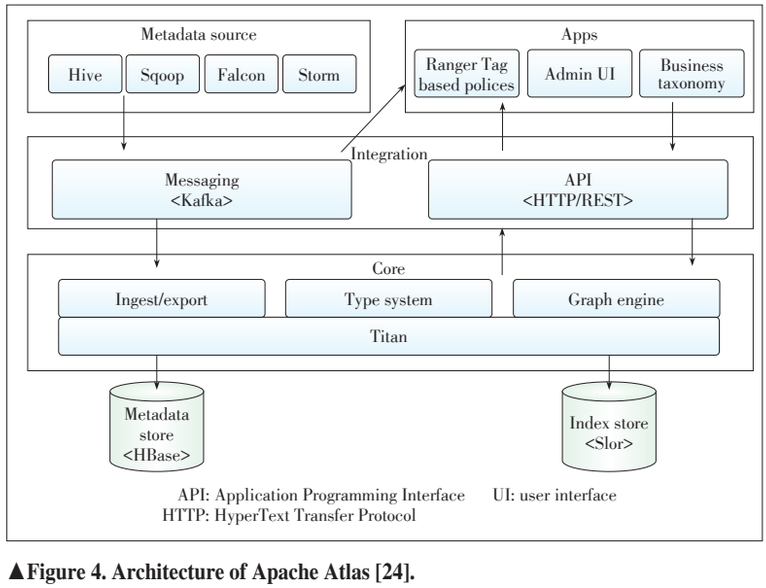
Apache Atlas is a scalable and extensible metadata management tool and provides core foundational governance services, including exchanging metadata with other components, changing the way of past metadata management, building a unified metadata definition standards, and integrating various components of the Hadoop ecosystem to establish a unified, highly scalable metadata management platform [23].

Metadata management can provide complete data definition information for data users, reduce data redundancy, help identify and search data, track data changes in the database, help users understand the data throughout the lifecycle, achieve a simple and efficient management of big data systems in the massive data, and find the value of data through the effective tracking of data resources. Atlas can efficiently integrate all ecosystem components of the enterprise platform with pre-defined requirements while enabling data visualization in Hadoop with pre-set models, providing easy-to-use functions data Audit, and enriching the business metadata by lineage search.

In the process of big data governance, data management tracks the entire lifecycle of data, including data sources, data modification and deletion, and the ability to quickly retrieve. The metadata model can better understand the data and lifecycle by combining labels and data attributes, which enables rapid modeling of data. Unified metadata standards provide a basis for establishing a unified meta-database that runs through the Hadoop ecosystem.

Apache Atlas provides five services: metadata exchange, data lineage, data lifecycle visualization, fast data modeling, and rich API in big data governance. Atlas also has the characteristics of data classification, centralized audit, search and data lineage, security and strategy engine, which plays an important role in the management of big data.

Fig. 4 shows the framework of Atlas. Its components can be grouped into four major categories: the Atlas core, integrations, applications (Apps), and metadata sources. Atlas supports ingesting and managing metadata from the following metadata sources: Hive, Sqoop, Falcon and Storm. After the metadata are acquired, both the API and message system can be used. In terms of metadata management, Atlas can be exposed to user through REST API, so that the user can perform corresponding operations. Meanwhile, the user can choose to integrate with Atlas using a messaging interface that is based on Kafka. Metadata managed by Atlas is used by various applications (Apps) to satisfy many governance use cases, including Admin User Interface (Admin UI), Ranger Tag Based Polices and Business Taxonomy. Atlas Admin UI is a web based application that allows data stewards and scientists to discover and annotate metadata. The Admin UI uses the REST API of Atlas for building its functionality. Tag Based Policies are used to integrate Ranger and Atlas. Ranger is notified by Atlas when metadata change. Meanwhile Atlas provides a business class taxonomy interface that allows the user to build a hierarchical set of terms for various terms in the business domain and integrate them into metadata entities that can be managed by Atlas. The core part of Atlas includes data import and export, type system, graph engine, and Titan. Atlas uses a graph model to represent metadata objects and then stores metadata objects through the Titan graph database. The Titan graph database uses metadata and index databases for data storage, and the metadata uses HBase and the index database uses Solr. Atlas de-



▲Figure 4. Architecture of Apache Atlas [24].

fines an original metadata model to represent various objects, providing the corresponding modules from these components to the metadata object. There are various metadata stored in the Atlas meta-database, and these metadata will be used by a wide variety of applications to meet the needs of a variety of display services and big data governance. Atlas can also be integrated with Apache Ranger, which allows administrators to customize the metadata-based security-driven policies for efficient management of big data.

Although Apache Atlas is still an Apache incubation project, it has been used in a production environment. Apache Atlas can efficiently integrate with all ecosystem components of the enterprise platform while meeting the enterprise's default requirements for the Hadoop ecosystem. At the same time, Atlas can use the pre-set model to visualize data in Hadoop, provide easy-to-use data auditing, and to enrich the metadata of enterprise's business through data collection. It also allows any metadata consumers to collaborate with each other without having to build a separate interface between them. In addition, the accuracy and security of metadata in Atlas is guaranteed by Apache Ranger, which prevents data access requests that do not have permissions at runtime.

4.3 New Progress of Falcon and Atlas Open Source Communities

With new feature requirements flowing in constantly, the Falcon project is making more frequent releases to ensure the features become available to the users as soon as possible. The latest in this string of releases is Falcon 0.10 that was announced on August 8, 2016, however the current stable version is still 0.9. There are many new features that the community is currently working on for product improvements in version 0.9, some of which are: 1) native time-based scheduling, 2) ability to import from and export to a database, and 3) additional API support in the Falcon unit.

Falcon huses Oozie as its scheduling engine. However, while Oozie works reasonably well, there are scenarios where Oozie scheduling is proved to be a limiting factor in version 0.9, e.g., simple periodic scheduling with no gating conditions, calendar based time triggers, scheduling based on data availability for periodic datasets, etc. To overcome these limitations, a native scheduler will be built and released over the next few releases of Falcon. In the 0.9 release, only time-based scheduling without data dependency is supported.

At present, the latest version of Atlas is 1.0 which was announced on May 2018. There are many new features in Atlas 1.0, some of which are: 1) new DSL implementation, using ANTLR instead of Scala; 2) removal of older type system implementation in atlas-type system library; 3) using fine grained authorization to add metadata security; 4) classification propagation via entity relationships; 5) adding HBase integration. Looking for the future, as the next version, Atlas has the aggressive design plans to support, e.g. Titan 1.0+, Spark Integration, NiFi Integration, etc.

5 Open Source Initiatives for Big Data Security

As the Hadoop ecosystem is becoming more and more mature, it is now able to support a complete data lake. A data lake is a storage repository that holds a vast amount of raw data in its native format until it is needed [25]. Enterprises can run multiple workloads in a multi-client environment on a Hadoop system. Enterprises need to support multi-user access to the data lake and data is an important asset for enterprises, so how to protect these different types of user data need to be solved. The solutions are big data distributed security frameworks, including Ranger, Sentry, and Kerberos.

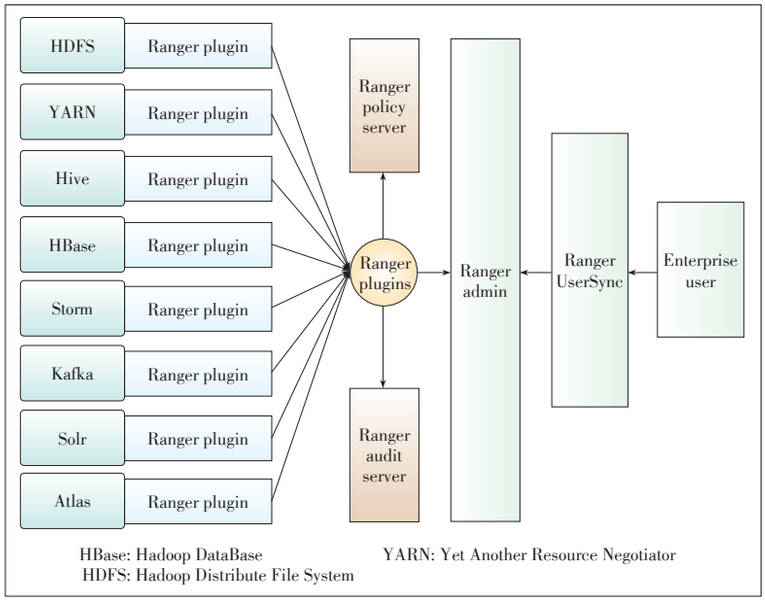
5.1 Apache Ranger

Apache Ranger is a centralized security management framework that provides centrally managed security policies and monitors user access. It supports fine grained authorization and auditing for Hadoop ecosystem components such as Hive and HBase. By operating the Ranger Web UI console, administrators can easily control the user's access rights by configuring policies. Compared to Apache Sentry, Ranger supports more services, including most of the Hadoop components like HDFS, HBase, Hive, Yarn, Strom, Kafka, Knox, Atlas, and Solr. Therefore, when a user needs to manage these frameworks, or when the entire big data ecosystem contains these frameworks, the user can use Ranger to control these frameworks' security.

As shown in **Fig. 5**, the Ranger architecture consists of three parts: Ranger Admin, Ranger Usersync and Ranger Plugins [26]. Ranger Admin is the core interface for security administration and is the center of the Ranger framework. Users can manage users' system rights on the Web UI provided by this service, and can create and update the authentication policies, which are stored in a policy database. Each component's plugin periodically monitors these policies. Ranger Admin also provides an audit service that collects data stored in HDFS or relational databases for auditing.

Ranger Plugins are the core of rights security management and is a lightweight Java program that can be embedded in each cluster component. For example, Ranger, for its highly supported Hive, provides a plugin that can be embedded in the Hiveserver2 service, which extracts the rights authentication policies for all Hives from the ranger admin service and stores them in a local file. When a user request comes through the component, the plugin intercepts the request and evaluates whether it meets the security policy. At the same time, the plugin can also collect data from user requests and create a separate thread to send the data back to the audit server.

Ranger Usersync is a very important tool for synchronizing users/groups from UNIX systems or Lightweight Directory Ac-



▲Figure 5. Ranger architecture.

cess Protocol (LDAP) to Ranger Admin. This stand-alone process can also be used as an authentication server to log into Ranger Admin by using a Linux user/password. The user or group information is stored in Ranger Admin for policy definition. Moreover, users can manually add/delete/modify user or group information to set permissions on these users or groups.

From the perspective of the rights model, Ranger controls the component's rights through centralized access controlling. Moreover, authorization is defined by "User – Resources – Permissions" between the three relations. Ranger abstracts this relationship and then extends users' own authority models. The "User–Resources–Permissions" has the following definitions:

1) User and group: User represents a user who is accessing the resource, while group refers to the one the user belongs to.
2) Resources: Using the tuple (Service, Resource), a policy only corresponds to a service, but can correspond to multiple resources.
3) Permissions: Expressed by the tuple (AllowACL, DenyACL), both of which contain two sets of AccessItem. AccessItem describes the relationship between a set of users and a set of accesses. AllowACL indicates that permission is allowed, and denyACL denies the permission.

Table 2 lists the model entity enumeration values for several common components.

In the recently released Ranger version, Apache Ranger has added Atlas components, integrating Atlas to support classification-based (tagging) and other dynamic policies (based on location, prohibition, and data lifecycle). This is the first time that Apache Ranger for security and Apache Atlas for data governance are integrated to authorize customers to define and implement security policies based on dynamic classification. Ranger's centralized platform enables data administrators to define security policies and apply the strategy to the entire hierarchy of data assets, including databases, tables, and columns, based on Atlas metadata tags or attributes. Ranger today has important features, but there are still some questions about how to adapt to the larger Hadoop security ecosystem. For example, some Ranger targets overlap with the targets of Apache Sentry (see the next section for details), and there seems to be little consensus about how the project synchronizes its work.

5.2 Apache Sentry

Apache Sentry, similar to Apache Ranger, performs fine-grained access control on the Hadoop ecosystem components, such as Hive and Impala. It also provides control and implementation of data for authenticated users and applications on the permission control function of Hadoop clusters. In the existing group mapping environment of the Hadoop ecosystem, it is easy to manage permissions by simply manipulating the unique role of Sentry.

Sentry mainly consists of three components: the server, data engine and plugin. The Sentry server manages the policy metadata, which supports the interface for safe retrieval and manipulation of metadata. The data engine is a data handler that requires authorization to access data or metadata, such as Hive and Impala. The data engine loads the Sentry plugin and intercepts all client requests that access the resource. Moreover, the requests are validated by the Sentry plugin. The Sentry Plugin runs in the data engine. It provides an interface to manipulate the authorization metadata stored in the Sentry server and includes an authorization policy engine that evaluates the access request by using the authorization metadata retrieved from the server.

In fact, the primary purpose of the Sentry server is to facilitate the management of metadata, and real authorization decisions are made by the policy engine in the Sentry plugin. The Sentry architecture (**Fig. 6**) has three important layers:

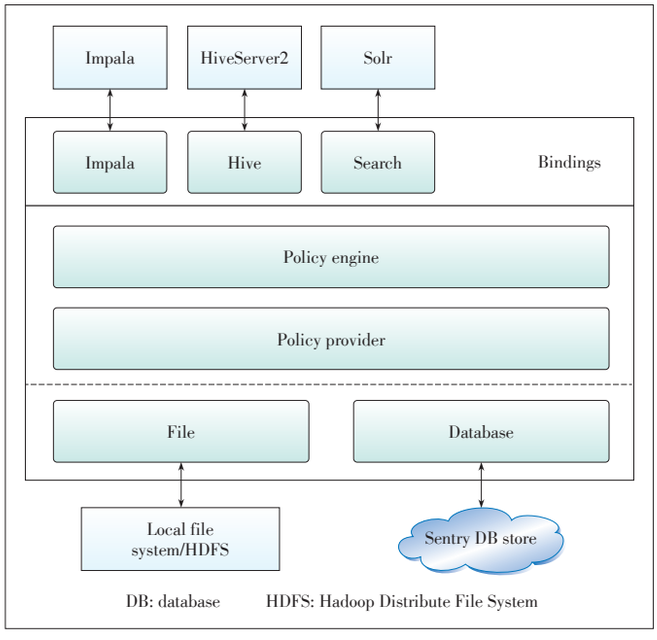
1) Bindings Layer

As mentioned earlier, Sentry's policy engine is part of the Sentry plugin, called by the Impala, Hive and Search components. The bindings layer is the bridge between the Sentry authorization and the invoking tools like Impala, HiveServer2,

▼Table 2. Ranger model entity enumeration values

Service	Resource	Authority
HDFS	Path	Read; Write; Execute
YARN	Queue	Submit; Admin
HBase	Table; Column Family; Column	Read; Write; Create; Admin
Hive	Database; Table; Column	Select; Update; Create; Drop; Alter; Index; Lock

HBase: Hadoop DataBase
HDFS: Hadoop Distribute File System
YARN: Yet Another Resource Negotiator



▲Figure 6. The architecture of Apache Sentry [27].

and Solr, which is responsible for converting the native format of authorization request to the request that can be processed by the Sentry policy engine.

2) Policy Engine

This is the heart of the Sentry, which obtains the privilege from the bindings layer and obtains the required privileges from the policy provider layer. It compares the requested and required permissions and determines whether the operation should be allowed.

3) Policy Provider

The policy provider is an abstraction that makes the authorization metadata available to the policy engine. It allows to use metadata regardless of how metadata is stored. Currently, Sentry supports file-based storage and relational database storage. A file-based scenario is to store metadata in a file format. The file can be stored in the local file system or HDFS. The file contains the group, role and privilege between the two groups of maps. However, it is difficult to use the program to modify the file, because there is competition for resources, and it is not conducive to maintenance. At the same time, Hive and Impala need to provide industry-standard SQL interface to manage the authorization strategy, requiring the use of programming the management.

Apache Sentry and Ranger are very similar in many functions, such as support for fine-grained access control, secure authorization mechanism, and support for multiple Hadoop components. The main difference is that Ranger was originally developed by Hortonworks, while Sentry was originally developed by Cloudera. However, both of them now belong to the Apache Foundation's incubation program. In contrast, Ranger is more comprehensive, which may be better for the industry. However, Cloudera has previously announced a major plan for security and Sentry is one of the beneficiaries of this "one platform" strategy. The prospects of Sentry are immeasurable.

5.3 Kerberos

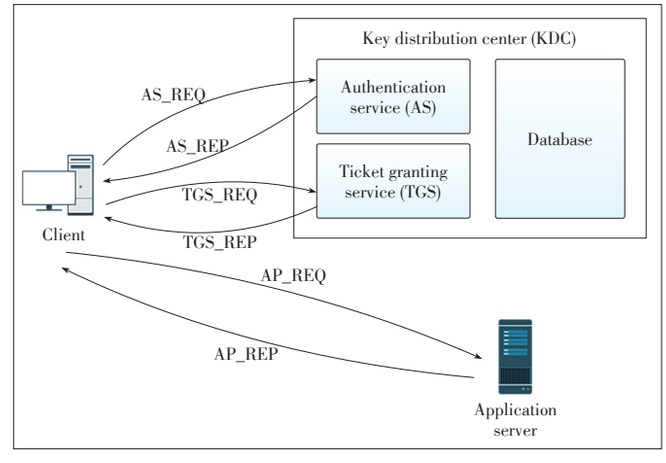
Kerberos is a computer network authorization protocol used to authenticate personal communications over non-secure network environments [28]. The implementation of the Kerberos authentication process does not depend on the authentication of the host operating system. It requires neither trust based on the host address nor the physical security of all the hosts on the network. Kerberos is based on the assumption that the data packets can be arbitrarily read, modified and inserted on the network. Kerberos supports the integration of Hadoop's multiple components, including HDFS, Yarn, Hive, Zookeeper, HBase, Sqoop, Hue, Spark, Solr, Kafka, Storm, Impala, etc.

The user uses the principal to authenticate through the Kerberos client. After the authentication is successful, the server will return the authenticated ticket to the user, who will use it for secure communication.

Kerberos supports Windows, Linux and Mac OS systems. It is often used in such systems as Web applications and enterprise networks, which require high security. Many companies such as Microsoft, Apple, and Red Hat have used Kerberos products; Kerberos also plays a very important role in the X-Box and cable television industry. Therefore, it can be said that Kerberos is one of the most widely used authentication methods in the history of computer networks.

Kerberos consists of the key distribution center (KDC) and the client and application server (**Fig. 7**). KDC provides the authentication service (AS) and ticket granting service (TGS). The specific process of certification is as follows:

1) A client sends an authentication request AS_REQ to AS. AS returns AS_REP, which includes a session key SK_TGS that is generated by the user and TGS and sends the ticket granting ticket(TGT) and SK_TGS encrypted with the user key.



▲Figure 7. The architecture of Kerberos.

2) The client sends TGS_REQ to TGS as requesting a service ticket (ST) for accessing an application server, and then sends TGT and the authenticator. The authenticator is used to verify that the user who sent the request is the user declared in the TGT.
3) If TGS judgment is correct, a new session key SK_Service will be generated for the user and the application server and then TGS_REP be sent to the user, including SK_Service and ST.
4) The user uses the session key SK_TGS to unlock the packet and get the session key SK_Service. SK_Service is then used to generate an authenticator andST and the authenticator are sent to the application server.
5) Authenticator is encrypted using the session key (SK_Service) between the user and the application server. The application server receives the key decryption ST or the session key SK_Service, and then use the session key (SK_Service) to decrypt the authenticator to verify that the user who sent the request is the user declared in the ticket.
6) The application server sends a packet to the user to prove his identity, which is encrypted using the session key (SK_Service). The client waits for the application server to send a confirmation message. Ifthe application server is not correct, it cannot unlock the ST, nor get the session key, so as to avoid the use of a wrong server. After that, the user and the application server can use SK_Service to communicate, and in the TGT validity period, the user will skip the first step of the authentication and directly jump to the second step by using TGT to prove their identity.

Although Kerberos is a high-performance security encryption system, it also has some problems if used improperly or its management is neglected. For example, if we observe the Kerberos authentication process, we can find that the Kerberos service is almost entirely dependent on the services on the KDC. Once the host of the KDC is down, all Kerberos-enabled services are not available.

5.4 Comparison of Security Frameworks

In this section, we compare the functionalities supported by Apache Ranger, Apache Sentry, and Kerberos frameworks.

As shown in **Table 3**, Ranger and Sentry have quite similar functionalities, both to provide audit log services, fine-grained authorization, unified authorization management strategy, and role-based management. They cannot support module authentication, nor the ticket grant services. However, Ranger supports more big data open source components than Sentry. Sentry only supported Apache HDFS, Apache Kafka, Apache Solr, Apache Sqoop, and Cloudera Impala by December 2016. As a comparison, Ranger also supports other components, i.e., HBase, Solr, Storm, and Atlas. Compared with Ranger and Sentry, Kerberos is mainly used to authenticate the above-mentioned components, but it does not support fine-grained permission control. Kerberos provides the ticket authorization service and the component authentication as well. It basically supports all big data open source components, as long as these components can use Kerberos authority for identification. To a certain extent, it is worth noting that Apache Ranger and Apache Sentry's functionalities overlap, and the users can choose them based on her own experience. However, Ranger supports more components than Sentry, and thus it may be a better choice in production environments.

▼Table 3. Comparisons between the three security frameworks

Services and Features	Ranger	Sentry	Kerberos
Audit log service	√	√	√
Fine grained authorization service	√	√	
Authentication service			√
Unified authorization policy	√	√	
Ticket granting service			√
Role-based management	√	√	
Supported components	9	5	12

5.5 New Progress of Ranger, Sentry and Kerboros

Ranger 0.6.0 was released in August 2016. It removes the support of database-based auditing, uses Solr as the index audit data, and HDFS to store audit data. The purpose of using HDFS is that the Ranger plugin can expand the audit log and index, when a new Ranger plugin is incorporated.

In early 2016, Sentry successfully graduated from the Apache incubation with six releases and continued to grow to provide unified authorization policy management across different Hadoop components. It is now targeting significant enhancements across the areas of 1) ease of Sentry enablement and management of permissions, 2) feature parity with access control capabilities of mature relational database systems, 3) attribute-based access control (ABAC), including permissions based on data sensitivity tags, and 4) integration with additional Hadoop ecosystem frameworks, so that existing permissions can be enforced across additional access paths [29].

Finally, Kerberos is designed to provide strong authentication for client/server applications by using secret-key cryptography. The availability of krb5-1.15.1 has been recently released. Now, the detached PGP signature is available without going through the download page, if one wishes to verify the authenticity of a distribution you have obtained elsewhere.

6 Future Work

Based on the above descriptions, the entire big data ecosystem can be somehow secured when we use Apache Falcon and Apache Atlas on the Hadoop ecosystem to govern the big data, use Apache Ranger and Apache Sentry on the application for security authentication and rights management, and use Kerberos to secure network transmission. However, there are still

some open problems and challenges in the process of big data governance and security.

6.1 Data Privacy Protection and Security

The existing data security protection methods are not effective enough to solve the multi-dimensional security of big data. The security frameworks described in this paper address neither the security of data semantics for access control, nor the access authorities for data owners. Moreover, conventional security scanning methods take too much time to process massive amount of data. Furthermore, current user data collection, storage, management and use are not standardized, and lack of supervisions, which mainly relies on self-disciplines of enterprises. Furthermore, end users cannot determine their own use of privacy information. Users have the right to decide how their information is used to achieve certain level of controllable privacy protections.

6.2 Secured Data Storage of Both Relation and Non-Relational Data

The data scale can easily reach the size of PB level, and therefore, massive data storage system should also have the appropriate level of scalability. The Internet has enforced the data to develop toward heterogeneous, unstructured, and other heterogeneous data such as images, video, audio, and text, growing at an alarming rate every day. The increasing heterogeneous data make the secured storage a challenging problem. That is, traditional Relational Database Management System (RDBMS) and emerging NoSQL databases have different security protection methods, at different scales, and of different applicability. However, certain degree of transparency is highly expected for companies because they do not care about which security protection method is used and how to store/retrieve the data. This requirement demands a unified middleware that supports secured data storage of both relational and non-relational databases.

6.3 Credibility of Big Data

There can be a general view of the big data that the data itself can explain everything, and the data itself is the fact. However, the reality is that the data will also be deceived. This is a threat to the security of big data. One of the threats to the credibility of big data is forgery or deliberate manufacturing data, while erroneous data often lead to erroneous conclusions. If a data application scenario is clear, someone may deliberately create data, create a false impression, and induce analysts to come to the conclusion that is beneficial to them. However, false information is often hidden inside the data, which prohibits the accurate identification of its authenticity and thus makes false judgments. Furthermore, the emergence and fast spread of false information through online social networks significantly increases the difficulties of identification, which cannot be solved by current security techniques. Another threat to the credibility of big data is that data may be distorted in the process of transmission. This is because a data acquisition process usually involves human intervention that may introduce errors, data distortion and deviation, and ultimately affect the accuracy of data analysis results. Therefore, it is highly expected that standardized transmission specification is enforced.

6.4 Optimizing Big Data Access Control

Access control is an effective way for data-controlled sharing, since data may be used for a variety of different scenarios. The key challenge comes from how to preset the roles, or to divide the different roles before the data system runs. Due to the wide range of big data applications, data are usually accessed by different organizations or individuals, with different purposes. However, their specific authorization requirements and access control rights are usually unknown as a priori.

7 Conclusions

This paper first gives a definition of data governance and security, and proposes that the management of big data can be carried out from three aspects: governance principle, governance scope, and implementation and evaluation of governance. The audit of big data governance mainly focuses on the supervision and evaluation of big data management in five aspects: the audit of big data governance, the audit of big data management content, the audit of big data management, the big data security audit, and the big data lifecycle audit. In this paper, we introduce the data lifecycle management framework (i.e., Apache Falcon), life cycle management and metadata management (i.e., Atlas), and three security authentication frameworks (i.e., Ranger, Sentry, and Kerberos). Detailed analysis of all these frameworks have been made. Moreover, we discuss how these frameworks carry out the data Lifecycle management, and protection of data security and privacy in the Hadoop ecosystem. Finally, we suggest the future works of the data governance and security and conclude this paper.

References

[1] Z. J. Dong, "Improving performance of cloud computing and big data technologies and applications," *ZTE Communications*, vol. 12, no. 4, pp. 1–2, Dec. 2014.
[2] L. Douglas. (2001, Feb. 6). *3D data management: controlling data volume, velocity and variety* [Online]. Available: https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf
[3] S. Iwata, "Big data era," *Journal of Information Processing and Management*, vol. 55, no. 8, pp. 543–551, Jan 2012. doi:10.1241/johokanri.55.543.
[4] K. Sarvakar, "big data security and privacy using data transformation with role based access control," *International Journal of Computer Science & Communication (IJCSC)*, vol. 7, no. 2, pp. 90–94, Jul. 2016. doi: 10.090592/IJCSC.2016.115.
[5] S. B. Scruggs, K. Watson, and A. I. Su, "Harnessing the heart of big data," *Circulation Research*, vol. 116, no. 7, pp. 1115–1119, Mar. 2015. doi: 10.1161/CIRCRESAHA.115.306013.

[6] M. Jensen, "Challenges of privacy protection in big data analytics," in *IEEE BigData Congress*, Santa Clara, USA, Jul. 2013, pp. 235–238. doi: 10.1109/BigData.Congress.2013.39.
[7] F. Cang, M. Zhang, and Y. Wu, "Preventing data leakage in a cloud environment," *ZTE Communications*, vol. 11, no. 4, pp. 27– 31, Dec. 2013. doi: 10.3969/j. issn.1673–5188.2013.04.004.
[8] K. Setty and R. Bakhshi, "What is big data and what does it have to do with it audit?," *ISACA Journal*, vol. 3, no. 14, pp. 1–3, 2013.
[9] M. Anup, R. Nimje, V. T. Gaikwad, and H. N. Datir, "A review of various trust management models for cloud computing storage systems," *International Journal of Engineering and Computer Science*, vol. 3, no. 2, pp. 3924–3928, Feb. 2014.
[10] M. Al-Ruithe, E. Benkhelifa, and K. Hameed, "Key dimensions for cloud data governance," in *IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Vienna, Austria, Sept. 2016, pp. 379–386. doi: 10.1109/FiCloud.2016.60.
[11] M. Felici, T. Koulouris, and S. Pearson, "Accountability for data governance in cloud ecosystems," in *IEEE 5th International Conference on Cloud Computing Technology and Science*, Bristol, UK, Dec. 2013, pp. 327–332. doi: 0.1109/CloudCom.2013.157.
[12] A. Corradi, L. Foschini, A. Zanni, et al., "A federation model to support semantic SPARQL queries for enterprise data governance," in *Eleventh International Conference on Digital Information Management (ICDIM)*, Porto, Portugal, Sept. 2016, pp. 96–100. doi: 10.1109/ICDIM.2016.7829778.
[13] T. Priebe and S. Markus, "Business information modeling: a methodology for data-intensive projects, data science and big data governance," in *IEEE International Conference on Big Data*, Santa Clara, USA, Dec. 2015, pp. 2056–2065. doi: 10.1109/BigData.2015.7363987.
[14] M. Al-Ruithe, S. Mthunzi, and E. Benkhelifa, "Data governance for security in IoT & cloud converged environments," in *IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, Agadir, Morocco, Dec. 2016, pp. 1–8. doi: 10.1109/AICCSA.2016.7945737.
[15] R. J. DeStefano, L. Tao, and K. Gai, "Improving data governance in large organizations through ontology and linked data," in *IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, Beijing, China, Jun. 2016, pp. 279–284. doi: 10.1109/CSCloud.2016.47.
[16] A. Yulfitri, "Modeling operational model of data governance in government: case study: government agency X in Jakarta," in *International Conference on Information Technology Systems and Innovation (ICITSI)*, Bandung, Indonesia, Oct. 2016, pp. 1–5. doi: 10.1109/ICITSI.2016.7858207.
[17] R. Thiel, K. A. Stroetmann, and P. D. Singleton, "Clinical data governance: legal and ethical challenges," in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, Valencia, Spain, Jun. 2014, pp. 597–600. doi: 10.1109/BHI.2014.6864435.
[18] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, "Information security in big data: privacy and data mining," *IEEE Access*, vol. 2, pp. 1149–1176 Oct. 2014. doi: 10.1109/ACCESS.2014.2362522.
[19] H. Ye, X. Cheng, M. Yuan, et al., "A survey of security and privacy in big data," in *16th International Symposium on Communications and Information Technologies (ISCIT)*, Qingdao, China, Sept. 2016, pp. 268–272. doi: 10.1109/ISCIT.2016.7751634.
[20] S. Tan, D. De, W. Z. Song, J. Yang, and S. K. Das, "Survey of security advances in smart grid: a data driven approach," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 397–422, Oct. 2016. doi: 10.1109/COMST.2016.2616442.
[21] Apache. (2016, Sept. 20). *What Falcon does* [Online]. Available: https://falcon.apache.org/FalconDocumentation.html
[22] Hortonworks. (2016, Sept. 20). *Falcon* [Online]. Available: https://zh.hortonworks.com/apache/falcon
[23] Apache. (2016, Oct. 16). *Data governance and metadata framework for hadoop* [Online]. Available: http://atlas.apache.org
[24] Apache. (2017, Mar. 16). *Atals architecture* [Online]. Available: http://atlas.apache.org/Architecture.html
[25] M. Rouse. (2015, May 23). *Data-lake* [Online]. Available: http://searchaws.techtarget.com/definition/data-lake
[26] Hortonworks. (2015, Dec. 1). *How Ranger works* [Online]. Available: https://hortonworks.com/apache/ranger
[27] Cloudera. (2016, May 23). *Sentry* [Online]. Available: http://www.cloudera.com/content/cloudera/en/products-adn-services/cdh/sentry.html
[28] J. Kohl and C. Neuman, "The kerberos network authentication service," Internet RFC 1510, Sept. 1993.
[29] Apache. (2016, Mar. 25). *Apache Sentry* [Online]. Available: https://blogs.apache.org/sentry/entry/sentry_graduates_to_a_ top?platform=hootsuite

Manuscript received: 2017-06-15

Biographies

HU Baiqing (baibenny@foxmail.com) received his B.Eng. degree in software engineering from Wuhan Textile University, China in 2016. He is pursuing an M.Eng. degree at Beijing Institute of Technology, China, with a major in software engineering. His research interests include cloud computing, big data, and the Internet of Things.

WANG Wenjie (wangwj1203962899@gmail.com) received his B.Eng. degree in software engineering from Chongqing University, China in 2016. He is pursuing an M.Eng. degree at Beijing Institute of Technology, China, with a major in software engineering. His research interest is the security of big data.

Chi Harold Liu (chiliu@bit.edu.cn) received his B.Eng. degree in electronic and information engineering from Tsinghua University, China in 2006, and Ph.D. degree in electrical engineering from Imperial College, UK. He is currently a full professor and Vice Dean of School of Computer Science, Beijing Institute of Technology, China. His research interests include big data and the Internet of Things.