Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Privacy Aware Access Control for Big Data: A Research Roadmap

Pietro Colombo*, Elena Ferrari

Department of Theoretical and Applied Sciences, University of Insubria, Via Mazzini 5, 21100 Varese, Italy

A R T I C L E   I N F O

Article history:
Received 15 April 2015
Received in revised form 12 June 2015
Accepted 3 August 2015
Available online xxxx

Keywords:
Big Data
NoSQL datastores
MapReduce systems
Data management
Data analytics
Privacy policies
Access control enforcement
MongoDB

A B S T R A C T

Big Data is an emerging phenomenon that is rapidly changing business models and work styles [1]. Big Data platforms allow the storage and analysis of high volumes of data with heterogeneous format from different sources. This integrated analysis allows the derivation of properties and correlations among data that can then be used for a variety of purposes, such as making predictions that can profitably affect decision processes. As a matter of fact, nowadays Big Data analytics are generally considered an asset for making business decisions. Big Data platforms have been specifically designed to support advanced form of analytics satisfying strict performance and scalability requirements. However, no proper consideration has been devoted so far to data protection. Indeed, although the analyzed data often include personal and sensitive information, with relevant threats to privacy implied by the analysis, so far Big Data platforms integrate quite basic form of access control, and no support for privacy policies. Although the potential benefits of data analysis are manifold, the lack of proper data protection mechanisms may prevent the adoption of Big Data analytics by several companies. This motivates the fundamental need to integrate privacy and security awareness into Big Data platforms. In this paper, we do a first step to achieve this ambitious goal, discussing research issues related to the definition of a framework that supports the integration of privacy aware access control features into existing Big Data platforms.

1. Introduction

In the recent years we have entered the Big Data era, which is rapidly and radically changing the way we live, work and think [1]. The Big Data term denotes a data management and analytics paradigm featuring 5V: huge data Volume, high Velocity (i.e., timely response requirements), high Variety of data formats, low Veracity (i.e., uncertainties in the data), and high Value [2]. Jin et al. [2] believe that Big Data are helping people to better understand the present and that this enhanced perception allows one to better predict the future. Big data play a key role for industrial upgrades. They will not uniquely sustain the growth of information industry, rather they will become a mean for improving their competitiveness. Big Data are source of business for IT giants, which are taking huge profit from the development of services and technologies (e.g., cloud-based storage and analysis services) providing the infrastructure to Big Data analytics and management. The availability of these services is continuously growing and is evolving towards real-time and ready-to-use solutions that radically change user experience wrt analysis and prediction. For instance, IBM Watson analytics[1] has been specifically designed to provide advanced, but easy to use, analytics and prediction services to managers.

Several companies which have integrated the use of Big Data analytics services into their processes, and which represent the users of the above mentioned services, are also significantly improving their business. For instance, companies that have invested on Internet of Things [3] technologies are radically changing the management of logistics and production processes getting benefit from the analysis of high volumes of sensed data and from advanced prediction features. According to Jin et al. [2], Big Data will even play a key role for national development, and according to their vision, data sovereignty of a country will be in the great power-game space together with resources such as land, sea and air.

Data managed by Big Data analytics platforms are of heterogeneous formats [4] and they can come from any source which can be sensed features of the physical world as well as information referring to social networks, internet, business, finance, economics or others. They can be structured, unstructured and semi structured data, such as transactions, electronic documents and emails. The joint analysis of data from different sources allows deriving

* Corresponding author. Tel.: +39 0332218927; fax: +39 0332218919.
E-mail addresses: pietro.colombo@uninsubria.it (P. Colombo), elena.ferrari@uninsubria.it (E. Ferrari).

[1] http://www-03.ibm.com/software/products/it/watson-content-analytics.

http://dx.doi.org/10.1016/j.bdr.2015.08.001
2214-5796/© 2015 Elsevier Inc. All rights reserved.

Big Data platform	Access control	Privacy policies
MongoDB (https://www.mongodb.org)	RBAC at database / collection level	No support
Cassandra (http://cassandra.apache.org)	RBAC at key-space / table level	No support
Redis (http://redis.io)	Access control can only be achieved at application level	No support
HBase (http://hbase.apache.org)	Access control lists at column family and/or table level	No support
CouchDB (http://couchdb.apache.org)	No native access control	No support
Hive (https://hive.apache.org)	Fine grained access control, relational model	No support
Hadoop (https://hadoop.apache.org)	Access control lists at HDFS resource level	No support
Spark (https://spark.apache.org)	Access control lists at resource level	No support

Fig. 1. Access control and privacy policies support within a selection of Big Data platforms.

valuable information, such as data correlations and properties that can be profitably used for making business decisions. Due to innovative computational paradigms that distribute data and analysis tasks over clusters of nodes, and simple but effective data models, Big Data analytics platforms outdo traditional Data Warehouses (DWs) and Database Management Systems (DBMSs) wrt scalability, performance, and high availability. Moreover, due to the high availability of cloud-based storage and analysis services, companies can either decide to handle their own private Big Data clusters within local server farms or use cloud-based services.

Big Data platforms can be classified according to the supported data model and system architecture into three main groups [5]: 1) massively parallel processing (MPP) systems, which include RDBMSs and DWs, which are basically distributed data management and analysis systems defined on top of the relational data model; 2) MapReduce based analytics platforms, denoted MapReduce systems in the rest of this paper for the sake of brevity, which comprise technologies supporting MapReduce based analysis (e.g., the Hadoop platform), and which operate by building key-value abstractions of raw data; 3) NoSQL datastores, which provide highly flexible, scalable, and efficient storage and analysis services based on different data models such as the key value, document oriented and graph based one.

Although the potential benefits of Big Data have attracted several companies, the massive diffusion of Big Data technologies is hindered by the lack of proper data protection tools [6]. As a matter of fact, today Big Data platforms integrate quite basic form of access control and data protection features [7], but no support for privacy policies. An overview of the current situation is shown in Fig. 1, which summarizes both the access control features and the support for privacy policies for a selection of popular NoSQL datastores and MapReduce systems. The poor support for security and privacy enforcement represents a severe shortcoming of Big Data platforms as the managed data typically include sensitive and personal information. Due to the huge volume of data typically stored and the advanced analysis and prediction services provided by these platforms, relevant privacy and security threats arise.

This issue cannot be easily addressed reusing privacy aware access control (PAAC) frameworks proposed for traditional data management systems (e.g., [8–10]). We believe that solutions developed for traditional data management systems can be adapted for MPP systems only, due to the many shared characteristics with traditional DBMSs (e.g., the use of the relational data model, and SQL as query language). In contrast, the innovative features of MapReduce systems and NoSQL datastores along with the variety of data models and query languages introduced for them make the definition of PAAC solutions a new and ambitious research goal, which we start investigating in this paper.

Privacy issues are far more difficult to be addressed within Big Data systems than in the context of traditional DBMSs and DWs. These aspects have been thoroughly analyzed in a report recently prepared by the President's Council of Advisors on Science and Technology for the President of United States [11], which analyzes the relation between Big Data and privacy from a technological perspective. In addition, enforcement techniques proposed for traditional DBMSs appear inadequate for the Big Data context, due to the strict performance requirements needed to handle large data volumes, the heterogeneity of the data, the speed at which data are generated and must be analyzed (analytics are moving from batch to real-time [6,12]), and the distributed nature of these systems. Moreover, so far no standard language and data model for Big Data platforms has yet emerged. The variety of query languages and data models proposed for different data stores make the development of a general privacy-aware access control enforcement solution even more ambitious.

The overall goal of this paper is thus exploring the definition of PAAC mechanisms suited for Big Data platforms and the related enforcement mechanisms. Policy specification, management and enforcement will be exemplified through a framework, whose architecture will be presented in Section 2. Finally, we discuss how the framework can be applied to MongoDB,[2] which is considered today the most popular NoSQL datastore.[3]

The remainder of the paper is organized as follows. Section 2 presents the reference architecture of a framework supporting PAAC for Big Data. Section 3 discusses research goals related to the definition of the framework, whereas Section 4 provides preliminary considerations related to the instantiation of the framework for MongoDB. Finally, Section 5 concludes the paper.

2. The reference architecture and roadmap

We believe that PAAC functionalities can be integrated into target Big Data platforms by means of a framework consisting of: 1) a language for policy specification, 2) tools for policy management, 3) proper enforcement mechanisms, and 4) suitable enforcement monitors implementing the devised mechanisms. As previously mentioned in Section 1, we focus on platforms belonging to the category of MapReduce systems and NoSQL datastores, since we believe that solutions targeting MPP systems can be defined through the extension of mechanisms already proposed for traditional data management systems. A high level view of the reference framework architecture for PAAC support into Big Data platforms is shown in Fig. 2. The framework has been conceived as an application running on a host separated from the cluster of a target Big Data platform. The application provides a dashboard of functionalities allowing the management of privacy policies and enforcement monitors specialized to different Big Data platforms. The dashboard functionalities include: 1) the specification/update/deletion of privacy policies for a target platform, 2) the management of enforcement monitors implementing proper policy enforcement mechanisms for the considered platform, and 3) the assessment of performance impact of policy enforcement.

Framework development can be articulated into several macro activities, each consisting of a set of tasks, which are summarized in Fig. 3.

First of all, an in depth review of existing Big Data platforms is required to identify, on the basis of their popularity, intrinsic char-

[2] http://www.mongodb.org.
[3] According to http://db-engines.com/en/ranking.

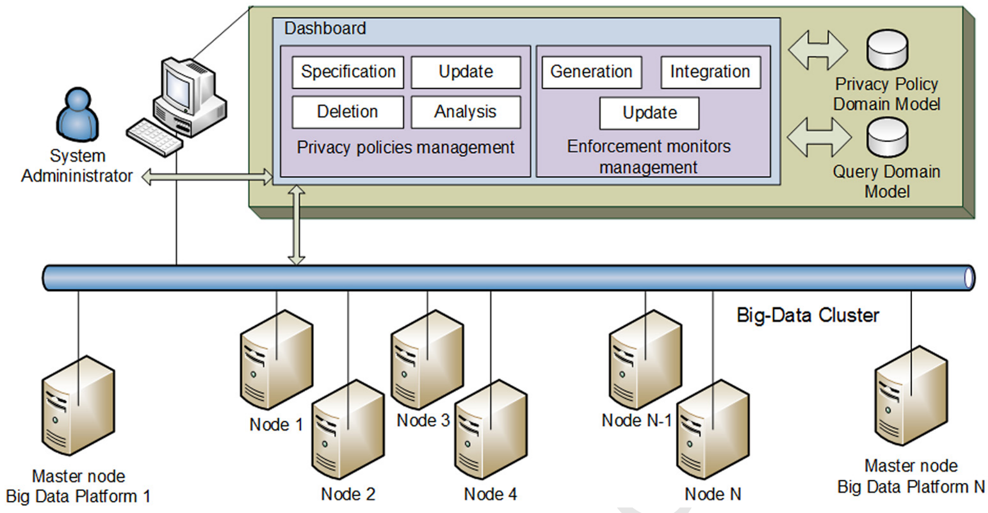

Fig. 2. A high level view of the framework architecture.

Main activities	Tasks
Platform selection and analysis	Selection of existing MapReduce systems and NoSQL datastores Platform classification based on data model, query language, use, and diffusion Analysis of native access control and data protection features
Identification of policy components	Review of the literature on privacy policies Identification of conceptual elements characterizing privacy policies for Big Data Specification of the privacy policy domain model
Identification of criteria for policies specification	Identification of policies specification granularity, data binding and policy encoding criteria
Definition of enforcement mechanisms	Definition of a model of the actions that can be performed on data by queries and analytics tasks Definition of privacy and action aware access control mechanisms based on state of the art or novel enforcement techniques
Framework development	Design and implementation of software modules supporting the management (specification, update, deletion, impact analysis) of privacy policies for the considered platforms Definition of systematic approach to the development and specification of enforcement monitors and the integration of such monitors into the target analytic platforms Design and implementation of software modules supporting the management (generation, integration, update, removal) of enforcement monitors

Fig. 3. The proposed roadmap.

acteristics, and functionalities, a selection of relevant MapReduce systems and NoSQL datastores. The identified platforms need to be classified wrt the supported data model, the query languages, the supported type of queries and analytics tasks, with the aim of identifying features characterizing classes of platforms. An in depth analysis of the native access control functionalities of the selected platforms is required to classify them also wrt the already provided data protection features. PAAC can then be defined for each of the identified classes.

Then, in order to identify which privacy policies for Big Data platforms should be supported, it is required to review the literature on privacy policies for data management systems. We believe that a domain model of privacy policies should be defined, which constrains which privacy policies can be expressed and how they can be supported in a target platform (see Section 3.2 for more details). The analysis of policy support requires reasoning on aspects like policy components specification granularity, data binding, and policy encoding.

An additional objective is related to the definition of proper enforcement mechanisms for the supported policies. We believe that a first point to achieve this goal is defining a model of the actions that queries/analytics functions can execute on data within platforms belonging to any of the previously identified classes. Indeed, as discussed in [13], the awareness of the actions that are executed on data can profitably contribute to support PAAC. PAAC enforcement mechanisms can then be defined exploiting state of the art techniques, such as query rewriting or filter-based access control to system resources, or defining novel techniques. The final goal is regulating the access on the basis of the compliance of the access that should be performed by queries/analytics tasks with the specified privacy policies.

Once all these basic research goals have been fulfilled, development activities related to the definition of the above mentioned dashboard functionalities can be easily achieved.

3. Research goals

In this section, we discuss relevant goals and research issues related to the definition of the framework presented in Section 2. More precisely, we present a selection of 11 research goals concerning the definition of specification, binding and enforcement mechanisms for privacy policies.

A summary of the selected goals is shown in Fig. 4. The figure also shows how goals achievement is related to the main activities of the roadmap introduced in Section 2 and illustrated in Fig. 3.

In what follows, we will discuss in details the identified goals for each main activity of the proposed roadmap.

3.1. Platform selection and analysis

The selection and analysis of the existing Big Data platforms aims at identifying classes of platforms with common characterizing features (e.g., data model, query language, and native access control mechanisms). The final objective of the classification is

Roadmap activity	Goal	Description
Platform selection and analysis	1	Leveraging native access control features
Identification of policy components	2	Purpose based access control
	3	Action aware access control
	4	Content and context based constraints
	5	Profile based constraints
Identification of criteria for policies specification	6	Beyond the granularity of data models
	7	Efficient policy binding strategies
Definition of enforcement mechanisms	8	Policy derivation on support of incremental analysis
	9	Definition of the query domain model
	10	Query rewriting for NoSQL datastores
	11	Resource based key value filters for MapReduce systems

Fig. 4. Allocation of the main research goals to roadmap activities.

researching general PAAC enforcement solution which can be efficiently used with all platforms belonging to the same class.

Goal 1: Leveraging native access control features

We believe that, independently from the specific supported privacy-aware access control features, a key requirement for PAAC is that it should be developed on top of the native protection mechanisms of the considered platforms, rather than being developed from scratch. Indeed, the basic protection functionalities of the considered Big Data platforms can be profitably reused and extended to provide more advanced data protection services. This will also make the developed solution more acceptable and in line with the business model of Big Data platform vendors, in that it does not require a complete re-design of the already supported access control and privacy preserving services. Therefore, the goal should be to study how the native access control mechanisms provided by some Big Data platforms can be enhanced with privacy aware mechanisms. Moreover, it is important to provide reliable solutions that can work with multiple versions of the same platforms.

For instance, let us consider the case of MongoDB, which enforces role based access control at collection level. It is well recognized (see e.g., the proposal by Byun and Li [14] for RDBMs), that a first step towards making a data management system privacy-aware is to be able to support purposed-based fine-grained access control (FGAC), since FGAC enables to enforce different privacy preferences for different users and for different access purposes. Therefore, the first step for making MongoDB privacy-aware is refining the granularity of access control to the level of field, and integrating the required support for purpose aware access control. In this case, all functionalities related to user authentication and role management, which are natively provided by MongoDB, can be reused within the enhanced purpose and role based model. The same reasoning can also be applied to other NoSQL datastores (e.g., Cassandra, CouchDB) and to MapReduce systems as well. We are not aware of any PAAC model for MapReduce systems, however, this non-substitution requirement has already been applied for integrating fine-grained access control (FGAC) into Hadoop [15]. Indeed, in [15], the basic access control features of Hadoop are still used to regulate the access to HDFS resources. However, field level access control is applied to fields of the key and value objects derived from the accessed resources by RecordReaders.

3.2. Identification of policy components

A fundamental research objective is identifying what privacy policies for Big Data should actually be supported.

We are not aware of any PAAC model for Big Data platforms. To the best of our knowledge, the only related contribution is a very recent work by Sen et al. [16], who proposes a framework to integrate privacy policy compliance checking by means of information flows analysis in Bing. Sen et al. [16] focus on auditing techniques rather than access control, and policy enforcement is based on the analysis of previously tracked execution flows. Policy specification is supported by an ad hoc defined language, called Legalease, which allows specifying allowed and denied flows by composing configurable domain specific attributes. As such, although this work represents a very relevant contribution for integrating privacy awareness with an auditing based approach, it does not help identifying the policy components that can be effectively used for access control purposes.

We believe that, in order to identify the target features of privacy policies, it is first required to check whether the policy models proposed for traditional data management systems can be used within Big Data platforms as well.

In the literature on data management systems, privacy policies are characterized by several dimensions, such as purposes, obligations, retention, conditions, and actions [17]. Well known privacy policies specification languages, such as XACML [18] and EPAL [19], integrate support for all these concepts. However, the purpose is considered the key component of any privacy policy [14], and several access control mechanisms and policy models have been defined around this concept.

For instance, [14] proposes a purpose-based access control model for RDBMSs, where the access is regulated based on the compliance of the purposes for which data have been collected with those for which queries aim at accessing them. This seminal work has inspired several PAAC models. For instance, [20] proposes a conditional purpose-based access control model (CPBAC) which extends [14] with conditional purposes. Some work propose to combine purpose-based access control with role based access control (RBAC) [21], to better customize access regulation. For example, [22] extends CPBAC to Role-involved Purpose-based Access Control (RPAC), whereas [23] proposes a family of models, called Conditional Privacy-aware Role Based Access Control (P-RBAC), which extend RBAC integrating concepts like purposes and obligations, allowing the specification of privacy policies. Also [24] proposes a purpose-based access control model that extends RBAC and is characterized by the dynamic association of access purposes to user queries, based on system and user attributes. In a previous work [10], we propose a framework for the automatic generation of enforcement monitors for purpose and role based privacy policies and their integration into RDBMSs. In [25] we have extended the framework in [10] to support obligations.

Goal 2: Purpose based access control

Purpose-based models are expected to be easily integrable into NoSQL datastores and MapReduce systems. Indeed, purposes can be modeled as independent privacy metadata, purpose-based policies can be specified as lists of authorized purposes, and enforcement mechanisms as simple checks that verify that a given purpose element belongs to the list of authorized purposes. As such, similar to the case of traditional data management systems, we believe that the concept of purpose should represent the core element of the privacy policy domain model for Big Data platforms.

Conversely, some other components of privacy policies that have been considered within traditional data management systems appear unsuited to analytics scenarios. For instance, retention policies require data deletion. However, due to data distribution and redundancy in a Big Data ecosystem, there is no way to ensure that data are completely destroyed [11]. It is worth noting that this issue may not affect specific scenarios, such as those where the Big Data datasets are static. For similar reasons, we also believe that obligations [26] appear as not suited to analytics platforms, as, according to [27], the enforcement of policies including obligations requires to keep track of the actions performed within a system and to update data accordingly, and this may be impractical within analytics systems, especially if temporal and periodical obligations are considered [25].

Goal 3: Action aware access control

A very recent PAAC model proposed for relational DBMSs supports policies that constrain specific actions performed by queries (e.g., aggregations) on data belonging to specific categories (e.g., sensitive, identifiable) [13]. For instance, it is possible to specify a policy which allows the joint access of sensitive data and identifiable data only if their value is not directly shown in the result set of queries.[4] These innovative features allow reaching high level of access control customization. We believe that similar mechanisms can be profitably used within NoSQL datastores as well.

However, relevant engineering issues make this goal very ambitious. Indeed, different from SQL queries where the actions can be identified through the analysis of the clauses of SQL statements, which are reasonably easy to parse and analyze, similar tasks are challenging within NoSQL datastores. For instance, the identification of basic actions may be a complex task for all NoSQL datastores that support MapReduce jobs (e.g., MongoDB, Cassandra). Indeed, the use of scripting languages for job definitions drastically complicate the identification of the actions executed on the accessed data. This situation is far more severe for MapReduce systems as typically the source code of MapReduce jobs is not available. For instance, in Hadoop a MapReduce job is a binary jar which aggregates the bytecode of all the job related Java classes.

Goal 4: Content and context based constraints

An additional aspect that should be considered is the possibility of supporting content and context based constraints. These constraints promise to play a key role within Big Data platforms as they allow reaching highly customized access control levels, also handling scalability issues that typically affect systems where authorizations are statically defined. Indeed, due to the huge number of policies that regulate the access to Big Data platform datasets and the number of users that may access these datasets, it can be hard to foresee and predefine all user authorizations, and manually assigning or revoking them when scenario dependent conditions are met. Context and content based constraint allows handling this complexity, regulating authorizations automatically on the basis of the satisfaction of conditions related to specific application scenarios.

Content-based constraints allow specifying access control conditions on intrinsic properties of the stored data. For instance, supposing that our target dataset is composed of all the emails exchanged in a company, a privacy policy could specify that the only accessible emails are those received by a given employee and the access is only granted for administrative purposes. In this case, the content based constraint restricts the access to a unique receiver, which is a mandatory field of every email header.

[4] Data can only be accessed if the result set shows aggregated values.

In contrast, context based constraints allow specifying access control conditions built on scenario specific contextual information. For instance, referring to the previous example, it could be required that the access is only allowed using devices with specific IP addresses. Obviously, to support context-based policies it is also required to support the modeling of contextual information for a specific application scenario as well as defining proper mechanisms for retrieving context information (e.g., the secure and trustworthy derivation of the used IP address). The wide literature on context based access control includes models designed for different application scenarios (e.g., [28,29]). However, to the best of our knowledge we are not aware of proposals targeting context awareness within Big Data platforms. A further issue is related to the selection of proper context modeling frameworks (see [30] for an overview of the existing ones) or the definition of new modeling solutions. It is worth noting that NoSQL datastores provide support for the execution of geo-spatial queries (e.g., MongoDB) and temporal queries. These features can be used for context aware access control enforcement. For instance, a policy may require that the position of the requester must be within a given area.

Goal 5: Profile based constraints

Besides using information related to the context within which the execution is requested, PAAC should also consider all information related to the requesting users. This may be achieved adopting solutions similar to those proposed for trust negotiation systems [31], where users are modeled by means of a profile, consisting of several attributes.

The characterization of users in terms of profiles allows specifying attributes based constraints which promise to reach highly customized access control levels. We believe that, in order to reach yet higher customization levels, all previously introduced elements should be combined together.

3.3. Identification of criteria for policies specification

Another challenging aspect related to the specification of privacy policies is identifying the granularity level at which the policies should be specified. As we mentioned before, FGAC has been recognized as a fundamental requirement for the support of privacy policies [17]. However, FGAC is not supported by the majority of Big Data platforms, and this is listed among the top challenging security issues identified for this application scenario [32].

Recently, some papers tried to fill this void. Kulkarni [33] proposed a FGAC model for key value systems, which operates at the granularity level of keyspace, column family, rows, and columns. FGAC is integrated into Cassandra by modifying the source code of this data store substituting the original access control model with an ad-hoc implemented one. The proposed solution suffers from severe portability issues even across different versions of the same platform. Shermin [34] presents the desired characteristics of a fine grained context aware RBAC model for NoSQL databases. However, the proposed mechanisms are only abstractly discussed and the author does not present or discuss implementation related aspects.

For what MapReduce systems are concerned, FGAC has been only recently considered, and although few Hadoop-based datastores have been specifically designed to integrate FGAC, such as Accumulo,[5] and Sentry,[6] to the best of our knowledge only seminal work (i.e., [35] and [15]) have targeted the direct integration of FGAC into Hadoop. More precisely, in [35,15] an aspect oriented technique is proposed to regulate the data records that are processed by the submitted MapReduce jobs. However, [35,15] are

[5] https://accumulo.apache.org/.
[6] https://sentry.incubator.apache.org/.

only suited to integrate FGAC into Hadoop based analytics platform, assuming that specific Hadoop Java classes are used. In addition no specific support for privacy policies is provided. As such, more general privacy aware solutions are still required.

Goal 6: Beyond the granularity of data models

Within the Big Data scenario, it may be quite challenging identifying the finest granularity of data at which the access should be regulated. The identification of the granularity level to be used for policy specification depends on the data model of the considered platforms. The data model of both MapReduce based systems and NoSQL data stores could be brought back to data records, therefore the corresponding granularity level reaches the level of record fields. However, the granularity could also go beyond the limits imposed by the data structures, and finest levels should also be considered. Indeed, an unstructured data field can be structured based on patterns. For instance, a raw textual field could be seen as the serialized content of a data record. Each field of such a record could be derived with regular expressions. Therefore, the structure of data that regulates the granularity level of privacy policies should be considered as a dynamic property derived from the actual content of the analyzed data. As such, a relevant research goal is supporting the specification of patterns which can be used to tune the granularity level of policy specification.

It is worth noting that this approach is not uniquely related to textual content, as patterns can be applied to any kind of binary representations, including images, audio, and multimedia streams. This may be achieved by specifying a policy stating that if a specific pattern classification algorithm applied to a given single resource r finds that r belongs to a known category, then specific access control actions should be taken. In other words, pattern classification algorithms may be dynamically associated with data resources on the basis of specific content based privacy policies. For instance, one may be interested to prune from a collection of images those that include human faces. This can be achieved by classifying images with face detection algorithms and then enforcing policies specified on the basis of the results of face detection.

Fine-grained policies have the potentiality to define personalized protection levels. However, at the same time, they complicate the specification and enforcement mechanisms requiring more computational effort and memory consumption. The identification of a proper trade-off appears as an additional challenging aspect to be investigated.

Goal 7: Efficient policy binding strategies

A relevant research related to policy specification is identifying how policies can be bound to data. Within relational DBMSs, privacy policies are specified as metadata, either stored in dedicated tables, or within table columns that are added to the scheme of the tables (e.g., see [14]). Different from traditional data management systems, within Big Data platforms, it may be quite challenging identifying where policies can be stored.

As far as MapReduce systems are concerned, we believe that access control metadata can hardly be stored together with the accessed data. Let us consider, for instance, the case of a file stored within the Hadoop file system, which is accessed by a MapReduce job. If policies are specified within the same file to be protected, only specific MapReduce jobs capable of recognizing the file format could be executed. The RecordReader should recognize and separate metadata from the file content. This would drastically restrict the admissible jobs to those integrating such RecordReaders, in this case the integration of PAAC would limit usability of the analysis. In addition, this solution would also require identifying proper encoding strategies. For instance, all metadata could be specified at the head/end of the file, or policies could be weaved together with the file content, meaning that the involved policies regulate the access to data included in the file portion that precedes/follows the policy specification. An alternative technique could be the one we have proposed in [15], where content based filters are applied to all key value pairs which are extracted from a given resource. Policies are stored separately from the accessed resource, and no constraint is imposed to the RecordReaders that can be used. Although in [15] the focus is on FGAC and not on privacy policies, we believe that a similar approach could possibly be used for privacy policies as well.

In contrast, NoSQL datastores are open to a variety of different possible solutions for policy binding. Similar to the above mentioned case, policies could be applied to data collections operating selection criteria at the level of single key-value pair/document. In other words, key-value pairs/documents referring to a specific collection can only be accessed if the privacy policy specified for the whole data collection is satisfied. An alternative solution is specifying privacy policies for single key-value pairs/documents composing data collections. In this case, similar to traditional data management systems (e.g., see [14]), the data unit (i.e., the document or key-value pair) could be simply extended with an additional field that keeps track of the privacy policies (e.g., using a dedicated field *policy*). Policies can be specified once, for all/group of fields, or can be specified for every field. Based on the number of fields and the hierarchical structure of fields, the policy specification can be quite complex. Thus, proper encoding choices need to be identified. The latter case can drastically complicate policy retrieval operations, as the data are typically schemaless, the structure of data units may change within the same collection, and the extensive analysis of the single data unit to derive its structure and where policies have been specified is expected to be a time demanding operation and probably not always achievable. For instance, to the best of our knowledge, with MongoDB the only way to derive the structure of a document is through Javascript code, which is only invokable within specific queries (e.g., MapReduce queries). As a consequence, a reasonable compromise among the considered alternatives could be an approach that combines policies specified at the level of data collections and data units. More precisely, policies at the level of data unit specify exceptions that refine the filtering operated at collection level by possibly altering the fields which are visible within an accessed document.

3.4. Definition of enforcement mechanisms

A key goal is the definition of proper enforcement techniques that will be implemented by the enforcement monitors. This objective is articulated into the definition of: 1) an approach for policy derivation and composition on support of complex forms of data analysis, 2) a model of the actions that queries/analytics tasks should execute on the resources to be accessed, and 3) mechanisms to regulate queries/analytics tasks execution on the basis of the specified policies.

Goal 8: Policy derivation on support of incremental analysis

Different from traditional database management systems, with NoSQL datastores or MapReduce systems, queries/analytics task result sets may not be entirely included in RAM, as they can be as big as the analyzed datasets. For this reason, Big Data platforms allow storing them on a storage support. For instance, MongoDB allows moving the result set of queries executed on a collection on a secondary output collection. These collections contain data which, either correspond to selected data items extracted from the original collection, or they are derived through data aggregation. In either cases, the resulting collection is stored and thus can be accessed by other queries. As such, besides regulating the access to data of the original collection, it is also required controlling the access to data derived from query execution.

This issue opens a further research goal, affecting both MapReduce systems and NoSQL datastores, which is related to the derivation of policies regulating the access to data composing the result set of queries.

Any possible derivation approach depends on policies granularity and criteria for policy assignment, i.e., where the policies are specified, as well as on the type of policies and the actions performed by the query. It is reasonable to assume that if the query performs a simple selection, the derived policies should correspond to the ones of the original data. Conversely, if any aggregation operation is performed (e.g., sum/count), any item in the result set corresponds to multiple items in the original dataset. Different straightforward composition options can be considered. For instance, the derived policy could be defined as the conjunction/disjunction of the original policies. However, both these options require policies to have specific characteristics, such as not to specify content or context based constraints, as both the context within which subsequent queries are executed and the data content can be completely different, and this may block the execution. Thus, the identification of proper derivation criteria appears as an open challenge.

Goal 9: Definition of the query domain model

Different queries that access the same data can threat privacy differently for the actions they execute on data [13]. For instance, let us consider an application scenario related to a MongoDB collection *employees* storing documents that keep track of employees salary. Let us consider the queries *qa : db.employees.find(*{},{ *name:1, salary:1*}*)*, which selects names and salaries of employees, and *qb : db.employees.aggregate(*[{*$group:* {*_id: null, avgsalary:* {*$avg : "$salary"* }}}]*)*, which calculates the average salary of the employees. These two queries might threat privacy in different ways as they disclose different pieces of information related to the stored data. *qa* shows an higher threatening level than *qb* as within the result set of *qa* the actual content of the fields *name* and *salary* of *employees* documents is explicitly shown, whereas the result set of *qb* only includes the average value of the field *salary* of *employees* documents.

In order to define proper PAAC enforcement mechanisms, it is first required to identify the actions that can be possibly executed by the queries/analysis tasks of the considered platforms.

A basic requirement for defining a PAAC model for Big Data platforms is defining the model of all types of actions that can be executed by queries of the target platform.

It is worth noting that the definition of the query model appears a complex task for BigData platforms due to characteristics of the data models. For instance, within RDBMSs, assuming that information related to tables schema is known, one can easily derive the columns that are/are not accessed by a query by means of straightforward SQL code analysis. In contrast, this cannot be easily achieved with schemaless NoSQL datastores. For instance, let us consider the case of MongoDB where the documents of a collection can all have a different structure. Even considering a simple query such as *db.collection.find()*, which selects all documents of collection, it is not possible to derive from the analysis of the query code which document fields will be accessed. Thus, other techniques must be defined, which derive these information at query execution time.

Another goal of the query model is identifying the actions that can be executed within different Big Data platforms by abstracting from syntactical aspects of the query languages. Indeed, we believe that the model should be defined for multiple NoSQL datastores and MapReduce systems generalizing as much as possible its definition.

Goal 10: Query rewriting for NoSQL datastores

The characteristics of the Big Data scenario, such as the distributed nature of the considered platforms, the use of proprietary query languages, and the focus on performance, make it impossible to reuse the same enforcement mechanisms that have been defined for RDBMSs. However, we believe that what has been defined for RDBMSs should be considered anyway, investigating how it can be adapted to the Big Data scenario or if completely novel approaches need to be defined.

The enforcement shall be defined in such a way that the complexity of the enforcement mechanisms should not compromise the usability of the hosting analytics frameworks. Based on the considered queries, the number of policies compliance checks to be executed during the query execution can match or be even greater than the number of data records, and in the Big Data scenario, data sets can include up to hundreds of millions of data records. The execution time overhead required for the enforcement shall be minimized. This requires the performance-oriented definition of the policy compliance mechanism.

Basically, the literature on relational DBMSs proposes two main approaches to the enforcement of fine grained access control, which can also support context and content based policies. With the first approach, users are only allowed to access portions of the target dataset that satisfy given policy conditions, whereas with the second one, each data item is labeled with metadata and the constrained elements can only be accessed if implicit conditions referring to these metadata are satisfied. The first approach, which is the one used by Oracle VPD,[7] is particularly suited to enforce content and context based policies specified at data collection level. The second approach, which is used by Byun and Li for their purpose-based access control model [14], is suited to enforce policies specified at the level of data unit. It is worth noting that, in this case, access purposes can be considered as the application context. For instance, a cell for which a given intended purpose is specified can only be accessed by a query if the access purpose of the query complies with such an intended purpose, thus, in this case, the implicit condition is purpose compliance.

The above mentioned approaches are implemented through the definition of authorized views or by means of query rewriting. The first technique requires to pre-calculate a view for each table of a target database and execute the submitted queries on these views rather than on the original tables. This quite naive approach suffers from a very high overhead, which hinders its applicability to the Big Data context. Indeed, Big Data datasets are huge by definition, and the storage of dataset replica per user or group of users would introduce hardly manageable storage and temporal overheads.

In contrast, query rewriting has been recognized as an effective enforcement mechanisms for several access control models (e.g., [14]). With this approach, a submitted query is rewritten in such a way that 1) it only accesses data that would have been accessed by the original query and 2) the access complies with the policies specified for such data. It is worth noting that the result set generated with the execution of a rewritten query is the same as the result set of the original query executed on an authorized view of the accessed tables. The advantages of the second approach is that it does not require any data duplication, and the cost of rewriting a query is generally negligible wrt the execution time. Thus, although the two approaches derive the same result sets, the latter has relevant advantages in terms of enforcement overhead.

As a matter of fact, query rewriting has been used with a variety of access control models, including privacy aware ones. For instance, the purpose-based model in [14] uses query rewriting

[7] http://www.oracle.com/technetwork/database/security/index-088277.html.

for enforcing policies, as well as the action aware purpose-based model that we proposed in [13].

Rizvi et al. [36] discuss an issue related to query rewriting, arguing that the approach returns non-consistent results in case the query performs data aggregations and the rewritten query does not aggregate all values which are aggregated by the original query. To address this issue, they propose an approach where several views are assigned to each user and a query is only executed (with no need of rewriting) if the authorized views include all data to be aggregated. Although the approach in [36] allows to achieve more accurate results, it suffers from a high overhead related to the management of multiple authorized views for every user, thus we do not consider it as an effective solution for Big Data platforms.

We believe that the only solutions that can work within Big Data are those that do not require storing additional data for enforcement purposes. As such, query rewriting appears as a possible option for NoSQL datastores.

Query rewriting techniques favor the preventive filtering of the accessed dataset and reduce the quantity of data to be processed. In addition they can be defined in such a way to exploit the distributed processing capabilities of host Big Data platforms, allowing the parallel computation among the nodes that compose the Big Data clusters (e.g., using Map Reduce jobs).

Goal 11: Resource based key value filters for MapReduce systems

Query rewriting is not suited for MapReduce systems. Let us consider, for instance, MapReduce Jobs of Hadoop platforms. MapReduce Jobs may be seen as queries performing data aggregation, but, different from SQL queries, their source code is not available as they are provided as input to Hadoop as binary JAR packages. In this case, it is hard to derive the instructions and rewrite them, other techniques should be considered. For instance, in [15], the enforcement is achieved by pre-filtering the key-value pairs to be analyzed using an aspect oriented approach which complement the Job instructions with policy enforcement operations. We believe that a mechanism similar to those proposed in [15] should also be defined to enforce PAAC policies.

Although the enforcement mechanisms varies according to the considered platforms, query languages and data models, some requirements are in common with any system. The proposed enforcement techniques must be preventive, blocking query execution in case of insufficient permissions, and filtering the accessible data of the considered data sources.

We believe that post-execution filtering mechanisms are unsuited to the Big Data scenario. The reasons are related to the complexity of the analytics tasks and queries that characterize this application domain and the enforcement overhead. Indeed, the data that are included in the result set of the queries are typically derived through the execution of operation pipelines where aggregation, selection, and projection functions are invoked within complex processes. Each datum in the result set can be derived from a variety of different sources and for each of these sources a different privacy policy may have been defined. When post execution filtering mechanisms are executed the result set is filtered after query execution. This implies that the query is first executed accessing all the data that satisfy the query selection criteria independently from the specified policies, and then the query result set is filtered by evaluating for each datum in the result set if all privacy policies specified for its data sources comply with the access. Conversely, when pre-filtering techniques are applied, queries only process those data for which the corresponding privacy policies grant the access. Thus, in this case policy selectivity can drastically cut the volume of data which is effectively processed. As a consequence, the execution time in scenarios where post execution filtering mechanisms are used is expected to be far higher than in scenarios where pre-filtering mechanisms preselect the data sources which can be used for query execution.

4. Towards the integration of PAAC into MongoDB

In this section, we shortly present some preliminary considerations related to the application of the roadmap introduced in Section 2 for integrating PAAC into MongoDB. The goal of this section is showing how the proposed general tasks and goals can be refined and specialized for a target platform. We do not aim at discussing development details or the specification of advanced mechanisms, rather we want to provide simple hints of how a selection of the goals[8] presented in Section 3 could be achieved with a popular NoSQL datastore.

In the remaining of this section a simple running example is used to ease the presentation of the proposed concepts and mechanisms. More precisely, we consider a MongoDB database *patients* including a collection *medical-records*, which groups the documents representing the medical charts of the patients of a clinic.

MongoDB natively integrates a role based access control model (RBAC). Thus, according to Goal 1, PAAC mechanisms should be defined on top of MongoDB RBAC by reusing the natively provided data protection mechanisms. Within MongoDB RBAC, access regulation is achieved by granting or denying the access to all documents of the collection accessed by the submitted queries. However, in order to support privacy policies, finer grained specification and enforcement mechanisms must be defined. For the sake of simplicity, in this preliminary analysis, let us assume MongoDB documents as the finest granularity level for PAAC.[9]

Therefore, in order to achieve Goal 7, policy binding should be defined at the level of collection and document. In the former case, policy can be specified as MongoDB documents and stored within a dedicated system collection *policyAtCollection*, whereas, in the latter case, they can be specified within a dedicated field *policy* of the protected documents. The former binding choice allows specifying policies regulating the access to all documents in a collection, or to documents with specific structural characteristics or content, whereas the latter allows regulating the access to single documents. Referring to our running example, privacy policies can regulate the access to the whole collection *medical-records*, to a subset of the collected medical charts with given structural characteristics (e.g., all medical charts of underage patients), or to the medical charts of individual patients.

According to Goal 2, the proposed enhanced model has to support purpose based policies. To this aim, borrowing the purpose modeling proposed in [14], the concept of *purpose* can be used for defining the reasons for which MongoDb collections, or single MongoDB documents, can be accessed, referred to as *intended purposes*, as well as for declaring the reasons for which users aim at accessing them through query execution, referred to as *access purposes*. The access to data resources is subject to purpose based authorizations granted to users or roles. A *purpose authorization* is a permit that authorizes a given user or role to access a data resource for a specified access purpose. For instance, considering the running example, role *physician* is authorized to access *medical-records* for *diagnostic* purposes. Similar to MongoDB role privileges, purpose authorizations for requesting users and roles can be stored within a dedicated system collection *purposeAuth*.

Let us now consider how Goal 4 can be achieved. The access purpose for which users aim at executing a query *q* can be modeled as a property of the context within which *q* is invoked for

[8] The proposed goals have been selected for exemplification purposes since we consider them among the most significant for common application scenarios where MongoDB is used.

[9] This granularity level is equivalent to the row level within relational DBMSs.

execution. As a consequence of this modeling choice, the intended purposes for which data resources (i.e., collections or documents) can be accessed can be specified as context-based constraints. For instance, we can assume that an administrator authorizes the access to *medical-records* for access purposes compliant with *diagnostic* and *scientific* purposes. Required condition to grant the access is the compliance of the access purposes belonging to the contextual information associated with the query execution request with the intended purposes specified for the accessed collections or documents. For instance, referring to the previous example, let us assume that Mary, a receptionist operating in the clinic, aims at executing a query that accesses *medical-records* for *reservation* purposes. On the basis of previous reasoning, the access purpose *reservation* needs to be checked for compliance with the intended purposes specified for *medical-records* and the query can only be executed in case of compliance.

In contrast, content based constraints of privacy policies regulating the access to a collection *cl* specify required structural properties of documents in *cl* for being accessed. Content based constraints are logic predicates built on top of document fields identifiers and fields values. For instance, let us assume that in our application scenario attending physicians can only access medical records of their patients, and let us also assume that *Bob* is a family doctor operating within the clinic. A content based constraint of a policy specified for *medical-records* can be defined to grant to *Bob* the access to only those documents of *medical-records* that includes the field *attending-physician* set to *Bob*.

A MongoDB deployment is a distributed system where several clients interact with a server using a communication protocol, called Wire. PAAC can be enforced by integrating an enforcement monitor operating as a proxy that analyzes and manipulates the Wire messages that are exchanged by the MongoDB clients and the server. In order to fulfill Goal 10, when a message carrying a query execution request is intercepted, the message must be rewritten in such a way that the query accesses a subset of the documents accessed by the original query, that is, only those documents for which the specified policies (at collection and document level) grant the access.

Let us denote with r a Wire message carrying a query q, and with u a user requiring r execution, and let us suppose that q accesses a database collection cl, by selecting documents on the basis of selection criteria sc. Let ap be the access purpose associated with q execution request. In order to check whether u is authorized for ap, the monitor has to verify if a proper authorization exists for u in *purposeAuth*. For instance, let us consider two different cases, the former where q represents a query execution request targeting *medical-records* and issued by *Mary* for *reservation* purposes, and the latter where the same query q is issued by *Bob* for *diagnostic* purposes. Based on previous reasoning, it is required to check that *Mary*/*Bob* is authorized to access *medical-records* for *reservation*/*diagnostic* purposes. If ap has not been authorized for the requesting user u, the execution of q is aborted (e.g., this may be the case of the request issued by *Mary*), otherwise r is rewritten as r' (e.g., for the case of the request issued by *Bob*), according to the following strategy. First, the monitor accesses *policyAtCollection* to derive the content based constraints specified for cl when accessed by queries issued by u. The rewriting is achieved by defining r' in such a way that while achieving all processing activities of q, the selection criteria sc of r are enhanced with content and context based constraints that regulate u's visibility on cl documents. More precisely, content based constraints specified for cl are conjuncted with a purpose based compliance checks which evaluates the compliance of ap with the intended purposes specified for each accessed document within the dedicated field *policy*. For instance, considering the above mentioned query invoked by *Bob*, due to the content based constraint of the policy specified for *medical-records*, sc is enhanced with criteria that select all documents including field *attending-physician* initialized to *Bob*. In addition, due to the context based constraints specified for the documents in *medical-records*, sc is also enhanced with checks that verify the compliance of the access purpose *diagnostic* with the intended purposes of each accessed document in *medical-records*.

Once rewritten, r' is forwarded to the server which executes the specified query. The result set of r' is only derived from documents whose access has been authorized for u. As such, considering our example, the result set of the rewritten query will only be derived from the medical charts of *Bob*'s patients who has granted the access to their medical-records for intended purposes compliant with the purpose *diagnostic*.

The interested reader can find additional information related to the implementation of the proposed approach for MongoDB in a previous work by us [37]. Although in [37] we focused on fine grained access control, several architectural, design, and implementation choices related to the development of the enforcement monitor can be reused for PAAC as well.

5. Conclusions

Nowadays Big Data represent one of the great new frontiers of IT [38]. IT managers recognize the use of Big Data platforms as an asset for their organizations, however, they also consider the poor security and privacy enforcement practices of these platforms as a top obstacle to the adoption of these solutions within their companies [6]. The definition of data security and privacy standards represent a challenging open issue for the research community [32], as traditional security and privacy mechanisms tailored to traditional data management systems are inadequate for Big Data [32].

This paper presents the foundations of a framework for the integration of PAAC into MapReduce systems and NoSQL datastores. A research and development roadmap is proposed and aspects related to the framework definition are discussed. The roadmap tasks cover a variety of activities, such as the selection of target platforms, the identification of policies suited for common application scenarios of BigData platforms, and the definition of policy specification and enforcement mechanisms. Several research goals related to the framework definition are discussed, such as the desired characteristics of privacy policies, policy binding and specification criteria for policies operating in MapReduce systems and NoSQL datastores, and general guidelines for the definition of enforcement mechanisms. Finally, in order to show the applicability of the proposed framework, some preliminary considerations related to the enhancement of a popular NoSQL datastore with PAAC features are provided.

The potential application scenarios of the proposed framework match those within which NoSQL datastores and MapReduce systems are currently used.

Acknowledgements

The research presented in this paper has been partially funded by the IBM Faculty Award "Privacy-preserving Big Data Analytics".

References

[1] V. Mayer-Schönberger, K. Cukier, Big Data: A Revolution That Will Transform How We Live, Work, and Think, Houghton Mifflin Harcourt, 2013.
[2] X. Jin, B.W. Wah, X. Cheng, Y. Wang, Significance and challenges of big data research, Big Data Res. 2 (2) (2015) 59–64, http://dx.doi.org/10.1016/j.bdr.2015.01.006.
[3] J. Holler, V. Tsiatsis, C. Mulligan, S. Avesand, S. Karnouskos, D. Boyle, From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, Academic Press, 2014.

[4] A. Gandomi, M. Haider, Beyond the hype: big data concepts, methods, and analytics, Int. J. Inf. Manag. 35 (2) (2015) 137–144, http://dx.doi.org/10.1016/j.ijinfomgt.2014.10.007.
[5] E. Begoli, A short survey on the state of the art in architectures and platforms for large scale data analysis and knowledge discovery from data, in: Proceedings of the WICSA/ECSA 2012 Companion Volume, WICSA/ECSA '12, ACM, New York, NY, USA, 2012, pp. 177–183, http://dx.doi.org/10.1145/2361999.2362039.
[6] I. Co, Intel's it manager survey on how organizations are using big data, http://www.intel.com/content/dam/www/public/us/en/documents/reports/data-insights-peer-research-report.pdf, 2013.
[7] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, J. Abramov, Security issues in nosql databases, in: 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2011, pp. 541–547.
[8] Q. Ni, A. Trombetta, E. Bertino, J. Lobo, Privacy-aware role based access control, in: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, ACM, 2007.
[9] Q. Ni, D. Lin, E. Bertino, J. Lobo, Conditional privacy-aware role based access control, in: Computer Security–ESORICS, 2007.
[10] P. Colombo, E. Ferrari, Enforcement of purpose based access control within relational database management systems, IEEE Trans. Knowl. Data Eng. 26 (11) (2014), http://dx.doi.org/10.1109/TKDE.2014.2312112.
[11] President's Council of Advisors on Science and Technology, Big data and privacy: a technological perspective, Tech. rep., Executive Office of the President of the United States, 2014.
[12] I. Co, Big data in the cloud: converging technologies, http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/big-data-cloud-technologies-brief.pdf, 2015.
[13] P. Colombo, E. Ferrari, Efficient enforcement of action aware purpose based access control within relational database management systems, IEEE Trans. Knowl. Data Eng. (2015), in press.
[14] J. Byun, N. Li, Purpose based access control for privacy protection in relational database systems, VLDB J. 17 (4) (2008).
[15] H. Ulusoy, P. Colombo, E. Ferrari, M. Kantarcioglu, E. Pattuk, GuardMR: fine-grained security policy enforcement for MapReduce systems, in: ACM ASIACCS, 2015, in press.
[16] S. Sen, S. Guha, A. Datta, S.K. Rajamani, J. Tsai, J.M. Wing, Bootstrapping privacy compliance in big data systems, in: Proceedings of the 2014 IEEE Symposium on Security and Privacy, SP '14, IEEE Computer Society, 2014.
[17] E. Bertino, J.-W. Byun, N. Li, Privacy-preserving database systems, in: Foundations of Security Analysis and Design III, Springer, 2005, pp. 178–206.
[18] OASIS, Extensible access control markup language (xacml), http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf, January 2013, ver. 3.0.
[19] W3C, Enterprise privacy authorization language (epal 1.2), http://www.w3.org/Submission/EPAL/, 2003.
[20] M.E. Kabir, H. Wang, Conditional purpose based access control model for privacy protection, in: Proceedings of the Twentieth Australasian Conference on Australasian Database, vol. 92, Australian Computer Society, Inc., 2009, pp. 135–142.
[21] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, R. Chandramouli, Proposed nist standard for role-based access control, ACM Trans. Inf. Syst. Secur. 4 (3) (2001).
[22] M. Kabir, H. Wang, E. Bertino, A role-involved conditional purpose-based access control model, in: M. Janssen, W. Lamersdorf, J. Pries-Heje, M. Rosemann (Eds.), E-Government, E-Services and Global Processes, in: IFIP Advances in Information and Communication Technology, vol. 334, Springer, Berlin, Heidelberg, 2010, pp. 167–180.
[23] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, A. Trombeta, Privacy-aware role-based access control, ACM Trans. Inf. Syst. Secur. 13 (3) (2010) 24.
[24] H. Peng, J. Gu, X. Ye, Dynamic purpose-based access control, in: International Symposium on Parallel and Distributed Processing with Applications, ISPA '08, 2008, pp. 695–700, http://dx.doi.org/10.1109/ISPA.2008.80.
[25] P. Colombo, E. Ferrari, Enforcing obligations within relational database management systems, IEEE Trans. Dependable Secure Comput. 11 (4) (2014), http://dx.doi.org/10.1109/TDSC.2013.48.
[26] Q. Ni, E. Bertino, J. Lobo, An obligation model bridging access control policies and privacy policies, in: ACM SACMAT, 2008.
[27] N. Li, H. Chen, E. Bertino, On practical specification and enforcement of obligations, in: ACM CODASPY, 2012.
[28] B. Shebaro, O. Oluwatimi, E. Bertino, Context-Based Access Control Systems for Mobile Devices, IEEE, 2014.
[29] M.L. Damiani, E. Bertino, B. Catania, P. Perlasca, Geo-rbac: a spatially aware rbac, ACM Trans. Inf. Syst. Secur. 10 (1) (2007) 2.
[30] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni, A survey of context modelling and reasoning techniques, Pervasive Mob. Comput. 6 (2) (2010) 161–180.
[31] E. Bertino, E. Ferrari, A. Squicciarini, Trust negotiations: concepts, systems, and languages, Comput. Sci. Eng. 6 (4) (2004) 27–34.
[32] C.S. Alliance, Top ten big data security and privacy challenges, https://cloudsecurityalliance.org/download/top-ten-big-data-security-and-privacy-challenges/, 2012.
[33] D. Kulkarni, A fine-grained access control model for key-value systems, in: Proceedings of the third ACM Conference on Data and Application Security and Privacy, ACM, 2013, pp. 161–164.
[34] M. Shermin, An access control model for nosql databases, Master's thesis, University of Western Ontario, 2013, Electronic Thesis and Dissertation Repository. Paper 1797, http://ir.lib.uwo.ca/etd/1797.
[35] H. Ulusoy, M. Kantarcioglu, K. Hamlen, E. Pattuk, Vigiles: fine-grained access control for mapreduce systems, in: IEEE BigData, 2014.
[36] S. Rizvi, A. Mendelzon, S. Sudarshan, P. Roy, Extending query rewriting techniques for fine-grained access control, in: ACM SIGMOD 2004, 2004, pp. 551–562.
[37] P. Colombo, E. Ferrari, Complementing mongodb with advanced access control features: concepts and research challenges, in: 23rd Italian Symposium on Advanced Database Systems (SEBD 2015), 2015, in press.
[38] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A.H. Byers, Big data: the next frontier for innovation, competition, and productivity, Tech. rep., McKinsey Global Institute, 2011.